(19) Deutsches Patent- und Markenamt

(10) **DE 10 2016 204 792 A1** 2016.09.29

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2016 204 792.2**
(22) Anmeldetag: **23.03.2016**
(43) Offenlegungstag: **29.09.2016**

(51) Int Cl.: ***G06F 11/10*** (2006.01)

(30) Unionspriorität:
14/672,128 **28.03.2015** **US**
14/948,273 **21.11.2015** **US**

(71) Anmelder:
International Business Machines Corporation, Armonk, N.Y., US

(74) Vertreter:
Richardt Patentanwälte PartG mbB, 65185 Wiesbaden, DE

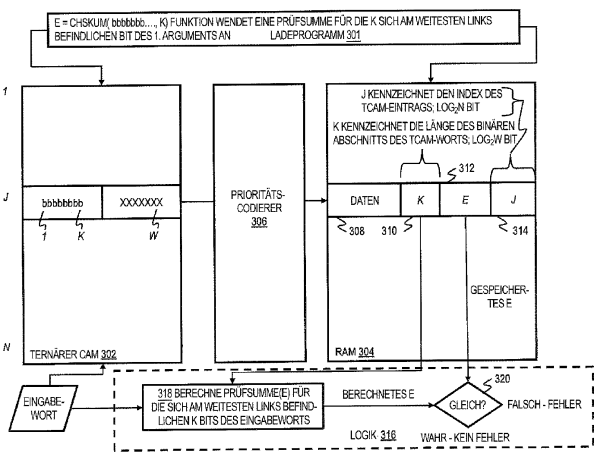
(72) Erfinder:
Abali, Bulent, Yorktown Heights, N.Y., US; Blaner, Bartholomew, Williston, Vt., US

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Gleichzeitig ablaufende Fehlererkennung in einer ternären inhaltsadressierbaren Speicher- (TCAM-) Einheit**

(57) Zusammenfassung: Eine Vielzahl von Datenwörtern werden in einen TCAM geschrieben; jedes weist Binärzeichen und Don't-Care-Zeichen auf. Gleichzeitig wird für jedes der Wörter: eine erste Prüfsumme für die Binärzeichen berechnet; und Folgendes wird in einem entsprechenden Abschnitt eines RAM gespeichert: eine Kennung der Binärzeichen und die erste Prüfsumme. Der ternäre inhaltsadressierbare Speicher wird mit einem Eingabewort abgefragt. Beim Ergeben einer Übereinstimmung in der Abfrage beinhalten weitere Schritte das Abrufen entsprechender Werte der Kennung der Binärzeichen und der ersten Prüfsumme aus dem Direktzugriffsspeicher; das Berechnen einer zweiten Prüfsumme für das Eingabewort unter Verwendung der Kennung der Binärzeichen; und das Ermitteln in Echtzeit, dass die Übereinstimmung falsch positiv ist, wenn die zweite und die erste Prüfsumme nicht gleich sind.



Beschreibung

Gebiet der Erfindung

[0001] Die vorliegende Erfindung betrifft die Fachgebiete der Elektrik, Elektronik und Computer und insbesondere Computerspeicher und dergleichen.

Hintergrund der Erfindung

[0002] In einem herkömmlichen Computerarbeitsspeicher (Direktzugriffsspeicher oder RAM, random access memory) stellt der Benutzer eine Speicheradresse bereit, und der RAM liefert das an dieser Adresse gespeicherte Datenwort. In inhaltsadressierbarem Speicher oder CAM (content-adressable memory) stellt der Benutzer ein Datenwort bereit, und der CAM sucht seinen gesamten Speicher ab, um herauszufinden, ob das Datenwort irgendwo darin gespeichert ist. Wenn dem so ist, liefert der CAM eine Liste der Speicheradresse(n), wo das Wort gefunden wurde (und in einigen Architekturen liefert er auch das Datenwort oder andere zugehörige Datenstücke).

[0003] Binärer CAM ist der einfachste Typ von CAM; er setzt Datensuchbegriffe ein, die lediglich Einsen und Nullen beinhalten. Ternärer CAM (TCAM) lässt für einen oder mehrere Bits einen dritten Übereinstimmungszustand zu, nämlich „Don't Care”.

[0004] Der Artikel „PEDS: A Parallel Error Detection Scheme for TCAM Devices” von Anat Bremler-Barr u. a., IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 18, NR. 5, OKTOBER 2010, 1665–75 offenbart ein Offline-(nicht gleichzeitig ablaufendes)Fehlererkennungsschema.

Kurzdarstellung der Erfindung

[0005] Grundgedanken der Erfindung stellen Techniken zur gleichzeitig ablaufenden Fehlererkennung in einer TCAM-Einheit bereit. In einem Aspekt beinhaltet ein beispielhaftes Verfahren den Schritt des Schreibens einer Vielzahl von Datenwörtern in einen ternären inhaltsadressierbaren Speicher. Jedes der Datenwörter weist Binärzeichen und „Don't-Care”-Zeichen auf. Ein weiterer Schritt beinhaltet gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern Folgendes: Berechnen einer ersten Prüfsumme für die Binärzeichen; und Speichern in einem entsprechenden Abschnitt eines Direktzugriffsspeichers von: einer Kennung der Binärzeichen des entsprechenden der Datenwörter; und der ersten Prüfsumme für das entsprechende der Datenwörter. Noch ein weiterer Schritt beinhaltet das Abfragen des ternären inhaltsadressierbaren Speichers, in den die Vielzahl von Datenwörtern geschrieben wurde, mit einem Eingabewort. Noch ein weiterer Schritt beinhaltet beim Ergeben einer Übereinstimmung in der Abfrage Folgendes: Abrufen entsprechender Werte der Kennung der Binärzeichen und der ersten Prüfsumme aus dem Direktzugriffsspeicher; Berechnen einer zweiten Prüfsumme für das Eingabewort unter Verwendung der Kennung der Binärzeichen; und Ermitteln in Echtzeit, dass die Übereinstimmung falsch positiv (false positive) ist, wenn die zweite und die erste Prüfsumme nicht gleich sind.

[0006] In einem anderen Aspekt beinhaltet eine beispielhafte Vorrichtung eine ternäre inhaltsadressierbare Speicheranordnung; eine Direktzugriffsspeicher-Anordnung; ein Ladeprogramm, das mit der ternären inhaltsadressierbaren Speicheranordnung und der Direktzugriffsspeicher-Anordnung gekoppelt ist; eine Abfrageeinheit, die mit der ternären inhaltsadressierbaren Speicheranordnung gekoppelt ist; und ein Übereinstimmungslogikmodul, das mit der Direktzugriffsspeicher-Anordnung gekoppelt ist. Das Ladeprogramm ist so konfiguriert, dass es eine Vielzahl von Datenwörtern in die ternäre inhaltsadressierbare Speicheranordnung schreibt, wobei jedes der Datenwörter Binärzeichen und Don't-Care-Zeichen aufweist. Das Ladeprogramm ist auch so konfiguriert, dass es gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern Folgendes durchführt: Berechnen einer ersten Prüfsumme für die Binärzeichen; und Speichern in einem entsprechenden Abschnitt der Direktzugriffsspeicher-Anordnung von: einer Kennung der Binärzeichen des entsprechenden der Datenwörter und der ersten Prüfsumme für das entsprechende der Datenwörter. Die Abfrageeinheit ist so konfiguriert, dass sie den ternären inhaltsadressierbaren Speicher, in den die Vielzahl von Datenwörtern geschrieben wurde, mit einem Eingabewort abfragt. Das Übereinstimmungslogikmodul ist so konfiguriert, dass es beim Ergeben einer Übereinstimmung in der Abfrage Folgendes durchführt: Abrufen entsprechender Werte der Kennung der Binärzeichen und der ersten Prüfsumme aus der Direktzugriffsspeicher-Anordnung; Berechnen einer zweiten Prüfsumme für das Eingabewort unter Verwendung der Kennung der Binärzeichen; und Ermitteln in Echtzeit, dass die Übereinstimmung falsch positiv ist, wenn die zweite und die erste Prüfsumme nicht gleich sind.

[0007] Die Verwendung von „Ermöglichen” einer Aktion hierin beinhaltet das Durchführen der Aktion, das Erleichtern der Aktion, das Helfen beim Ausführen der Aktion oder das Verursachen des Durchführens der Aktion. Somit könnten auf einem Prozessor ausgeführte Anweisungen beispielhaft und nicht als Einschränkung eine Aktion ermöglichen, die durch auf einem fernen Prozessor ausgeführte Anweisungen ausgeführt wird, indem geeignete Daten oder Befehle gesendet werden, um das Durchführen der Aktion zu verursachen oder zu unterstützen. Zur Vermeidung von Missverständnissen, wenn ein Akteur eine Aktion durch etwas anderes als das Durchführen der Aktion

ermöglicht, wird die Aktion trotzdem von irgendeiner Einheit oder einer Kombination aus Einheiten durchgeführt.

[0008] Zumindest ein Teil einer oder mehrerer Ausführungsformen der Erfindung oder Elemente davon können in Form eines Computerprogrammprodukts umgesetzt werden, das ein durch einen Computer lesbares Speichermedium mit durch einen Computer verwendbarem Programmcode zum Durchführen der angegebenen Verfahrensschritte beinhaltet. Des Weiteren kann zumindest ein Teil einer oder mehrerer Ausführungsformen der Erfindung oder Elemente davon in Form eines Systems (oder einer Vorrichtung) umgesetzt werden, das/die einen Hauptspeicher und mindestens einen Prozessor, der mit dem Hauptspeicher verbunden ist, enthält und so betrieben werden kann, dass es/sie die beispielhaften Verfahrensschritte durchführen kann. Ferner können unter einem anderen Aspekt eine oder mehrere Ausführungsformen der Erfindung oder Elemente davon in Form eines Mittels zum Ausführen eines oder mehrerer der hierin beschriebenen Verfahrensschritte umgesetzt werden; wobei das Mittel (i) ein oder mehrere Hardware-Module, (ii) ein oder mehrere auf einem durch eine Computer lesbaren Speichermedium (oder mehreren derartigen Medien) gespeicherte und in einem Hardware-Prozessor umgesetzte Software-Module oder (iii) eine Kombination aus (i) und (ii) beinhalten kann; jedes beliebige von (i) bis (iii) setzt die spezifischen hierin dargelegten Techniken um.

[0009] Techniken der vorliegenden Erfindung können erhebliche vorteilhafte technische Auswirkungen bereitstellen; zum Beispiel;

- die Fähigkeit des Bereitstellens einer Online- (auch bezeichnet als gleichzeitig ablaufende oder Echtzeit-)Fehlererkennung in einem TCAM;
- kann mit bestehenden TCAM-Schaltungen umgesetzt werden.

[0010] Diese und weitere Merkmale und Vorteile der vorliegenden Erfindung ergeben sich aus der folgenden ausführlichen Beschreibung von veranschaulichenden Ausführungsformen deren, die in Verbindung mit den beigefügten Zeichnungen zu lesen ist.

Kurzbeschreibung der Zeichnungen

[0011] Fig. 1 zeigt ein gleichzeitig ablaufendes TCAM-Fehlererkennungsschema in Übereinstimmung mit einem Aspekt der Erfindung;

[0012] Fig. 2 zeigt eine alternative Ausführungsform eines gleichzeitig ablaufenden TCAM-Fehlererkennungsschemas in Übereinstimmung mit einem Aspekt der Erfindung;

[0013] Fig. 3 zeigt ein nicht einschränkendes Beispiel von Codes und deren entsprechende Symbole gemäß dem Format aus **Fig.** 1;

[0014] Fig. 4 zeigt einen Ablaufplan eines beispielhaften Verfahrens gemäß einem Aspekt der Erfindung; und

[0015] Fig. 5 zeigt ein Computersystem, das bei der Umsetzung zumindest eines Teils von einem oder mehreren Aspekten und/oder Elementen der Erfindung nützlich sein kann.

Ausführliche Beschreibung von bevorzugten Ausführungsformen

[0016] Eine oder mehrere Ausführungsformen stellen vorteilhafterweise eine gleichzeitig ablaufende Fehlererkennung und/oder -korrektur für TCAMs bereit. In einer oder mehreren Ausführungsformen werden mehrere TCAM-Zeilen parallel durchsucht. Mit einer einzelnen Übereinstimmungsverbindung pro Zeile wie in aktuellen Techniken ist es schwierig (oder sogar unmöglich), eine gleichzeitig ablaufende Fehlererkennung und -korrektur auszuführen. Techniken nach dem Stand der Technik sind somit auf eine nicht gleichzeitig ablaufende Fehlererkennung beschränkt; d. h. Bereinigen von TCAM-Speicher im Hintergrund. Deshalb könnte es sein, dass Fehler übersehen werden. Versuche, damit umzugehen, indem zwei TCAMs mit dupliziertem Inhalt verwendet werden, um jegliche CAM/RAM-(Direktzugriffsspeicher-)Fehler zu erkennen, und/oder drei TCAMs mit dreifachem Inhalt verwendet werden, um jegliche CAM/RAM-Fehler zu erkennen und zu korrigieren, sind aufwendig in Bezug auf Fläche und Leistung.

