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

(10) **DE 601 17 066 T2** 2006.08.24

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 160 987 B1**
(21) Deutsches Aktenzeichen: **601 17 066.0**
(96) Europäisches Aktenzeichen: **01 108 612.1**
(96) Europäischer Anmeldetag: **05.04.2001**
(97) Erstveröffentlichung durch das EPA: **05.12.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **08.02.2006**
(47) Veröffentlichungstag im Patentblatt: **24.08.2006**

(51) Int Cl.[8]: ***H03M 13/00*** (2006.01)

(30) Unionspriorität:
562133 **01.05.2000** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Hewlett-Packard Development Co., L.P., Houston, Tex., US

(72) Erfinder:
Das Sharma, Debendra, Santa Clara, CA 95050, US; Wolf, Elizabeth S., Cupertino, CA 95014, US

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

(54) Bezeichnung: **Verfahren und Vorrichtung zum Überprüfen von fehlerkorrigierenden Codes**

Beschreibung

Technisches Gebiet

[0001] Das technische Gebiet ist ein Fehlerkorrekturcode für Speicher- oder Kommunikationssysteme.

Hintergrund

[0002] Kommunikations- und Speichersysteme sind Fehlern unterworfen, die eine Funktionsweise angeschlossene Systeme beeinflussen könnten. Ein typischer Fehler könnte resultieren, wenn ein bestimmter Speicherort einem oder mehreren α-Teilchen ausgesetzt wird. Eine derartige Strahlung könnte bewirken, dass ein an dem Speicherort gespeichertes Datenbit von einer „1" in eine „0" umgedreht wird.

[0003] Fehlerkorrekturcodes (ECC) werden verwendet, um eine Zuverlässigkeit und Zustandsintegrität von Kommunikations- und Speichersystemen zu verbessern. Fehlerkorrekturcodes sind bekannt, die einen Einzelfehler korrigieren und einen Doppelfehler erfassen, jedoch nicht korrigieren. Andere ECCs erfassen und korrigieren Mehrfachfehler. Für ECC-Anwendungen könnten Speicherarraychips so organisiert sein, dass in einem Chip erzeugte Fehler durch den ECC korrigiert werden können.

[0004] Die Korrektur von Einzelbitfehlern und die Erfassung von Doppelbitfehlern könnten durch die Verwendung von Prüfbits erzielt werden. Eine typische ECC-Implementierung hängt eine Anzahl von Prüfbits an jedes Datenwort an. Die angehängten Prüfbits werden durch ECC-Logikschaltungen verwendet, um Fehler innerhalb des Datenworts zu erfassen. Die einfachste und üblichste Form der Fehlersteuerung wird durch die Verwendung von Paritätsbits implementiert. Ein Einzelparitätsbit wird an ein Datenwort angehängt und als eine 0 oder eine 1 zugewiesen, um so die Anzahl von 1en in dem Datenwort in dem Fall gerader Paritätscodes gerade oder in dem Fall ungerader Paritätscodes ungerade zu machen.

[0005] Vor einer Übertragung des Datenworts in einem Computersystem wird der Wert des Paritätsbits an dem Quellenpunkt des Datenworts berechnet und an das Datenwort angehängt. Auf einen Empfang des übertragenen Datenworts hin berechnet eine Logik an dem Zielpunkt erneut das Paritätsbit und vergleicht dieses mit dem empfangenen, zuvor angehängten Paritätsbit. Wenn das neu berechnete und das empfangene Paritätsbit nicht gleich sind, wurde ein Bitfehler erfasst. Die Verwendung von Paritätscodes hat jedoch den Nachteil, dass man nicht in der Lage ist, Bitfehler zu korrigieren, und nicht in der Lage ist, gerade Anzahlen von Bitfehlern zu erfassen. Wenn sich z. B. ein Datenbit von einer 0 in eine 1 verändert und ein weiteres Datenbit von einer 1 in eine 0 verändert (ein Doppelbitfehler), verändert sich die Parität des Datenworts nicht und der Fehler bleibt unerfasst.

[0006] Durch das Anhängen zusätzlicher Paritätsbits an das Datenwort, wobei jedes einem Teilsatz von Datenbits innerhalb des Datenworts entspricht, könnte das Paritätsbitkonzept erweitert werden, um eine Erfassung von Mehrfachbitfehlern bereitzustellen oder den Ort von Einzel- oder Mehrfachbitfehlern zu bestimmen. Sobald ein Datenbitfehler erfasst wurde, könnten Logikschaltungen verwendet werden, um das fehlerhafte Bit zu korrigieren, was eine Einzelfehlerkorrektur bereitstellt.

[0007] Ein bekannter Fehlerkorrekturcode ist der Hamming-Code, der z. B. ein SEC-DED-Code sein könnte. Der ECC hängt eine Serie von Prüfbits an das Datenwort an, wenn dieses in einem Speicher gespeichert wird. Auf eine Leseoperation hin werden die wiedergewonnenen Prüfbits verglichen, um Prüfbits neu zu berechnen, um einen Einzelbitfehler zu erfassen und zu lokalisieren (d. h. korrigieren). Durch ein Hinzufügen mehrerer Prüfbits und ein geeignetes Überlappen der Teilsätze von Datenbits, die durch die Prüfbits dargestellt werden, könnten weitere Fehlerkorrekturcodes für eine Mehrfachfehlerkorrektur und -erfassung sorgen.

