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

(10) **DE 603 03 895 T2** 2006.10.05

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 424 631 B1**
(21) Deutsches Aktenzeichen: **603 03 895.6**
(96) Europäisches Aktenzeichen: **03 256 792.7**
(96) Europäischer Anmeldetag: **28.10.2003**
(97) Erstveröffentlichung durch das EPA: **02.06.2004**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **08.03.2006**
(47) Veröffentlichungstag im Patentblatt: **05.10.2006**

(51) Int Cl.[8]: ***G06F 11/10*** (2006.01)

(30) Unionspriorität:
421911 P **28.10.2002** **US**

(84) Benannte Vertragsstaaten:
AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LI, LU, MC, NL, PT, RO, SE, SI, SK, TR

(73) Patentinhaber:
SanDisk Corp., Sunnyvale, Calif., US

(74) Vertreter:
Schwabe, Sandmair, Marx, 81677 München

(72) Erfinder:
Chang, Robert C, Danville, CA 94506, US; Qawami, Bahman, San Jose, CA 95138, US; Sabet-Sharghi, Farshid, San Jose, CA 95138, US

(54) Bezeichnung: **Hybridimplementierung von Fehlerkorrekturkoden eines nichtflüchtigen Speichersystems**

Beschreibung

HINTERGRUND DER ERFINDUNG **1**. Gebiet der Erfindung

[0001] Die vorliegende Erfindung betrifft im Allgemeinen Massenspeichersysteme für digitale Daten. Insbesondere betrifft die vorliegende Erfindung Systeme und Verfahren, die ermöglichen, Algorithmen zu verwenden, um Daten zu codieren, die in Blöcke eines dynamisch zu schaltenden nichtflüchtigen Speichers geschrieben werden sollen.

2. Beschreibung des Standes der Technik

[0002] Die Verwendung von nichtflüchtigen Speichersystemen wie z. B. Flash-Speichersystemen nimmt aufgrund der kompakten physikalischen Größe solcher Speichersysteme und der Fähigkeit, dass der nichtflüchtige Speicher wiederholt umprogrammiert wird, zu. Die kompakte physikalische Größe von Flash-Speichersystemen erleichtert die Verwendung von solchen Speichersystemen in Vorrichtungen, die immer weiter verbreitet werden. Vorrichtungen, die Flash-Speichersysteme verwenden, umfassen Digitalkameras, Digitalcamcorder, digitale Musikabspielgeräte, in der Hand gehaltene Personalcomputer und globale Positionsbestimmungsvorrichtungen, ohne jedoch darauf eingeschränkt zu sein. Die Fähigkeit, einen nichtflüchtigen Speicher, der in Flash-Speichersystemen enthalten ist, wiederholt umzuprogrammieren, ermöglicht, dass Flash-Speichersysteme verwendet und wieder verwendet werden.

[0003] Im Allgemeinen können Flash-Speichersysteme Flash-Speicherkarten und Flash-Speicherchipsätze umfassen. Flash-Speicherchipsätze umfassen im Allgemeinen Flash-Speicherkomponenten und Steuereinheitskomponenten. Typischerweise kann ein Flash-Speicherchipsatz dazu vorgesehen sein, in einem eingebetteten System montiert zu werden. Die Hersteller solcher Baugruppen oder Hauptrechnersysteme erhalten typischerweise einen Flash-Speicher in Komponentenform sowie andere Komponenten und montieren dann den Flash-Speicher und die anderen Komponenten in einem Hauptrechnersystem.

[0004] Um die Genauigkeit von Daten, die in physikalischen Blöcken eines Flash-Speichers gespeichert werden, sicherzustellen, kann häufig ein Fehlerkorrekturcode-Algorithmus (ECC-Algorithmus) oder ein Fehlerprüf- und -korrekturcode-Algorithmus verwendet werden, um die Daten zur Speicherung zu codieren und um die gespeicherten Daten zu decodieren. Typischerweise verwenden ECC-Algorithmen eine zweckgebundene Schaltung oder Software, um die Daten zu codieren und zu decodieren. Viele ECC-Algorithmen oder -Verfahren können ein Paritätsbit oder Paritätsbits hinzufügen, die verwendet werden, um Fehler, die zu gespeicherten Daten gehören, sowohl zu erkennen als auch zu korrigieren.

[0005] Einige ECC-Algorithmen, die verwendet werden, um Daten zur Speicherung zu codieren und zu decodieren, sind als 1-Bit-ECC-Algorithmen und 2-Bit-ECC-Algorithmen bekannt. 1-Bit-ECC-Algorithmen ermöglichen, dass eine Menge von Symbolen derart dargestellt wird, dass, wenn ein Bit der Darstellung falsch ist, z. B. gekippt ist, die Symbole korrigiert werden, und wenn zwei Bits falsch sind, z. B. gekippt sind, die Symbole immer noch korrekt identifiziert werden können. 2-Bit-ECC-Algorithmen ermöglichen, dass eine Menge von Symbolen derart dargestellt wird, dass, wenn zwei Bits der Darstellung gekippt oder anderweitig falsch sind, die zwei Bits korrigiert werden, und wenn mehr als zwei Bits gekippt sind, die Symbole immer noch korrekt identifiziert werden können.