[0017] Eine Fehlererkennung im Hintergrund bedeutet, dass ein Bereinigungsprozess während Leerlaufzyklen kontinuierlich nach Fehlern in der TCAM-Anordnung sucht; es handelt sich um keine gleichzeitig ablaufende Fehlererkennung. Zum Ermöglichen einer Fehlererkennung im Hintergrund speichert jedes TCAM-Wort Fehlererkennungsbits wie zum Beispiel ein Paritätsbit, einen zyklischen Mehrbit-Redundanzprüfungscode (CRC, cyclic redundancy check) oder einen anderen Fehlererkennungscode. Angenommen, die ternären Datenbits eines TCAM-Worts sind jeweils mit zwei Binärzeichen codiert, sodass eine 0 durch Speichern des Zwei-Bit-Binärcodes (0,1) dargestellt wird, eine 1 als (1,0) gespeichert ist, ein X (Don't Care) als (0,0) gespeichert ist (der Code (1, 1) wird nicht verwendet) und ein einzelnes Paritätsbit zu dem Wort hinzugefügt wird, um eine Einzelbit-Fehlererkennung für die ternären Datenbits zu ermöglichen, so ist ein Einzelbitfehler in dem Paritätsschema, elf Spangen von null auf X (Don't Care), erkennbar, wie auch ein Springen von eins auf X (Don't Care). Ein Springen von null auf eins wird jedoch nicht erkannt, da dies einen Zwei-Bit-Sprung in einer

einzelnen Wahrheitstafel einer TCAM-Zelle bedeuten würde (d. h. (0,1) → (1,0)). In einem CRC-Schema werden möglicherweise mehrere Bit-Fehler erkannt. Das Bereinigungsprogramm sucht parallel nach Fehlern in sämtlichen TCAM-Zeilen. Für eine TCAM-Anordnung mit N Eingaben kann das Bereinigungsprogramm nach 2N Abfragen in der TCAM-Anordnung jeden einzelnen Fehler in sämtlichen TCAM-Zeilen erkennen. In den Bereinigungsabfragen werden Prüfzeichen verwendet. Für den normalen TCAM-Betrieb wird eine Prüfzeicheneingabe mit X (Don't Care) bereitgestellt, um Nichtübereinstimmungen zu beseitigen.

[0018] Der vorstehend erwähnte PEDS-Artikel stellt ein Offline-(nicht gleichzeitig ablaufendes)Fehlererkennungsschema bereit. Im Gegensatz dazu stellen eine oder mehrere Ausführungsformen eine Online-Fehlererkennung bereit, d. h. gleichzeitig mit dem Suchen im TCAM.

[0019] In der Tat stellen eine oder mehrere Ausführungsformen ein gleichzeitig ablaufendes TCAM-Fehlererkennungsschema bereit; z. B. eine gleichzeitig ablaufende Fehlererkennung in einer TCAM-Anordnung. Unter Bezugnahme auf **Fig.** 1 ist eine beispielhafte Ausführungsform darauf angewiesen, dass sich „Don't-Care-"(X)Zeichen in vielen TCAM-Anwendungen auf der rechten Seite des ternären Worts befinden. Insbesondere beinhaltet ein TCAM 302 N Wörter 1 bis N. Das J. Wort beinhaltet insgesamt W Bits mit K sich am weitesten links befindlichen Bits von 1 bis K, die Werte von null oder eins aufweisen (in dem nicht einschränkenden Beispiel aus **Fig.** 1 beträgt K = 8 und W = 15); die verbleibenden sieben Bits ganz rechts sind „Don't-Care"-Bits. In einer oder mehreren Ausführungsformen weist die Huffman-Decodierung diese Form auf, und ein Adressumsetzpuffer (TLB, translation lookaside buffer) und Netzwerkanwendungen haben wahrscheinlich auch dieselbe ternäre Form.

[0020] Eine oder mehrere Ausführungsformen fügen zu dem entsprechenden RAM-Eintrag **308** in dem RAM **304** auch eine zusätzliche Fehlerkennungsinformation **310**, **312**, **314** hinzu, wie nachstehend näher erörtert wird.

[0021] Eine oder mehrere Ausführungsformen erkennen lediglich falsch positive Meldungen (falsche Treffer (false hits)). Falsche Treffer sind sachdienlicher als übersehene Fehler (false misses), wie nachfolgend ebenfalls näher erörtert wird.

[0022] Eine oder mehrere Ausführungsformen benötigen keine Änderungen an bestehenden TCAM-Anordnungsschaltungen. Eine oder mehrere Ausführungsformen setzen die zusätzlichen RAM-Bits **310**, **312**, **314** und eine externe Zufallslogik zum Codieren und Decodieren von Fehlerprüfbits ein. **Fig.** 1 zeigt die zusätzlichen RAM-Bits **310**, **312** und **314** in einem von dem TCAM **302** getrennten RAM **304**. Eine Logik **316**, die sich außerhalb der Elemente **302**, **304** befindet, verarbeitet die Felder **310**, **312**, **314** in einer TCAM-Suche. In dem einfachen Fall, in dem E ein Paritätsbit ist, handelt es sich bei dieser externen Logik um eine Paritätsprüfung der Bits 1 bis K. Unter weiterer Bezugnahme auf **Fig.** 1 nimmt die Logik **316** das Eingabewort für den TCAM **302** und wendet K an, um bei **318** eine Fehlererkennungsprüfsumme (CKSUM(E) zu ermitteln. In dem Entscheidungsblock **320** wird dieses berechnete E mit dem gespeicherten Wert von E **312** verglichen, das aus dem RAM des TCAM-Eintrags abgerufen wurde, der mit der Abfrage übereingestimmt hat. Wenn die beiden gleich sind, gibt es keinen Fehler (WAHR-Verzweigung). Wenn die beiden nicht gleich sind, gibt es einen Fehler (FALSCH-Verzweigung).

[0023] Im Gegensatz zu aktuellen Techniken führen wiederum eine oder mehrere Ausführungsformen eine gleichzeitig ablaufende (Echtzeit-)Fehlererkennung aus.

[0024] Wie in **Fig.** 1 gezeigt ist, codiert ein Ladeprogramm **301** das J. Wort in dem TCAM **302** in einen entsprechenden Eintrag in dem RAM **304**, darunter DATEN **308** sowie die zusätzlich hinzugefügte Fehlererkennungsinformation **310**, **312**, **314**. „DATEN" **308** sind die Daten, die der TCAM **302** für eine Übereinstimmung mit Wort J ausgibt. Zum Beispiel gibt es in einem TLB-Eintrag die übergeordneten virtuellen Adress-Bits (VA), die bei einem Nachschlagen übereinstimmen, und der TLB gibt die reale Adresse (RA) des übereinstimmenden Eintrags aus. Die RA entspricht „DATEN" in **Fig.** 1, während die VA dem Abschnitt „bbbb...b" entsprechen. In Bezug auf die Information **310** kennzeichnet K die Länge des binären Abschnitts des TCAM-Worts; er enthält CEILING($\log_2$W) Bit (das heißt, aufrunden). Zum Beispiel wird mit W = 15, binär 00001111, $\log_2$W auf die ganze Zahl 4 aufgerundet. In Bezug auf die Information **312** wendet die Prüfsummenfunktion E, E = CKSUM (bbbbbbb..., K) für die sich am weitesten links befindlichen K Bit des ersten Arguments eine Prüfsumme an. Hier ist das erste Argument der binäre Übereinstimmungswert in dem TCAM-Eintrag, der von Interesse ist (bei den XXX...X handelt es sich um ternäre Don't-Care-Werte). Siehe vorstehenden Kommentar bezüglich TLB. In Bezug auf die Information **314** kennzeichnet J den Index des TCAM-Eintrags; er enthält CEILING($\log_2$W) Bit. Mit N = 512, binär 0000001000000000, ist $\log_2$N zum Beispiel ganzzahlig 9 Bit.

[0025] Man betrachte nun einen geeigneten TCAM-Schreib-Algorithmus und nehme Bezug auf den Ablaufplan aus **Fig.** 4, der bei **4002** beginnt. Mit Bezug auf Schritt **4004** berechne zum Zeitpunkt des Schreibens in den TCAM eine erste Prüfsumme E für den

binären Abschnitt bbbbbbb des ternären Worts. Einer einfacheren Umsetzung der Prüfsumme halber können „X" Zeichen durch Nullen ersetzt werden. Angesichts der Lehre hierin wird der Fachmann in der Lage sein, ein geeignetes Fehlererkennungsschema für die Prüfsumme auszuwählen – zu Beispielen gehören: Einzelbitparität, CRC und dergleichen. Zusammen mit den RAM-Daten **308** speichere auch:

- Die Länge des binären Abschnitts K des ternären Worts **310** (allgemeiner ausgedrückt, irgendeine Kennung, die einige „Don't-Care"-Bits aufweist, die signifikante Binärwerte haben).
- Die Prüfsumme E **312**,
- (optional) den TCAM Speicherortindex J **314**, der den Speicherort des übereinstimmenden Eintrags bereitstellt, der zum Auffinden des fehlerhaften Eintrags verwendet werden kann.

[0026] Im Vergleich zu aktuellen Techniken erfordert dieser Ansatz zusätzlich $\log_2 W + \log_2 N$ + Prüfbit(s) in der RAM-Anordnung, z. B. 4 + 9 + 1 = 14 zusätzliche Bits für die Fehlererkennung für ein 16-stelliges ternäres Wort, einen TCAM mit 512 Einträgen und Einzelbitparität.

[0027] Wie bei dem Entscheidungsblock **4006** angegeben ist, fahre solange mit dem Schreibprozess fort, bis alle gewünschten Daten geladen wurden.

[0028] Man betrachte nun einen geeigneten TCAM-Abfrage-Algorithmus. Frag in Schritt **4008** den TCAM **302** mit dem Eingabewort INP ab; INP weist Binärzeichen auf; es sind keine X zulässig. Wenn es keine Übereinstimmung gibt (der Entscheidungsblock **4010** ergibt ein NEIN), beende wie in Schritt **4012**. Wenn die Abfrage einen oder mehrere Einträge trifft (der Entscheidungsblock **4010** ergibt ein JA), ermittle in Entscheidungsblock **416**, ob es mehrere Treffer gibt. Wenn dem so ist, löst der Prioritätscodierer **306** dies in Schritt **4018** in genau einen übereinstimmenden Eintrag bei Adresse J auf. Im Fall lediglich eines einzelnen Treffers verwende diese einzige Übereinstimmung wie in Schritt **4016**. In Schritt **4020** gibt der RAM **304** die DATEN **308** aus, falls vorhanden, sowie die Fehlererkennungsbits K, E und J (falls vorhanden). In Schritt **4022** berechne die zweite Prüfsumme (d. h. für INP), und in Schritt **4024** prüfe auf die folgende Bedingung: CKSUM(INP, K) != E, wobei „!=" der Boole'sche Operator NICHT GLEICH ist. Wenn sie gleich sind, gibt es gemäß Schritt **4026** keinen Fehler. Wenn sie nicht gleich sind, wurde gemäß Schritt **4028** in dem binären Abschnitt bbbbbbb des gespeicherten ternären Worts bei Speicherort J ein Fehler erkannt.

[0029] Eine oder mehrere Ausführungsformen erkennen lediglich falsche Treffer, die hierin auch „falsch positiv" genannt werden. Wenn die Abfrage zum Beispiel bbbbbXXXX lautet und bbebbXXXX geliefert wird (d. h. ein falscher Treffer; „e" ist das fehlerhafte Bit in dem TCAM Wort), wird dies erkannt. Im Falle eines Fehlers in einem „Don't-Care"-Bit, wie zum Beispiel wenn die Abfrage bbbbbXXXX lautet und bbbbbXeXX geliefert wird, wird dies andererseits möglicherweise nicht erkannt (d. h. ein übersehender Fehler). Eine oder mehrere Ausführungsformen stellen eine sofortige Fehlererkennung für jede beliebige Anwendung bereit, die einen Schutz vor falschen Treffern benötigt. Ein Schutz vor falschen Treffern ist in vielen Anwendungen wichtiger als übersehene Fehler. In den TLB- und Netzwerk-Anwendungen wären falsche Treffer Sicherheitslücken, die Zugriff auf nicht gestattete Seiten gewähren oder Pakete an nicht beabsichtigte Anschlüsse senden würden. Im Fall der Huffman-Decodierung verursacht ein falscher Treffer eine mehrfache Übereinstimmung und kann im schlimmsten Fall die decodierten Daten beschädigen (was später mit CRC erkannt werden kann).

[0030] Übersehene Fehler werden andererseits nicht erkannt, aber diese sind weniger problematisch. In der TLB-Anwendung würde ein übersehener Fehler zu einem erneuten Laden des TLB von der Seitentabelle führen, wodurch im Prinzip der Fehler korrigiert wird. In der Netzwerk-Anwendung führt ein übersehener Fehler dazu, dass Pakete fallengelassen werden; üblicherweise kann eine Anwendung auf höherer Ebene verlorene Pakete erneut senden, um den Fehler zu beheben. Im Fall der Huffman-Decodierung liefert ein übersehener Fehler aufgrund der präfixfreien Eigenschaft den Fehler „das Code-Wort wurde in dieser Tabelle nicht gefunden", was die die Huffman-Decodierung verwendende Anwendung sofort zum Abbrechen kennzeichnet.

