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

(10) **DE 102 55 142 B4** 2008.01.03

(12) # Patentschrift

(21) Aktenzeichen: **102 55 142.1**
(22) Anmeldetag: **26.11.2002**
(43) Offenlegungstag: **09.10.2003**
(45) Veröffentlichungstag
der Patenterteilung: **03.01.2008**

(51) Int Cl.[8]: ***H04L 12/56*** (2006.01)
H04L 29/14 (2006.01)
H04L 12/26 (2006.01)

Innerhalb von drei Monaten nach Veröffentlichung der Patenterteilung kann nach § 59 Patentgesetz gegen das Patent Einspruch erhoben werden. Der Einspruch ist schriftlich zu erklären und zu begründen. Innerhalb der Einspruchsfrist ist eine Einspruchsgebühr in Höhe von 200 Euro zu entrichten(§ 6 Patentkostengesetz in Verbindung mit der Anlage zu § 2 Abs. 2 Patentkostengesetz).

(30) Unionspriorität:
10/099335 14.03.2002 US

(73) Patentinhaber:
Agilent Technologies, Inc. (n.d.Ges.d. Staates Delaware), Santa Clara, Calif., US

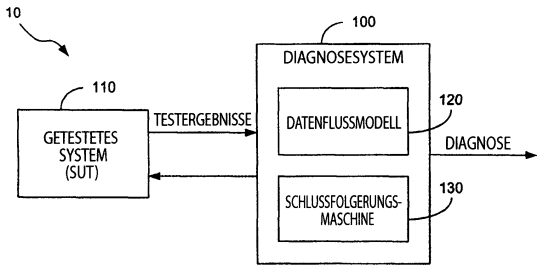
(74) Vertreter:
Schoppe, Zimmermann, Stöckeler & Zinkler, 82049 Pullach

(72) Erfinder:
Manley, Douglas R., Fort Collins, Col., US; Barford, Lee A., San Jose, Calif., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
US 58 08 919 A
U6 63 27 545 B1

(54) Bezeichnung: **Diagnose von Datenpaketübertragungs-Fehlern unter Verwendung von Einschränkungen**

(57) Hauptanspruch: Verfahren zum Diagnostizieren von Datenpaketübertragungs-Fehlern in einem getesteten System (SUT) (110), wobei das SUT Datenübertragungswege definiert, durch die Datenpakete übertragen werden, wobei das Verfahren folgende Schritte aufweist:
Identifizieren von zumindest einigen Abschnitten der Datenübertragungswege das SUT, die Fehler in die Datenpaketübertragung einbringen können;
Bereitstellen von Einschränkungen, die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege, die identifiziert wurden, definiert;
Diagnostizieren des SUT im Hinblick auf die Einschränkungen;
wobei ein Teststimulus, der als Eingabe für das Diagnostizieren des SUT verwendet wird, ein Modell einer tatsächlichen Eingabe, die dem SUT während des Betriebes zugeführt wird, ist.



Beschreibung

[0001] Die vorliegende Erfindung bezieht sich allgemein auf eine Systemfehlerdiagnose. Speziell bezieht sich die vorliegende Erfindung auf Systeme und Verfahren, die die Diagnose von Fehlern bei mehreren diskreten Datenübertragungen zwischen Abschnitten eines Systems involvieren.

[0002] Verschiedene Systeme und Verfahren sind zum Diagnostizieren von Fehlern, die bei SUTs (SUT = systems under test = getestete Systeme) vorkommen, verwendet worden. Dazu sind beispielsweise eine manuelle Diagnose, eine automatisierte Diagnose basierend auf einer testmodellbasierten Technologie, kundenspezifischen Software und Fehlersimulation verwendet worden. Tendenziell weisen diese Techniken jedoch einen oder mehrere festgestellte Nachteile auf, die deren Anwendbarkeit tendenziell einschränken können.

[0003] Im Hinblick auf die manuelle Diagnose ist diese Technik typischerweise eine wissensintensive Technik, die ein hohes Niveau an Wissen bezüglich SUT und Testfolge erfordert. Der Erwerb derartigen Wissens durch einen Operator kann zeitaufwendig und daher kostspielig sein. Zusätzlich sind Ergebnisse, die während einer Diagnose erhalten werden, typischerweise dahingehend nicht wiederholbar, daß die Ergebnisse von Operator zu Operator und/oder Standort zu Standort variieren können. Eine solche Technik kann auch gewissermaßen fehleranfällig sein, da eine unsachgemäße Anwendung der Technik eine ungenaue Fehlerdiagnose zur Folge haben kann.

[0004] Viele Formen einer testmodellbasierten Diagnose, obgleich dieselben für das Diagnostizieren von statischen Fehlern als angemessen erachtet werden, erweisen sich tendenziell als zur Verwendung beim Diagnostizieren von zeitweise auftretenden Fehlern unwirksam. Ein statischer Fehler ist ein Fehler, der während eines gesamten Tests vorliegt und typischerweise alle Datenübertragungen während des Tests beeinträchtigt, wohingegen ein zeitweise auftretender Fehler typischerweise nur einen Teil der Datenübertragungen beeinträchtigt. Im Vergleich zum Verfolgen eines speziellen Abschnitts und/oder einer Komponente des Testwegs verfolgen testmodellbasierte Techniken tendenziell einen gesamten Testweg, wenn ein Fehler in bezug auf diesen Testweg diagnostiziert wird. Zusätzlich erfordert die testmodellbasierte Diagnose typischerweise die Entwicklung eines detaillierten Modells der Tests für ein System. Ein Beispiel von testmodellbasierten Systemen ist in der US-Patentanmeldung 5,808,919 A offenbart.

[0005] Kundenspezifische Software wird ebenfalls zum Diagnostizieren von Systemen verwendet. Leider ist die kundenspezifische Software typischerweise geschrieben, um nur ein spezifisches System zu diagnostizieren. Dieser Lösungsansatz ist tendenziell aufwendig und daher in seiner Implementierung kostspielig.

[0006] Wie ebenfalls bekannt ist, können bei einer Systemdiagnose Fehlersimulatoren verwendet werden. Die Fehlersimulatoren arbeiten typischerweise durch Erzeugen eines Fehlerwörterbuchs. Die Fehlersimulation erfordert jedoch typischerweise eine große Menge an Modellierzeit und relativ umfangreiche Ausführungszeiten, speziell wenn komplexe Schaltungen durch das SUT genutzt werden. Dies ist darin begründet, daß die Simulation typischerweise eine Bit-um-Bit-Analyse des SUT-Betriebs involviert. Aufgrund dessen gilt eine Fehlersimulation für die Verwendung bei komplexen, handelsüblichen Anwendungen typischerweise als unpraktisch. Zusätzlich ist eine Fehlersimulation nichtexistent oder gilt anderweitig als zur Diagnose von zeitweise auftretenden Ausfällen unpraktikabel.

[0007] Anhand des vorstehenden wird darauf hingewiesen, daß ein Bedarf an verbesserten Systemen und Verfahren besteht, die die oben erwähnten und/oder andere festgestellte Nachteile des Stands der Technik angehen.

[0008] US 6,327,545 B1 offenbart ein System zum Testen einer Schaltung. Um die Komplexität des Tests zu reduzieren, kann die Schaltung in Gruppen von Komponenten, sogenannte Cluster, unterteilt werden, die getrennt voneinander getestet werden. Zum Testen werden die Cluster weiter unterteilt, bis hinab auf die zu testenden Knoten. Diese können mit speziell optimierten Teststimuli getestet werden, die in Bezug auf Testdurchsatz oder Testabdeckung optimiert sind.

[0009] Es ist eine Aufgabe der vorliegenden Erfindung, Systeme und Verfahren zur Diagnose von Datenpaketübertragungs-Fehlern unter Verwendung von Einschränkungen zu schaffen.

[0010] Diese Aufgabe wird durch ein Verfahren gemäß Anspruch 1 sowie ein System gemäß den Ansprüchen 13, 20 oder 22 gelöst.

[0011] Die vorliegende Erfindung bezieht sich auf die Diagnose von Fehlern in Datenpaketübertragungen eines SUT. Typischerweise verwendet die Erfindung Einschränkungen, um Datenpaketübertragungs-Beziehungen aus verschiedenen Abschnitten des SUT zu definieren. Diese Einschränkungen können dann im Hinblick auf die Testergebnisse, die vom SUT erhalten werden, ausgewertet werden.

[0012] Bei einigen Ausführungsbeispielen wird ein Datenflußmodell verwendet, um jene Abschnitte eines SUT zu identifizieren, die Datenpaketübertragungs-Fehler einführen können. Einschränkungen werden dann entwickelt, um die Datenpaketübertragungs-Beziehungen aus den identifizierten Abschnitten zu definieren. Daher, wenn dem SUT entsprechende Testergebnisse empfangen werden und ein Datenpaketübertragungs-Fehler erfaßt wird, können die Einschränkungen im Hinblick auf die Testergebnisse unter Verwendung des Datenflußmodells ausgewertet werden, um Komponenten und/oder Teilkomponenten des SUT, die die Datenpaketübertragungs-Fehler erzeugt haben könnten, zu identifizieren und/oder auszuschließen.

[0013] Verschiedene Techniken können zum Bestimmen einer Diagnose verwendet werden. Beispielsweise kann eine lineare Programmierung, wie z. B. eine Integer-Programmierung, eine regelbasierte Kantenklassifizierung und/oder eine flußereignisbasierte Kantenklassifikation verwendet werden.

[0014] Bei einigen Ausführungsbeispielen können jene Abschnitte eines SUT, die Daten, z. B. Datenpakete, zählen können und/oder eine Operation im Hinblick auf die Daten ausführen können, ebenfalls identifiziert werden. Zum Beispiel könnte eine solche Operation ein Reproduzieren (Bussing), Aufteilen, Kombinieren, Fallenlassen und/oder Routen (Schalten) von Daten umfassen.

[0015] Diesbezüglich können die Ausführungsbeispiele der Erfindung als Verfahren zum Diagnostizieren von Datenpaketübertragungs-Fehlern in einem SUT ausgelegt werden. Speziell umfaßt ein solches Verfahren folgende Schritte: ein Identifizieren von zumindest einigen Abschnitten der Datenübertragungswege des SUT, die Fehler bei einer Datenpaketübertragung einführen kann; ein Bereitstellen von Einschränkungen, die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege definieren, und ein Diagnostizieren des SUT im Hinblick auf die Einschränkungen.

[0016] Ausführungsbeispiele der Erfindung können auch als Systeme zum Diagnostizieren von Datenpaketübertragungs-Fehlern in einem SUT ausgelegt werden. Ein solches System umfaßt ein Datenflußmodell und eine Schlußfolgerungsmaschine. Das Datenflußmodell stellt Datenübertragungsfähigkeiten des SUT dar. Die Schlußfolgerungsmaschine ist angepaßt, um Testergebnisse entsprechend des SUT in Relation zum Datenflußmodell auszuwerten.

[0017] Ein weiteres System zum Diagnostizieren von Fehlern umfaßt eine Einrichtung zum Empfangen von Testergebnissen entsprechend den Übertragungen von Datenpaketen durch zumindest einige Abschnitte der Datenübertragungswege des SUT und eine Einrichtung zum Diagnostizieren des SUT im Hinblick auf die Einschränkungen, die die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege des SUT definieren.

[0018] Noch weitere Ausführungsbeispiele der Erfindung können als Diagnosesysteme ausgelegt werden, wobei zumindest einige von ihnen auf einem computerlesbaren Medium gespeichert sein können. Ein solches Diagnosesystem umfaßt eine Logik, die konfiguriert ist, um zumindest einige Abschnitte der Datenübertragungswege des SUT zu identifizieren, die zum Einbringen von Fehlern in die Datenpaketübertragung fähig sind; eine Logik, die konfiguriert ist, um Einschränkungen zu liefern, die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege zu definieren; und eine Logik, die konfiguriert ist, um das SUT im Hinblick auf die Einschränkungen zu diagnostizieren.

[0019] Es ist klar, daß die Ausführungsbeispiele der Erfindung Merkmale und/oder Vorteile neben oder anstelle jener, die oben aufgeführt sind, aufweisen können. Zusätzlich werden einem Fachmann nach einer Untersuchung der nachstehenden Zeichnungen und ausführlichen Beschreibung weitere Systeme, Verfahren, Merkmale und/oder Vorteile der vorliegenden Erfindung offenbar. Solche zusätzlichen Systeme, Verfahren, Merkmale und/oder Vorteile sollen innerhalb dieser Beschreibung umfaßt sein, sich innerhalb des Schutzbereichs der vorliegenden Erfindung befinden und durch die beigefügten Ansprüche geschützt sein.

[0020] Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend Bezug nehmend auf die beiliegenden Zeichnungen, die nicht als maßstabsgerecht zu verstehen sind, näher erläutert. Es zeigen:

[0021] Fig. 1 ein schematisches Diagramm, das ein Ausführungsbeispiel eines Systems der vorliegenden Erfindung darstellt, das ein Ausführungsbeispiel eines Diagnosesystems umfaßt, das verwendet wird, um ein getestetes System zu testen,

[0022] Fig. 2 ein Flußdiagramm, das eine Funktionalität eines Ausführungsbeispiels des Diagnosesystems der vorliegenden Erfindung darstellt,

[0023] Fig. 3 ein computer- oder prozessorbasiertes System, das zum Implementieren eines Ausführungsbeispiels des Diagnosesystems der vorliegenden Erfindung verwendet werden kann,

[0024] Fig. 4 ein Flußdiagramm, das eine Funktionalität des Ausführungsbeispiels des Diagnosesystems von Fig. 4 darstellt,

[0025] Fig. 5 einen gerichteten Graphen, der ein Ausführungsbeispiel eines Datenflußmodells darstellt, das durch ein Diagnosesystem der vorliegenden Erfindung verwendet werden kann,

[0026] Fig. 6 ein Blockdiagramm, das ein repräsentatives getestetes System darstellt,

[0027] Fig. 7 einen gerichteten Graphen, der ein Ausführungsbeispiel eines Datenflußmodells darstellt, das durch ein Diagnosesystem der vorliegenden Erfindung verwendet werden kann, um das SUT von Fig. 6 zu diagnostizieren,

[0028] Fig. 8 einen weiteren gerichteten Graphen, der ein Ausführungsbeispiel eines Datenflußmodells darstellt, das durch ein Diagnosesystem der vorliegenden Erfindung verwendet werden kann, um das SUT von Fig. 6 zu diagnostizieren.

[0029] Wie hierin ausführlicher beschrieben ist, ermöglichen die Systeme und Verfahren der vorliegenden Erfindung potentiell die Fehlerdiagnose von getesteten Systemen, die der Datenübertragung zugeordnet sind. Speziell können die Einschränkungen, die die Beziehungen zwischen den verschiedenen Abschnitten von Datenübertragungswegen eines SUT darstellen, verwendet werden, um Fehlerkandidaten- oder Abschnitte des SUT, die möglicherweise für die erfaßten Fehler verantwortlich sind, abzuleiten und/oder auszuschließen. Die Einschränkungen, die Datenflußfunktionalität des SUT definieren, können verwendet werden, um Regeln und/oder Gleichungen abzuleiten, die z. B. beschreiben, wie Daten durch das SUT fließen sollen. Typischerweise wird ein Datenflußmodell, das das fehlerfreie Datenpaketübertragungs-Verhalten des SUT darstellt, verwendet. Bei einem solchen Ausführungsbeispiel kann das SUT unter Verwendung des Datenflußmodells und einer zugeordneten Schlußfolgerungsmaschine diagnostiziert werden. Bei einigen Ausführungsbeispielen können die diagnostizierten Fehler im SUT mit Geschwindigkeit und/oder zeitweise auftreten.