[0008] Ein Verifizieren der Korrektheit des Fehlerkorrekturcodes umfasst zwei Schritte: Verifizieren des zugrundeliegenden Algorithmus des Fehlerkorrekturcodes und Verifizieren der Implementierung des Fehlerkorrekturcodes an einer Hardware-Vorrichtung oder an einer Simulation der Hardware-Vorrichtung. Gegenwärtige Verfahren zum Verifizieren des Fehlerkorrekturcodes verbinden diese beiden Schritte nicht und liefern so keine vollständige Verifizierung. Ein Beispiel dieses Problems könnte in Bezug auf lineare Codes gezeigt werden. Lineare Codes sind unter Verwendung von Eigenschaften aufgebaut, die auf einer Galois-Feld-Arithmetik basieren. Der Beweis der Eigenschaften in einem Konzept könnte innerhalb des mathematischen Rahmens von Galois-Feldern durchgeführt werden. Basierend auf diesem Konzept werden eine Erzeugermatrix (als eine G-Matrix bekannt), eine Paritätsmatrix (als eine H-Matrix bekannt) und unterschiedliche Syndromvektoren, die verschiedenen Fehlerszenarien entsprechen, entweder von Hand oder durch ein Computerprogramm erzeugt. Ein Einzelfehlerkorrektur-Doppelfehlererfassungs(SEC-DED-) Code hätte eine H-Matrix, bei der keine zwei Spalten identisch sind, und bei der die Galois-Feld-Addition beliebiger zwei Spalten nicht gleich einer beliebigen Spalten in der H-Matrix ist. Der mathematische Beweis des Konzeptes erfasst keinen Fehler, der während der Erzeugung der G- und der H-Matrix und der Syndromvektoren eingeführt wird. Die G- und die H-Matrix und die Syndromvektoren werden dann in einer Sprache auf hoher Ebene verwendet, um den Fehlerkorrektur-

codeschaltungsaufbau zu erzeugen, der als eine Hardwarevorrichtung oder als eine Simulation der Hardwarevorrichtung implementiert sein könnte. Eine Verifizierung der Implementierung wird durch ein Prüfen, ob die Implementierung erwartete Ausgaben liefert, basierend auf der G- und der H-Matrix und den Syndromvektoren, vervollständigt.

[0009] Ein Problem bei diesem herkömmlichen Ansatz stammt von Fehlern, die während der Erzeugung der G- und der H-Matrix und der Syndromvektoren auftreten könnten. Derartige Fehler könnten unerfasst bleiben, da kein automatisiertes Werkzeug vorliegt, um direkt den Fehlerkorrekturcodeschaltungsaufbau aus den mathematischen Eigenschaften zu erzeugen.

[0010] R. G. Cooper u. a. offenbaren in „Diagnostic error forcing circuit", IBM Technical Disclosure Bulletin, Bd. 19, Nr. 5, Oktober 1976, eine Schaltungsunterstützung, um sicherzustellen, dass die Fehlerprüf- und Korrekturschaltungen eines Feldeffekttransistorspeichers betriebsfähig sind. Die Fehlererzwingungsschaltung besteht aus einem FET-Speicher zum Speichern von Wörtern der Datenbits eines Worts plus einer Gruppe von Prüfbits. Die Datenbits werden von einer zentralen Steuereinheit empfangen und durch einen ECC-Generator, der wirksam zum Erzeugen von Prüfbits für die Daten ist. Wenn das Datenwort von dem Speicher gelesen wird, werden die Daten- und Prüfbits durch einen ECC-Decodierer empfangen, der Prüfbits neu aus den Datenbits erzeugt und dieselben mit den ursprünglichen Prüfbits aus dem Speicher prüft und ferner Datenbits, in denen ein Fehler gefunden wurde, korrigiert. Die korrigierten Datenbits werden dann an die Steuereinheit gesendet. Die Steuereinheiten können verwendet werden, um Fehler einzuführen, um zu bestimmen, ob dieselben korrigiert sind. Für diese Einführung wird ein Diagnoseregister angewendet.

[0011] Die US 5,502,732 A offenbart ein Verfahren zum Testen einer ECC-Logik. Insbesondere wird die in einem Computerspeichersystem enthaltene Testlogik derart geprüft, dass mögliche Fehler vor dem Beginn der Verarbeitungsoperation bestimmt und für die Systemsoftware verfügbar gemacht werden können. Eine CPU vergleicht die Daten, die in den Speicher geschrieben werden, mit den Daten, die rückgelesen werden. Da bekannt ist, dass aufgrund des induzierten Fehlers ein Fehler auftritt, werden identisches Daten verifizieren, dass die ECC-Korrekturlogik ordnungsgemäß arbeitet. Deshalb ist ein Multiplexer in dem Datenschreibpfad vorgesehen, der den konstanten Satz identischer Bits für die tatsächlichen Daten, die durch die CPU erzeugt werden, ersetzt und so wird ein Fehler eingeführt. ECC-Bits werden dann basierend auf den tatsächlichen erzeugten Testdaten, und nicht den eingeführten identischen Bits erzeugt. Die ersetzten Datenbits und die erzeugten ECC-Bits werden dann in dem Speicher gespeichert. Dieses Verfahren offenbart ferner, dass Doppelbitfehler injiziert werden können.

[0012] Die Aufgabe der vorliegenden Erfindung besteht darin, eine Vorrichtung und ein Verfahren zum Verifizieren einer Korrektheit eines Fehlerkorrekturcodealgorithmus und einer Korrektheit einer Fehlerkorrekturcodeimplementierung in einem realen Übertragungsszenario bereitzustellen, in dem ein codiertes Signal, das Datenbits und Prüfbits aufweist, durch Fehler verfälscht werden könnte.

[0013] Diese Aufgabe wird durch eine Vorrichtung zum Verifizieren eines Fehlerkorrekturcodes gemäß Anspruch 1 oder durch das Verfahren zum Verifizieren des Fehlerkorrekturcodes gemäß Anspruch 3 gelöst.

[0014] Ein Verfahren und eine Vorrichtung verifizieren die Korrektheit des Fehlerkorrekturcodealgorithmus und die Korrektheit der Fehlerkorrekturcodeimplementierung. Ein Fehlerinjektionsmodul wird verwendet, um zufällige Fehler in eine ECC-Schaltung zwischen einem Codierer und einem Decodierer zu injizieren. Der Codierer codiert Datenbits mit Prüfbits, um ein codiertes Signal zu erzeugen. Ein Decodierer decodiert das codierte Signal nach einer Modifizierung durch das Fehlerinjektionsmodul. Das Fehlerinjektionsmodul könnte Null-Fehler injizieren. Die Fehlerinjektionsschaltung könnte Einzelfehler oder Mehrfachfehler injizieren. Die Ausgabe des Decodierers ist ein Null-Fehler-Signal, ein Einzelfehlersignal, ein Mehrfehlersignal und ein Fehlerortssignal. Weitere Signale sind ebenso möglich. Die Ausgabe des Decodierers wird unter Verwendung eines Überwachungsmoduls mit erwarteten Werten für jedes Signal verglichen. Mögliche Unterschiede zwischen den Ausgangssignalen und den erwarteten Werten zeigen einen Fehler in dem ECC oder in der Schaltung, die zur Implementierung des ECC verwendet wird, an.