[0031] Nun betrachte man beispielhafte Fehlerarten. Man nehme die folgenden Fehlerübergänge an, wobei die Werte in Klammern die interne Darstellung von ternären Logikwerten sind. Bei Annahme eines Einzelbitfehlers ist der Übergang 0↔1 nicht möglich. Der einzige Fehler, der eine falsch positive Meldung erzeugt, ist ein Null- oder Eins-zu-X-Fehler. Auch wird ein nicht verwendeter Zustand namens Y eingeführt. Er erzeugt abhängig von der TCAM-Logik entweder eine Übereinstimmung oder eine Nichtübereinstimmung; beide Fälle werden gemäß der vorstehenden Erörterung abgedeckt.

- 0 = (0,1) → Einzelbitfehler führt entweder zu X oder Y
- 1 = (1,0) → Einzelbitfehler führt entweder zu X oder Y
- X = (0,0) → Einzelbitfehler führt entweder zu 0 oder 1
- Y = (1,1) undefiniert; diese Zustandskombination wird nicht verwendet. Wenn sie aufgrund eines Bitfehlers vorliegt, wird sie als Fehler angese-

hen. Unter Verwendung derselben Argumentation wie vorstehend verursacht sie einen übersehenen Fehler, der weniger problematisch als ein falscher Treffer ist.

[0032] Man betrachte nun eine beispielhafte Verwendung der Ausführungsform aus **Fig.** 1. Huffman-Codes sind Codes mit variabler Länge. Ein komprimierter Datenstrom kann eine Verkettung mehrerer Huffman-Codes mit keinen abgrenzenden Markierungen zwischen den Symbolen enthalten. Ein wie in **Fig.** 1 gezeigter TCAM **302** kann Huffman-codierte Datenströme decodieren. In einem nicht einschränkenden Beispiel nehme man an, dass die Buchstabensymbole A, B, C, D durch die Huffman-Codes 0, 10, 110 bzw. 111 dargestellt sind. Man beachte, dass die Codes Bit-Längen von 1, 2, 3 bzw. 3 Bits aufweisen. Man nehme an, dass ein komprimierter Datenstrom die Symbole der Reihe nach beinhaltet:
ABABDCA...

[0033] Da die Folge jedoch Huffman-codiert ist, beinhaltet der Strom jedoch:
0 10 0 10 111 110 0;
mit der Erklärung halber eingefügten Leerzeichen; in der Praxis haben die codierten Datenströme keine Leerzeichen:
0100101111100.

[0034] Es ist die Aufgabe des TCAM **302** in diesem Beispiel, die Codes in diesem Bit-Strom zu beschreiben und die Buchstabensymbole A, B, C, D aus dem Binärstrom wiederzugewinnen. Von daher hätte der Decoder den TCAM mit den Codes und deren entsprechenden Symbolen gemäß dem Format aus **Fig.** 1 geladen. Dies ist in der Tabelle aus **Fig.** 3 gezeigt. Hier enthält jeder TCAM-Eintrag ein ternär codiertes 15-Bit-Wort, wobei X das ternäre „Don't-Care"-Symbol darstellt, das der TCAM **302** für jeden beliebigen gegebenen TCAM-Eingabe-Bit-Wert von 0 oder 1 als wahr zuordnet.

[0035] Die Datenfelder enthalten die Buchstabensymbole A, B, C bzw. D. Das K-Feld gibt die Bit-Länge jedes Huffman-codierten Worts auf der linken Seite an. Das E-Feld in diesem Beispiel ist eine gerade Paritätsprüfung des ternären 15-Bit-Worts; wie bereits erwähnt, kann einer einfacheren Umsetzung halber ein X-Bit für die Paritätsberechnung als 0 angenommen werden. Schließlich enthält das J-Feld in diesem nicht einschränkenden Beispiel die TCAM-Zeilenindizes der Einträge A, B, C, D.

[0036] Angenommen, die Folge 0100101111100 ist mit dieser Tabelle zu decodieren. Die Anfangseingabefolge wird dem TCAM **302** in Schritt 1 wie nachfolgend gezeigt zugeführt. Man beachte, dass die einzige übereinstimmende Zeile in diesem Beispiel die erste Zeile ist, im Wesentlichen stimmen die ersten Bits der Eingabe mit dem Zeileneintrag 0xxxx... überein. Dann gibt der RAM-Abschnitt **304** die folgenden Daten aus:
A 1 0 0.

[0037] Dies ist nämlich das Buchstabensymbol A und dessen Länge von 1 Bit. Jetzt, da der Decoder weiß, dass der erste Huffman-Code 1 Bit lang ist, verschiebt er die Eingabefolge in Schritt 2 um 1 Bit nach links und wiederholt den Vorgang. Dieses Mal wird die zweite Reihe, nämlich:
B 2 1 1
aus dem RAM ausgegeben, was auch anzeigt, dass der Code B 2 Bit lang ist. Deshalb verschiebt der Decoder in Schritt 3 die Eingabe um 2 Bit, um den nächsten Code abzurufen. Die Eingabe stimmt nun wiederum mit der ersten Zeile überein, nämlich dem Buchstaben A. Man beachte, dass der Decoder bereits die Folge A B A aus dem codierten Eingabestrom extrahiert hat (siehe vorstehende Folge). Deshalb fährt der Decodierprozess so lange auf diese Weise fort, bis sämtliche Eingabe-Bits verbraucht sind.

[0038] Beziehen wir uns weiterhin auf **Fig.** 3.
0100101111100XX **<<<<** Eingabe Ausgabe **>>>>** A 1 0 0 (übereinstimmende erste Zeile, Code 1 Bit lang) Schritt 1
um 1 Bit nach links verschieben
100101111100XXX **<<<<** Eingabe Ausgabe **>>>>** B 2 0 0 (übereinstimmende zweite Zeile, Code 2 Bit lang) Schritt 2
um 2 Bit nach links verschieben
0101111100XXXXX **<<<<** Eingabe Ausgabe **>>>>** A 1 0 0 (übereinstimmende erste Zeile, Code 1 Bit lang) Schritt 3

[0039] Nun nehme man an, dass in dem TCAM **302** ein Bit-Fehler aufgetreten ist. In einem ersten Fehlerszenario nehme man an, dass eines der ternären X Bits in der ersten Zeile einen Einzelbitfehler aufweist: ein X wurde fälschlicherweise zu dem Binärwert 1, wie nachfolgend in der dritten Spalte gezeigt:
0x1xxxxxxxxxxx A 1 0 0
und die Eingabe in Schritt 1 ist:
0100101111100XX **<<<<** Eingabe

[0040] Wenn dem TCAM **302** die Eingabe präsentiert wird, erzeugt er keine Übereinstimmung, da es keine TCAM-Zeile mehr gibt, die mit der Eingabe übereinstimmt. Deshalb bricht der TCAM-basierende Decoder den Decodierprozess ab, da kein Code gefunden wurde. Somit wurde ein fehlerhaftes Ergebnis vermieden (andernorts hierin als „übersehener Fehler" bezeichnet).

[0041] In dem zweiten Fehlerszenario (falscher Treffer) nehme man an, dass das erste Bit der dritten Zeile fälschlicherweise umgesprungen und wie gezeigt zu einem X geworden ist:
x10xxxxxxxxxxx C 3 0 2

und man nehme an, dass der Decoder Schritt 1 durchführt:
0100101111100XX <<<< Eingabe

[0042] In diesem Fall würden sowohl die erste als auch die dritte Zeile mit der Eingabe übereinstimmen. Wenn die TCAM-Logik ein mehrfach übereinstimmendes Signal hat, hätte der Decoder den Decodierprozess abgebrochen, da dies in der Huffman-Decoder-Anwendung nicht gestattet ist. Wenn kein mehrfach übereinstimmendes Ausgabesignal vorhanden ist, könnte eine Ausführungsform eines TCAM eine der beiden übereinstimmenden Zeilen ausgeben. Wenn die erste Zeile ausgegeben wird, wäre das Decodieren erfolgreich. Wenn die dritte Zeile ausgegeben wird, würde der Decoder diese Ausgabe als fehlerhaft kennzeichnen, da das K-Feld aussagt, dass der Code 3 Bit lang ist, und deshalb berechnet der Decoder die gerade Parität mit den sich am weitesten links befindlichen 3 Bits der Eingabe EVENPARITY(010) = 1, was nicht mit der gespeicherten Parität E = 0 übereinstimmt, wobei das Ergebnis wiederum als fehlerhaft gekennzeichnet wird. Einige TCAM-Umsetzungen geben im Fall einer mehrfachen Übereinstimmung logische ODER für alle Felder aus. Die Ausgabewerte können unsinnige Werte sein; zum Beispiel nicht existierende Buchstabensymbole. Zur Sicherstellung der Fehlererkennung in diesen Fällen würde man anstelle einer einfachen 1-Bit-Parität stärkere Fehlercodes in der E-Spalte verwenden.

[0043] **Fig.** 2 zeigt eine alternative Ausführungsform. Vorteilhafterweise lässt die Ausführungsform aus **Fig.** 2 zu, dass Don't-Care-Bits X in jeder beliebigen Spalte vorhanden sind, wenngleich auf Kosten der Verwendung einer größeren Anzahl von RAM-Bits (RAM-Bits sind billiger als TCAM-Bits). Ähnlich wie die Ausführungsform aus **Fig.** 1 stellt die Ausführungsform aus **Fig.** 2 eine gleichzeitig ablaufende Fehlererkennung (Echtzeit) bereit, erkennt lediglich falsche Treffer und erfordert keine Änderungen an den TCAM-Schaltungen (kann unter Verwendung bestehender TCAM-Anordnungsschaltungen umgesetzt werden).

[0044] Der TCAM **402** beinhaltet N Wörter 1 bis N. Das J. Wort beinhaltet insgesamt W Bits mit K sich am weitesten links befindlichen Bits von 1 bis K, die Werte von null oder eins und in dieser Ausführungsform auch Don't-Care-Bits aufweisen können (in dem nicht einschränkenden Beispiel aus **Fig.** 2 beträgt K = 6 und W = 13); die verbleibenden sieben Bits ganz rechts sind „Don't-Care"-Bits.

[0045] Wie in **Fig.** 2 gezeigt ist, codiert ein Ladeprogramm **401** das J. Wort in dem TCAM **402** in einen entsprechenden Eintrag in dem RAM **404**, darunter Daten **408** sowie die zusätzlich hinzugefügte Fehlererkennungsinformation **410**, **412**, **414**. „DATEN" **408** sind die Daten, die der TCAM **402** für eine Übereinstimmung mit Wort J ausgibt. Zum Beispiel gibt es in einem TLB-Eintrag die übergeordneten virtuellen Adress-Bits (VA), die bei einem Nachschlagen übereinstimmen, und der TLB gibt die reale Adresse (RA) des übereinstimmenden Eintrags aus. Die RA entspricht „DATEN" in **Fig.** 2, während die VA dem Abschnitt „bbbb...b" mit zwei höchstwertigen und sieben als X angegebenen niedrigstwertigen Bits entsprechen. In Bezug auf die Information **410** kennzeichnet die Maske M, wo sich die X Bit in der TCAM-Anordnung in dem gezeigten Beispiel befinden (z. B. 1100001111111), und gibt an, dass das erste, das zweite und das siebte bis dreizehnte Bit X-Bits sind. In Bezug auf die Information **412** wendet die Prüfsummenfunktion E, E = CKSUM(INPUT, M) eine Prüfsumme für das maskierte (M) INPUT an, d. h. nur für den Abschnitt bbbb der Eingabe. In Bezug auf die Information **414** kennzeichnet J den Index des TCAM-Eintrags; er enthält $\log_2 N$ Bit. Mit N = 512 ist $\log_2 N$ zum Beispiel 9 Bit.

[0046] Die Schreib- und Abfrageprozesse sind für die zweite Ausführungsform in **Fig.** 2 dieselben wie für die erste Ausführungsform in **Fig.** 1. Der Prioritätscodierer **406** löst mehrfache Übereinstimmungen wie vorstehend für die erste Ausführungsform beschrieben. Die Logik **416** führt die Berechnung der zweiten Prüfsumme in Schritt **418** aus (hier unter Verwendung des Maskenwerts) und prüft in Block **420** auf Gleichheit mit der ersten Prüfsumme.

[0047] In Bezug auf die TCAM-Anordnungsanforderungen wird die CAM-Anordnungsbreite in einem nicht einschränkenden Beispiel wie folgt bestimmt. Es wird Bezug genommen auf DEFLATE, einen Datenkomprimierungsalgorithmus, der eine Kombination aus dem LZ77-Algorithmus und der Huffman-Codierung verwendet, wie in RFC 1951 spezifiziert ist. Das folgende Beispiel bezieht sich auf den in dem DEFLATE RFC 1951 festgelegten Huffman-Decoder. Ein 16b-TCAM-Übereinstimmungsfeld wird benötigt.
Deflate-Maximal-Bit-Folgen-Länge = 15 Bit;
1 Bit Buchstabe/Länge versus Entfernungswähler;
Gesamt-TCAM-Breite = 16 Zeichen.