[0030] Unter Bezugnahme auf die Zeichnungen, bei denen identische Bezugszeichen entsprechende Komponenten in den verschiedenen Ansichten anzeigen, ist Fig. 1 ein schematisches Diagramm, das ein Ausführungsbeispiel eines Systems **10** der vorliegenden Erfindung darstellt. Spezieller umfaßt das System ein Diagnosesystem **100**, das mit einem SUT **110** kommuniziert. Das Diagnosesystem **100** umfaßt ein Datenflußmodell **120** und eine Schlußfolgerungsmaschine **130**. Das Datenflußmodell **120** beschreibt die Flüsse von Daten, die dem SUT zugeordnet sind, und die Schlußfolgerungsmaschine **130** interpretiert die Testergebnisse relativ zum Datenflußmodell, wie später ausführlich beschrieben ist. Vorzugsweise umfaßt eine Ausgabediagnose des Diagnosesystems **100** einen Hinweis über Komponenten und/oder Teilkomponenten, deren Ausfall zu den beobachteten Testergebnissen geführt haben könnte.

[0031] Bei einigen Ausführungsbeispielen kann das Diagnosesystem indirekt mit dem SUT kommunizieren. Zum Beispiel könnte das SUT Informationen, z. B. Testergebnisse, an ein anderes System oder Programm liefern, wobei die Informationen dann an das Diagnosesystem zur Analyse geliefert werden.

[0032] Ein Flußdiagramm, das die Funktionalität eines Ausführungsbeispiels des Systems von **Fig.** 10 der vorliegenden Erfindung darstellt, ist in Fig. 2 gezeigt. Wie in Fig. 2 gezeigt ist, kann das System oder Verfahren **10** so ausgelegt werden, als ob es bei Block **210** beginnt, wo zumindest einige Abschnitte der Datenübertragungswege eines SUT identifiziert werden. Spezieller können die identifizierten Abschnitte des SUT Fehler in die Datenübertragung einbringen. Bei Block **220** werden Einschränkungen, die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege definieren, geliefert. Anschließend, wie in Block **230** dargestellt ist, wird das SUT im Hinblick auf die Einschränkungen diagnostiziert.

[0033] Die Diagnosesysteme **100** können in einer Software, Firmware, Hardware oder einer Kombination aus denselben implementiert sein. Wenn das Diagnosesystem **100** in einer Hardware implementiert ist, kann es mit einer beliebigen oder einer Kombination aus verschiedenen Technologien implementiert sein. Die nachstehenden Technologien, die in der Technik hinreichend bekannt sind, können beispielsweise verwendet werden als: diskrete logische Schaltungen mit logischen Gattern zum Implementieren von logischen Funktionen auf Datensignale hin, eine ASIC (ASIC = application specific integrated circuit = anwendungsspezifische integrierte Schaltung) mit entsprechenden kombinatorischen logischen Gattern, ein PGA (PGA = programmable gate array = programmierbares Gatterarray) und ein FPGA (FPGA = field programmable gate array = feldprogrammierbares Gatterarray).

[0034] Wenn das Diagnosesystem **100** in einer Software implementiert ist, kann es ein Programm sein, das durch einen Computer oder eine prozessorbasierte Vorrichtung ausführbar ist. Ein Beispiel eines solchen Computers oder einer prozessorbasierten Vorrichtung wird nun unter Bezugnahme auf das schematische Diagramm von **Fig. 3** beschrieben.

[0035] Im Hinblick auf die Hardwarearchitektur umfaßt der Computer **300** von **Fig. 3** allgemein einen Prozessor **302**, einen Speicher **304** und eine oder mehrere I/O-Vorrichtungen (I/O = input/output = Eingabe/Ausgabe) **306** (oder Peripheriegeräte), die über eine lokale Schnittstelle **308** kommunikativ gekoppelt sind. Die lokale Schnittstelle **308** kann z. B. ein oder mehrere Busse oder andere verdrahtete oder drahtlose Verbindungen sein, wie in der Technik bekannt ist. Die lokale Schnittstelle **308** kann zusätzliche Elemente umfassen, die zur Vereinfachung der Beschreibung ausgelassen worden sind. Diese zusätzlichen Elemente können beispielsweise Steuerungen, Puffer (Caches), Treiber, Repeater (Zwischenverstärker) und/oder Empfänger sein. Ferner kann die lokale Schnittstelle Adreß-, Steuerungs- und/oder Datenverbindungen umfassen, um entsprechende Kommunikation unter den Komponenten des Computers **300** zu ermöglichen.

[0036] Der Prozessor **302** kann eine Hardwarevorrichtung sein, die konfiguriert ist, um eine Software auszuführen, die im Speicher **304** gespeichert ist. Der Prozessor **302** kann ein beliebiger kundenspezifisch gefertigter oder im Handel erhältlicher Prozessor, eine CPU (CPU = central processing unit = zentrale Verarbeitungseinheit) oder ein Hilfsprozessor von verschiedenen Prozessoren sein. Zusätzlich kann der Prozessor beispielsweise ein halbleiterbasierter Mikroprozessor (in der Form eines Mikrochips) sein.

[0037] Der Speicher **304** kann eine beliebige Kombination aus flüchtigen Speicherelementen (z. B. ein Direktzugriffsspeicher (RAM, wie DRAM, SRAM etc.)) und/oder nichtflüchtigen Speicherelementen (z. B. ein ROM, Festplattenlaufwerk, Band, CDROM etc.) umfassen. Außerdem kann der Speicher **504** elektronische, magnetische, optische und andere Typen von Speicherungsmedien umfassen. Es ist zu beachten, daß der Speicher **304** eine verteilte Architektur aufweisen kann, wohingegen die verschiedenen Komponenten entfernt voneinander angeordnet sind, jedoch durch einen Prozessor **302** zugreifbar sind.

[0038] Die Software im Speicher **304** kann ein oder mehrere separate Programme umfassen, von denen ein jedes eine geordnete Auflistung von ausführbaren Anweisungen zum Implementieren von logischen Funktionen aufweist. Die Software im Speicher **304** umfaßt das Diagnosesystem **100** und ein geeignetes O/S (O/S = operating system = Betriebssystem) **310**. Es ist zu beachten, daß das Diagnosesystem eine oder mehrere von verschiedenen Funktionen wie ein Testen **100A**, Modellieren **100B** und ein Schlußfolgern **100C** aufweisen kann, die später beschrieben werden. Bei einigen Ausführungsbeispielen können eine oder mehrere dieser Funktionen als separate Programme bereitgestellt sein. Das Betriebssystem **310** steuert die Ausführung von anderen Computerprogrammen, wie das Diagnosesystem **100**. Das Betriebssystem **310** kann auch einen Planungs-, Eingabe-Ausgabe-Steuerungs-, Datei- und Datenverwaltungs-, Speicherverwaltungs- und Kommunikationssteuerungs- und verwandte Dienste liefern.

[0039] Die I/O-Vorrichtungen **306** können Eingabevorrichtungen wie z. B. ein Tastenfeld umfassen. Die I/O-Vorrichtungen **306** können auch Ausgabevorrichtungen wie z. B. eine Anzeigevorrichtung umfassen. Die I/O-Vorrichtungen **306** können ferner Vorrichtungen umfassen, die konfiguriert sind, um sowohl Eingänge als auch Ausgänge, wie z. B. einen Kommunikationsport, zu kommunizieren.

[0040] Wenn der Computer **300** in Betrieb ist, ist der Prozessor **302** konfiguriert, um eine Software, die im Speicher **304** gespeichert ist, auszuführen, um Daten an den oder vom Speicher **304** zu kommunizieren und um allgemein die Operationen des Computers zu steuern. Das Diagnosesystem **100** und das Betriebssystem **310** werden im ganzen oder teilweise durch den Prozessor **302** gelesen, eventuell im Prozessor **302** gepuffert und dann ausgeführt.

[0041] Wenn das Diagnosesystem **100** in einer Software implementiert ist, wird darauf hingewiesen, daß das Diagnosesystem auf einem beliebigen computerlesbaren Medium zur Verwendung durch oder in Verbindung mit einem computerverwandten System oder Verfahren gespeichert sein kann. Im Zusammenhang mit diesem Dokument ist das computerlesbare Medium eine elektronische, magnetische, optische oder andere physische Vorrichtung oder Einrichtung, die ein Computerprogramm zur Verwendung durch oder in Verbindung mit einem computerverwandten System oder Verfahren enthalten oder speichern kann. Das Diagnosesystem **100** kann in einem beliebigen computerlesbaren Medium zur Verwendung durch oder in Verbindung mit einem Anweisungsausführungs-System, -Vorrichtung oder -Gerät, wie einem computerbasierten System, einem prozessorenthaltenden System oder einem anderen System verkörpert sein, das die Anweisungen vom Anweisungsausführungs-System, -Vorrichtung oder -Gerät holen und die Anweisungen ausführen kann.

[0042] Das hierin verwendete computerlesbare Medium kann eine beliebige Einrichtung sein, die ein Programm zur Verwendung durch oder in Verbindung mit einem Anweisungsausführungs-System, -Vorrichtung oder -Gerät speichern, kommunizieren, ausbreiten oder transportieren kann. Daher kann ein computerlesbares Medium beispielsweise ein elektronisches, magnetisches, optisches, elektromagnetisches, Infrarot- oder Halbleiter-System, -Gerät, -Vorrichtung oder Ausbreitungsmedium sein, muß aber nicht auf dieselben beschränkt sein. Spezifischere Beispiele (eine unerschöpfliche Liste) eines computerlesbaren Mediums umfassen folgende Medien: eine elektrische Verbindung (elektronisch) mit einem oder mehreren Drähten, eine tragbare Computerdiskette (magnetisch), einen Direktzugriffsspeicher (RAM) (elektronisch), einen Nur-Lese-Speicher (ROM) (elektronisch), einen löschbaren programmierbaren Nur-Lese-Speicher (EPROM, EEPROM oder Flash-Speicher) (elektronisch), eine optische Faser (optisch) und einen tragbaren Kompaktdisk-Nur-Lese-Speicher (CDROM) (optisch). Es ist zu beachten, daß das computerlesbare Medium sogar Papier oder ein anderes geeignetes Medium sein könnte, auf das das Programm gedruckt ist, da das Programm über optisches Abtasten des Papiers oder eines anderen Mediums elektronisch erfaßt, dann kompiliert, interpretiert oder anderweitig nach Bedarf in einer geeigneten Weise verarbeitet und dann in einem Computerspeicher gespeichert werden könnte.

[0043] Es wird nun Bezug auf das Flußdiagramm von Fig. 4 genommen, das die Funktionalität eines repräsentativen Ausführungsbeispiels des Diagnosesystems **100** darstellt. Diesbezüglich stellt jeder Block des Flußdiagramms ein Modulsegment- oder Abschnitt des Codes dar, der eine oder mehrere ausführbare Anweisungen oder eine Logik zum Implementieren der spezifizierten logischen Funktionen aufweist. Es wird ebenfalls darauf hingewiesen, daß bei einigen alternativen Implementierungen die in den verschiedenen Blöcken von Fig. 4 oder einem beliebigen anderen der beigefügten Flußdiagramme angemerkten Funktionen außerhalb der Reihenfolge auftreten können, in der sie dargestellt sind. Zum Beispiel können zwei in Fig. 4 in Folge gezeigte Blöcke tatsächlich im wesentlichen gleichzeitig ausgeführt werden. Bei anderen Ausführungsbeispielen können die Blöcke manchmal in der umgekehrten Reihenfolge, abhängig von der involvierten Funktionalität, ausgeführt werden.

[0044] Wie in Fig. 4 gezeigt ist, kann das Diagnosesystem oder -verfahren **100** ausgelegt werden, als ob es am Block **410** beginnt, wo ein Datenflußmodell, das das SUT darstellt, vorgesehen ist. Vorzugsweise umfaßt das Datenflußmodell Informationen entsprechend den Datenpaketübertragungs-Beziehungen, die zumindest einem Abschnitt des SUT zugeordnet sind. Im Block **420** wird das SUT im Hinblick auf das Datenflußmodell diagnostiziert. Typischerweise umfaßt dies ein Erwerben von Testergebnissen wie z. B. durch Verwenden einer Testlogik (siehe Testen **100A** von Fig. 3), und ein Analysieren der Testergebnisse mit einer Schlußfolgerungsmaschine (siehe Schlußfolgern **100C** von Fig. 3). Wie vorstehend erwähnt ist, können die Testergebnisse durch ein separates System erworben werden, das die Testergebnisse an das Diagnosesystem liefert.

[0045] Typischerweise ist die Datenflußsemantik, die in einem Datenflußmodell verkörpert ist, allgemeiner Natur und kann auf verschiedene Systeme angewendet werden. Typischerweise ist das Datenflußmodell eines speziellen SUT ein gerichteter Graph, der Scheitelpunkte und Kanten umfaßt. Ein Scheitelpunkt stellt den Abschluß einer Kante dar, d. h. die Scheitelpunkte werden verwendet, um die Enden einer Kante zu definieren. Zusätzlich kann ein Scheitelpunkt einer Position oder einem Abschnitt eines Datenübertragungswegs entsprechen, wo auf die Daten reagiert werden kann. Mittels eines Beispiels kann ein Scheitelpunkt einem Abschnitt eines Datenübertragungswegs entsprechen, der Daten aussortiert, die falsch übertragen worden sind, d. h. der Scheitelpunkt läßt Datenpakete fallen, teilt die Daten in mehrere Abschnitte auf, kombiniert die Daten, routet die Daten und/oder reproduziert die Daten. Mittels eines weiteren Beispiels kann ein Scheitelpunkt einer Position entsprechen, wo die Messungen, z. B. das Zählen von Daten, erfolgen und/oder wo die Güte oder Schlechtheit von Daten bestimmt werden kann, z. B. können CRC (CRC = cyclical redundancy check = zyklische Redundanzprüfung) ausgeführt werden. Es ist zu beachten, daß ein Aufspüren von Daten das Aufspüren von Daten eines Typs, die nicht gut oder schlecht sind, umfassen kann. So können Ausführungsbeispiele der

Erfindung angepaßt sein, um andere Charakteristika von Daten abhängig von der speziellen Anwendung zu berücksichtigen.

[0046] Die Kanten repräsentieren die Datenübertragungswege oder Abschnitte derselben durch ein SUT von einem Scheitelpunkt zu einem anderen. Spezieller sind die Kanten direktionale Komponenten, die als Gelegenheiten zum Einführen von Datenübertragungsfehlern betrachtet werden. Zum Beispiel repräsentiert eine Kante (A, B) die konditionale Übertragung von guten oder schlechten Daten, z. B. einem Datenpaket vom Scheitelpunkt A zum Scheitelpunkt B. Eine Selbstschleife, z. B. (A, A), ist typischerweise nicht zulässig.