[0015] Der ECC könnte durch ein Implementieren der Verifizierungsvorrichtung in einer tatsächlichen Hardwarevorrichtung verifiziert werden. Bei diesem Ausführungsbeispiel könnten das Fehlerinjektionsmodul und das Überwachungsmodul auf einem gleichen Chip wie der Decodierer und der Codierer angeordnet sein. Alternativ könnten das Fehlerinjektionsmodul und das Überwachungsmodul auf Chips separat von dem Decodierer und dem Codierer angeordnet sein. Die ECC-Verifizierungsvorrichtung könnte auch als eine Simulation der tatsächlichen Hardwarevorrichtung oder in einem formalen Verifizierungsmodell der tatsächlichen Hardware implementiert sein.

[0016] Die detaillierte Beschreibung nimmt Bezug auf die folgenden Figuren, in denen gleiche Bezugszeichen sich auf gleiche Gegenstände beziehen, und

in denen:

[0017] Fig. 1A Blockdiagramme einer Fehlerkorrekturschaltung und Fig. 1B sind;

[0018] Fig. 2 ein Blockdiagramm einer Vorrichtung zum Verifizieren eines Fehlerkorrekturcodes und einer -schaltung ist; und

[0019] Fig. 3A Flussdiagramme sind, die Prozesse zeigen, die und Fig. 3B auf der Vorrichtung aus Fig. 2 ausgeführt werden.

Detaillierte Beschreibung

[0020] Fehlerkorrekturcode- (ECC-) Schaltungen werden häufig in Halbleiterspeicherentwürfen verwendet, um Einzelbitfehler zu korrigieren und Doppelbitfehler zu erfassen. Ein üblicher ECC-Code ist der SEC-DED- (Einzelfehlerkorrektur-Doppelfehlererfassungs-) Code. Andere ECC-Codes sind in der Lage, mehr als zwei Fehler zu erfassen und mehr als Einzelfehler zu korrigieren.

[0021] Die ECC-Schaltungen führen ihre Fehlerprüffunktionen durch ein Erzeugen einer Anzahl von Prüfbits für eine spezifische Anzahl von Datenbits und ein folgendes Schreiben der Prüfbits mit den Datenbits in einen Speicher durch. Die Prüfbits werden dann während nachfolgenden Lese-Schreib-Zyklen oder anderen Speicherzugriffen verwendet, um die korrekten Werte für die Datenbits zu verifizieren. Die Anzahl von Prüfbits, die zur Implementierung des ECCs erforderlich ist, hängt von der Anzahl gerade gelesener Datenbits ab. Wie in Tabelle 1 gezeigt ist, nimmt mit zunehmender Anzahl gerade gelesener Datenbits auch die Anzahl erforderlicher ECC-Bits zu.

Tabelle 1

Datenbits	ECC-Bits
16 - 31	6
32 - 63	7
64 - 127	8
128 - 255	9

[0022] Eine Hardware zur Implementierung von ECC-Prüfbits unter Verwendung gegenwärtiger Systeme ist in Fig. 1A dargestellt. Eine Fehlerkorrekturcodeschaltung **10** umfasst eine Speicherzeile **11**, die in Fig. 1A als 30 Datenbits umfassend gezeigt ist. Der Speicherzeile **11** zugeordnet ist eine ECC-Zelle **12**. Bezug nehmend auf die Tabelle 1 oben müssen sechs ECC-Bits in der ECC-Zelle **12** gespeichert werden, um eine Einzelbitfehlerkorrektur und eine Doppelbitfehlererfassung in der Speicherzeile **11** zu erzielen. Ein ECC-Block **13** wird verwendet, um die ECC-Bits zu erzeugen und die Fehlerkorrektur-/Erfassungscodeoperationen, einschließlich eines Prüfens der Datenbits in der Speicherzeile **11** während Lese- und Schreiboperationen durchzuführen.

[0023] Fig. 1B ist ein Blockdiagramm eines Abschnitts des ECC-Blocks **13**, der Prüfbits und Syndrombits erzeugt. Wie in der Technik bekannt ist, sind Syndrombits das Produkt eines Vergleichs der ECC-Bits, die ursprünglich mit den Daten während einer Datenspeicheroperation in dem Speicher gespeichert wurden, und eines neuen Satzes von ECC-Bits, die basierend auf den Daten erzeugt werden, die aus dem Speicher geholt wurden, wie während der Ausführung eines Lesebefehls oder eines Speicherzugriffs in einem Computersystem auftreten würde. Dies bedeutet, dass ein Syndrombit einfach XOR eines entsprechenden empfangenen ECC-Bits mit einem neu erzeugten ECC-Bit ist. Wenn die Kombination des wiedergewonnenen und des neu erzeugten ECC-Bits Nicht-Null-Syndrombits erzeugt, wurde ein Fehler innerhalb der wiedergewonnenen Daten erfasst.

[0024] In Fig. 1B umfasst eine Schaltung **20** einen XOR-Baum **21** und ein Bitweise-XOR-Modul **22**. Bei einer Schreiboperation werden die ECC-Bits gleichzeitig durch ein Verarbeiten der Datenbits z. B. unter Verwendung einer Paritätsprüfmatrix erzeugt. Eine derartige Erzeugung von ECC-Bits ist in der Technik bekannt. Bei einer Leseoperation werden die Syndrombits gleichzeitig gemäß Standarddecodierungsverfahren aus den gelesenen Datenbits erzeugt. Der gleiche XOR-Baum **21** könnte für sowohl die ECC-Bits als auch die Syndrombits verwendet werden, wie in Fig. 1B gezeigt ist.

[0025] Gegenwärtige Ansätze zum Erzeugen des ECC und eines zugeordneten Schaltungsaufbaus (Hardware oder Hardwaresimulation) berücksichtigen mögliche Fehler in dem zugrundeliegenden Algorithmus nicht. So könnte eine Anwendung des ECC bei einer Implementierung unter Umständen nicht sicherstellen, dass alle Fehler korrekt korrigiert oder erfasst werden. Dies könnte besonders zutreffen, wenn der ECC eine Kombination linearer Codes und arithmetischer Codes oder bestimmter anderer kundenspezifischer Codes ist, die keinen Standardprozeduren folgen.