[0048] Die TCAM-Einträge lauten wie folgt:

288 Buchstabe/Länge + 32 Entfernungssymbole = 320 Einträge → 512 TCAM-Einträge (Aufrunden auf die nächste Zweierpotenz; die anderen Parameter kommen aus DEFLATE).

[0049] Betrachten wir die RAM-Breite. In Bezug auf Symbole gibt es neun Bits für 288 Buchstabe/Länge-Symbole, fünf Bit für Entfernungssymbole und ein Ein-Bit-Präfix zur Unterscheidung von Buchstabe/Länge und Entfernung. Somit gibt es MAX(9,5) + 1 = 10 Bit zum Kennzeichnen von Symbolen.

[0050] In Bezug auf die Verschiebungswerte beträgt die maximale Bit-Länge des CAM-Worts (Huffman-Code) fünfzehn, und es werden dreizehn zusätzliche Bits für die Entfernungscodes verwendet. Somit sind 15 + 13 = 28 Bit. Deshalb kann die Eingabe um 28 Stellen oder weniger verschoben werden → $\log_2(28)$ = 5 Bit des in dem RAM-Eintrag gespeicherten Verschiebungswerts. Es gibt somit insgesamt = 10 + 5 = 15 Bit für den RAM ohne Fehlererkennung.

[0051] Eine oder mehrere Ausführungsformen fügen eine gleichzeitig ablaufende Fehlererkennung hinzu, somit beträgt die RAM-Wort-Breite $\log_2 W$ + $\log_2 N$ + Fehlerprüfbit. Unter Zuordnung von drei Bit für die gleichzeitig ablaufende Fehlererkennung fügt das RAM-Wort $\log_2$ 16 Zeichen breiten CAM + $\log_2$ 512 + 3 Prüfbit = 16 Bit hinzu (Logarithmen beziehen sich auf Basis 2, normale arithmetische Addition 4 + 9 + 3 = 16).

[0052] Somit beträgt die Gesamt-RAM-Breite 15 + 16 = 29 Bit; Aufrunden auf die nächste Zweierpotenz bedeutet einen 32-Bit breiten RAM.

[0053] Zusammengefasst beträgt die beispielhafte TCAM-Größe 512 × {16 TCAM-Zeichen, 32 RAM-Bit}.

[0054] Mehrfache Übereinstimmungen treten im Nichtfehlerfall nicht auf, können aber im Fehlerfall auftreten. Mehrfache Übereinstimmungen aufgrund von falsch positiven Meldungen sollten einen TCAM nicht beschädigen. Vorzugsweise sollte eine mehrfache Übereinstimmung eins der übereinstimmenden Ergebnisse liefern.

[0055] Herkömmliche Fehlererkennungs- und -korrekturcodes (ECC) für SRAM- und DRAM-Einheiten sind üblicherweise nicht anwendbar auf TCAM (ternäre inhaltsadressierbare Speicher). Die Inhalte von TCAMs werden nicht nach Adresse sondern nach Eingabedaten abgerufen. Von daher wird das Eingabewort parallel in der TCAM-Anordnung gesucht, wobei auf ganze TCAM-Zeilen auf einmal zugegriffen wird, wodurch die Verwendung einer herkömmlichen ECC-Logik nicht praktikabel ist. Aktuelle TCAM-Fehlererkennungstechniken beinhalten eine Fehlererkennung und -korrektur im Hintergrund durch Bereinigen; Codieren von ternären Wörtern (0, 1,X) in der TCAM-Anordnung; und/oder ein wesentliches Verändern von TCAM-Schaltungen zum Erkennen und Korrigieren von Fehlern.

[0056] Eine oder mehrere Ausführungsformen stellen vorteilhafterweise ein Verfahren zur gleichzeitig ablaufenden Fehlererkennung in einer TCAM-Anordnung bereit, wobei falsch positive Meldungen unmittelbar beim Zugreifen erkannt werden, ohne dass die interne Organisation der TCAM-Schaltungen geändert werden muss. Eine oder mehrere Ausführungsformen nutzen den Vorteil der anwendungsspezifischen Eigenschaft der TCAM-Inhalte; nämlich die Tatsache, dass in vielen TCAM-Anwendungen die binären Werte 0,1 in einem in TCAM gespeicherten Wort in den sich am weitesten links befindlichen Bits des Worts gefunden werden und die „Don't-Care"-Werte in den sich am weitesten rechts befindlichen Bits des Worts gefunden werden. Ein Fehlererkennungscode wird auf die sich am weitesten links befindlichen Binärzeichen des Worts angewendet. Die sich ergebende Fehlererkennungsprüfsumme wird auch in dem RAM-Abschnitt des entsprechenden TCAM-Eintrags gemeinsam mit der Länge des binären Abschnitts und dem Index der TCAM-Zeile gespeichert, der später zum Berechnen einer Prüfsumme für die Eingabedaten und zum Erkennen eines fehlerhaften Eintrags verwendet wird. Wenn in dem TCAM Eingabedaten abgefragt werden und diese mit einem der TCAM-Einträge übereinstimmen, werden die sich ergebenden RAM-Daten abgerufen, welche die früher gespeicherte Prüfsumme, die Länge des binären Abschnitts des gespeicherten Worts und den TCAM-Zeilenindex beinhalten. Der Fehlererkennungsalgorithmus wird so auf das Eingabewort angewendet, wie es für das früher gespeicherte TCAM-Wort durchgeführt wurde. Wenn die beiden Prüfsummen nicht übereinstimmen, ist dies ein Hinweis auf eine falsch positive Meldung aufgrund eines Fehlers in der TCAM-Anordnung. Der fehlerhafte Eintrag kann unter Verwendung des Zeilenindex ersetzt werden, der auch als Teil der Eingabeabfrage abgerufen wurde. Eine oder mehrere Ausführungsformen werden preiswert außerhalb des TCAM umgesetzt, ohne die interne Organisation der TCAM-Schaltungen ändern zu müssen.

[0057] Eine oder mehrere Ausführungsformen erfordern keine zweidimensionale Suchfähigkeit.

[0058] Eine oder mehrere Ausführungsformen erfordern es nicht, dass ein CAM in eine spezielle Betriebsart versetzt wird, wenn sich der CAM im Leerlauf befindet, wo keine Vergleiche angestellt werden; eine oder mehrere Ausführungsformen stellen stattdessen eine gleichzeitig ablaufende (Echtzeit-)Fehlererkennung bereit.

[0059] Gleichzeitig ablaufende (Echtzeit-)Fehlererkennungstechniken in Übereinstimmung mit Aspekten der Erfindung erfordern vorteilhafterweise kein regelmäßiges Abtasten des TCAM und erfordern keine separate Prüffolge zum Erkennen von Fehlern.

[0060] Eine oder mehrere Ausführungsformen werden unter Verwendung von Hardware-Speicherelementen, digitaler Logik-Hardware und optional auf einem Universalprozessor laufender Software umgesetzt. Die hierin beschriebenen Logik- und Speicheraspekte können von einem Fachmann für jede bekannte Familie von digitaler Logikschaltung und unter

Verwendung von bekannten Speicherelementen umgesetzt werden.

[0061] Angesichts der bisherigen Erörterung wird man verstehen, dass allgemein ein beispielhaftes Verfahren in Übereinstimmung mit einem Aspekt der Erfindung den Schritt **4004** (der nach Bedarf wie durch den Entscheidungsblock **4006** angegeben wiederholt wird) des Schreibens einer Vielzahl von Datenwörtern in einen ternären inhaltsadressierbaren Speicher **302** oder **402** beinhaltet. Jedes der Datenwörter weist Binärzeichen und „Don't-Care"-(X-)Zeichen auf. Der Fachmann wird verstehen, dass dieser Schritt mit dem TCAM **302**, **402** einhergeht, bei dem es sich um ein Stück Hardware handelt. Das Schreiben in den TCAM wird durch das Ladeprogramm **301**, **401** ausgeführt. Bei dem Ladeprogramm kann es sich zum Beispiel um Software handeln, die Schreibvorgänge in den Arbeitsspeicher durchführt; eine Hardware-Zustandsmaschine, die Daten aus irgendeinem Arbeitsspeicher oder irgendeinem Netzwerk in den TCAM lädt; oder dergleichen. Zum Beispiel kopiert eine geeignete Hardware die Daten von einem dynamischen Direktzugriffsspeicher (DRAM), in den durch die eben beschriebene Software oder Zustandsmaschine Daten geladen wurden. Das heißt, ein nicht einschränkendes Beispiel ist ein Hardware-Laden aus einem DRAM, wobei in den DRAM wiederum Daten durch eine auf einem Universalprozessor laufende Software, eine in digitalen Logikschaltungen umgesetzte Zustandsmaschine usw. geladen wurden.

[0062] Noch immer unter Bezugnahme auf die Schritte **4004** und **4006** beinhaltet ein weiterer Schritt gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern das Berechnen einer ersten Prüfsumme **312**, **412** für die Binärzeichen; und das Speichern eines Datenabschnitts (optional) eines entsprechenden der Datenwörter (DATEN **308**, **408**) in einem entsprechenden Abschnitt eines Direktzugriffsspeichers **304**, **404**; einer Kennung der Binärzeichen des entsprechenden der Datenwörter (K oder Maske); und der ersten Prüfsumme **312**, **412** für das entsprechende der Datenwörter. Der Schritt kann auch durch das Ladeprogramm **301**, **401** ausgeführt werden. Zum Beispiel berechnet die Ladeprogramm-Software K, E und J. Die Software, die Daten in den TCAM lädt, weist einen Zähler 1, 2, 3, ...J...N in sequenziellen Einträgen in dem TCAM auf. Für alle bbb-Bits, die sie speichert, erstellt sie K und E und kennt J, da dies der Index ist, den sie zum Wiederholen des Ladens von Daten in den TCAM verwendet. Dasselbe gilt, wenn das Ladeprogramm eine Hardware-Zustandsmaschine ist. In der Ausführungsform von **Fig.** 2 wird anstelle von K der Maskenwert verwendet.

[0063] Ein weiterer Schritt **4008** beinhaltet das Abfragen des ternären inhaltsadressierbaren Speichers **302**, **402**, in den die Vielzahl von Datenwörtern geschrieben wurde, mit einem Eingabewort. Dieser Schritt kann zum Beispiel durch die Ladeprogrammeinheit ausgeführt werden, die den Schreibvorgang durchführt, oder es könnte sich um eine andere Hardware-Einheit handeln, z. B. eine Hardware-Zustandsmaschine wie zum Beispiel einen Huffman-Decoder, der versucht, Huffman-codierte Eingabeströme zu decodieren. Zum Beispiel erhält die Einheit ein Eingabedatum und stellt es dem TCAM bereit, um zu sehen, wie die Antwort lautet. Wenn eine falsch positive Meldung geliefert wird, entscheidet diese andere Hardware-Einheit (z. B. eine Hardware-Zustandsmaschine wie zum Beispiel ein Huffman-Decoder), was mit dem Fehler zu tun ist. In dem vorstehenden nicht einschränkenden Beispiel wird das Decodieren beendet. Die Zustandsmaschine, die das Abfragen durchführt, kann dieselbe wie oder eine andere als die Zustandsmaschine sein, die das Laden durchführt.

[0064] Noch eine weitere Folge von Schritten kann durch die in Hardware umgesetzte Logik **316**, **416** ausgeführt werden. Beim Ergeben einer Übereinstimmung in der Abfrage (JA-Verzweigung von Block **4010**, mit allen Mehrfachtreffern wie beschrieben gelöst), ruf in Schritt **4020** entsprechende Werte des Datenabschnitts (optional), die Kennung der Binärzeichen und die erste Prüfsumme aus dem Direktzugriffsspeicher ab; berechne in Schritt **4022** eine zweite Prüfsumme für das Eingabewort unter Verwendung der Kennung der Binärzeichen; und ermittle in dem Entscheidungsblock **4024** in Echtzeit, dass die Übereinstimmung falsch positiv ist, wie bei **4028** angegeben ist, wenn die zweite und die erste Prüfsumme nicht gleich sind.

[0065] Optional beinhaltet der Schritt **4004** das Speichern des Zeilenindex J für das entsprechende der Datenwörter, wobei eine Zeile in dem ternären inhaltsadressierbaren Speicher **302**, **402** ermittelt wird, in der das entsprechende der Datenwörter gespeichert ist; und der Abrufschritt **4020** beinhaltet ferner das Abrufen eines entsprechenden Werts aus dem Zeilenindex. Ein weiterer Schritt in diesem Fall beinhaltet das Ersetzen des fehlerhaften Eintrags unter Verwendung des Zeilenindex. Das heißt, Überschreiben der Fehlerzeile mit dem korrekten Wert des Datenworteintrags J, um den Bit-Fehler zu beseitigen. Den korrekten Wert erhält man von dem Ladeprogramm, das zu Beginn Daten in den TCAM geladen hat. Man betrachte einen TLB – der Teil in dem TCAM sind einige Bits der virtuellen Adresse. Der Teil, bei dem es sich um die Daten in dem RAM handelt, ist die reale Adresse nach der Übersetzung. Es gibt ein gültiges Bit, das der virtuellen Adresse zugehörig ist. Es befindet sich wahrscheinlich nicht in dem TCAM; es kann sich in dem RAM befinden; es kann in einem separaten Zwischenspeicher gespeichert sein. Eine Antwort auf einen Fehler besteht darin, das gültige Bit von eins auf null zu ändern, sodass der Eintrag nicht mehr gültig ist und keine Übereinstimmung er-

gibt, wenn der TCAM abgefragt wird. Das Programm, das diese Adressenübersetzung für einen Arbeitsspeicherzugriff haben wollte, erhält einen TLB-Fehltreffer. Bei der Lösung des TLB-Fehltreffers berechnet das Ladeprogramm die Zuordnung der virtuellen Adresse zu der realen Adresse neu und schreibt den Eintrag erneut zurück in den TCAM. Man beachte aus Genauigkeitsgründen, dass die DATEN **308**, **408** nicht genau dasselbe sind wie die Vielzahl von Datenwörtern, die anfangs in den TCAM geschrieben wurden.