[0006] Im Allgemeinen kann die Verwendung eines 2-Bit-ECC-Algorithmus gegenüber einem 1-Bit-ECC-Algorithmus aufgrund der Fähigkeit, dass ein 2-Bit-ECC-Algorithmus mehr als zwei fehlerbehaftete Bits erkennt und zwei Bits korrigiert, während ein 1-Bit-ECC-Algorithmus zwei fehlerbehaftete Bits erkennen und ein Bit korrigieren kann, bevorzugt sein. Die Implementierung eines 2-Bit-ECC-Algorithmus beinhaltet jedoch im Allgemeinen, obwohl er erhöhte Fehlerkorrekturfähigkeiten an gespeicherten Daten bereitstellt, mehr Berechnungen und daher mehr Rechen-Overhead als die Implementierung eines 1-Bit-ECC-Algorithmus. Wenn mehr Rechen-Overhead erforderlich ist, kann mehr Leistung, z. B. Batterieleistung, von einem nichtflüchtigen Speicher verbraucht werden. Folglich kann die Gesamtleistung eines Speichersystems gefährdet werden, insbesondere da die Integrität von Daten, die in Blöcken gespeichert sind, die eine relativ niedrige Anzahl von Malen gelöscht wurden, im Allgemeinen relativ hoch ist.

[0007] Um die Rechen- und Leistungsanforderungen zu verringern, die mit dem Implementieren eines 2-Bit-ECC-Algorithmus verbunden sind, können einige Systeme 1-Bit-ECC-Algorithmen verwenden, um Daten zu codieren und zu decodieren. 1-Bit-ECC-Algorithmen sind jedoch häufig weniger genau als 2-Bit-ECC-Algorithmen. Wenn sich Blöcke, in denen Daten gespeichert werden, dem Ende ihrer Nutzlebensdauern nähern, enthalten ferner die in solchen Blöcken gespeicherten Daten wahrscheinlicher Fehler. Wenn 1-Bit-ECC-Algorithmen verwendet werden, um Daten zu codieren, die in Blöcken gespeichert sind, die eine relativ hohe Anzahl von Malen gelöscht wurden, und um solche Daten zu decodieren, kann an sich die Integrität der Daten gefährdet werden und die den Blöcken zugeordnete Leistung kann nachteilig beeinflusst werden.

[0008] Im Stand der Technik beschreibt US-A-6041001 ein Verfahren zum Erhöhen der Datenzuverlässigkeit der Flash-Speichervorrichtung ohne Gefährdung der Kompatibilität. Das Verfahren modifiziert Bits in verschiedenen definierten Bereichen eines Sektors innerhalb der Spezifikationen des existierenden Speicherformats, so dass die Bits als Indikatoren dienen. US-A-6085339 beschreibt ein System zum Handhaben eines Speicherfehlers unter Verwendung von Modulen, die zum Schreiben von Daten in und Lesen von Daten aus ihren zugehörigen Speicherblöcken gemäß einem unterschiedlichen Fehlerbehandlungszustand strukturiert sind.

[0009] Was daher erforderlich ist, ist ein Verfahren und eine Vorrichtung, die ermöglichen, dass die Leistung von Blöcken, die eine relativ hohe Anzahl von Malen gelöscht wurden, verbessert wird, ohne einen relativ hohen Rechen-Overhead und Leistungseinbußen zu erfordern, um Daten zu codieren und zu decodieren, die in Blöcken gespeichert sind, die eine relativ niedrige Anzahl von Malen gelöscht wurden. Das heißt, was erwünscht ist, ist ein Verfahren und eine Vorrichtung, die ermöglichen, dass der Inhalt von Blöcken unter Verwendung von verschiedenen ECC-Algorithmen codiert wird, die auf der Basis der Anzahl von Malen, die die Blöcke gelöscht wurden, ausgewählt werden können.

[0010] Es ist eine Aufgabe zumindest einer bevorzugten Ausführungsform der vorliegenden Erfindung, zumindest einen der Nachteile des Standes der Technik zu beseitigen oder zu verbessern.

ZUSAMMENFASSUNG DER ERFINDUNG

[0011] Die Erfindung ist in den beigefügten unabhängigen Ansprüchen 1 und 13 definiert. Bevorzugte Merkmale der Aspekte sind in ihren entsprechenden abhängigen Ansprüchen definiert.

[0012] Die vorliegende Erfindung betrifft ein System und ein Verfahren zur Verwendung von verschiedenen Fehlerkorrekturcode-Algorithmen, um den Inhalt von Blöcken innerhalb eines nichtflüchtigen Speichers zu codieren und zu decodieren. Gemäß einem Aspekt der vorliegenden Erfindung umfasst ein Verfahren zum Speichern von Daten innerhalb eines nichtflüchtigen Speichers das Identifizieren eines ersten Blocks, in dem die Daten gespeichert werden sollen, und das Erhalten eines dem ersten Block zugeordneten Indikators. Dann kann eine Bestimmung hinsichtlich dessen durchgeführt werden, ob der Indikator angibt, dass die Daten unter Verwendung eines ersten Algorithmus codiert werden sollen. Die Daten werden unter Verwendung des ersten Algorithmus codiert, wenn festgestellt wird, dass die Daten unter Verwendung des ersten Algorithmus codiert werden sollen, nach welchem Punkt die unter Verwendung des ersten Algorithmus codierten Daten in den ersten Block geschrieben werden.