[0047] Im Hinblick auf ein SUT sind Fehlererfassungsfähigkeiten den Komponenten zugeordnet, die angepaßt sind, um Prüfungen auszuführen, um die Integrität von Daten während und/oder nach der Operation, wie Erzeugen, Speichern, Übertragen und Empfangen, zu bestimmen. Solche Prüfungen umfassen CRC- und Nachrichten-Zusammenfassungsverfahren wie MD5. Verständlicherweise gilt dies für jene SUTs, die paketbasierte Architekturen umfassen. Zum Beispiel kann die Datenübertragungsintegrität bei einem solchen SUT sichergestellt werden, indem ein CRC-Code an einer Position des SUT erzeugt wird, der CRC-Code an einer anderen Position erneut berechnet wird und dann die zwei CRC-Codes verglichen werden.

[0048] Durch Aufspüren von Daten, wie durch Verwenden von Fehlererfassungsfähigkeiten, kann ein Abschnitt oder eine Komponente eines SUT Informationen in bezug darauf erfassen, ob fehlerenthaltende Daten, z. B. ein schlechtes Datenpaket, empfangen worden sind, übertragen worden sind oder im Begriff sind, übertragen zu werden und/oder ob schlechte Daten fallen gelassen oder nach unten ausgebreitet worden sind. Zusätzlich kann bei einigen Ausführungsbeispielen der Zustand der Komponenten und/oder ein Zeitpunkt, der der Fehlererfassung zugeordnet ist, bestimmt werden.

[0049] Bei einigen Ausführungsbeispielen wird davon ausgegangen, daß die Fehlerregistrierungsfähigkeit des SUT vollkommen ist. Das heißt, daß typischerweise davon ausgegangen wird, daß das SUT den korrekten Status von eingehenden Daten an allen Kanten unter allen Bedingungen registrieren kann. Dies ist bei typischen Anwendungen natürlich falsch, kann jedoch ermöglichen, daß eine effizientere Diagnose mit einer höheren Auflösung ausgeführt wird. Verständlicherweise könnten bei einigen Ausführungsbeispielen zusätzliche Variablen verwendet werden, um z. B. eine unvollkommene Fehlerregistrierung zu berücksichtigen.

[0050] Wie vorstehend erwähnt ist, verwenden die Diagnosesysteme der vorliegenden Erfindung Einschränkungen, um Fehlerkandidaten von SUTs zu diagnostizieren. Spezieller nutzen die Ausführungsbeispiele des Diagnosesystems das Prinzip, daß der Datenpaketfluß durch das SUT gemäß der Funktionalität des SUT eingeschränkt ist. Dies ist typischerweise als ein Datenflußgraph eines speziellen Testwegs dargestellt. Das SUT und der Datenflußgraph erfassen auch einen Vorrichtungsstatus, Zähler etc. Es kann jedoch auch eine andere Schlußfolgerungsmaschinen-Funktionalität verwendet werden, und diese wird später beschrieben.

[0051] Ungeachtet der speziellen Funktionalität verwenden die Ausführungsbeispiele der Schlußfolgerungsmaschine die gleiche Definition von einer Diagnose, d. h. die Ausgabe der Schlußfolgerungsmaschine. Zusätzlich verwenden die Schlußfolgerungsmaschinen Testergebnisse, die den Vorrichtungsstatus, Paketzählwerte etc. umfassen, und Datenflußgraphen, die den Testweg und die zugeordnete Vorrichtungsfunktionalität beschreiben, als Eingabe. Schlußfolgerungsstrategien erzeugen eine Diagnose als Ausgabe, die verdächtige Kanten und die Fehlertypen/Mengen, die jeder Kante zugeordnet sind, umfassen kann. Zum Beispiel kann eine Kante als auffällig gelten, wenn ein Fehler auf dieser Kante mit den Testergebnissen übereinstimmt. Im Gegensatz dazu kann eine Kante nicht als verdächtig gelten, d. h. gut, wenn ein beliebiger Ausfall auf dieser Kante mit den Testergebnissen nicht übereinstimmt. Gute und verdächtige Kanten können dann nach Wunsch in physische Komponenten des SUT abgebildet werden.

[0052] Diesbezüglich können Ausführungsbeispiele der Schlußfolgerungsmaschine der Erfindung eine oder mehrere Techniken wie ein lineares Programmieren, eine regelbasierte und flußereignisbasierte Fehlersimulation, verwenden, um die Einschränkungen anzuwenden. Die Ausführungsbeispiele der Diagnosesysteme, die ein lineares Programmieren nutzen, um die SUTs auszuwerten, verwenden typischerweise Einschränkungsgleichheiten und/oder -ungleichheiten von denen ein Beispiel später beschrieben wird, um eine Diagnose zu bestimmen.

[0053] Mittels eines Beispiels kann das lineare Programmieren verwendet werden, um eine machbare Diagnose angesichts der SUT-Funktionalitätseinschränkungen und Einschränkungen, die dem Test zugeordnet sind, z. B. Gesamtanzahl von versuchten Daten, z. B. Datenpaketübertragungen und/oder Einschränkungen, die dem beobachteten Verhalten (Testergebnissen) zugeordnet sind, zu finden. Speziell kann das lineare Pro-

grammieren bei einigen Ausführungsbeispielen verwendet werden, um die Anzahl von Datenpaketen, die an jeder Kante schlecht gemacht wurden, zu optimieren/maximieren.

[0054] Man nehme z. B. an, daß ein gerichteter Graph G = (V, E), der den zulässigen Fluß von Daten, z. B. Datenpakete, modelliert, vorgesehen ist. Ein Scheitelpunkt v ∈ V von G ist eine Position, wo z. B. Messungen stattfinden können, die Güte oder Schlechtheit von Paketen getestet werden kann, z. B. durch Überprüfen eines CRC-Codes, und/oder schlechte Pakete z. B. fallengelassen werden können.

[0055] Ein Scheitelpunkt kann mit Informationen über bestimmte Verhaltenscharakteristika des Scheitelpunkts etikettiert sein. Zum Beispiel ist ein „Ausbreitungs"- (bzw. prop) Scheitelpunkt ein Scheitelpunkt, der schlechte Pakete ausbreitet, und ein „Nicht-Ausbreitungs"- (bzw. noprop) Scheitelpunkt ist ein Scheitelpunkt, der schlechte Pakete, die erfaßt wurden, fallenläßt. Zusätzlich stellt ein „Bus"-Scheitelpunkt einen physischen Bus dar, d. h. alle guten Pakete, die empfangen wurden, werden auf allen „Aus-Kanten" eines solchen Scheitelpunkts übertragen. „Uneingeschränkte" Scheitelpunkte können ebenfalls verwendet werden. Bezüglich der Beziehungen zwischen der Anzahl von Paketen, die empfangen wurden, und der Anzahl von Paketen, die durch diesen Typ von Scheitelpunkt übertragen wurden, ist kein Wissen verfügbar. Ein solcher Scheitelpunkt kann verwendet werden, um komplexe, datenabhängige Operationen des SUT darzustellen, wo die Mengen von guten und schlechten Paketen, die in den Scheitelpunkt hineinfließen und aus demselben herausfließen, beispielsweise schwierig zu beschreiben sind.

[0056] Man lasse Λ = {Ausbreitung, Nicht-Ausbreitung, Bus, uneingeschränkt} den Satz von möglichen Scheitelpunktetiketten sein. Jeder Scheitelpunkt v Ausbreitung ε V weist einen zugeordneten Satz von Etiketten auf, der durch die Funktion $T:V \rightarrow 2^{\Lambda}$ gegeben ist. Die gerichteten Kanten E ⊆ V × V sind Kommunikationswege zwischen den Scheitelpunkten. Ohne die Allgemeingültigkeit zu mindern, werden typischerweise einzelne Richtungskanten, d. h. Kanten mit nur einem Pfeil, verwendet. Ansonsten kann eine bidirektionale Kante durch zwei einzelne direktionale Kanten ersetzt werden. Erinnern wir uns, daß die Kanten (j, i) ∈ E als die „Ein-Kanten" von i bezeichnet werden und daß die Kanten (j, i) ∈ E als die „Aus-Kanten von i bezeichnet werden.

[0057] Von der nachstehenden Semantik von Kanten wird typischerweise ausgegangen: Ein Paket, das in einen Scheitelpunkt v von einer beliebigen seiner Ein-Kanten fließt, kann aus einer beliebigen Aus-Kante fließen. Wenn ein System oder Test bekannt ist, um den Fluß von Paketen einzuschränken, die in einen Scheitelpunkt v an einer speziellen Kante oder Kanten eintreten, um aus einer speziellen Kante oder Kanten auszutreten, dann sollte der Scheitelpunkt v in zwei oder mehrere Scheitelpunkte aufgeteilt werden. Zusätzlich wird ein Scheitelpunkt als eine „Quelle" bezeichnet, wenn er keine Ein-Kanten aufweist. ER wird als eine „Senke" bezeichnet, wenn er keine Aus-Kanten aufweist.

[0058] Neben dem Graphen G wird davon ausgegangen, daß ein Satz von Zählern Ψ und eine Abbildung M : E × {t, r} × {good, bad} → Ψ vorhanden ist. Die Abbildung M gibt die Semantik der Zähler an. Sie sollte wie folgt interpretiert werden:
Man nehme an M((i, j), t, good) = ψ. Dann wird ψ inkrementiert, immer wenn ein gutes Paket vom Scheitelpunkt i auf eine Kante (i, j) übertragen wird.
Man nehme an M((i, j), t, bad) = ψ. Dann wird ψ inkrementiert, immer wenn ein schlechtes Paket vom Scheitelpunkt i auf die Kante (i, j) übertragen wird.
Man nehme an M((i, j), r, good) = ψ. Dann wird ψ inkrementiert, immer wenn ein gutes Paket durch den Scheitelpunkt j über die Kante (i, j) empfangen wird.
Es ist anzunehmen daß M((i, j), t, bad) = ψ. Dann wird ψ inkrementiert, immer wenn ein schlechtes Paket vom Scheitelpunkt j über die Kante (i, j) empfangen wird.

[0059] Es ist zu beachten, daß eine Abbildung M darauf gerichtet sein sollte, jedoch nicht Eins-zu-Eins sein kann. Man nehmen z. B. an, daß ein Scheitelpunkt v drei Ein-Kanten (x, v), (y, v) und (z, v) aufweist. Es ist wünschenswert, daß ψ alle guten Pakete, die bei v ankommen, zählt. Dann setze man:

M(((x, v)r, good)) = M(((y, v), r, good)) = M(((z, v), r, good)) = ψ

[0060] In der gleichen Weise kann ein einzelner Zähler verwendet werden, um eine große Vielzahl von unterschiedlichen Ereignissen, die an verschiedenen Kanten stattfinden, zu zählen. Ein Satz von speziellen gemessenen Werten für jeden Zähler wird als Syndrom bezeichnet.

[0061] Die allgemeine Prämisse der SUT-Diagnose unter Verwendung des linearen Programmierens umfaßt ein Codieren verfügbarer Informationen, z. B. Informationen bezüglich der Einschränkung der Pakete, Zähler-

semantik und gemessener Zählerwerte, zu einem Optimierungsproblem, wobei die optimale Lösung desselben bestimmt, ob eine spezielle Kante fehlerhaft sein kann.

[0062] Diesbezüglich können die Ausführungsbeispiele einer Schlußfolgerungsmaschine eines Diagnosesystems, das das lineare Programmieren verwendet, allgemein als drei Teilabschnitte umfassend bezeichnet werden: (1) Einschränkungsextraktion, (2) Hinzufügen von Syndromeinschränkungen und (3) Bestimmen, welche Fehlerkandidaten angesichts der Einschränkungen und des Syndroms möglich sind. Typischerweise kann der erste Teilabschnitt für ein gegebenes SUT im voraus berechnet werden. Zusätzlich müssen typischerweise nur der zweite und der dritte Teilabschnitt für jedes Syndrom erneut durchlaufen werden.

[0063] Im Hinblick auf die Einschränkungsextraktion wird ein Satz von Variablen $U_{(i,j)\in E}${g(i, j), b(i, j), mb(i, j) gd(i, j), bd(i, j)} erzeugt. Die Variable g(i, j) stellt die Anzahl von guten Paketen, die auf die Kante (i, j) übertragen werden, dar. Die Variable b(i, j) stellt die Anzahl von schlechten Paketen dar, die auf der Kante (i, j) durch den Scheitelpunkt i übertragen werden. Die Variable mb(i, j) stellt die Anzahl von Paketen dar, die auf der Kante (i, j) schlecht gemacht wurden, d. h. die Pakete, die auf der Kante als gut übertragen wurden, jedoch als schlecht empfangen wurden. Die Variable gd(i, j) stellt die Anzahl von guten Paketen dar, die auf der Kante (i, j) übertragen wurden, die verschwand. Es ist zu beachten, daß ein Paket verschwinden kann, wenn es so verfälscht wird, daß eine Empfangsvorrichtung das Paket nicht als Paket erkennen kann. Die Variable bd(i, j) stellt die Anzahl von schlechten Paketen dar, die auf der Kante (i, j) übertragen wurden, die verschwand.

[0064] Allgemein wird ein anfänglich leerer Satz von Einschränkungen C erzeugt. Für jeden Scheitelpunkt i mit uneingeschränktem $\notin$ T(i), der zumindest eine Ein-Kante und zumindest eine Aus-Kante aufweist, ist zu C aus den Einschränkungen, die nachstehend definiert sind, folgendes hinzuzufügen,

- die Einschränkung KG(i) wenn bus $\notin$ T(i),
- die Einschränkung KGB(i, j) für jede Aus-Kante j von i wenn bus $\in$ T(i).

[0065] Für jeden Scheitelpunkt i mit uneingeschränkter $\notin$ T(i), die zumindest eine Aus-Kante aufweist, ist zu C

- die Einschränkung KBP(i), wenn prop $\in$ T(i) und bus $\notin$ T(i),
- die Einschränkung KBPB(i, j) für jede Aus-Kante j von i, wenn prop $\in$ T(i) und bus $\in$ T(i),
- eine KBNP-Einschränkung, wenn prop $\notin$ T(i),

hinzuzufügen.

[0066] Für jede Kante (i, j) $\in$ E ist eine Einschränkung EDGECONsERVE (i, j) hinzuzufügen.

[0067] Für jeden Zähler $\psi \in \Psi$ ist eine Einschränkung COUNTER (Ψ) hinzuzufügen.

[0068] Die vorstehend erwähnten Einschränkungen sind wie folgt definiert:

KG(i)

(Die Kirchhoff-ähnliche Einschränkung auf den guten Paketen, Scheitelpunkt kein Bus):

$$\sum_{(k,i)\in E} g(k,i) - \sum_{(k,i)\in E} gd(k,i) - \sum_{(k,i)\in E} mb(k,i) - \sum_{(i,j)\in E} g(i,j) = 0$$

[0069] Die Einschränkung KG sagt aus, daß die Anzahl von guten Paketen, die an den Scheitelpunkt i übertragen werden, minus der Anzahl von Paketen, die auf den Ein-Kanten von i verschwanden, minus der Anzahl von Paketen, die innerhalb der Ein-Kanten von i schlecht gemacht wurde, gleich der Anzahl von guten Paketen sein muß, die aus i fließen.