[0066] Unter Bezugnahme auf die erste Ausführungsform aus **Fig.** 1 beinhaltet die Kennung der Binärzeichen des entsprechenden der Datenwörter in einigen Fällen beim Speicherschritt **4004** eine Länge (K) eines sich am weitesten links befindlichen binären Abschnitts des entsprechenden der Datenwörter; und das Berechnen der zweiten Prüfsumme beinhaltet das Berechnen der zweiten Prüfsumme für den sich am weitesten links befindlichen binären Abschnitt des Eingabeworts.

[0067] Unter Bezugnahme auf die zweite Ausführungsform aus **Fig.** 2 beinhaltet die Kennung der Binärzeichen des entsprechenden der Datenwörter in einigen Fällen beim Speicherschritt eine Maske (M), die einen binären Abschnitt des entsprechenden der Datenwörter kennzeichnet; und das Berechnen der zweiten Prüfsumme beinhaltet das Berechnen der zweiten Prüfsumme für einen binären Abschnitt des Eingabeworts, der aus der Maske ermittelt wurde.

[0068] M maskiert die Bits, die den X in dem TCAM-Wort entsprechen. Das Eingabeabfragewort ist vollständig binär, aber die den X in dem TCAM-Wort entsprechenden Bits werden zum Zwecke des Berechnens der zweiten Prüfsumme maskiert.

[0069] Bei der Prüfsumme kann es sich zum Beispiel um eine Paritätsprüfung handeln.

[0070] Angesichts der bisherigen Erörterung wird man des Weiteren verstehen, dass eine beispielhafte Vorrichtung gemäß einem anderen Aspekt der Erfindung eine ternäre inhaltsadressierbare Speicheranordnung **302**, **402**; eine Direktzugriffsspeicher-Anordnung **304**, **404**; und ein Ladeprogramm **301**, **401**, das mit der ternären inhaltsadressierbaren Speicheranordnung und der Direktzugriffsspeicher-Anordnung gekoppelt ist, beinhaltet. Auch beinhaltet ist eine Abfrageeinheit, die mit der ternären inhaltsadressierbaren Speicheranordnung gekoppelt ist; und ein Übereinstimmungslogikmodul **316**, **416**, das mit der Direktzugriffsspeicher-Anordnung gekoppelt ist. Die Elemente sind zusammenwirkend so konfiguriert, dass sie die hierin beschriebenen Verfahrensschritte ausführen.

[0071] Wie andernorts angegeben, kann es sich bei dem Ladeprogramm zum Beispiel um ein Software-Programm handeln, das physisch auf einem nichtflüchtigen, durch einen Computer lesbaren Medium enthalten ist und das auf mindestens einem Hardware-Prozessor ausgeführt wird, oder um eine Hardware-Zustandsmaschine.

[0072] Bei der Abfrageeinheit kann es sich um dieselbe Einheit handeln wie das Ladeprogramm oder sie könnte separat sein; zum Beispiel eine Hardware-Zustandsmaschine, die separat und getrennt von dem Ladeprogramm ist. Letzterer Fall ist nicht veranschaulicht, um ein Durcheinander bei den Zeichnungen zu vermeiden. In einem nicht einschränkenden Beispiel beinhaltet die Abfrageeinheit einen Huffman-Decoder.

[0073] In einer oder mehreren Ausführungsformen ist ein mit der ternären inhaltsadressierbaren Speicheranordnung und der Direktzugriffsspeicher-Anordnung gekoppelter Prioritätscodierer **306**, **406** so konfiguriert, dass er, wenn das Abfragen mehrere vermeintliche Übereinstimmungen ergibt, selbige löst, um die Übereinstimmung zu erhalten. Der Prioritätscodierer kann zum Beispiel unter Verwendung von digitalen Logikschaltungen umgesetzt werden. Somit geht der Prioritätscodierer das Problem an, das auftritt, wenn mehr als ein Eintrag in dem TCAM übereinstimmt – der Prioritätscodierer entscheidet darüber, welcher zu melden ist – er meldet normalerweise denjenigen mit dem niedrigsten Index, wobei dies aber ein nicht einschränkendes Beispiel ist.

[0074] Zumindest ein Teil von einer oder mehreren Ausführungsformen der Erfindung bzw. Elemente davon können in Form einer Vorrichtung umgesetzt werden, die einen Speicher und mindestens einen mit dem Speicher verbundenen Prozessor enthält, der so betrieben werden kann, dass er beispielhafte Verfahrensschritte durchführen kann.

[0075] Eine oder mehrere Ausführungsformen verwenden Software, die auf einem Universalcomputer oder einer Workstation läuft. Unter Bezugnahme auf **Fig.** 5 kann eine derartige Umsetzung zum Beispiel einen Prozessor **502**, einen Speicher **504** und eine zum Beispiel durch eine Anzeige **506** und eine Tastatur **508** gebildete Eingabe/Ausgabe-Schnittstelle gebildet werden. In der Verwendung hierin soll der Begriff „Prozessor" jede beliebige Verarbeitungseinheit beinhalten, wie zum Beispiel eine, die eine CPU (zentrale Verarbeitungseinheit) und/oder andere Arten von Verarbeitungsschaltungen enthält. Ferner kann sich der Begriff „Prozessor" auf mehr als einen einzelnen Prozessor beziehen. Der Begriff „Speicher" soll einem Prozessor oder einer CPU zugewiesenen Speicher beinhalten, z. B. RAM (Direktzugriffsspeicher), ROM (Nur-Lese-Speicher), eine feste Speichereinheit (zum Beispiel ein Festplattenlauf-

werk), eine austauschbare Speichereinheit (zum Beispiel eine Diskette), einen Flash-Speicher und dergleichen. Außerdem soll der Ausdruck „Eingabe/Ausgabe-Schnittstelle" in der Verwendung hierin zum Beispiel einen oder mehrere Mechanismen zum Eingeben von Daten in die Verarbeitungseinheit (zum Beispiel eine Maus) sowie einen oder mehrere Mechanismen zum Bereitstellen von der Verarbeitungseinheit zugewiesenen Ergebnissen (zum Beispiel einen Drucker) beinhalten. Der Prozessor **502**, der Speicher **504** und die Eingabe/Ausgabe-Schnittstelle wie die Anzeige **506** und die Tastatur **508** können zum Beispiel über den Bus **510** als Teil einer Datenverarbeitungseinheit **512** miteinander verbunden sein. Geeignete Verbindungen zum Beispiel über den Bus **510** können auch einer Netzwerkschnittstelle **514** wie einer Netzwerkkarte bereitgestellt werden, die als Schnittstelle zu einem Computernetzwerk bereitgestellt werden kann, und einer Medienschnittstelle **516** wie einem Disketten- bzw. CD-ROM-Laufwerk, das als Schnittstelle zu den Medien **518** bereitgestellt werden kann.

[0076] Entsprechend kann eine Anweisungen oder Code zum Durchführen der hierin beschriebenen Methodiken der Erfindung enthaltende Computer-Software in einer oder mehreren der zugehörigen Speichereinheiten (zum Beispiel ROM, fester oder austauschbarer Speicher) gespeichert sein und, wenn sie bereit zur Verwendung ist, teilweise oder ganz (zum Beispiel in RAM) gespeichert und durch eine CPU ausgeführt werden. Derartige Software könnte Firmware, im Arbeitsspeicher befindliche Software, Mikrocode und dergleichen enthalten, ist aber nicht darauf beschränkt.

[0077] Ein zur Speicherung und/oder Ausführung von Programmcode geeignetes Datenverarbeitungssystem enthält zumindest einen Prozessor **502**, der direkt oder indirekt über einen Systembus **510** mit Speicherelementen **504** verbunden ist. Zu den Speicherelementen können ein lokaler Speicher, der während der eigentlichen Umsetzung des Programmcodes eingesetzt wird, Massenspeicher sowie Cachespeicher gehören, die eine vorübergehende Speicherung von zumindest etwas Programmcode bereitstellen, um die Häufigkeit zu verringern, mit der Code während der Umsetzung von dem Massenspeicher abgerufen werden muss.

[0078] Eingabe/Ausgabe- bzw. E/A-Einheiten (darunter Tastaturen **508**, Anzeigen **506**, Zeigegeräte und dergleichen, jedoch nicht darauf beschränkt) können entweder direkt (zum Beispiel über den Bus **510**) oder über mitbeteiligte E/A-Steuereinheiten (der Klarheit halber weggelassen) mit dem System verbunden werden.

[0079] Es können auch Netzwerkadapter wie zum Beispiel die Netzwerkschnittstelle **514** mit dem System verbunden werden, um es dem Datenverarbeitungssystem zu ermöglichen, über mitbeteiligte private oder öffentliche Netzwerke mit anderen Datenverarbeitungssystemen oder entfernt angeordneten Druckern oder Speichervorrichtungen verbunden zu werden. Modems, ein Kabelmodem sowie Ethernet-Karten sind nur einige wenige der momentan verfügbaren Arten von Netzwerkadaptern.

[0080] In der Verwendung hierin, darunter die Ansprüche, enthält ein „Server" ein physisches Datenverarbeitungssystem (zum Beispiel das in **Fig.** 5 gezeigte System **512**), das ein Server-Programm ausführt. Es wird darauf hingewiesen, dass ein derartiger physischer Server eine Anzeige und eine Tastatur enthalten kann, aber nicht muss.

[0081] Es sei bemerkt, dass jedes beliebige hierin beschriebene Verfahren einen zusätzlichen Schritt des Bereitstellens eines eigene auf einem durch einen Computer lesbaren Medium enthaltene Software-Module umfassenden Systems enthalten kann; die Module können zum Beispiel ein beliebiges oder alle der in den Blockschaltbildern bzw. Schaubildern oder anderen Figuren und/oder hierin als durch Software umgesetzt beschriebenen Elemente beinhalten. Die Verfahrensschritte können dann unter Verwendung der eigenen Software-Module und/oder Untermodule des oben beschriebenen Systems ausgeführt werden, die auf dem einen oder mehreren Prozessoren **502** ausgeführt werden. Ferner kann ein Computerprogrammprodukt ein durch einen Computer lesbares Speichermedium mit Code enthalten, der so gestaltet ist, dass er ausgeführt werden kann, um einen oder mehrere hierin beschriebene Verfahrensschritte auszuführen, darunter das Bereitstellen des Systems mit eigenen Software-Modulen.

Beispielhafte integrierte Schaltungsdetails

[0082] Eine oder mehrere hierin beschriebene Ausführungsformen können zumindest teilweise unter Verwendung von integrierten Schaltungschips umgesetzt werden. Die integrierten Schaltungschips können von dem Hersteller in Roh-Wafer-Form (das heißt, auf einem einzelnen Wafer mit mehreren nicht verpackten Chips), als Nacktchip (bare die) oder in verpackter Form vertrieben werden. In letzterem Fall ist der Chip in einer Einzelchipverpackung (wie zum Beispiel einem Kunststoffträger mit an einer Steuerplatine befestigten Leitungen oder einem anderen übergeordneten Träger) oder in einer Mehrchipverpackung (wie zum Beispiel einem keramischen Träger mit Oberflächen- und/oder vergrabenen Verbindungen) angebracht. Auf jeden Fall wird der Chip dann mit anderen Chips, separaten Schaltungselementen und/oder anderen Signalverarbeitungseinheiten entweder als Teil eines (a) Zwischenprodukts wie zum Beispiel einer Steuerplatine oder (b) eines Endprodukts integriert. Das Endprodukt kann je-

des Produkt sein, das integrierte Schaltungschips beinhaltet.

Einzelheiten über das beispielhafte System und Herstellungsprodukt

[0083] Bei der vorliegenden Erfindung kann es sich um ein System, ein Verfahren und/oder ein Computerprogrammprodukt handeln. Das Computerprogrammprodukt kann (ein) durch einen Computer lesbare(s) Speichermedium (oder -medien) beinhalten, auf dem/denen durch einen Computer lesbare Programmanweisungen gespeichert ist/sind, um einen Prozessor dazu zu veranlassen, Aspekte der vorliegenden Erfindung auszuführen.