[0026] Um dieses Problem zu überwinden, unterwerfen eine Vorrichtung und ein Verfahren die Implementierung der ECC-Schaltung den verschiedenen Fehlern, die die ECC-Schaltung vermutlich korrigiert/erfasst. Die Vorrichtung und das Verfahren verifizieren das ECC-Konzept, den Algorithmus und die Implementierung gleichzeitig.

[0027] Fig. 2 ist ein Blockdiagramm, das die Vorrichtung und das Verfahren zum Verifizieren eines

ECC darstellt. In Fig. 2 umfasst eine Vorrichtung **100** einen Sender **110** mit einem Codierer **115**. Der Sender **110** und der Codierer **115** sind durch eine Fehlerinjektionsschaltung **120** mit einem Empfänger **130** mit einem Decodierer **135** gekoppelt. Ebenso mit dem Codierer **115** und dem Decodierer **135** gekoppelt ist ein Überwachungsmodul **140**. Wie oben angemerkt wurde, könnte die Vorrichtung **100** als eine tatsächliche Hardwarevorrichtung implementiert sein oder könnte als eine Simulation einer Hardwarevorrichtung implementiert sein, unter Verwendung einer Hardwarebeschreibungssprache, wie z. B. VHDL oder Verilog, die beide in der Technik bekannt sind.

[0028] Das Verfahren und die Vorrichtung **100** arbeiten, um den ECC durch das Koppeln des Codierers **115** und des Decodierers **135** und ein darauffolgendes Injizieren möglicher Fehler vollständig zu testen. In Betrieb werden Daten in den Codierer **115** eingegeben. Der Codierer **115** codiert die Daten, um einen Ausgangsvektor **112** zu erzeugen. Bei dem in Fig. 2 dargestellten Beispiel sind die eingegebenen Daten **64** Bits breit. Bezug nehmend auf die Tabelle 1 codiert der Codierer zusätzliche acht Bits zu den eingegebenen Daten, derart, dass der Ausgangsvektor **112** 72 Bits breit ist. Der Ausgangsvektor **112** wird durch eine Fehlerinjektionsschaltung **120** gesendet, die Fehler einführt, die der ECC korrigieren oder erfassen kann. Die Fehlerinjektionsschaltung **120** testet außerdem den ECC und dessen Implementierung durch Nicht-Einführung von Fehlern (ein Null-Fehler-Fall). Die modifizierten Daten werden dann direkt dem Decodierer **135** zugeführt.

[0029] Der Decodierer **135** decodiert die modifizierten Daten und erzeugt mehrere Ausgangssignale. Die Ausgangssignale könnten ein Daten-Aus-Signal, ein Kein-Fehler-Signal, ein Einzelfehlersignal und ein Doppel- (Mehrfach-) Fehlersignal umfassen. Der Decodierer **135** könnte außerdem ein Fehler_Ort-Signal bereitstellen, das einen Ort eines Bits in einem Fehler anzeigt. Das Fehler_Ort-Signal könnte dem oben erwähnten Syndrom ähneln. Andere Ausgangssignale könnten ebenso bereitgestellt werden. Diese Ausgangssignale werden an das Überwachungsmodul **140** geliefert. Das Überwachungsmodul **140** bestimmt, ob die bereitgestellten Ausgangssignale wie erwartet sind. Wenn die Ausgangssignale nicht wie erwartet sind, könnte ein Problem bei dem ECC oder der ECC-Schaltung vorliegen. Für das Beispiel eines SEC-DED-ECC sind, wenn keine Fehler injiziert wurden, die erwarteten Ergebnisse: ein Ausgangssignal Kein_Fehler ist gleich 1 gesetzt; Ausgangssignale Einzel_Fehler und Mehrfach_Fehler sind gleich 0 gesetzt und ein 64-Bit-Signal Daten_Aus = Daten_Ein.

[0030] Die Fehlerinjektionsschaltung **120** injiziert dann Einzelfehler, nämlich einen für jedes der 72 Bits. Wieder werden die Ausgangssignale aus dem Decodierer **135** an das Überwachungsmodul **140** geliefert, das bestimmt, ob die bereitgestellten Ausgangssignale mit den erwarteten Ausgangssignalen übereinstimmen. Für das Beispiel eines SEC-DED-ECC sind die erwarteten Ergebnisse: Daten_Aus = Daten_Ein (zeigt an, dass der Fehler korrigiert wurde), Einzel_Fehler = 1 und Kein_Fehler = Mehrfachfehler = 0. Ein Fehler_Ort-Signal könnte ebenso ausgegeben werden.

[0031] Die Vorrichtung **100** prüft außerdem auf eine ordnungsgemäße Operation des ECC bei Vorliegen von Mehrfachfehlern. Zur Prüfung nach Doppelfehlern injiziert die Fehlerinjektionsschaltung **120** Doppelfehler (es gibt bei diesem Beispiel 2.556 Möglichkeiten). Das erwartete Ergebnis ist Kein_Fehler = Einzel_Fehler = Null; Mehrfach_Fehler = 1. Da der ECC bei diesem Beispiel ein SEC-DED ist, vergleicht das Überwachungsmodul nicht Daten_Aus = Daten Ein.

[0032] Die Vorrichtung **100** könnte abhängig von der ECC-Verifizierungsmethodik in verschiedenen Weisen implementiert sein. Die Fehlerinjektionsschaltung **120** könnte als XOR der Daten_Aus-Bits mit einem binären Fehlervektor mit der gleichen Breite wie die Daten_Aus-Bits implementiert sein. Der binäre Fehlervektor könnte in einer Simulationsumgebung zufällig für alle unterschiedlichen Fehlertypen erzeugt werden. Diese Fehlertypen umfassen z. B. keinen Fehler, Einzelfehler und Doppelfehler. Der binäre Fehlervektor könnte auch handkopiert und an die Fehlerinjektionsschaltung **120** geliefert werden. Ähnlich könnte ein formales Verifizierungsmodul alle Fehlerszenarien umfassen.