KGB(i, j)

(Kirchhoff-ähnliche Einschränkung auf guten Paketen, Scheitelpunkt ist ein Bus):

$$\sum_{(k,i)\in E} g(k,i) - \sum_{(k,i)\in E} gd(k,i) - \sum_{(k,i)\in E} mb(k,i) - g(i,j) = 0$$

[0070] Die Einschränkung KG sagt aus, daß die Anzahl von guten Paketen, die an den Scheitelpunkt i übertragen werden, minus der Anzahl von Paketen, die auf den Ein-Kanten von i schwanden, minus der Anzahl von

Paketen ist, die innerhalb der Ein-Kanten von i schlecht gemacht wurden, gleich der Anzahl von guten Paketen sein müssen, die aus der Aus-Kante j von i herausfließen.

KBP(i)

(Kirchhoff-ähnliche Einschränkung auf guten Paketen, Ausbreitungs-Scheitelpunkt, Scheitelpunkt kein Bus):

$$\sum_{(k,i)\in E} b(k,i) - \sum_{(k,i)\in E} bd(k,i) - \sum_{(k,i)\in E} mb(k,i) - \sum_{(i,j)\in E} b(i,j) = 0$$

[0071] Die Einschränkung KBP sagt aus, daß bei einem Ausbreitungs-Scheitelpunkt i die Anzahl von schlechten Paketen, die an i übertragen wurden, plus der Anzahl von Paketen, die auf den Ein-Kanten von i verschwanden, plus der Anzahl von Paketen, die innerhalb der Ein-Kanten von i schlecht gemacht wurden, gleich der Anzahl von schlechten Paketen sein muß, die aus i herausfließen.

KBPB(i, j)

(Kirchhoff-ähnliche Einschränkung auf schlechte Pakete, Ausbreitungs-Scheitelpunkt, Scheitelpunkt ist ein Bus):

$$\sum_{(k,i)\in E} b(k,i) - \sum_{(k,i)\in E} bd(k,i) + \sum_{(k,i)\in E} mb(k,i) - b(i,j) = 0$$

[0072] Die Einschränkung KBP sagt aus, daß bei einem Ausbreitungs-Scheitelpunkt i die Anzahl von schlechten Paketen, die an i übertragen wurden, minus der Anzahl von Paketen, die auf den Ein-Kanten von I verschwanden, plus der Anzahl von Paketen, die innerhalb der Ein-Kanten des i schlecht gemacht wurden, gleich der Anzahl von schlechten Paketen sein muß, die aus jeder Aus-Kante j von i fließen.

KBNP(i)

(Kirchhoff-ähnliche Einschränkung auf schlechte Pakete, Nicht-Ausbreitungs-Scheitelpunkt):

$$\sum_{(i,j)\in E} b(j,i) = 0$$

[0073] Die Einschränkung KBNP sagt aus, daß keine schlechten Pakete von einem Nicht-Ausbreitungs-Scheitelpunkt übertragen werden.

EDGECONSERVE(i, j)

(Konservierung von Paketen auf Kanten):

gd(i, j) + mb(i, j) g(i, j)
bd(i, j) b(i,j)

[0074] Diese Ungleichheiten sagen aus, daß nicht mehr Pakete verschwinden können oder auf einer Kante schlecht gemacht werden können als auf der Kante übertragen wurden. Die EDGECONSERVE-Einschränkungen sind typischerweise notwendig. Ohne sie könnten Lösungen gefunden werden, wo mehr Pakete verschwinden als übertragen wurden.

COUNTER(ψ) (Die Ereignisse spezifizieren, die ψ zählt):

$$\sum_{M((i,j),t,good)=\psi} g(i,j)$$

+

$$\sum_{M((i,j),r,good)=\psi} (g(i,j) - gd(i,j) - mb(i,j))$$

+

$$\sum_{M((i,j),t,bad)=\psi} b(i,j)$$

+

$$\sum_{M((i,j),r,bad)=\psi} (b(i,j) - bd(i,j) + mb(i,j))$$

$$= counter\_value(\psi)$$

[0075] Es ist zu beachten, daß es typischerweise auch notwendig ist, alle Variablen einzuschränken, um nichtnegativ zu sein, d. h. es sind keine negativen Paketflüsse vorhanden. Zusätzlich ist es in einigen Situationen wünschenswert, alle oder einige Variablen einzuschränken, um Ganzzahlen zu sein.

[0076] Setzt man den Vorgang mit dem Hinzufügen von Syndromeinschränkungen fort, umfaßt ein Syndrom typischerweise Werte, die den verschiedenen Zählern oder verschiedenen anderen beobachteten SUT-Vorrichtungsstatussen zugeordnet sind, die nach der Testausführung gesammelt werden. Für jeden solchen Zähler ist eine Gleichheit zu C hinzuzufügen, die den Wert des Zählers spezifiziert. Zum Beispiel, wenn der gemessene Wert eines Zählers, der ψ_{11} zugeordnet ist, den Wert 127 aufweist, und ein gemessener Wert eines Zähler, der ψ_{17} zugeordnet ist, den Wert 1001 aufweist, sind die Einschränkungen counter_value (ψ_{11}) = 127 und counter_value (ψ_{17}) = 1001 hinzuzufügen. Diese Syndromeinschränkungen werden als S bezeichnet.

[0077] Im Hinblick auf die Bestimmung von möglichen Fehlerkandidaten, ist es die Aufgabe zu bestimmen, welche Fehlerkandidaten möglicherweise die schlechten Pakete, die erfaßt wurden, bewirkt haben könnten, z. B. welche Fehlerkandidaten die beobachteten Testergebnisse, wie die Fehlerwerte, in korrekter Weise berücksichtigen. Vorzugsweise umfaßt jeder Fehlerkandidat einen Fehlertyp, z. B. mb, gd, bd, etc., und eine Quantität des Fehlertyps und entspricht einer zugeordneten Kante $(i, j) \in E$.

[0078] Zum Beispiel können n Pakete auf spezielle Weise auf einer spezielle Kante falsch übertragen worden sein, und mehr als ein Fehlerkandidat kann dieser Kante zugeordnet sein. Dieser Satz von Fehlerkandidaten wird als FC(i, j) bezeichnet. Zusätzlich kann das SUT mehr als eine fehlerhafte Kante aufweisen, und mehr als ein Fehlerkandidat kann einem gegebenen beobachteten Testergebnis zugeordnet sein.

[0079] Bei diesem Ausführungsbeispiel kann ein gegebener Fehlerkandidat fehlerhaft sein, wenn nur eine einzige Lösung für das System von Einschränkungsgleichungen vorhanden ist, wo zumindest eine oder mehrere der zugeordneten Fehlervariablen größer als 0 ist. Die Einschränkungen C und S sind alle linear. Da die Variablenwerte typischerweise auch alle Ganzzahlen sind, können die Einschränkungsgleichungen als ein IP-Problem (IP = integer programming = ganzzahliges Programmierungen) gelöst werden.

[0080] Verschiedene Routinen können zum Lösen von IP-Problemen verwendet werden. Zum Beispiel gibt es viele Bibliotheksroutinen, wie z. B. lp_solve, die zum Lösen von IP-Problemen verfügbar sind. Der Quellencode für lp_solve ist hierin durch Bezugnahme aufgenommen. Es ist zu beachten, daß bei lp_solve die Variablen per Vorgabe nichtnegativ sind, so daß die Variablen nicht explizit als nichtnegativ einschränkt sein müssen. Zusätzlich löst lp_solve die IP-Probleme unter Verwendung des „Branch and Bound"-Verfahrens.

[0081] Durch Auswählen einer objektiven Funktion und Iterieren durch mehrere IP-Formulierungen unter Verwendung verschiedener Einschränkungen können alle Fehlertypen effizient für jede mögliche Fehlerkante nu-

meriert werden. Speziell lautet die objektive Funktion typischerweise:

$$\max \sum_{(i,j)\in E} mb(i,j),$$

d. h., um die Summe von Paketen, die auf allen Kanten in E schlecht gemacht wurden, zu maximieren. Diese Funktion erzwingt eine Lösung für alle Fehlervariablen, so daß jedes nicht-leere FC(i, j) zumindest einen Fehlerkandidat enthält. Es ist zu beachten, daß eine einzelne Optimierung nicht alle Mitglieder von jedem FC(i, j) erzeugt, sondern nur einen möglichst großen Satz eines gleichzeitig erfüllten FC(i, j). Diese objektive Funktion liefert Lösungen an die Fehlervariablen, wo mehr als eine Kante fehlerhaft sein kann.

[0082] Um zusätzliche Fehlerkandidaten zu erzeugen, kann das IP ferner eingeschränkt werden, und zusätzliche Optimierungen können betrieben werden, wie in der folgenden Weise. Zum Beispiel lasse man

$$UFC1 = \bigcup_{(i,j)\in E} FC(i,j)$$

und

UFC = UFC1.

[0083] Für jedes fc ∈ UFC1:

1. Füge zu C einen Einschränkung hinzu, die diese Fehlervariable auf 0 setzt, d. h. die diesen Fehlertyp aus zukünftigen Lösungen eliminiert. Dies bewirkt effektiv, daß neue Fehlertypen als Lösungen auftauchen.
2. Optimiere dieses neue IP.
3. Wenn eine machbare Lösung existiert, füge einen oder mehrere resultierende eindeutige Fehlerkandidaten zu UFC hinzu.
4. Wenn eine machbare Lösung nicht existiert, entferne aus C die Einschränkung, die in Schritt 1 hinzugefügt wurde.

[0084] Das UFC sollte nun einen Fehlerkandidaten von jedem machbaren Typ für jede möglicherweise fehlerhafte Kante enthalten. Es ist zu beachten, daß es für einige IP-Löser bei Schritt 1 effizienter sein kann, einen variablen Satz aus dem Problem zu entfernen, indem alle Referenzen auf denselben in allen Einschränkungen gelöscht werden, als eine Einschränkung, die erfordert, daß er 0 ist, hinzuzufügen.

[0085] Bei einigen Anwendungen kann es wünschenswert sein, eine Anzahl von gleichzeitigen Ausfällen zu erzwingen. Zum Beispiel kann aufgrund von A priori-Kenntnissen oder einer Kundenpräferenz eine Anzahl von gleichzeitigen defekten Kanten erzwungen werden. Alternativ ist nach Occam Razor anzunehmen, daß es wünschenswert ist, zu einer Diagnose mit einer minimalen Anzahl von defekten Kanten zu gelangen. Eine solche Diagnose kann festgestellt werden, indem zuerst versucht wird, eine einzelne defekte Kante zu finden, die die verfügbaren Daten erörtert. Anschließend, wenn keine existiert, soll versucht werden, ein Paar von effektiven Kanten zu finden, die das Verfügbare erklären. Dieser Prozeß kann fortgesetzt werden, bis eine Mehrfachdefekt-Hypothese gefunden wird, die das Syndrom erklärt.

Fall 1.

[0086] Es wird nun auf das Datenflußmodell von **Fig. 5** Bezug genommen. Jeder Scheitelpunkt, z. B. Scheitelpunkt 1, Scheitelpunkt 2 und Scheitelpunkt 3, weist vordefinierte Verhaltenscharakteristika auf. Speziell ist ein Scheitelpunkt 1 in der Lage, gute Pakete, die übertragen werden, zu zählen, Scheitelpunkt 2, schlechte Pakete, die empfangen wurden, zu zählen und Scheitelpunkt 3 in der Lage, gute Pakete, die empfangen werden, zu zählen. Zusätzlich breiten sowohl der Scheitelpunkt 1 als auch der Scheitelpunkt 3 keine empfangenen schlechten Pakete aus, und der Scheitelpunkt 2 breitet empfangene schlechte Pakete aus.

[0087] Basierend auf dem Datenflußmodell **500** können drei Zähler verwendet werden: $\Psi = \{\psi_1, \psi_2, \psi_3\}$.

[0088] Die Abbildung M ist gegeben durch:

$M((1, 2), t, good) = \psi_1$

$M((1, 2), r, bad) = \psi_2$

$M((2, 3), r, good) = \psi_3$

[0089] Die Einschränkungen C, die aus dem Datenflußmodell **500** entstehen, sind:

b_1_2 = 0; (KBNP auf Scheitelpunkt 1)

g_1_2 – gd_1_2 – mb_1_2 – g_2_3 – 0; (KG (2))

b_1_2 + bd_1_2 = mb_1_1 – b_2_3 = 0; (KBP (2))

g_1_2 = psi_1; (COUNTER (ψ_1))

b_1_2 – bd_1_2 + mb_1_2 = psi_2; (COUNTER auf (ψ_2))

g_2_3 – gd_2_3 – mb_2_3 = psi_3; (COUNTER auf (ψ_3))

gd_1_2 + mb_1_2 g_1_2; (EDGECONSERVE (1, 2))

bd_1_2 ≤ b_1_2; (EDGECONSERVE (1, 2))

gd_2_3 + mb_2_3 ≤ g_2_3; (EDGECONSERVE (2, 3))

bd_2_3 ≤ b_2_3; (EDGECONSERVE (2, 3))

[0090] Es ist anzunehmen, daß basierend auf erworbenen Testergebnissen der Scheitelpunkt 1 20 gute Pakete zählte, der Scheitelpunkt 2 einen CRC-Fehler zählte und der Scheitelpunkt 3 19 gute Pakete zählte. Die Einschränkungen S, die aus diesem Syndrom entstehen, sind:

psi_1 = 20

psi_2 = 1

psi_3 = 19

[0091] Das Ganzzahlenprogramm ist maximal {mb_1_2 + mb_2_3)|C, S}. Die Fehlervariablen mb(1, 2), gd(1, 2), bd(1, 2), mb(2, 3), gd(2, 3), bd(2, 3) sind größer oder gleich 1, wenn nur ihre entsprechende Kante fehlerhaft sein kann.

[0092] Nach dem Lösen des IP-Problems, das vorstehend beschrieben ist, ist mb(1, 2) gleich 1 und alle anderen Fehlervariablen sind 0. Daher ist die Kante (1, 2) defekt und ein Paket wurde schlecht gemacht.

Fall 2.

[0093] Es wird nun Bezug auf Fig. 6 genommen, die ein Blockdiagramm eines repräsentativen SUT darstellt. Wie in Fig. 6 gezeigt ist, umfaßt das SUT **600** fünf Komponenten, d. h. START, N2PB, PBIF, BUF und CBOC. Jede Komponente weist vordefinierte Verhaltenscharakteristika auf. Speziell ist jede der dargestellten Komponenten des SUT **600** in der Lage, empfangene Daten, z. B. Datenpakete, zu zählen und CRC-Prüfungen auszuführen. Zusätzlich wird darauf hingewiesen, daß sich mehrere der Komponenten in bezug aufeinander, beim Empfangen von schlechten Daten anders verhalten. Speziell breiten sowohl N2PB und BUF empfangene schlechte Daten aus, und sowohl START als auch PBIF breiten keine empfangenen schlechten Daten aus. Es gibt auch zwei unterschiedliche Arten von BUFF-Einheiten. Der „smart buff" (bzw. intelligenter Puffer) zählt die guten Pakete, die empfangen wurden, der „dumb buff" (bzw. dummer Puffer) tut dies nicht.

[0094] In Falle des „dumb buff" können vier Zähler verwendet werden: $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4\}$. Die Abbildung M ist gegeben durch:

$M((start, n2pb), t, good) = \psi_1$;

$M((start, n2pb), r, good) = \psi_2$;

M((n2pb, pbif), r, good) = M((buff, pbif), r, good) = ψ_3; und

M((pbif, cboc), r, good) = ψ_4.