[0084] Bei dem durch einen Computer lesbaren Speichermedium kann es sich um eine physische Einheit handeln, die Anweisungen zur Verwendung durch ein System zur Ausführung von Anweisungen behalten und speichern kann. Bei dem durch einen Computer lesbaren Speichermedium kann es sich zum Beispiel um eine elektronische Speichereinheit, eine magnetische Speichereinheit, eine optische Speichereinheit, eine elektromagnetische Speichereinheit, eine Halbleiterspeichereinheit oder jede geeignete Kombination daraus handeln, ohne auf diese beschränkt zu sein. Zu einer nicht erschöpfenden Liste spezifischerer Beispiele des durch einen Computer lesbaren Speichermediums gehören die Folgenden: eine tragbare Computerdiskette, eine Festplatte, ein Direktzugriffsspeicher (RAM), ein Nur-Lese-Speicher (ROM), ein löschbarer programmierbarer Nur-Lese-Speicher (EPROM bzw. Flash-Speicher), ein statischer Direktzugriffsspeicher (SRAM), ein tragbarer Kompaktspeicherplatte-Nur-Lese-Speicher (CD-ROM), eine DVD (digital versatile disc), ein Speicher-Stick, eine Diskette, eine mechanisch kodierte Einheit wie zum Beispiel Lochkarten oder gehobene Strukturen in einer Rille, auf denen Anweisungen gespeichert sind und jede geeignete Kombination daraus. Ein durch einen Computer lesbares Speichermedium soll in der Verwendung hierin nicht als flüchtige Signale an sich aufgefasst werden, wie zum Beispiel Funkwellen oder andere sich frei ausbreitende elektromagnetische Wellen, elektromagnetische Wellen, die sich durch einen Wellenleiter oder ein anderes Übertragungsmedium ausbreiten (z. B. durch ein Glasfaserkabel geleitete Lichtimpulse) oder durch einen Draht übertragene elektrische Signale.

[0085] Hierin beschriebene durch einen Computer lesbare Programmanweisungen können von einem durch einen Computer lesbaren Speichermedium auf jeweilige Datenverarbeitungs-/Verarbeitungseinheiten oder über ein Netzwerk wie zum Beispiel das Internet, ein lokales Netzwerk, ein Weitverkehrsnetz und/oder ein drahtloses Netzwerk auf einen externen Computer oder eine externe Speichereinheit heruntergeladen werden. Das Netzwerk kann Kupferübertragungskabel, Lichtwellenübertragungsleiter, drahtlose Übertragung, Leitwegrechner, Firewalls, Vermittlungseinheiten, Gateway-Computer und/oder Edge-Server aufweisen. Eine Netzwerkadapterkarte oder Netzwerkschnittstelle in jeder Datenverarbeitungs-/Verarbeitungseinheit empfängt durch einen Computer lesbare Programmanweisungen aus dem Netzwerk und leitet die durch einen Computer lesbaren Programmanweisungen zur Speicherung in einem durch einen Computer lesbaren Speichermedium innerhalb der entsprechenden Datenverarbeitungs-/Verarbeitungseinheit weiter.

[0086] Bei durch einen Computer lesbaren Programmanweisungen zum Ausführen von Arbeitsschritten der vorliegenden Erfindung kann es sich um Assembler-Anweisungen ISA-Anweisungen (Instruction-Set-Architecture), Maschinenanweisungen, maschinenabhängige Anweisungen, Mikrocode, Firmware-Anweisungen, zustandssetzende Daten oder entweder Quellcode oder Objektcode handeln, die in einer beliebigen Kombination aus einer oder mehreren Programmiersprachen geschrieben werden, darunter objektorientierte Programmiersprachen wie Smalltalk, C++ o. ä. sowie herkömmliche prozedurale Programmiersprachen wie die Programmiersprache „C” oder ähnliche Programmiersprachen. Die durch einen Computer lesbaren Programmanweisungen können vollständig auf dem Computer des Benutzers, teilweise auf dem Computer des Benutzers, als eigenständiges Software-Paket, teilweise auf dem Computer des Benutzers und teilweise auf einem fernen Computer oder vollständig auf dem fernen Computer oder Server ausgeführt werden. In letzterem Fall kann der entfernt angeordnete Computer mit dem Computer des Benutzers durch eine beliebige Art Netzwerk verbunden sein, darunter ein lokales Netzwerk (LAN) oder ein Weitverkehrsnetz (WAN), oder die Verbindung kann mit einem externen Computer hergestellt werden (zum Beispiel über das Internet unter Verwendung eines Internet-Dienstanbieters). In einigen Ausführungsformen können elektronische Schaltungen, darunter zum Beispiel programmierbare Logikschaltungen, im Feld programmierbare Gatter-Anordnungen (FPGA, field programmable gate arrays) oder programmierbare Logikanordnungen (PLA, programmable logic arrays) die durch einen Computer lesbaren Programmanweisungen ausführen, indem sie Zustandsinformationen der durch einen Computer lesbaren Programmanweisungen nutzen, um die elektronischen Schaltungen zu personalisieren, um Aspekte der vorliegenden Erfindung durchzuführen.

[0087] Aspekte der vorliegenden Erfindung sind hierin unter Bezugnahme auf Ablaufpläne und/oder Blockschaltbilder bzw. Schaubilder von Verfahren, Vorrichtungen (Systemen) und Computerprogrammprodukten gemäß Ausführungsformen der Erfindung beschrieben. Es wird darauf hingewiesen, dass jeder

Block der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder sowie Kombinationen von Blöcken in den Ablaufplänen und/oder den Blockschaltbildern bzw. Schaubildern durch durch einen Computer lesbare Programmanweisungen ausgeführt werden können.

[0088] Diese durch einen Computer lesbaren Programmanweisungen können einem Prozessor eines Universalcomputers, eines Spezialcomputers oder einer anderen programmierbaren Datenverarbeitungsvorrichtung bereitgestellt werden, um eine Maschine zu erzeugen, so dass die über den Prozessor des Computers bzw. der anderen programmierbaren Datenverarbeitungsvorrichtung ausgeführten Anweisungen ein Mittel zur Umsetzung der in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder festgelegten Funktionen/Schritte erzeugen. Diese durch einen Computer lesbaren Programmanweisungen können auch auf einem durch einen Computer lesbaren Speichermedium gespeichert sein, das einen Computer, eine programmierbare Datenverarbeitungsvorrichtung und/oder andere Einheiten so steuern kann, dass sie auf eine bestimmte Art funktionieren, so dass das durch einen Computer lesbare Speichermedium, auf dem Anweisungen gespeichert sind, ein Herstellungsprodukt aufweist, darunter Anweisungen, welche Aspekte der/des in dem Block bzw. den Blöcken des Ablaufplans und/oder der Blockschaltbilder bzw. Schaubilder angegebenen Funktion/Schritts umsetzen.

[0089] Die durch einen Computer lesbaren Programmanweisungen können auch auf einen Computer, eine andere programmierbare Datenverarbeitungsvorrichtung oder eine andere Einheit geladen werden, um das Ausführen einer Reihe von Prozessschritten auf dem Computer bzw. der anderen programmierbaren Vorrichtung oder anderen Einheit zu verursachen, um einen auf einem Computer ausgeführten Prozess zu erzeugen, so dass die auf dem Computer, einer anderen programmierbaren Vorrichtung oder einer anderen Einheit ausgeführten Anweisungen die in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder festgelegten Funktionen/Schritte umsetzen.

[0090] Die Ablaufpläne und die Blockschaltbilder bzw. Schaubilder in den Figuren veranschaulichen die Architektur, die Funktionalität und den Betrieb möglicher Ausführungen von Systemen, Verfahren und Computerprogrammprodukten gemäß verschiedenen Ausführungsformen der vorliegenden Erfindung. In diesem Zusammenhang kann jeder Block in den Ablaufplänen oder Blockschaltbildern bzw. Schaubildern ein Modul, ein Segment oder einen Teil von Anweisungen darstellen, die eine oder mehrere ausführbare Anweisungen zur Ausführung der bestimmten logischen Funktion(en) aufweisen. In einigen alternativen Ausführungen können die in dem Block angegebenen Funktionen in einer anderen Reihenfolge als in den Figuren gezeigt stattfinden. Zwei nacheinander gezeigte Blöcke können zum Beispiel in Wirklichkeit im Wesentlichen gleichzeitig ausgeführt werden, oder die Blöcke können manchmal je nach entsprechender Funktionalität in umgekehrter Reihenfolge ausgeführt werden. Es ist ferner anzumerken, dass jeder Block der Blockschaltbilder bzw. Schaubilder und/oder der Ablaufpläne sowie Kombinationen aus Blöcken in den Blockschaltbildern bzw. Schaubildern und/oder den Ablaufplänen durch spezielle auf Hardware beruhende Systeme umgesetzt werden können, welche die festgelegten Funktionen oder Schritte durchführen, oder Kombinationen aus Spezial-Hardware und Computeranweisungen ausführen.

[0091] Die hierin verwendete Terminologie dient lediglich dem Zweck des Beschreibens bestimmter Ausführungsformen und soll die Erfindung nicht einschränken. Die Verwendung der Singularform „ein", „eine" bzw. „der", „die", „das" hierin soll ebenfalls die Pluralformen einschließen, es sei denn, etwas anderes ergibt sich deutlich aus dem Zusammenhang. Es wird ferner darauf hingewiesen, dass die Begriffe „aufweisen" und/oder „aufweisend", wenn sie in dieser Beschreibung verwendet werden, das Vorhandensein von aufgeführten Eigenschaften, ganzen Zahlen, Schritten, Operationen, Elementen und/oder Komponenten angeben, jedoch nicht das Vorhandensein oder das Hinzufügen einer oder mehrerer anderer Eigenschaften, ganzer Zahlen, Schritte, Operationen, Elemente, Komponenten und/oder Gruppen hiervon ausschließen.

[0092] Die in den nachfolgenden Ansprüchen etwa vorhandenen, entsprechenden Strukturen, Materialien, Schritte und Entsprechungen aller Mittel oder Step-plus-function-Elemente verstehen sich dahingehend, dass sie jede beliebige Struktur, jedes beliebige Material bzw. jeden beliebigen Schritt zur Durchführung der Funktion in Kombination mit anderen beanspruchten Elementen nach Maßgabe der konkreten Beanspruchung aufweisen.

[0093] Die Beschreibungen der verschiedenen Ausführungsformen der vorliegenden Erfindung wurden zum Zwecke der Veranschaulichung aufgeführt, sollen jedoch nicht gesamthaft stehen für bzw. begrenzt sein auf die offenbarten Ausführungsformen. Für Fachleute werden viele Abänderungen und Abweichungen ersichtlich sein, ohne von dem Umfang und dem Gedanken der beschriebenen Ausführungsformen abzuweichen. Die hierin verwendete Terminologie wurde gewählt, um die Grundgedanken der Ausführungsformen, die praktische Anwendung oder technische Verbesserung von auf dem Markt vorgefundenen Technologien bestmöglich zu erläutern

oder um es anderen Fachleuten zu ermöglichen, die hierin offenbarten Ausführungsformen zu verstehen.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

Diese Liste der vom Anmelder aufgeführten Dokumente wurde automatisiert erzeugt und ist ausschließlich zur besseren Information des Lesers aufgenommen. Die Liste ist nicht Bestandteil der deutschen Patent- bzw. Gebrauchsmusteranmeldung. Das DPMA übernimmt keinerlei Haftung für etwaige Fehler oder Auslassungen.

Zitierte Nicht-Patentliteratur

- Artikel „PEDS: A Parallel Error Detection Scheme for TCAM Devices” von Anat Bremler-Barr u. a., IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 18, NR. 5, OKTOBER 2010, 1665–75 [0004]

Patentansprüche

1. Verfahren, aufweisend:
Schreiben einer Vielzahl von Datenwörtern in einen ternären inhaltsadressierbaren Speicher, wobei jedes der Datenwörter Binärzeichen und Don't-Care-Zeichen aufweist;
gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern:
Berechnen einer ersten Prüfsumme für die Binärzeichen;
Speichern in einem entsprechenden Abschnitt eines Direktzugriffsspeichers von:
einer Kennung der Binärzeichen des entsprechenden der Datenwörter; und
der ersten Prüfsumme für das entsprechende der Datenwörter;
Abfragen des ternären inhaltsadressierbaren Speichers, in den die Vielzahl von Datenwörtern geschrieben wurde, mit einem Eingabewort;
beim Ergeben einer Übereinstimmung in der Abfrage:
Abrufen entsprechender Werte der Kennung der Binärzeichen und der ersten Prüfsumme aus dem Direktzugriffsspeicher;
Berechnen einer zweiten Prüfsumme für das Eingabewort unter Verwendung der Kennung der Binärzeichen; und
Ermitteln in Echtzeit, dass die Übereinstimmung falsch positiv ist, wenn die zweite und die erste Prüfsumme nicht gleich sind.

2. Verfahren nach Anspruch 1, wobei:
das Speichern ferner das Speichern eines Datenabschnitts eines entsprechenden der Datenwörter in dem entsprechenden Abschnitt des Direktzugriffsspeichers aufweist; und
das Abrufen ferner das Abrufen eines entsprechenden Werts des Datenabschnitts aus dem Direktzugriffsspeicher aufweist.

3. Verfahren nach Anspruch 2, wobei:
das Speichern ferner das Speichern eines Zeilenindex für das entsprechende der Datenwörter aufweist, wobei eine Zeile in dem ternären inhaltsadressierbaren Speicher ermittelt wird, in der das entsprechende der Datenwörter gespeichert ist; und
das Abrufen ferner das Abrufen eines entsprechenden Werts des Zeilenindex aufweist;
ferner aufweisend das Ersetzen des fehlerhaften Eintrags unter Verwendung des Zeilenindex.