[0033] Zur Verifizierung einer ordnungsgemäßen Operation des ECC, einschließlich des zugrunde liegenden Algorithmus und der ECC-Schaltung, könnte die Vorrichtung **100** mit jedem beliebigen Typ Speicher in einem Computersystem verwendet werden. Die ECC-Schaltung **100** könnte z. B. mit einem Cache-Speicher und mit einem Hauptspeicher verwendet werden. Die Vorrichtung könnte mit einem beliebigen ECC verwendet werden. Während die vorangegangene Beschreibung eine Operation der Vorrichtung **100** mit einem SEC-DED beschrieben hat, würde ein Fachmann auf diesem Gebiet verstehen, dass das Verfahren und die Vorrichtung **100** mit ECCs verwendet werden könnten, die in der Lage sind, Mehrfachfehler zu erfassen und zu korrigieren (z. B. DEC-TED-Codes).

[0034] Die Vorrichtung **100** könnte bei einer Dual-In-Line-Speichermodul- (DIMM-) Karte gemeinsam mit einem oder mehreren Speicherchips enthalten sein und könnte z. B. innerhalb eines ASIC-Chips implementiert sein. Der ASIC-Chip würde dazu dienen, einen Datenbus (nicht gezeigt) des Computersystems mit dem Speicherchip zu verbinden. Daten, die während der Ausführung einer Schreiboperation

von dem Datenbus zu den Speicherchips laufen, würden vor einer Speicherung in den Speicherchips durch die Vorrichtung **100** laufen. Ähnlich würden auch Daten, die von den Speicherchips zu dem Datenbus laufen, durch die Vorrichtung **100** laufen. So arbeitet ein Fehlererfassungs- und Korrekturmechanismus an den Daten, wenn die Daten gerade durch das Computersystem in den Speicherchips gespeichert werden.

[0035] Bei dem in Fig. 2 gezeigten Ausführungsbeispiel weist der Datenbus, der den Sender **110** und den Empfänger **130** koppelt, eine ausreichende Bandbreite auf, um alle 72 Bits bei einem Taktzyklus zu tragen. Die Vorrichtung **100** jedoch könnte auch mit Systembussen verwendet werden, die kleinere Bandbreiten aufweisen. In diesem Fall könnten mehrere Zyklen benötigt werden, um alle Daten an Prüfbits zu übertragen.

[0036] Die Fig. 3A und Fig. 3B stellen Verfahren dar, die unter Verwendung der in Fig. 2 gezeigten Vorrichtung **100** ausgeführt werden könnten. Fig. 3A stellt ein Verfahren **200** dar, wenn ein Kein-Fehler-Signal injiziert wird. Das Verfahren beginnt bei einem Block **210**. Bei einem Block **220** codiert der Codierer **115** eine Transaktion mit einem ECC. Die Transaktion wird dann in einer Fehlerinjektionsschaltung **120** verarbeitet und ein Kein-Fehler-Signal wird bei einem Block **230** injiziert.

[0037] Bei einem Block **240** wird die Transaktion unter Verwendung des ECC decodiert. Bei einem Block **250** überwacht das Überwachermodul **140** die decodierte Transaktion. Bei einem Block **260** vergleicht das Überwachermodul die decodierte Transaktion mit den erwarteten Ergebnissen. In diesem Fall ist, wenn der ECC-Code und die Schaltung korrekt arbeiten, das 64-Bit-Signal Daten_Aus = Daten_Ein, das Ausgangsignal Kein_Fehler ist gleich 1 gesetzt und die Ausgangssignale Einzel_Fehler und Mehrfach_Fehler sind gleich 0. Wenn kein Fehler bei der Operation des ECC oder der ECC-Schaltung bemerkt wird, bewegt sich das Verfahren zu einem Block **280** und endet. Andernfalls bewegt sich das Verfahren zu einem Block **270** und ein Fehler wird deklariert. Das Verfahren bewegt sich dann zu Block **280** und endet.

[0038] Fig. 3B stellt ein Verfahren **300** dar, bei dem die Fehlerinjektionsschaltung **120** einen Einzelbitfehler einfügt. Das Verfahren **300** ähnelt dem Verfahren **200**, mit der Ausnahme, dass die erwarteten Ausgangssignale Daten_Aus = Daten_Ein (wobei der Einzelbitfehler durch den ECC korrigiert wird), Einzel_Fehler = 1 und Mehrfach_Fehler und Kein_Fehler gleich 0 sind.

Patentansprüche

1. Eine Vorrichtung (**100**) zum Verifizieren eines Fehlerkorrekturcodes ECC, der in einer ECC-Schaltung wirkt, mit folgenden Merkmalen:
einem Codierer (**115**), der ein Dateneingangssignal empfängt und die Datenbits desselben codiert, um ein codiertes Signal zu erzeugen, wobei das codierte Signal die Datenbits und Prüfbits aufweist;
einem Fehlerinjektionsmodul (**120**), das mit dem Codierer (**115**) gekoppelt ist und in der Lage ist, ein Fehlersignal in das codierte Signal zu injizieren, um ein modifiziertes Signal zu erzeugen, wobei das Fehlersignal kein Fehler, ein Einzelfehler oder ein Mehrfachfehler ist;
einem Decodierer (**135**), der mit dem Fehlerinjektionsmodul (**120**) gekoppelt ist, der das modifizierte Signal decodiert, um eine Mehrzahl von Ausgangssignalen zu erzeugen, wobei die Mehrzahl von Ausgangssignalen ein Datenaisgangssignal, ein Kein-Fehler-Signal, ein Einzelfehlersignal und ein Mehrfachfehlersignal aufweist; und
einem Überwachermodul (**140**), das mit dem Fehlerinjektionsmodul (**120**), dem Codierer (**115**) und dem Decodierer (**135**) gekoppelt ist, wobei das Überwachermodul (**140**) das Dateneingangssignal und die Mehrzahl von Ausgangssignalen empfängt, wobei das Überwachermodul (**140**) bestimmt, ob die Mehrzahl von Ausgangssignalen, die von dem Decodierer empfangen werden, mit erwarteten Ausgangssignalen für die Mehrzahl von Ausgangssignalen übereinstimmt, und das Überwachermodul (**140**) in der Lage ist, Fehler in dem ECC und der ECC-Schaltung auf der Basis der Bestimmung zu deklarieren,
wobei das Fehlerinjektionsmodul (**120**) angepasst ist, um für unterschiedliche Tests des ECC und der ECC-Schaltung keine Fehler, Einzelfehler oder Mehrfachfehler in das codierte Signal zu injizieren, wobei die erwarteten Ausgangssignale folgende sind:
– wenn keine Fehler durch das Fehlerinjektionsmodul (**120**) injiziert werden, ist das Kein-Fehler-Signal 1, das Einzelfehlersignal und das Mehrfachfehlersignal sind 0 und das Dateneingangssignal ist gleich dem Datenausgangssignal;
– wenn Einzelfehler durch das Fehlerinjektionsmodul (**120**) injiziert werden, ist das Einzelfehlersignal 1, das Kein-Fehler-Signal und das Mehrfachfehlersignal sind 0 und das Dateneingangssignal ist gleich dem Datenausgangssignal; und
– wenn Mehrfachfehler durch das Fehlerinjektionsmodul (**120**) injiziert werden, ist das Mehrfachfehlersignal 1 und das Kein-Fehler-Signal und das Einzelfehlersignal sind 0, wobei das Dateneingangssignal und das Datenausgangssignal nicht verglichen werden.