[0095] Es ist zu beachten, daß zwei unterschiedliche Argumente zu M auf ψ_3 abgebildet werden. So wird ψ_3 inkrementiert, immer wenn ein gutes Paket durch pbif auf einer ihrer beiden Ein-Kanten empfangen wird. Im Falle des „smart buff" ist ein zusätzlicher Zähler ψ_5 typischerweise erforderlich, und M(pbif, buff)), r, good) = ψ_5.

[0096] Das Datenflußmodell **700** von Fig. 7 kann basierend auf den Informationen, die bezüglich des SUT **600** von Fig. 6 präsentiert sind, konstruiert sein. Es ist zu beachten, daß das Blockdiagramm von Fig. 6 und das Datenflußmodell **700** von Fig. 7 eine Datenflußambiguität aufweisen. Das heißt, daß das Blockdiagramm und das Datenflußmodell **700** jeweils nicht beschreiben, wie die Daten tatsächlich von PBIF zu CBOC fließen. Speziell ist es dahingehend zweideutig, ob die Daten, die bei PBIF ankommen, zuerst zu BUF und zurück fließen, bevor sie zu CBUC übertragen werden, oder BUF irgendwie umgangen wird. Aufgrund dieser Ambiguität kann das Datenflußmodell **700**, das direkte Analogien für die fünf Komponenten des Blockdiagramms von Fig. 6 liefert, weniger nützlich sein als andere Datenflußmodelle, die keine solche Ambiguität beinhalten. Zum Beispiel, wenn Informationen bezüglich des tatsächlichen Flusses von Daten von PBIF zu CBOC erfaßt werden, kann ein unzweideutiges Datenflußmodell, das die Übertragung von Daten durch das SUT darstellt, konstruiert werden. Ein Ausführungsbeispiel eines solchen Datenflußmodells wird später im Hinblick auf Fig. 8 beschrieben.

[0097] Zurückkehrend zum Datenflußmodell von Fig. 7, wurden fünf Syndrome erzeugt, von denen ein jedes ein mögliches Syndrom ist, das aus einem zeitweise auftretenden Ausfall von einer der fünf Kanten im Datenflußmodell entsteht. Die Syndrome sind in Tabelle 1 gezeigt.

Tabelle 1: Syndrome, die in Fall 1 und 2 verwendet werden

Zähler	Syn. 1	Syn. 2	Syn. 3	Syn. 4	Syn. 5
defekt	start→n2pb	n2pb→pbif	pbif→buff	buff→pbif	pbif→cboc
ψ_1	10	10	10	10	10
ψ_2	9	10	10	10	10
ψ_3	18	18	19	19	20
ψ_4	9	9	9	9	9
ψ_5	9	9	9	10	10

[0098] Die Ergebnisse zum Lösen der Probleme des linearen Programmierens sind in Tabelle 2 und Tabelle 3 gezeigt. Man erinnere sich, daß ein Nicht-Nulleintrag impliziert, daß die entsprechende Fehlerhypothese eine machbare Ausfallursache sein kann. Der Wert ist die Anzahl von schlechten Paketen, die dieser Ausfallursache zugeschrieben sind.

Tabelle 2: Ergebnisse des LP-Lösens für Fall 2, dummer Puffer.

Fehlerhypo.	Syn. 1	Syn. 2	Syn. 3	Syn. 4	Syn. 5
start→n2pb	1	0	0	0	0
n2pb→pbif	0	1	1	1	1
pbif→buff	0	1	1	1	1
buff→pbif	0	1	1	1	1
pbif→cboc	0	1	1	1	1

Tabelle 3: Ergebnisse des LP-Lösens für Fall 2, intelligenter Puffer.

Fehlerhypo.	Syn. 1	Syn. 2	Syn. 3	Syn. 4	Syn. 5
start→n2pb	1	0	0	0	0
n2pb→pbif	0	1	0	1	0
pbif→buff	0	0	1	0	1
buff→pbif	0	1	0	1	0
pbif→cboc	0	0	1	0	1

Fall 3.

[0099] In diesem Beispiel wird eine weitere Annahme zu der vorstehenden, in bezug auf Fall 2 beschriebenen hinzugefügt. Speziell ist anzunehmen, daß eine zusätzliche Einschränkung bekannt ist, d. h., daß die Pakete von n2pb zu pbif zum buff zu pbif zu cboc fließen müssen. Anschließend kann ein genaueres Datenflußmodell für das SUT konstruiert werden. Ein solches Datenflußmodell ist in Fig. 8 dargestellt.

[0100] Wie in Fig. 8 gezeigt ist, umfaßt das Datenflußmodell **800** die Scheitelpunkte START, N2PB, PBIF1, BUF, PBIF2 und CBOC. Die Kanten START→N2PB, N2PB→PBIF1, PBIF1→BUF, BUF→PBIF2 und PBIF2→CBOC sind durch die Scheitelpunkte definiert. So ist die Komponente IF von Fig. 6 zum Zwecke des Datenflußmodells **800** als zwei getrennte Scheitelpunkte umdefiniert worden, d. h. PBIF1 und PBIF2, wodurch die Datenflußambiguität aufgehoben worden ist.

[0101] Wie in Fall 2 können vier Zähler verwendet werden: $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4\}$. Die Abbildung M ist angegeben durch:

$M((start, n2pb), t, good) = \psi_1$

$M((start, n2pb), r, good) = \psi_2$,

$M((n2pb, pbif1), r, good) = M((buff, pbif2), r, good) = \psi_3$,

$M((pbif2, cboc), r, good) = \psi_4$.

[0102] Im Falle des intelligenten Puffers ist ein zusätzlicher ψ_5 erforderlich, und $M((pbif1, buff) r, good) = \psi_5$. Es ist zu beachten, daß ψ_3 inkrementiert wird, wenn ein gutes Paket durch entweder pbif1 oder pbif2 empfangen wird. Dies ist darin begründet, daß beim ursprünglichen Datenflußmodell von Fig. 7 pbif alle ankommenden gute Pakete, die auf einer der beiden Kanten ankommen, zählt.

[0103] Die Einschränkungen C sind:

g_start_n2pb-gd_start_n2pb-mb_start_n2pb_pbifl=0;

b_start_n2pb-bd_start_n2pb=mbjstart-b_n2pb_pbifl=0;

g_n2pb_pbifl-gd_n2pb_pbif-mb_n2pb_pbifl-g_pbifl_buff=0;

b_pbifl_buff=0;

g_pbifl_buff-gd_pbif_buff-mb_pbif_buff-g_buff_pbif2=0;

b_pbifl_buff-bd_pbifl_buff+mb_pbif_buff-b_buff_pbif2=0;

g_buff_pbif2-gd_buff_pbif2_mb_buff-pbif-g_pbif2_cboc=0;

b_pbif2_cboc=0;

gd_start_n2pb+mb_start_n2pb≤g_start_n2pb;

bd_start_n2pb<b_start_n2pb;

gd_2npb_pbifl+mb_n2pb_pbif≤g_n2pb_pbifl;

bd_n2pb_pbif1≤b_n2pb_pbifl;

gd_pbifl_buff+mb_pbif_buff≤g_pbif_buff;

bd_pbifl_buff≤b_pbifl_buff;

gd_buff_pbif2+mb_buff_pbif≤g_buff_pbif2;

bd_biff_pbif2≤b_buff1_pbif;

g_start_n2pb=psi_1;

g_start_n2pb-gd_start_n2pb-mb_start_n2pb=psi_2;

g_n2pb_pbifl-gd_n2pb_pbifl-mb_n2pb_pbif+ g_buff_pbif2-gd_buff_pbif2-mb_buff_pbif=psi_3;

g_pbif2_choc-gd_pbif2_choc-mb_pbif_choc=psi_4;

g_pbifl_buff-gd_pbifl_buff-mb_pbif_buff=psi_5

[0104] Die Ergebnisse zum Lösen der LP-Probleme erscheinen in Tabelle 4 und 5. In diesem Fall sind die Variablen zusätzlich eingeschränkt, um Ganzzahlen zu sein.

Tabelle 4: Ergebnisse des LP-Lösens für Fall 3, dummer Puffer.

Fehlerhypo.	Syn. 1	Syn. 2	Syn. 3	Syn. 4	Syn. 5
start→n2pb	1	0	0	0	0
n2pb→pbifl	0	1	0	0	0
pbifl→buff	0	0	1	1	0
buff→pbif2	0	0	1	1	0
pbif2→cboc	0	0	0	0	1

Tabelle 5: Ergebnisse des LP-Lösens für Fall 3, smarter Puffer.

Fehlerhypo.	Syn. 1	Syn. 2	Syn. 3	Syn. 4	Syn. 5
start→n2pb	1	0	0	0	0
n2pb→pbifl	0	1	0	0	0
pbifl→buff	0	0	1	0	0
buff→pbif2	0	0	0	1	0
pbif2→cboc	0	0	0	0	1

[0105] Wie zuvor erwähnt wurde, können die Ausführungsbeispiele des Diagnosesystems Schlußfolgerungsmaschinen umfassen, die verschiedene Techniken zum Diagnostizieren von Fehlern verwenden. Mittels eines Beispiels kann ein Algorithmus oder eine regelbasierte Kantenklassifizierung und eine Kantenklassifizierung durch eine ereignisbasierte Fehlersimulation verwendet werden.

[0106] Im Hinblick auf die regelbasierte Kantenklassifizierung können, anstelle des Verarbeitens des Graphen und der Testeinschränkungen und eines Datenflußmodells in Sätze von Gleichungen zur Optimierung (vorstehend beschrieben), die gleichen Informationen unter Verwendung der Regeln ausgewertet werden. Diese Re-

geln können angepaßt sein, um Kanten eines Datenflußmodells als gut oder verdächtig zu klassifizieren. Speziell könnte die regelbasierte Kantenklassifizierung als ein Algorithmus durch eine Programmiersprache wie C oder Prolog implementiert sein. Als weiteres Beispiel könnte die regelbasierte Kantenklassifizierung durch eine einschränkungsbasierte Technologie wie CLP implementiert sein.

[0107] Typischerweise sind die Einschränkungen gemäß dem Graphen G(V, E) und der Abbildung M relevant. Zum Beispiel hält ein Busscheitelpunkt bestimmten Flußeinschränkungen, wie vorstehend angegeben ist, ein; ein Nicht-Ausbreitungs-Scheitelpunkt hält bestimmten Einschränkungen ein, alle Kanten halten die EDGE-CONSERVE-Einschränkung usw. ein. Die Einschränkungen dienen als eine präzise Definition der Bedeutung des Datenflußgraphen und sind nicht vom Ausführungsbeispiel, das zum Erzeugen einer Diagnose verwendet wird, abhängig.

[0108] Die Einschränkungen, die den Scheitelpunkten, Kanten und Zählern zugeordnet sind, werden untersucht, um Diagnosen zu bestimmen. Typischerweise weist jeder Scheitelpunkt einen relevanten Satz von Flußeinschränkungen, die durch Lambda bestimmt werden, auf. Zusätzlich umfaßt jede Kante typischerweise einen zugeordneten Satz von Einschränkungen, die die Konservierung von Datenpaketen über dieser Kante, z. B. EDGECONSERVE, beschreiben. Ferner umfaßt jeder Zähler typischerweise einen Satz von Einschränkungen, die durch G, die Abbildung M und durch die gemessenen Testergebnisse aus dem SUT definiert sind.

[0109] Bei einem regelbasierten Ausführungsbeispiel wird ein lineares Programm für die allgemeinen Einschränkungen nicht allgemein verwendet, sondern stattdessen wird ein graphunabhängiger Algorithmus verwendet, um G zu durchlaufen und die notwendigen Einschränkungen anzuwenden, um eine Diagnose zu bestimmen, die mit dem SUT und den Testergebnissen übereinstimmt.

[0110] Es ist zu beachten, daß die allgemeinen Einschränkungen auch als Regeln, die zusammen mit G und der Abbildung M in eine Regelverarbeitungsmaschine eingegeben werden, ausgedrückt werden können. Eine solche Regelverarbeitungsmaschine durchläuft dann G, wendet die Einschränkungen an und bestimmt eine Diagnose, die mit G, M und den Testergebnissen übereinstimmt.

[0111] Zurückkehrend zum Datenflußmodell **800** von Fig. 8 und den Ergebnissen des Syndroms 1 von Tabelle 5 werden die Informationen, die dem Datenflußmodell **800** zugeordnet sind und Syndrom 1 nun unter Verwendung einer exemplarischen regelbasierten Kantenklassifizierungstechnik analysiert.

[0112] Man erinnere sich, daß fünf Zähler im Falle des intelligenten Puffers verwendet werden können: $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4, \psi_5\}$. Die Abbildung M ist gegeben durch:

$M((start, n2pb), t, good) = \psi_1$

$M((start, n2pb), r, good) = \psi_2$,

$M((n2pb, pbifl), r, good) = M((buff, pbif2), r, good) = \psi_3$,

$M((pbif2, cboc), r, good) = \psi_4$ und

$M((pbif, buff), r, good) = \psi_5$ und

wobei die Zählerwerte folgende sind: $\psi_1 = 10$, $\psi_2 = 9$, $\psi_3 = 18$, $\psi_4 = 9$, $\psi_5 = 9$.

[0113] Wird die Analyse mit dem Kantenstart → n2pb begonnen, kann bestimmt werden, daß die Zähler 1 und 2 Informationen enthalten, die dieser Kante entsprechen. Speziell enthält der Zähler 1 Informationen bezüglich der Anzahl von guten Paketen, die auf der Kante übertragen wurden, und der Zähler 2 enthält Informationen bezüglich der Anzahl von guten Paketen, die von der Kante empfangen werden. Es ist zu beachten, daß im Hinblick auf eine beliebige Kante, wenn die Anzahl von guten Paketen, die zur Kante übertragen wurden, gleich der Anzahl von guten Paketen ist, die von der Kante empfangen wurden, die Kante nicht verdächtig ist. Im Hinblick auf den Kantenstart → n2pb ist jedoch die Anzahl von guten Paketen, die von dieser Kante empfangen wurden, nicht gleich der Anzahl von guten Paketen, die an diese Kante übertragen wurden, d. h. Zähler 1 – Zähler 2 = 1. Man erinnere sich, daß die Anzahl von guten Paketen, die von einer Kante empfangen wurden, gleich der Anzahl von guten Paketen, die an die Kante übertragen wurden, minus der Anzahl von guten Paketen, die auf der Kante übertragen wurden, die verschwanden, minus der Anzahl von guten Paketen, die auf der Kante schlecht gemacht wurden, ist. Daher ist 2-gd-mb = 1 oder, da nur Ganzzahlen verwendet werden, ent-

weder mb(start, n2pb) oder gd(start, n2pb) gleich 1.