4. Verfahren nach Anspruch 3, wobei:
in dem Speicherschritt die Kennung der Binärzeichen des entsprechenden der Datenwörter eine Länge eines sich am weitesten links befindlichen binären Abschnitts des entsprechenden der Datenwörter aufweist; und
das Berechnen der zweiten Prüfsumme das Berechnen der zweiten Prüfsumme für den sich am weitesten links befindlichen binären Abschnitt des Eingabeworts aufweist.

5. Verfahren nach Anspruch 3, wobei:
in dem Speicherschritt die Kennung der Binärzeichen des entsprechenden der Datenwörter eine Maske aufweist, die einen sich am weitesten links befindlichen binären Abschnitt des entsprechenden der Datenwörter kennzeichnet; und
das Berechnen der zweiten Prüfsumme das Berechnen der zweiten Prüfsumme für einen binären Abschnitt des Eingabeworts aufweist, der aus der Maske ermittelt wurde.

6. Verfahren nach Anspruch 1, wobei die in dem Berechnungsschritt berechnete Prüfsumme eine Paritätsprüfung aufweist.

7. Vorrichtung, aufweisend:
eine ternäre inhaltsadressierbare Speicheranordnung;
eine Direktzugriffsspeicher-Anordnung;
ein Ladeprogramm, das mit der ternären inhaltsadressierbaren Speicheranordnung und der Direktzugriffsspeicher-Anordnung gekoppelt ist;
eine Abfrageeinheit, die mit der ternären inhaltsadressierbaren Speicheranordnung gekoppelt ist; und
ein Übereinstimmungslogikmodul, das mit der Direktzugriffsspeicher-Anordnung gekoppelt ist;
wobei:
das Ladeprogramm so konfiguriert ist, dass es eine Vielzahl von Datenwörtern in die ternäre inhaltsadressierbare Speicheranordnung schreibt, wobei jedes der Datenwörter Binärzeichen und Don't-Care-Zeichen aufweist;
das Ladeprogramm so konfiguriert ist, dass es gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern Folgendes durchführt:
Berechnen einer ersten Prüfsumme für die Binärzeichen; und
Speichern in einem entsprechenden Abschnitt der Direktzugriffsspeicher-Anordnung von:
einer Kennung der Binärzeichen des entsprechenden der Datenwörter; und
der ersten Prüfsumme für das entsprechende der Datenwörter;
wobei die Abfrageeinheit so konfiguriert ist, dass sie den ternären inhaltsadressierbaren Speicher, in den die Vielzahl von Datenwörtern geschrieben wurde, mit einem Eingabewort abfragt; und
wobei das Übereinstimmungslogikmodul so konfiguriert ist, dass es beim Ergeben einer Übereinstimmung in der Abfrage Folgendes durchführt:
Abrufen entsprechender Werte der Kennung der Binärzeichen und der ersten Prüfsumme aus der Direktzugriffsspeicher-Anordnung;
Berechnen einer zweiten Prüfsumme für das Eingabewort unter Verwendung der Kennung der Binärzeichen; und

Ermitteln in Echtzeit, dass die Übereinstimmung falsch positiv ist, wenn die zweite und die erste Prüfsumme nicht gleich sind.

8. Vorrichtung nach Anspruch 7, wobei:
das Ladeprogramm ferner so konfiguriert ist, dass es gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern in dem entsprechenden Abschnitt des Direktzugriffsspeichers einen Datenabschnitt des entsprechenden der Datenwörter speichert; und
das Übereinstimmungslogikmodul darüber hinaus so konfiguriert ist, dass es beim Ergeben einer Übereinstimmung in der Abfrage einen entsprechenden Wert des Datenabschnitts aus dem Direktzugriffsspeicher abruft.

9. Vorrichtung nach Anspruch 8, wobei:
das Ladeprogramm darüber hinaus einen Zeilenindex für das entsprechende der Datenwörter in der Direktzugriffsspeicher-Anordnung speichert, wobei eine Zeile in der ternären inhaltsadressierbaren Speicheranordnung ermittelt wird, in der das entsprechende der Datenwörter gespeichert ist;
das Übereinstimmungslogikmodul einen entsprechenden Wert des Zeilenindex abruft; und
das Ladeprogramm ferner so konfiguriert ist, dass es den fehlerhaften Eintrag unter Verwendung des Zeilenindex ersetzt.

10. Vorrichtung nach Anspruch 8, wobei das Ladeprogramm eine Hardware-Zustandsmaschine aufweist.

11. Vorrichtung nach Anspruch 8, wobei die Abfrageeinheit und das Ladeprogramm identisch sind.

12. Vorrichtung nach Anspruch 8, wobei die Abfrageeinheit eine Hardware-Zustandsmaschine aufweist, die separat und getrennt von dem Ladeprogramm ist.

13. Vorrichtung nach Anspruch 8, wobei die Abfrageeinheit einen Huffman-Decoder aufweist.

14. Vorrichtung nach Anspruch 8, ferner aufweisend einen mit der ternären inhaltsadressierbaren Speicheranordnung und der Direktzugriffsspeicher-Anordnung gekoppelten Prioritätscodierer, wobei der Prioritätscodierer so konfiguriert, dass er, wenn das Abfragen mehrere vermeintliche Übereinstimmungen ergibt, selbige löst, um die Übereinstimmung zu erhalten.

15. Vorrichtung, aufweisend:
ein Mittel zum Schreiben einer Vielzahl von Datenwörtern in einen ternären inhaltsadressierbaren Speicher, wobei jedes der Datenwörter Binärzeichen und Don't-Care-Zeichen aufweist;
ein Mittel, um gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern Folgendes auszuführen:
Berechnen einer ersten Prüfsumme für die Binärzeichen;
Speichern in einem entsprechenden Abschnitt eines Direktzugriffsspeichers von:
einer Kennung der Binärzeichen des entsprechenden der Datenwörter; und
der ersten Prüfsumme für das entsprechende der Datenwörter;
ein Mittel zum Abfragen des ternären inhaltsadressierbaren Speichers, in den die Vielzahl von Datenwörtern geschrieben wurde, mit einem Eingabewort; und
ein Mittel, um beim Ergeben einer Übereinstimmung in der Abfrage Folgendes auszuführen:
Abrufen entsprechender Werte der Kennung der Binärzeichen und der ersten Prüfsumme aus dem Direktzugriffsspeicher;
Berechnen einer zweiten Prüfsumme für das Eingabewort unter Verwendung der Kennung der Binärzeichen; und
Ermitteln in Echtzeit, dass die Übereinstimmung falsch positiv ist, wenn die zweite und die erste Prüfsumme nicht gleich sind.

16. Nichtflüchtiges, durch einen Computer lesbares Medium, das auf einem Computer ausführbare Anweisungen enthält, die beim Ausführen durch einen Computer den Computer dazu veranlassen, das folgende Verfahren durchzuführen:
Schreiben einer Vielzahl von Datenwörtern in einen ternären inhaltsadressierbaren Speicher, wobei jedes der Datenwörter Binärzeichen und Don't-Care-Zeichen aufweist; und
gleichzeitig mit dem Schreiben für jedes aus der Vielzahl von Datenwörtern:
Berechnen einer ersten Prüfsumme für die Binärzeichen;
Speichern in einem entsprechenden Abschnitt eines Direktzugriffsspeichers von:
einer Kennung der Binärzeichen des entsprechenden der Datenwörter; und
der ersten Prüfsumme für das entsprechende der Datenwörter.