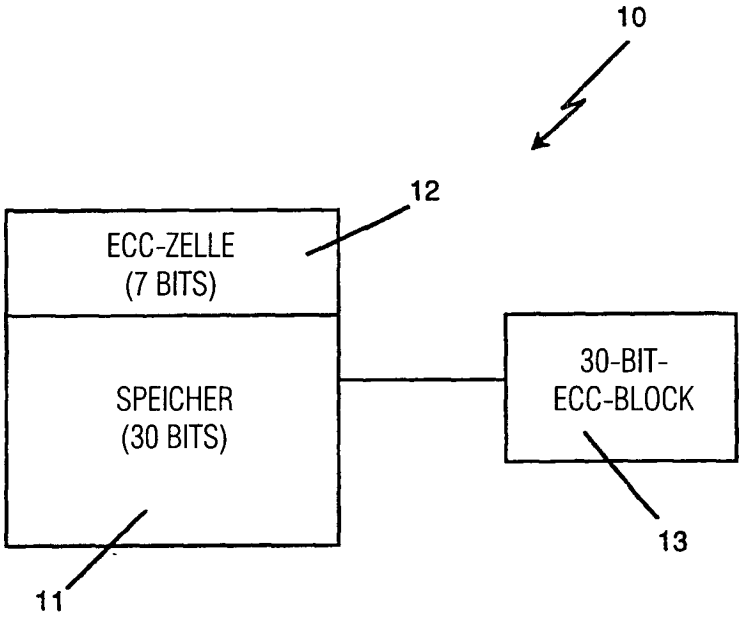
2. Die Vorrichtung gemäß Anspruch 1, wobei die Vorrichtung als eine tatsächliche Hardware-Vorrichtung ausgeführt ist.

3. Ein Verfahren zum Verifizieren eines Fehlerkorrekturcodes ECC, der an einer ECC-Schaltung wirkt, mit folgenden Schritten:
Bereitstellen eines Dateneingangssignals an einen Datencodierer (**115**);
Erzeugen (**220**, **320**) eines ECC-codierten Datensignals, das Datenbits und Prüfbits aufweist;
Bereitstellen (**230**, **330**) eines Fehlerinjektionssignals, wobei das Fehlerinjektionssignal das codierte Datensignal modifiziert, um ein modifiziertes Datensignal zu erzeugen, wobei das Fehlerinjektionssignal kein Fehler, ein Einzelfehler oder ein Mehrfachfehler ist;
Decodieren (**240**, **340**) des modifizierten Datensignals, um eine Mehrzahl von Ausgangssignalen zu erzeugen, wobei die Mehrzahl von Ausgangssignalen ein Datenausgangssignal, ein Kein-Fehler-Signal, ein Einzelfehlersignal und ein Mehrfachfehlersignal aufweist;
Bestimmen, ob die Mehrzahl von Ausgangssignalen, die von dem Decodierer empfangen werden, mit erwarteten Ausgangssignalen für die Mehrzahl von Ausgangssignalen übereinstimmt; und
wenn die Mehrzahl von Ausgangssignalen und die entsprechenden erwarteten Signale nicht übereinstimmen, Deklarieren (**270**, **370**) eines Fehlers in dem ECC oder der ECC-Schaltung,
wobei für unterschiedliche Tests des ECC und der ECC-Schaltung keine Fehler, Einzelfehler oder Mehrfachfehler in das codierte Signal injiziert werden, wobei die erwarteten Signale folgende sind:
– wenn keine Fehler injiziert werden, ist das Kein-Fehler-Signal 1, das Einzelfehlersignal und das Mehrfachfehlersignal sind 0 und das Dateneingangssignal ist gleich dem Datenausgangssignal;
– wenn Einzelfehler injiziert werden, ist das Einzelfehlersignal 1, das Kein-Fehler-Signal und das Mehrfachfehlersignal sind 0 und das Dateneingangssignal ist gleich dem Datenausgangssignal; und
– wenn Mehrfachfehler injiziert werden, ist das Mehrfachfehlersignal 1 und das Kein-Fehler-Signal und das Einzelfehlersignal sind 0, wobei das Dateneingangssignal und das Datenausgangssignal nicht verglichen werden.