[0114] Im Hinblick auf die Kante n2pb-pbifl, sind die Zähler 2 und 3 relevant. Sich daran erinnernd, daß der Zähler 3 alle guten Pakete zählt, die bei pbifl und pbif2 empfangen wurden, sollte der Zähler 3 während einer fehlerfreien Operation einen Wert enthalten, der zweimal so groß wie der Zählerwert von Zähler 2. Die Anwendung dieser Regel offenbart, daß der Wert des Zählers 3 zweimal so groß ist wie der Wert von Zähler 2, daher sollte die Kante n2pb pbifl nicht verdächtig sein. Dies ist eine weitere Anwendung der Zählerregel, die feststellt, daß die Anzahl von guten Paketen, die von einer Kante empfangen wurden, gleich der Anzahl von guten Paketen, die zu der Kante übertragen wurden, minus der Anzahl von guten Paketen, die auf der Kante verschwanden, minus der Anzahl von guten Paketen, die auf der Kante schlecht gemacht wurden, ist. Speziell, da bekannt ist, daß 18 Pakete beim Zähler 3 als gut empfangen wurden, und der Zähler 3 nur zweimal so groß sein kann wie der Wert von Zähler 2, muß die Operation der Kante n2pb, pbifl fehlerfrei gewesen sein. Es ist zu beachten, daß die verbleibenden Kanten in einer ähnlichen Weise klassifiziert werden könnten, wie einem Fachmann offenkundig wäre.

[0115] Wie vorstehend erwähnt ist, kann die Kantenklassifizierung durch eine flußereignisbasierte Fehlersimulation auch verwendet werden, um eine Diagnose zu liefern. Speziell können ein Datenflußgraph, zugeordnete Einschränkungen und ein Fehlermodell verwendet werden, um ein Verhaltensmodell zu konstruieren. Die flußereignisbasierte Fehlersimulation dieses Verhaltensmodells kann dann im Hinblick auf ein zeitweise-auftretender-Fehlermodell durchgeführt werden, wobei die Ergebnisse in einem Fehlerwörterbuch gespeichert werden. Dieses Fehlerwörterbuch kann eine Abbildung zwischen den Testergebnissen und den zugeordneten Diagnosen für zeitweise auftretenden Ausfälle in Paketvorrichtungen liefern.

[0116] Obgleich die Verhaltensmodelle und zugeordneten Simulatoren und Fehlersimulatoren für einige analoge und digitale Schaltungen existieren, können diese praktisch verwendet werden, um eine Diagnose von zeitweise auftretenden Fehlern von Komplexes-Paket-Architekturvorrichtungen wie Routern zu erzeugen. Dies ist darin begründet, daß solche Verhaltensmodelle und Simulatoren eine Bit-um-Bit-Beschreibung eines Teststimulus für ein komplexes SUT verwenden, das auf Millionen von Paketen arbeitet, und daher nicht kommerziell praktikabel sind.

[0117] Die Ausführungsbeispiele der Schlußfolgerungsmaschine, die eine flußereignisbasierte Fehlersimulation verwenden, verwenden Verhaltensmodelle, die die Elemente, z. B. Kanten und Scheitelpunkte, eines Datenflußgraphen betriebsmäßig darstellen. Das resultierende Modell auf Teil-, Platinen- oder Systemebene kann praktisch entwickelt und fehlersimuliert werden, um ein Fehlerwörterbuch und so eine Diagnose für zeitweise auftretende Fehler in Paketvorrichtungen zu erzeugen.

[0118] Der logische Prozeß der Fehlersimulation dient allgemein zum Simulieren eines eingebrachten Fehlers, Anwenden einer Beschreibung eines Teststimulus auf die Vorrichtung und Beobachten der Vorrichtungsantwort unter der eingebrachten Fehlerbedingung. Ein Fehlerwörterbuch zeichnet dann die Entsprechung des eingebrachten Fehlers zu einem sichtbaren Ergebnis auf. Der Prozeß wird für jeden Fehlertyp im Fehlermodell wiederholt.

[0119] Im Gegensatz zur herkömmlichen Fehlersimulation stellt der Teststimulus, der durch die Ausführungsbeispiele der Schlußfolgerungsmaschine geliefert wird, nicht die tatsächliche Eingabe in die Vorrichtung in der Form von Einsen und Nullen dar. Speziell ist der verwendete Teststimulus ein Modell oder eine Abstraktion der Eingaben, z. B. die Anzahl von Paketen und ihr Typ. Zusätzlich werden die Ereignisse, die dem SUT-Betrieb entsprechen, simuliert. Als weiterer Unterscheidungspunkt können die Ausführungsbeispiele der Schlußfolgerungsmaschine Fehlermodelle für zeitweise auftretende Ausfälle verwenden.

[0120] Da die flußsignifikanten Ergebnisse, z. B. Anzahl von Paketen, Pakettyp, Inhalt von internen Zählern oder ein anderer Zustand, von Interesse sind, können die Wirkungsgrade erreicht werden, indem die Betriebsmittel nicht zugeordnet werden müssen, um eine Aktivität eines Systems auf Bitebene aufzuspüren. Ein gegebenes flußsignifikantes Testergebnis kann mit simulierten Testergebnissen verglichen werden. Wenn die Ergebnisse übereinstimmen, dann können die simulierten Fehler, die den simulierten Testergebnissen entsprechen, durch Konsultieren des Fehlerwörterbuchs bestimmt werden.

[0121] Nachstehend folgt eine allgemeine Beschreibung eines Ausführungsbeispiels einer Schlußfolgerungsmaschine, die eine ereignisbasierte Fehlersimulation verwendet. Zuerst wird ein Ausführungsbeispiel des Verhaltensmodell beschrieben. Zum Beispiel nimmt ein entsprechendes Verhaltensmodell für einen Busscheitelpunkt jedes paketempfangene Ereignis auf einer beliebigen seiner Ein-Kanten an und erzeugt ein Paket, das

sogar auf allen Aus-Kanten übertragen wurde. Diese Nachbildung umfaßt ein Reproduzieren eines Ereignisses, das anzeigt, das ein Paket auf jeder Aus-Kante übertragen wurde. Wenn der Busscheitelpunkt ein Fallenlassen aller schlechten Pakete (Nicht-Ausbreitungs-Mitglied T(i)) erforderte, dann würde ein beliebiges ankommendes Schlechtes-Paket-Ankunftsereignis aussortiert werden.

[0122] Im Hinblick auf einen Ausbreitungs-Nicht-Bus-Scheitelpunkt mit mehreren Aus-Kanten wird jedes Paket-Ankunfts-Ereignis auf einer beliebigen Kante reproduziert, jedoch nicht notwendigerweise auf allen Kanten. Das zugeordnete Verhaltensmodell implementiert dies durch ein nichtdeterministisches Erzeugen eines paketübertragenen Ereignisses auf einer und nur einer Aus-Kante. In ähnlicher Weise existiert eine Abbildung zwischen allen Scheitelpunkttypen und T(i) zu einem zugeordneten Verhaltensmodell, das gemäß den Einschränkungen arbeitet, die dem Scheitelpunkt und seinen Eigenschaften zugeordnet sind.

[0123] Zusätzlich kann jeder Quellenscheitelpunkt für jeden Test eine gegebene Anzahl von Paketen eines gegebenen Typs liefern. Dies ist als ein Verhaltensmodell realisiert, das die zugeordnete Anzahl von paketübertragenen Ereignissen auf seinen Aus-Kanten erzeugt. Im Hinblick auf eine Senke erzeugt jede Senke, jeder Scheitelpunkt keine neuen Ereignisse, weil sie keine Aus-Kanten aufweist.

[0124] Die Kanteneinschränkungen werden auch auf Verhaltensmodelle abgebildet. Das Kantenmodell wandelt paketübertragene Ereignisse in paketangekommene Ereignisse für den Bestimmungsscheitelpunkt oder das zugeordnete Zählermodell um. Unter den Fehlersimulationsbedingungen kann die Kante paketübertragene Ereignisse gemäß dem Fehlermodell in Schlechtes-Paket-angekommen-Ereignisse oder Gutes-Paket-verschwunden-Ereignisse etc. umwandeln.

[0125] Die Zählereinschränkungen werden auch als Verhaltensmodelle dargestellt. Man erinnere sich, daß die Abbildung M einem Zähler einen Pakettyp (gut/schlecht) und einem Ereignis (Paket tx, Paket rx) mit einer Kante zuordnet. Das Paket tx ist das gleiche wie das übertragene Paket, das Paket rx ist das empfangene Paket. Ein gegebener Zähler überwacht seine zugeordneten Kanten für relevante Ereignisse. Wenn ein relevantes Ereignis eintritt, wird der Zähler inkrementiert.

Ereignisbasierte Fehlersimulation – Beispiel 1 (Einzelnes-Gutes-Paket-Simulation)

[0126] Gemäß dem Testentwurf gibt der Startscheitelpunkt im Verlauf des Tests zehn Gutes-Paket-übertragen-Ereignisse auf der Kante (Start, n2pb) aus. Die Ereignisketten durch die Simulation von einem Paket sind wie folgt:

1. Start signalisiert, daß Paket 1 auf (start, n2pb) übertragen wurde;
2. Zähler ψ_1 sieht sein relevantes Ereignis (gutes Paket auf (start, n2pb) übertragen) und inkrementiert sich selbst;
3. ie Kante (Start, n2pb) sieht das Paket-übertragen-auf-Ereignis und konvertiert dasselbe in ein Paket-empfangen-von-Ereignis für die Kante (start, n2pb);
4. Zähler ψ_2 sieht sein relevantes Ereignis (gutes Paket empfangen von (start, n2pb)) und inkrementiert sich selbst;
5. Scheitelpunkt n2pb sieht ein gutes Paket, empfangen von (start, n2pb) und erzeugt ein gutes Paket, das auf (n2pb, pbif) übertragen wurde;
6. Kante (n2pb, pbif) sieht ein Gutes-Paket-übertragen-auf-Ereignis und wandelt es in ein Gutes-Paket-empfangen-von-Ereignis um;
7. Zähler ψ_3 sieht sein relevantes Ereignis und inkrementiert sich selbst;
8. Knoten pbif sieht ein Gutes-Paket-empfangen-von-Ereignis und erzeugt ein Gutes-Paket-übertragen-auf (pbif, cboc). Für ein anschließendes Paket kann sich pbif dazu entschließen, stattdessen (pbif, buff) zu signalisieren; jedoch kann es per definitionem kein Ereignis für beide Kanten signalisieren;
9. Kante (pbif, cboc) sieht dann ein Gutes-Paket-übertragen-auf-Ereignis und wandelt es in ein Gutes-Paket-empfangen-von-Ereignis um;
10. Zähler ψ_4 sieht dann ein Gutes-Paket-empfangen-von-Ereignis und inkrementiert sich selbst;
11. Dann erzeugt der Senkenknoten cboc keine weiteren Ereignisse, da die Lebensdauer dieses Pakets komplett ist.

[0127] Am Ende der guten SUT-Simulation von allen 10 Paketquellen ist $\psi_1 = 10$, $\psi_2 = 10$, $\psi_3 = 20$, $\psi_4 = 10$.

Ereignisbasierte Fehlersimulation – Beispiel 2 (Einzelnes Schlechtes-Paket-Fehlersimulation)

[0128] Wie in dem vorhergehenden Beispiel sind die zehn Pakete Quellen. Jedoch ist eines der zehn Pakete

auf (n2pb, pbif) verfälscht. Die Iteration der flußereignisbasierten Fehlersimulation erfolgt wie folgt. Es ist zu beachten, daß sich der Fehlersimulator dazu entschließt, das Fehlerereignis „ein gutes Paket, das auf Kante (n2pb, pbif) schlecht gemacht wurde", einzubringen.

1. Start signalisiert, daß Paket 1 auf (start, n2pb) übertragen wurde;
2. Zähler ψ_1 sieht sein relevantes Ereignis (gutes Paket das auf (start, n2pb) übertragen wurde) und inkrementiert sich selbst;
3. Kante (n2pb, pbiff) erkennt ihr relevantes Fehlerereignis „ein gutes Paket, das schlecht gemacht wurde" und wandelt das Paket-übertragen-auf-Ereignis in ein Schlechtes-Paket-empfangen-von-Ereignis für (n2pb, pbif) um;
4. Zähler ψ_2 sieht nicht sein relevantes Ereignis (gutes Paket erhalten von (start, n2pb)) und inkrementiert sich daher nicht selbst;
5. Scheitelpunkt n2pb sieht dann ein schlechtes Paket, das von (n2pb, pbif) empfangen wurde und sortiert das Ereignis aus, weil pbif per Modelldefinition keine schlechten Pakete ausbreitet;
6. Fehlersimulation simuliert die Übertragung der verbleibenden neun guten Pakete, ohne ein weiteres Fehlerereignis zu umfassen.

[0129] Die resultierenden Zählwerte sind $\psi_1 = 10$, $\psi_2 = 10$, $\psi_3 = 18$, $\psi_4 = 9$. Der zugeordnete Fehlerwörterbucheintrag umfaßt diese Informationen und das Fehlerereignis, das den Anlaß dazu gab. Wenn die Testergebnisse aus dem SUT mit diesem Eintrag in dem Fehlerwörterbuch übereinstimmen, lautet die Diagnose „Gutes Paket, das auf (n2pb, pbif) schlecht gemacht wurde." Erwähnenswert ist, daß eine anschließende Simulation des Ereignisses „Ein Paket, das schlecht gemacht wurde" auf (buff, pbif) die gleichen simulierten Ergebnisse erzeugt. In diesem Fall umfaßt die Diagnose beide Fehlerereignisse, da beide eine vernünftige Erklärung für die Ergebnisse sein können.

[0130] Der vorstehende Prozeß kann für jeden der Fehlertypen auf jeder der Kanten wiederholt werden. Die Quantitäten der Fehlerereignisse pro Kante können variiert werden und die Anzahl der gleichzeitigen Kantenfehlerer kann ebenfalls gemäß den Bedürfnissen der Anwendung variiert werden. Die Anzahl von Iterationen des Fehlersimulators kann eingestellt werden, um einen Nicht-Determinismus im Paketfluß, wie durch die Definition der Scheitelpunkte angezeigt ist, zu kompensieren. Dies führt zum Erzeugen von Fehlerwörterbucheinträgen für die verschiedenen Arten und Weisen, auf die sich das SUT verhalten könnte.

Patentansprüche

1. Verfahren zum Diagnostizieren von Datenpaketübertragungs-Fehlern in einem getesteten System (SUT) (**110**), wobei das SUT Datenübertragungswege definiert, durch die Datenpakete übertragen werden, wobei das Verfahren folgende Schritte aufweist:
Identifizieren von zumindest einigen Abschnitten der Datenübertragungswege das SUT, die Fehler in die Datenpaketübertragung einbringen können;
Bereitstellen von Einschränkungen, die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege, die identifiziert wurden, definiert;
Diagnostizieren des SUT im Hinblick auf die Einschränkungen;
wobei ein Teststimulus, der als Eingabe für das Diagnostizieren des SUT verwendet wird, ein Modell einer tatsächlichen Eingabe, die dem SUT während des Betriebes zugeführt wird, ist.

2. Verfahren gemäß Anspruch 1, bei dem das Identifizieren ein Bereitstellen eines Datenflußmodells (**120**) entsprechend dem SUT aufweist, wobei das Datenflußmodell Kanten und Scheitelpunkte umfaßt, wobei jede der Kanten, die einem Abschnitt von einem der Datenübertragungswege des SUT entspricht, der Fehler in die Datenübertragung einbringen kann, wobei jede der Kanten zwischen zwei der Scheitelpunkte definiert ist, wobei jeder der Scheitelpunkte zumindest entweder einen Abschluß einer Kante oder eine Position darstellt, wo eine Operation im Hinblick auf die Datenpakete geschehen kann.

3. Verfahren gemäß Anspruch 2, bei dem die Operation, die einem Scheitelpunkt entspricht, zumindest entweder ein Fallenlassen von Daten, ein Aufteilen von Daten, ein Routen von Daten, ein Reproduzieren von Daten, ein Kombinieren von Daten, ein Zählen von Daten und ein Identifizieren eines Typs von Daten umfaßt.