Es folgen 5 Seiten Zeichnungen

Anhängende Zeichnungen

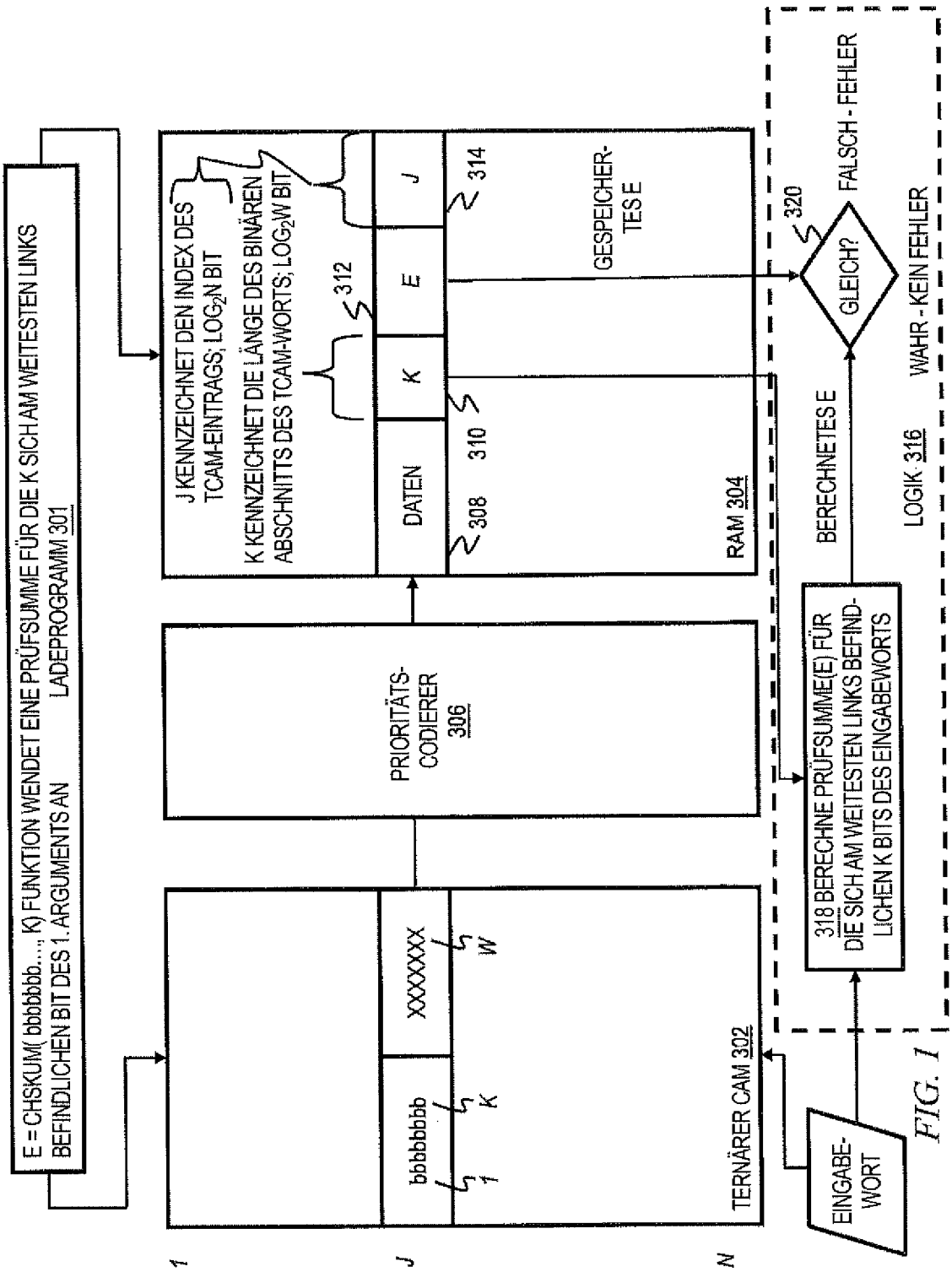


FIG. 1

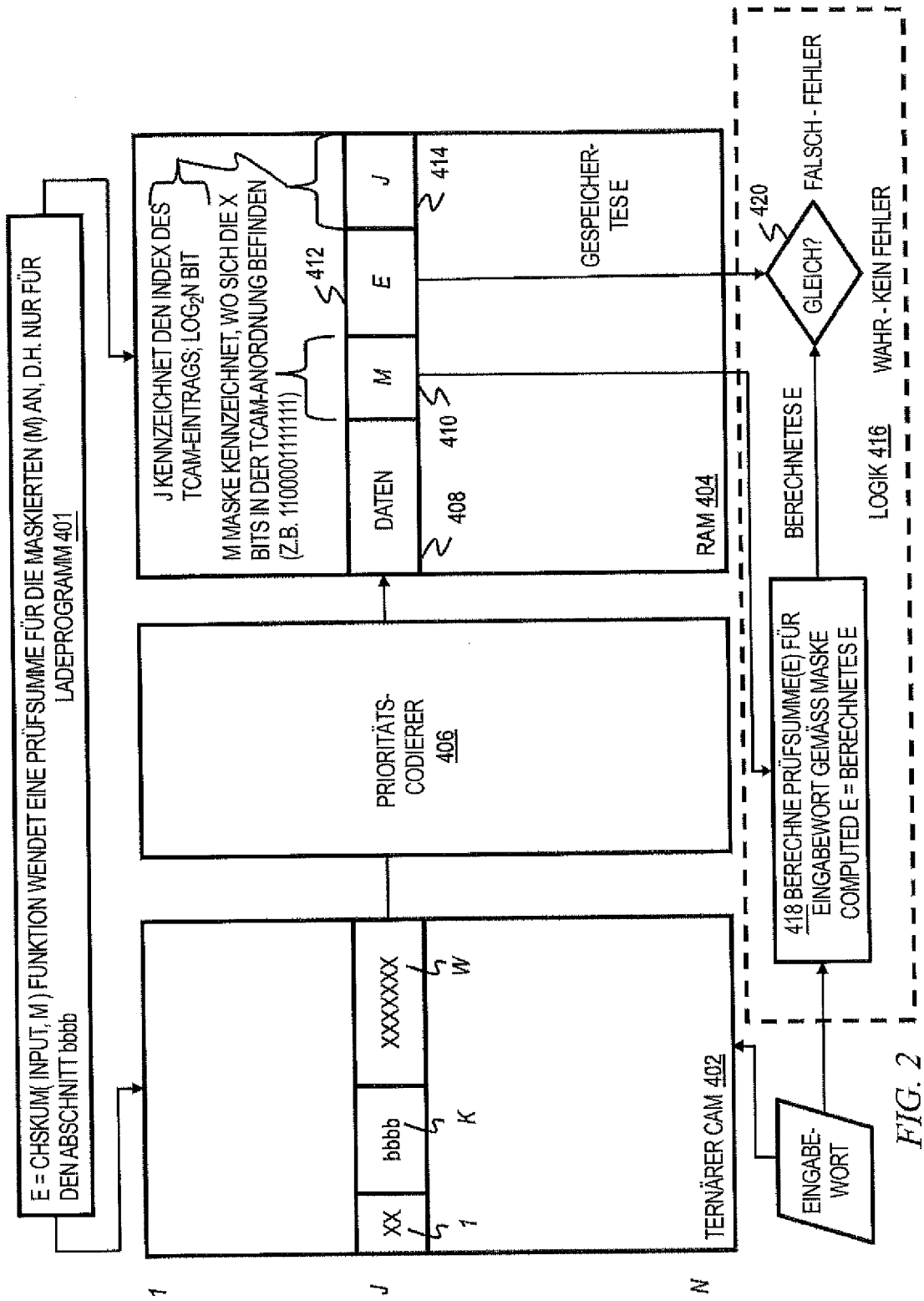

E = CHSKUM(INPUT, M) FUNKTION WENDET EINE PRÜFSUMME FÜR DIE MASKIERTEN (M) AN, D.H. NUR FÜR DEN ABSCHNITT bbbb
LADEPROGRAMM 401
J KENNZEICHNET DEN INDEX DES TCAM-EINTRAGS; LOG2N BIT
M MASKE KENNZEICHNET, WO SICH DIE X BITS IN DER TCAM-ANORDNUNG BEFINDEN (Z.B. 11000011111111)
412
DATEN
M
E
J
408
410
414
GESPEICHER-TES E
RAM 404
PRIORITÄTS-CODIERER 406
XX
bbbb
XXXXXXX
1
K
W
TERNÄRER CAM 402
1
J
N
EINGABE-WORT
418 BERECHNE PRÜFSUMME(E) FÜR EINGABEWORT GEMÄSS MASKE COMPUTED E = BERECHNETES E
BERECHNETES E
420
GLEICH?
FALSCH - FEHLER
WAHR - KEIN FEHLER
LOGIK 416

FIG. 2

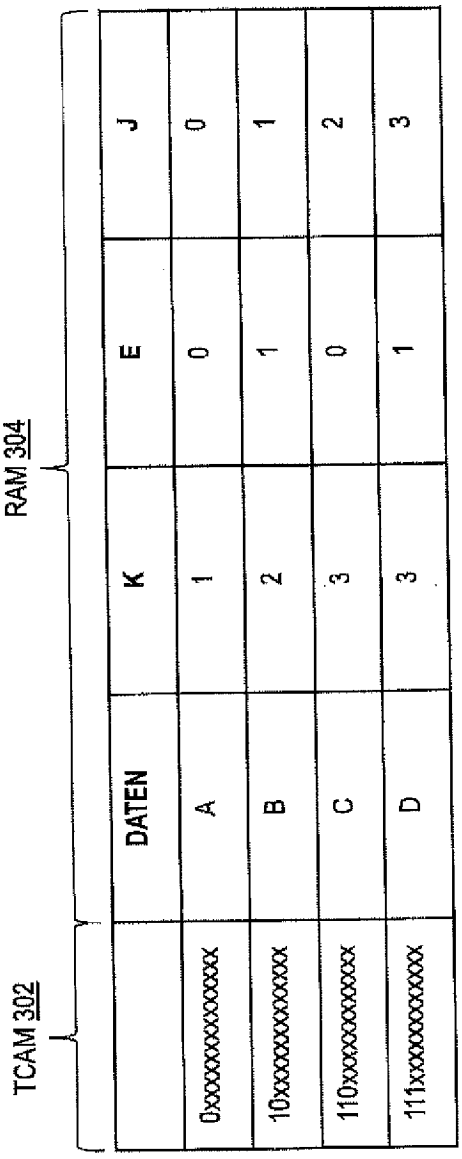

TCAM 302
RAM 304
DATEN
K
E
J
0xxxxxxxxxxxxx
A
1
0
0
10xxxxxxxxxxxx
B
2
1
1
110xxxxxxxxxxx
C
3
0
2
111xxxxxxxxxxx
D
3
1
3

FIG. 3

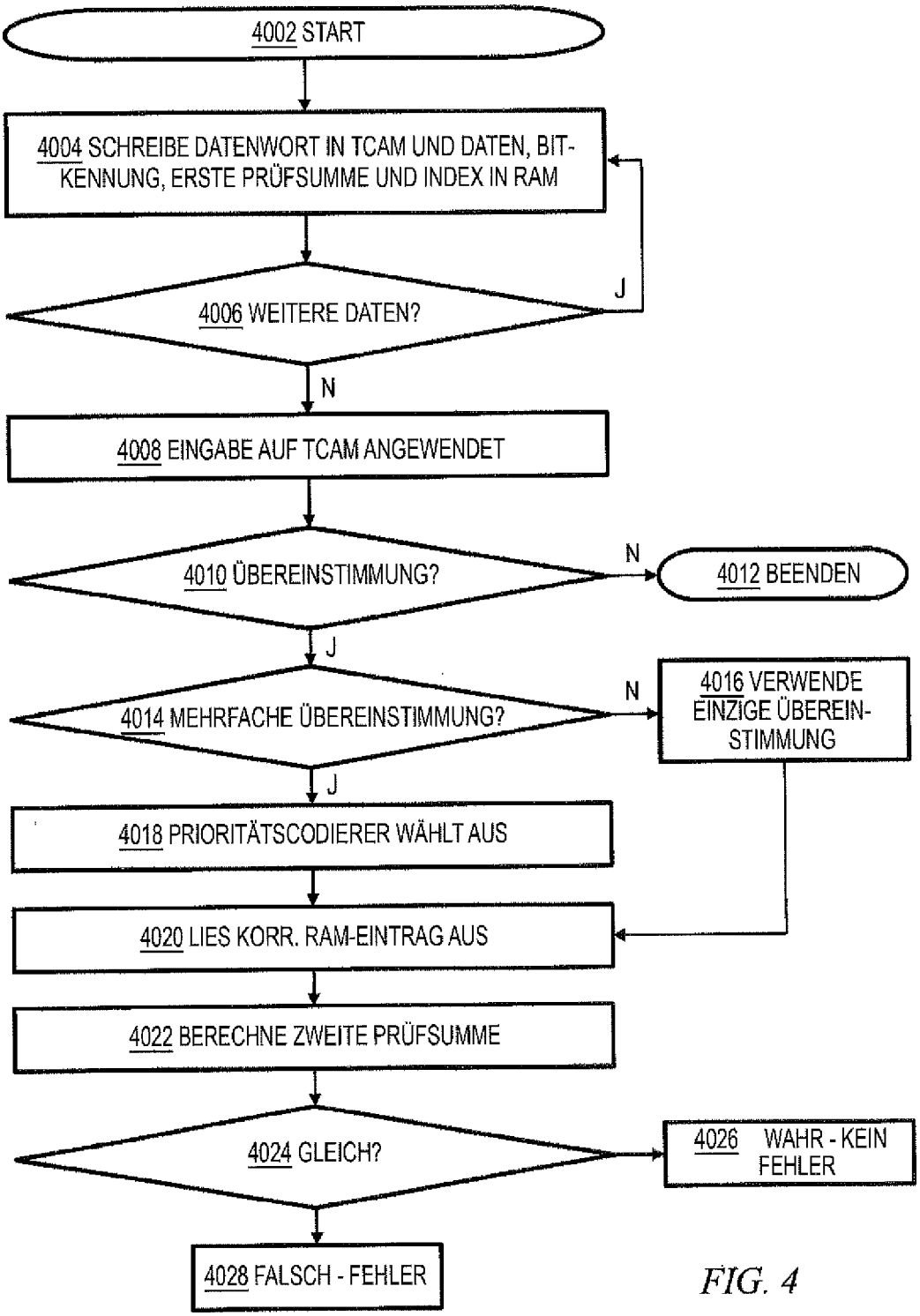

4002 START
4004 SCHREIBE DATENWORT IN TCAM UND DATEN, BIT-KENNUNG, ERSTE PRÜFSUMME UND INDEX IN RAM
4006 WEITERE DATEN?
J
N
4008 EINGABE AUF TCAM ANGEWENDET
4010 ÜBEREINSTIMMUNG?
N
4012 BEENDEN
J
4014 MEHRFACHE ÜBEREINSTIMMUNG?
N
4016 VERWENDE EINZIGE ÜBEREIN-STIMMUNG
J
4018 PRIORITÄTSCODIERER WÄHLT AUS
4020 LIES KORR. RAM-EINTRAG AUS
4022 BERECHNE ZWEITE PRÜFSUMME
4024 GLEICH?
4026 WAHR - KEIN FEHLER
4028 FALSCH - FEHLER

FIG. 4

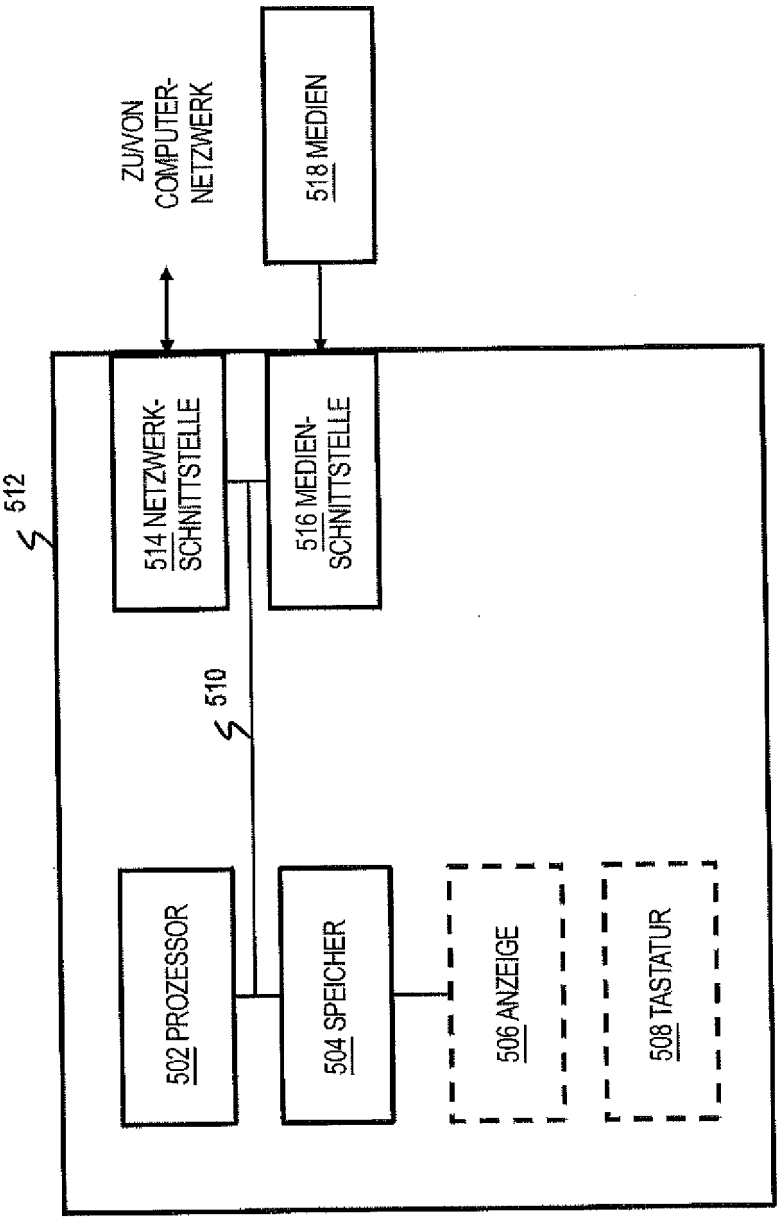

ZU/VON COMPUTER-NETZWERK
518 MEDIEN
514 NETZWERK-SCHNITTSTELLE
516 MEDIEN-SCHNITTSTELLE
512
510
502 PROZESSOR
504 SPEICHER
506 ANZEIGE
508 TASTATUR

FIG. 5