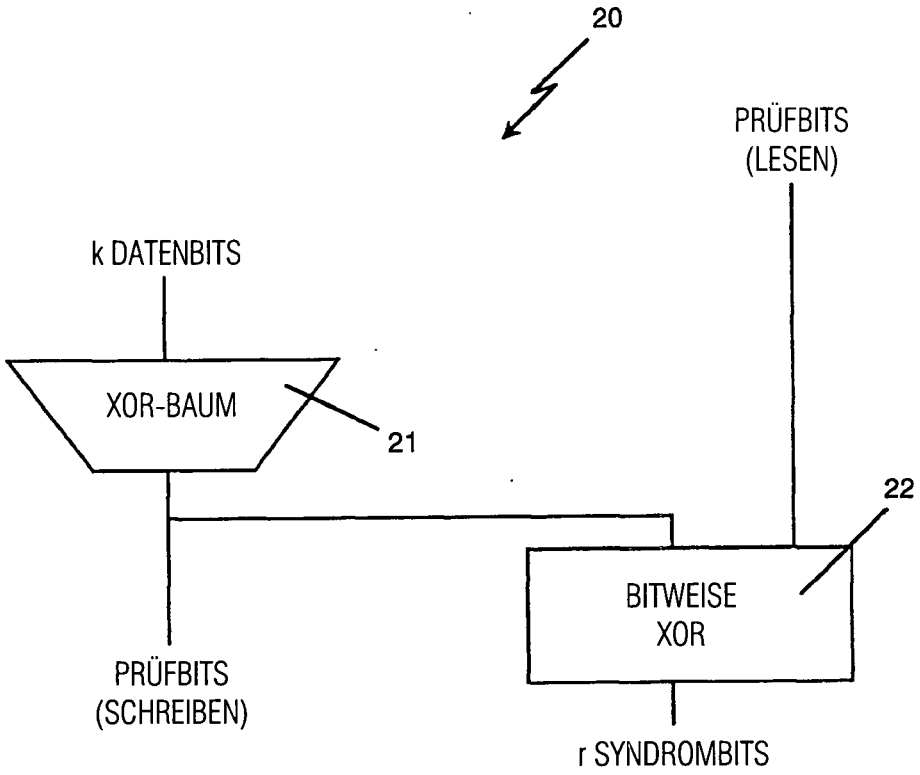
4. Das Verfahren gemäß Anspruch 3, bei dem die Verifizierung eine Simulation ist.

Es folgen 5 Blatt Zeichnungen

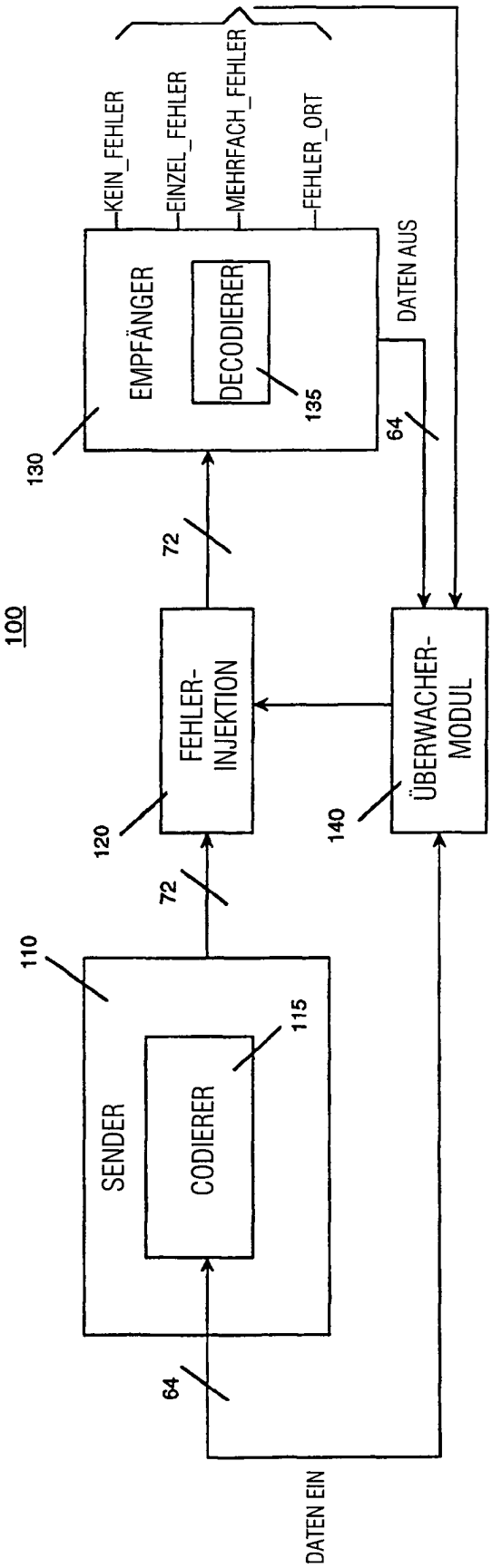
Anhängende Zeichnungen



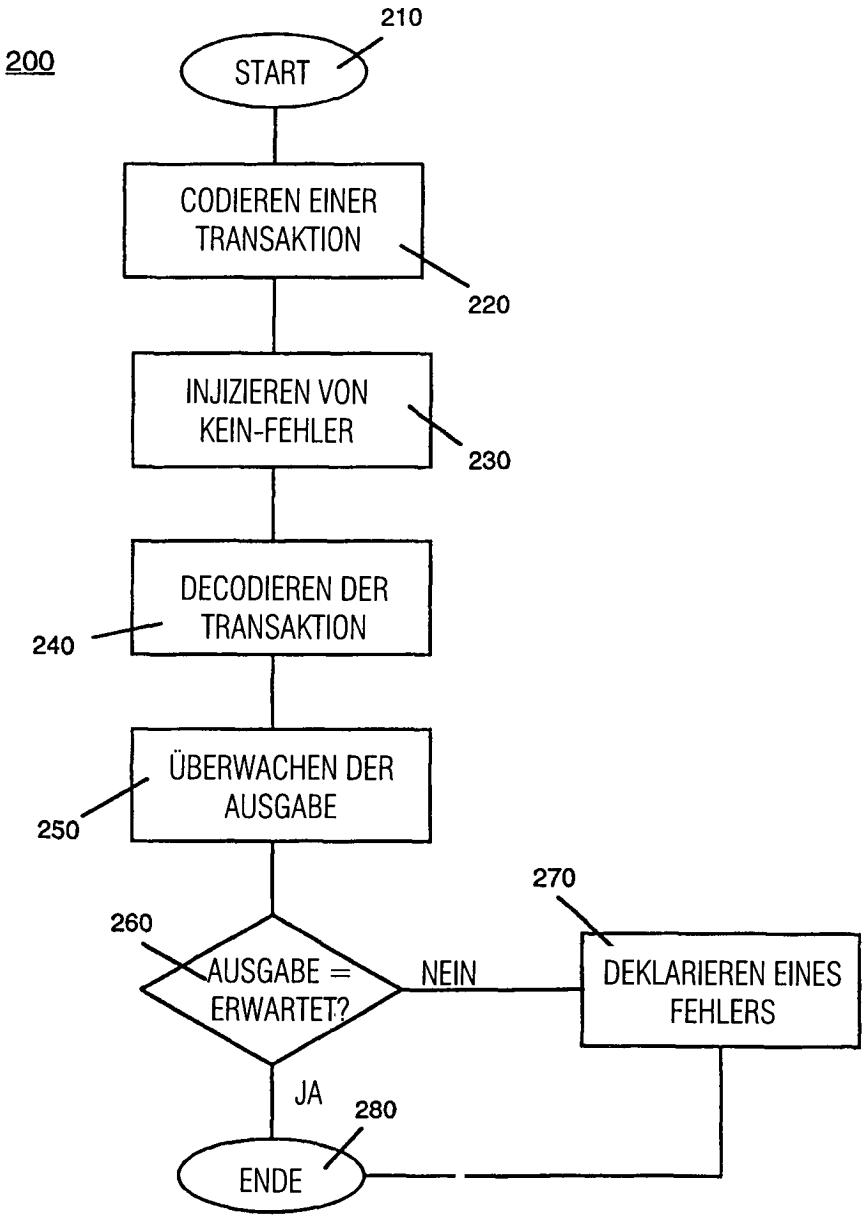
FIGUR 1A
(STAND DER TECHNIK)