4. Verfahren gemäß Anspruch 2 oder 3, das ferner folgende Schritte aufweist:
Empfangen von Testergebnissen, die dem SUT entsprechen;
wobei das Diagnostizieren ein Analysieren der Testergebnisse im Hinblick auf das Datenflußmodel aufweist.

5. Verfahren gemäß Anspruch 4, bei dem das SUT Zähler umfaßt, die zumindest einigen der Kanten des

Datenflußmodels entsprechen, wobei jeder der Zähler zumindest einem der Scheitelpunkte zugeordnet ist; wobei das Verfahren ferner folgenden Schritt aufweist:
Empfangen von Informationen, die den Testergebnissen von zumindest einigen der Zähler entsprechen.

6. Verfahren gemäß Anspruch 4 oder 5, bei dem das Analysieren der Testergebnisse folgenden Schritt aufweist:
Identifizieren eines Fehlertyps, der einer fehlgeschlagenen Datenübertragung zugeordnet ist.

7. Verfahren gemäß einem der Ansprüche 4 bis 6, bei dem das Analysieren der Testergebnisse folgende Schritte aufweist:
Empfangen von Informationen, die den fehlgeschlagenen Datenübertragungen entsprechen;
Identifizieren von Abschnitten des SUT, die den fehlgeschlagenen Datenübertragungen potentiell zugeordnet sind.

8. Verfahren gemäß Anspruch 7, bei dem das Analysieren der Testergebnisse folgenden Schritt aufweist:
Ausschließen von Abschnitten des SUT, die anfänglich als den fehlgeschlagenen Datenübertragungen zugeordnet identifiziert wurden, wenn bestimmt worden ist, daß jene Abschnitte des SUT nicht zumindest eine der fehlgeschlagenen Datenübertragungen initiiert haben.

9. Verfahren gemäß Anspruch 7 oder 8, bei dem das Identifizieren ein Identifizieren der fehlgeschlagenen Datenübertragungen unter Verwendung einer regelbasierten Kantenklassifizierungstechnik aufweist, die einen graphunabhängigen Algorithmus verwendet, um die Einschränkungen zu durchlaufen und zumindest einige der Einschränkungen anzuwenden, um eine Diagnose zu bestimmen.

10. Verfahren gemäß Anspruch 7 oder 8, bei dem das Identifizieren ein Identifizieren der fehlgeschlagenen Datenübertragungen unter Verwendung einer flußereignisbasierten Fehlersimulationstechnik aufweist, die Verhaltensmodelle verwendet, die Abschnitte des SUT darstellen, um ein Fehlerwörterbuch zu konstruieren.

11. Verfahren gemäß Anspruch 7 oder 8, bei dem das Identifizieren ein Identifizieren der fehlgeschlagenen Datenübertragungen unter Verwendung einer linearen Programmierung aufweist.

12. Verfahren gemäß Anspruch 11, bei dem das lineare Programmieren ein ganzzahliges Programmieren ist.

13. System zum Diagnostizieren von Datenpaketübertragungs-Fehlern in einem getesteten System (SUT) (**110**), wobei das System folgende Merkmale aufweist:
ein Datenflussmodell (**120**), das zumindest einige Abschnitte der Datenübertragungswege des SUT darstellt; und
eine Schlussfolgerungsmaschine (**130**), die dem Datenflußmodell zugeordnet ist, wobei die Schlußfolgerungsmaschine angepaßt ist, um die Testergebnisse, die dem SUT entsprechen, bezüglich des Datenflußmodells auszuwerten, wobei die Schlussfolgerungsmaschine angepasst ist, um die Testergebnisse des SUT bezüglich Einschränkungen auszuwerten, wobei die Einschränkungen Beziehungen von zumindest einigen der Abschnitte des Datenflussmodells definieren; und
wobei ein Teststimulus, der als Eingabe für das Diagnostizieren des SUT verwendet wird, ein Modell einer tatsächlichen Eingabe, die dem SUT während des Betriebes zugeführt wird, ist.

14. System gemäß Anspruch 13, bei dem das Datenflußmodell ein gerichteter Graph (**500**, **700**, **800**) ist, der Kanten und Scheitelpunkte umfaßt, wobei jede der Kanten zumindest einem Abschnitt des Datenübertragungswegs des SUT entspricht, durch den ein Fehler eingebracht werden kann, wobei jede der Kanten durch zwei der Scheitelpunkte definiert ist.

15. System gemäß Anspruch 13, bei dem die Schlußfolgerungsmaschine die fehlgeschlagenen Datenübertragungen unter Verwendung einer regelbasierten Kantenklassifizierungstechnik identifiziert, die einen graphunabhängigen Algorithmus verwendet, um die Einschränkungen zu durchlaufen und zumindest einen Teil der Einschränkungen anzuwenden, um eine Diagnose zu bestimmen.

16. System gemäß Anspruch 13 oder 15, bei dem die Schlußfolgerungsmaschine die fehlgeschlagenen Datenübertragungen unter Verwendung einer ereignisbasierten Fehlersymulationstechnik identifiziert, die Verhaltensmodelle verwendet, die Abschnitte des SUT darstellen, um ein Fehlerwörterbuch zu konstruieren.

17. System gemäß einem der Ansprüche 13 bis 16, bei dem die Schlußfolgerungsmaschine ein Identifizieren der fehlgeschlagenen Datenübertragungen unter Verwendung einer linearen Programmierung aufweist.

18. System gemäß Anspruch 17, bei dem das lineare Programmieren ein ganzzahliges Programmieren ist.

19. System gemäß einem der Ansprüche 13 bis 18, bei dem die Schlußfolgerungsmaschine angepaßt ist, um Informationen zu empfangen, die den fehlgeschlagenen Datenübertragungen entsprechen, und um Abschnitte des SUT zu identifizieren, die den fehlgeschlagenen Datenübertragungen potentiell zugeordnet sind.

20. System zum Diagnostizieren von Datenpaketübertragungs-Fehlern in einem getesteten System (SUT), wobei das System folgende Merkmale aufweist:
eine Einrichtung zum Empfangen von Testergebnissen, die Übertragungen von Datenpaketen durch zumindest einige der Abschnitte der Datenübertragungswege des SUT entsprechen; und
eine Einrichtung zum Diagnostizieren des SUT im Hinblick auf die Einschränkungen, die die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege des SUT definieren;
wobei ein Teststimulus, der als Eingabe für das Diagnostizieren des SUT verwendet wird, ein Modell einer tatsächlichen Eingabe, die dem SUT während des Betriebes zugeführt wird, ist.

21. System gemäß Anspruch 20, das ferner folgendes Merkmal aufweist:
eine Einrichtung zum Testen des SUT, um die Testergebnisse zu erzeugen.

22. Diagnosesystem, das auf einem computerlesbaren Medium gespeichert ist, wobei das Diagnosesystem angepaßt ist, um Datenpaketübertragungs-Fehler in einem getesteten System (SUT) zu diagnostizieren, wobei das Diagnosesystem folgende Merkmale aufweist:
eine Logik, die konfiguriert ist, um zumindest einige Abschnitte der Datenübertragungswege des SUT zu identifizieren, die zum Einbringen von Fehlern in die Datenpakets-Übertragung fähig sind;
eine Logik, die konfiguriert ist, um Einschränkungen zu liefern, die Datenpaketübertragungs-Beziehungen von zumindest einigen der Abschnitte der Datenübertragungswege definieren;
eine Logik, die konfiguriert ist, um das SUT bezüglich der Einschränkungen zu diagnostizieren;
wobei ein Teststimulus, der als Eingabe für das Diagnostizieren des SUT verwendet wird, ein Modell einer tatsächlichen Eingabe, die dem SUT während des Betriebes zugeführt wird, ist.

23. Diagnosesystem gemäß Anspruch 22, bei dem die Logik, die zum Diagnostizieren konfiguriert ist, folgende Merkmale aufweist:
eine Logik, die konfiguriert ist, um ein Datenflußmodell zu liefern; und
eine Logik, die konfiguriert ist, um das SUT bezüglich eines Datenflußmodells zu analysieren.

24. Diagnosesystem gemäß Anspruch 22 oder 23, bei dem die Logik, die zum Diagnostizieren konfiguriert ist, eine Logik umfaßt, die konfiguriert ist, um Informationen zu erzeugen, die den Fluß von Daten anzeigen, die einem Zeitpunkt der Fehlererfassung zugeordnet sind.

25. Diagnosesystem gemäß einem der Ansprüche 22 bis 24, bei dem die Logik, die zum Diagnostizieren konfiguriert ist, eine Logik umfaßt, die zum Identifizieren von Abschnitten des SUT konfiguriert ist, die den fehlgeschlagenen Datenübertragungen potentiell zugeordnet sind.

26. Diagnosesystem gemäß Anspruch 25, bei dem die Logik, die zum Diagnostizieren konfiguriert ist, eine Logik umfaßt, die konfiguriert ist, um Komponenten auszuschließen, die anfänglich als den fehlgeschlagenen Datenübertragungen zugeordnet identifiziert waren.

Es folgen 5 Blatt Zeichnungen

Anhängende Zeichnungen

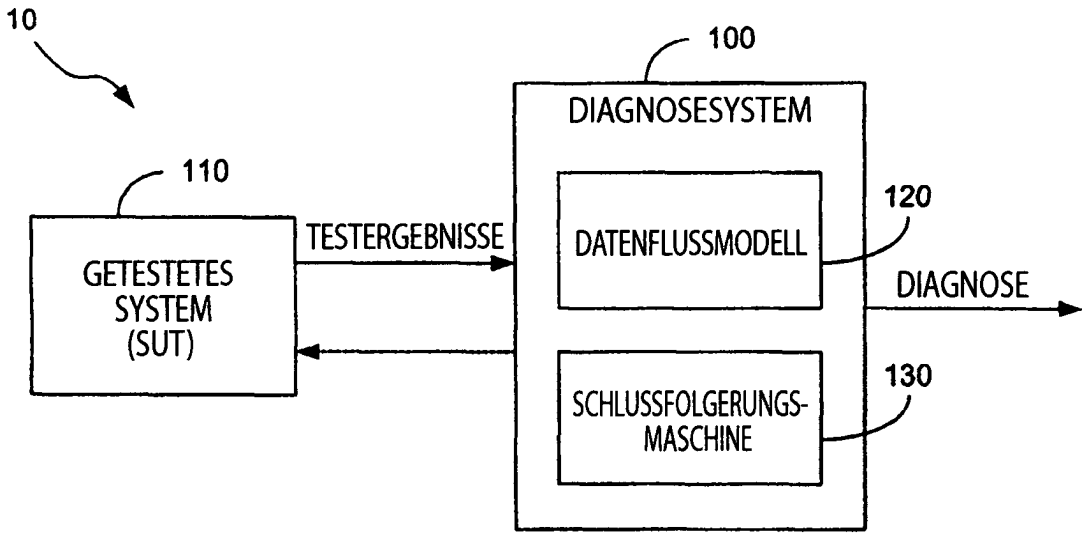


FIG. 1

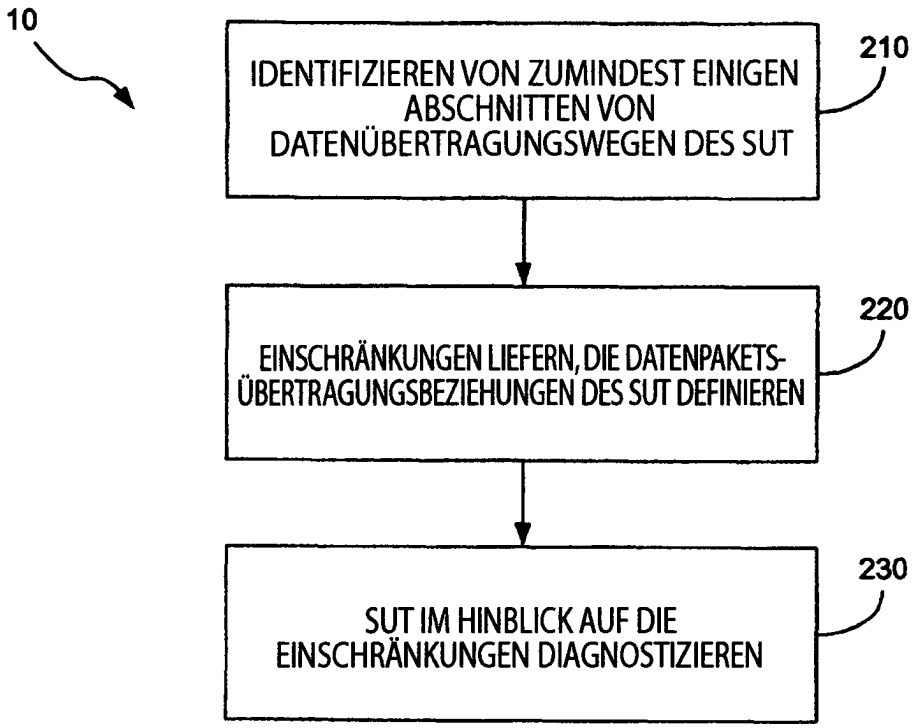


FIG. 2

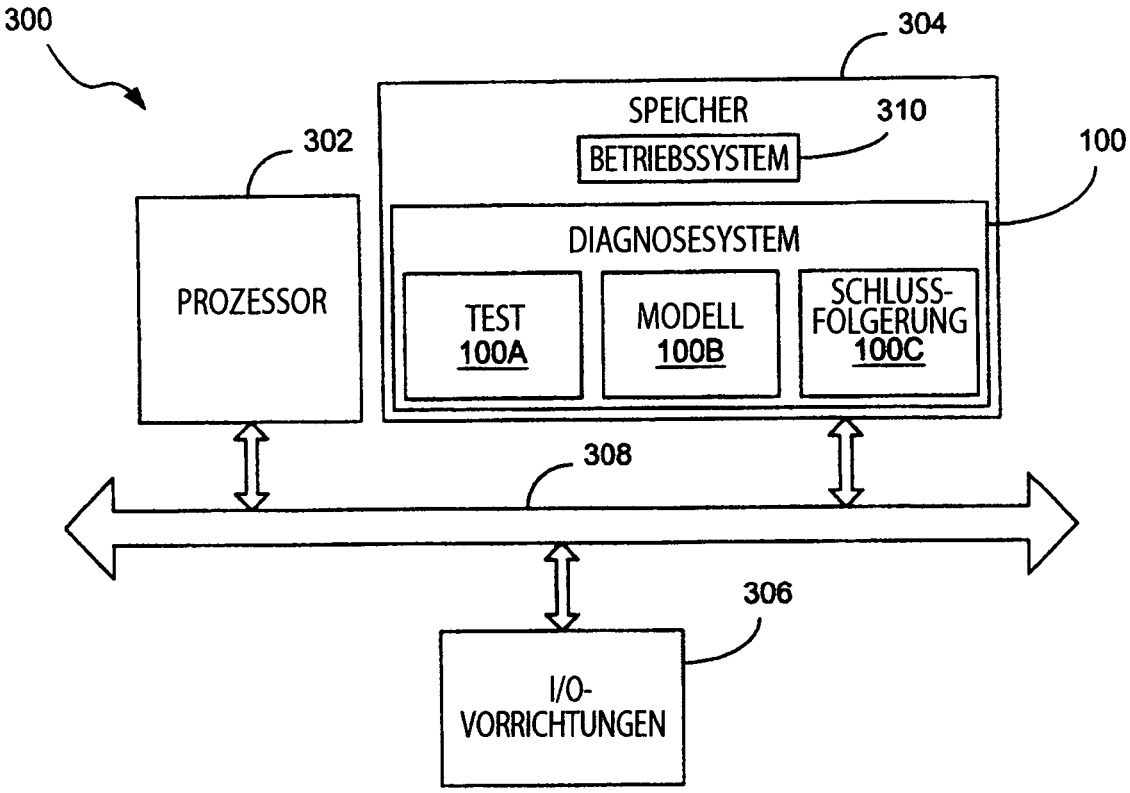

300
304
SPEICHER
310
BETRIEBSSYSTEM
302
100
DIAGNOSESYSTEM
PROZESSOR
TEST
100A
MODELL
100B
SCHLUSS-
FOLGERUNG
100C
308
306
I/O-
VORRICHTUNGEN

FIG. 3

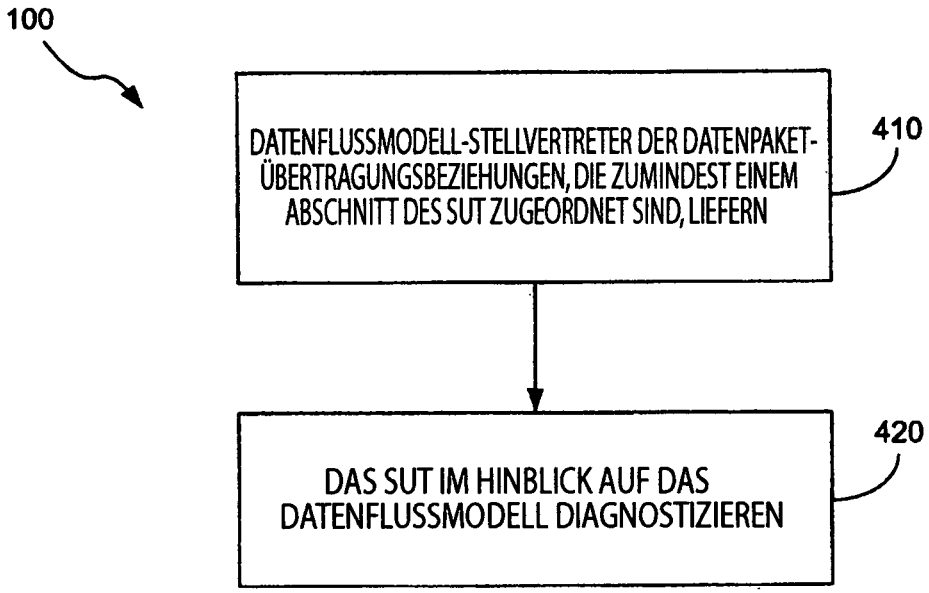

100
DATENFLUSSMODELL-STELLVERTRETER DER DATENPAKET-ÜBERTRAGUNGSBEZIEHUNGEN, DIE ZUMINDEST EINEM ABSCHNITT DES SUT ZUGEORDNET SIND, LIEFERN
410
DAS SUT IM HINBLICK AUF DAS DATENFLUSSMODELL DIAGNOSTIZIEREN
420

FIG. 4

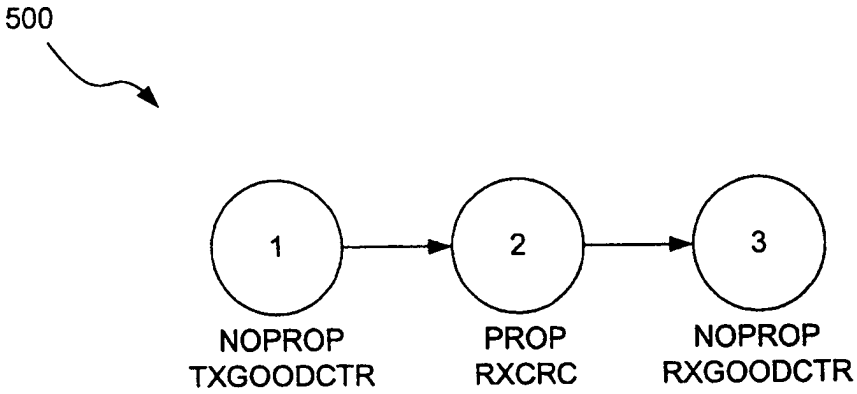

500
1
2
3
NOPROP
TXGOODCTR
PROP
RXCRC
NOPROP
RXGOODCTR

FIG. 5

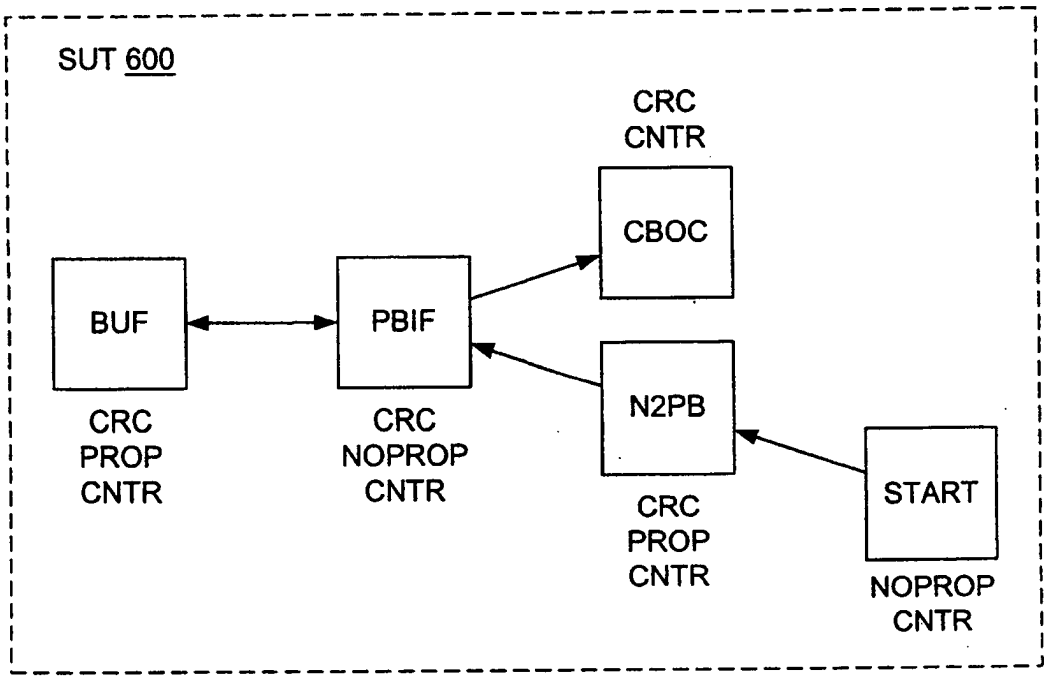

SUT 600
CRC
CNTR
CBOC
BUF
PBIF
N2PB
START
CRC
PROP
CNTR
CRC
NOPROP
CNTR
CRC
PROP
CNTR
NOPROP
CNTR

FIG. 6

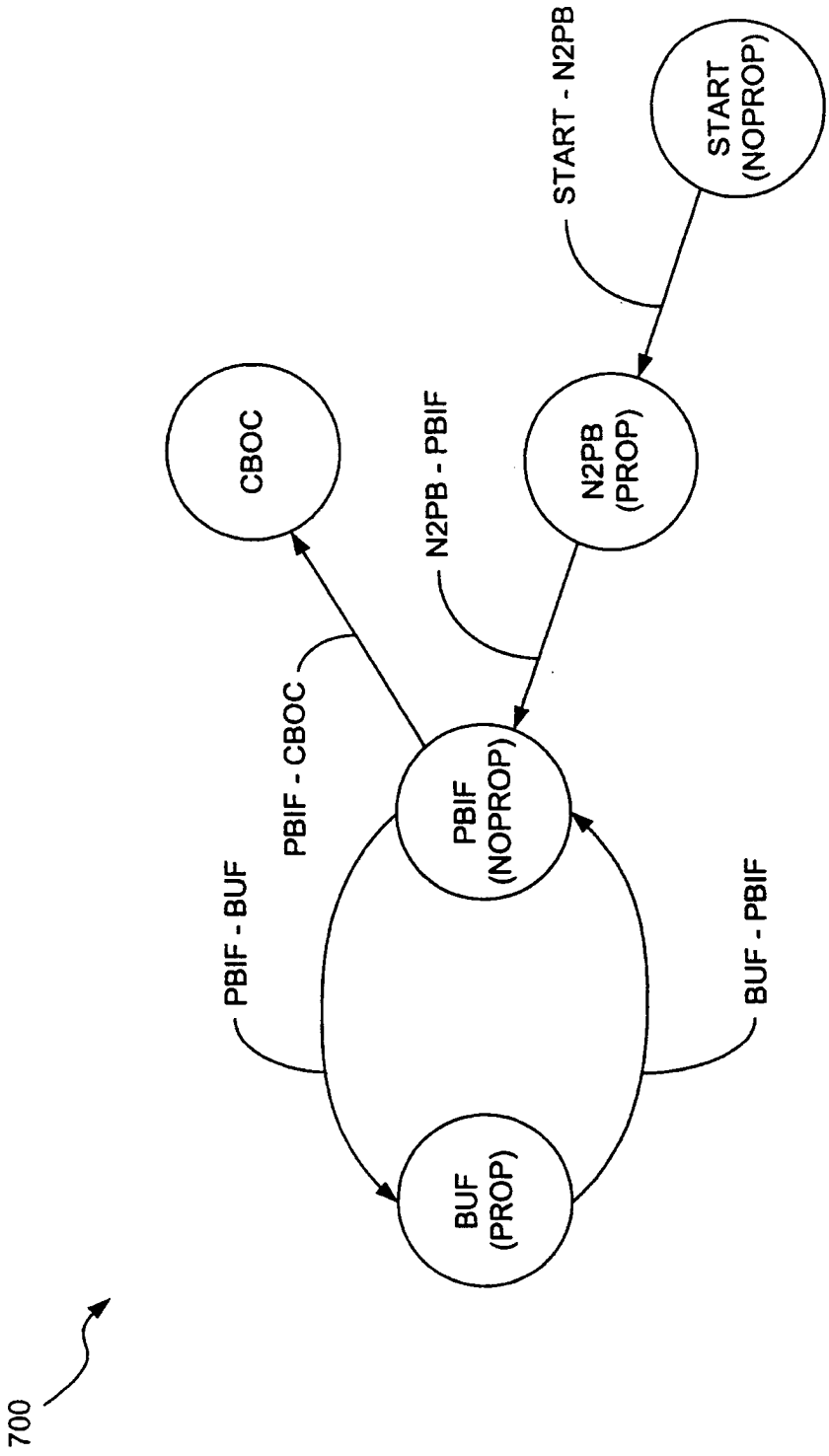

700
START (NOPROP)
START - N2PB
N2PB (PROP)
N2PB - PBIF
CBOC
PBIF - CBOC
PBIF (NOPROP)
PBIF - BUF
BUF - PBIF
BUF (PROP)

FIG. 7

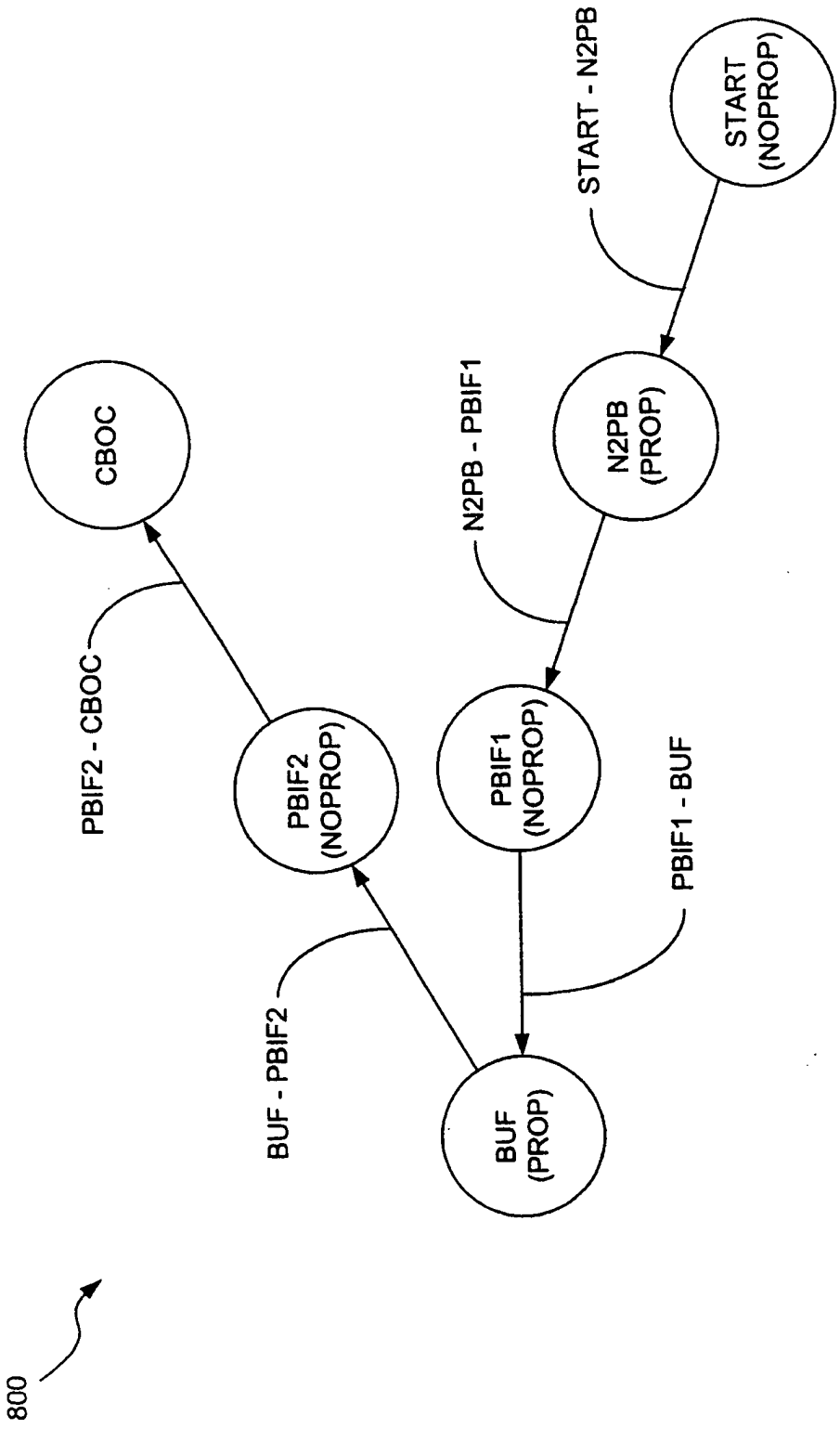

800
START (NOPROP)
START - N2PB
N2PB (PROP)
N2PB - PBIF1
PBIF1 (NOPROP)
PBIF1 - BUF
BUF (PROP)
BUF - PBIF2
PBIF2 (NOPROP)
PBIF2 - CBOC
CBOC

FIG. 8