20
PRÜFBITS
(LESEN)
k DATENBITS
XOR-BAUM
21
22
BITWEISE
XOR
PRÜFBITS
(SCHREIBEN)
r SYNDROMBITS

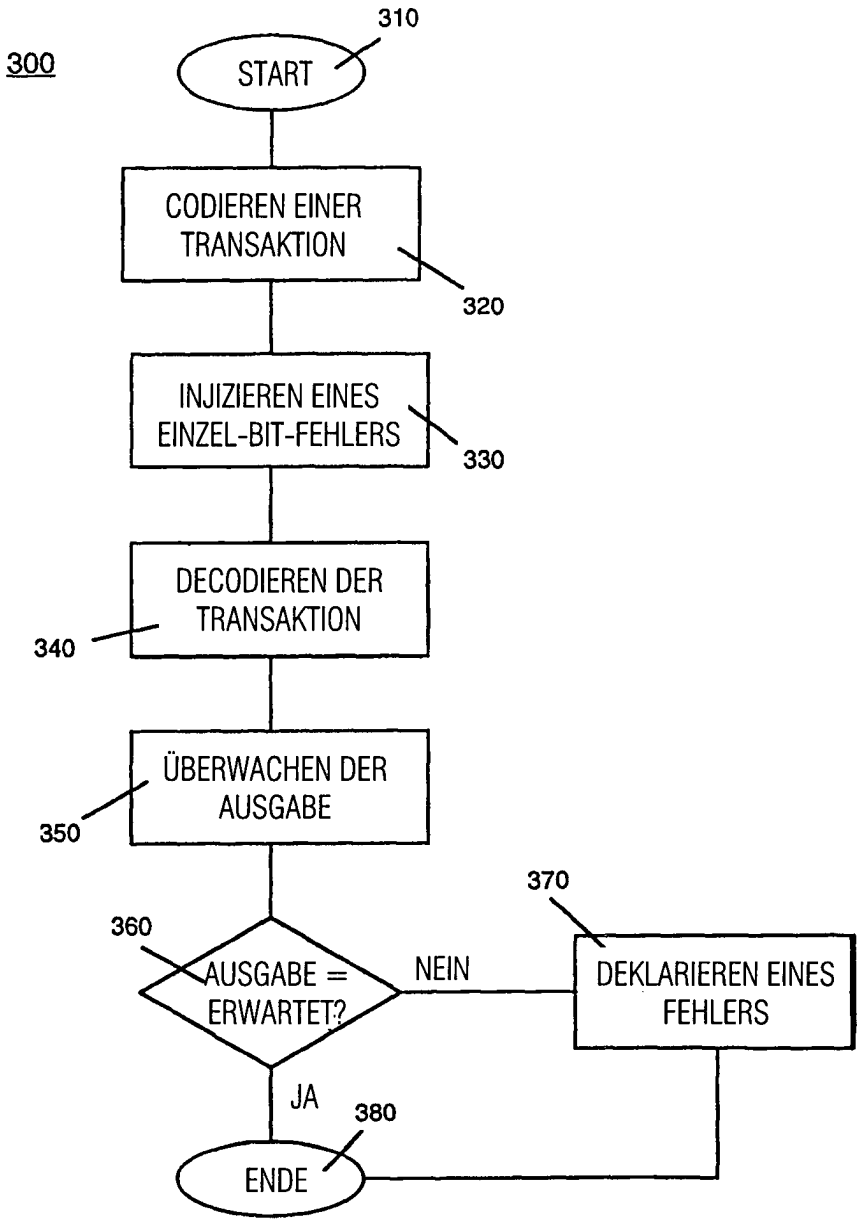
FIGUR 1B


100
KEIN_FEHLER
EINZEL_FEHLER
MEHRFACH_FEHLER
FEHLER_ORT
EMPFÄNGER
DECODIERER
135
130
DATEN AUS
64
72
FEHLER-
INJEKTION
120
ÜBERWACHER-
MODUL
140
72
110
SENDER
CODIERER
115
64
DATEN EIN

FIGUR 2


200
210
START
CODIEREN EINER TRANSAKTION
220
INJIZIEREN VON KEIN-FEHLER
230
DECODIEREN DER TRANSAKTION
240
ÜBERWACHEN DER AUSGABE
250
270
260
AUSGABE = ERWARTET?
NEIN
DEKLARIEREN EINES FEHLERS
JA
280
ENDE

FIGUR 3A


300
START
310
CODIEREN EINER TRANSAKTION
320
INJIZIEREN EINES EINZEL-BIT-FEHLERS
330
DECODIEREN DER TRANSAKTION
340
ÜBERWACHEN DER AUSGABE
350
AUSGABE = ERWARTET?
360
NEIN
DEKLARIEREN EINES FEHLERS
370
JA
ENDE
380

FIGUR 3B