[0013] In einer Ausführungsform umfasst das Verfahren auch das Codieren der Daten unter Verwendung eines zweiten Algorithmus, wenn festgestellt wird, dass die Daten nicht unter Verwendung des ersten Algorithmus codiert werden sollen, und das Schreiben der unter Verwendung des zweiten Algorithmus codierten Daten in den ersten Block. In einer solchen Ausführungsform kann der erste Algorithmus ein 1-Bit-Fehlerkorrekturcode-Algorithmus (ECC-Algorithmus) sein und der zweite Algorithmus kann ein 2-Bit-ECC-Algorithmus sein.

[0014] Indem ermöglicht wird, dass die in Blöcken eines nichtflüchtigen Speichersystems gespeicherten Daten unter Verwendung von verschiedenen ECC-Algorithmen auf einer blockweisen Basis codiert werden, können einige Blöcke innerhalb des Speichersystems unter Verwendung eines 1-Bit-ECC-Algorithmus codiert werden, während andere Blöcke unter Verwendung eines 2-Bit-ECC-Algorithmus codiert werden können. Ein 1-Bit-ECC-Algorithmus oder ein weniger rechenintensiver Algorithmus wird verwendet, um Daten zu codieren, die in Blöcken gespeichert werden sollen, die keine relativ hohe Anzahl von Löschoperationen durchlaufen haben, während ein 2-Bit-ECC-Algorithmus oder ein rechenintensiverer Algorithmus verwendet wird, um Daten zu codieren, die in Blöcken gespeichert werden sollen, die eine relativ hohe Anzahl von Löschoperationen durchlaufen haben. Da ein 2-Bit-ECC-Algorithmus im Allgemeinen ermöglicht, dass Daten genauer codiert und decodiert werden, als ein 1-Bit-ECC-Algorithmus, ermöglicht die Verwendung eines 2-Bit-ECC-Algorithmus zum Codieren von Daten, die in Blöcken gespeichert werden sollen, die sich dem Ende ihrer Nutzbarkeit nähern, dass die den Blöcken zugeordnete Leistung verbessert wird. Wenn ein 1-Bit-ECC-Algorithmus nicht verwendet wird, wenn ein Block nicht nahe dem Ende seiner Nutzbarkeit liegt, kann die Anzahl von Berechnungen, die erforderlich sind, um Daten zu speichern und aus einem solchen Block zu lesen, verringert werden, wodurch ermöglicht wird, dass die Gesamtgeschwindigkeit von Lese- und Schreibprozessen verbessert wird, und Leistungsanforderungen, die mit dem Speichersystem verbunden sind, verringert werden.

[0015] Gemäß einem weiteren Aspekt der vorliegenden Erfindung umfasst ein Verfahren zum Lesen von Daten innerhalb eines nichtflüchtigen Speichers eines Speichersystems das Identifizieren eines ersten Blocks, aus dem Daten gelesen werden sollen, das Erhalten eines dem ersten Block zugeordneten Indikators und das Feststellen, wenn der Indikator angibt, dass die im ersten Block gespeicherten Daten unter Verwendung eines ersten Algorithmus codiert wurden. Das Verfahren umfasst auch das Decodieren der Daten unter Verwendung des ersten Algorith-

mus, wenn festgestellt wird, dass die Daten unter Verwendung des ersten Algorithmus codiert wurden. In einer Ausführungsform umfasst das Verfahren ferner das Decodieren der Daten unter Verwendung eines zweiten Algorithmus, wenn festgestellt wird, dass die Daten nicht unter Verwendung des ersten Algorithmus codiert wurden.

[0016] In einer weiteren Ausführungsform ist der Indikator dazu beschaffen anzugeben, wenn der Block ein zurückgewonnener Block ist. Wenn der Block ein zurückgewonnener Block ist, ist der Indikator ferner so beschaffen, dass er angibt, dass die Daten unter Verwendung des zweiten Algorithmus codiert wurden. In noch einer weiteren Ausführungsform ist der Indikator so beschaffen, dass er eine Anzahl von Malen angibt, die der Block gelöscht wurde. In einer solchen Ausführungsform kann das Bestimmen, wenn der Indikator angibt, dass die Daten unter Verwendung des ersten Algorithmus codiert wurden, auch das Bestimmen, wenn der Indikator geringer ist als ein Schwellenwert, umfassen. Wenn der Indikator geringer ist als der Schwellenwert, besteht die Implikation darin, dass die Daten unter Verwendung des ersten Algorithmus codiert wurden.

[0017] Gemäß einem nochmals weiteren Aspekt der vorliegenden Erfindung umfasst ein Speichersystem einen nichtflüchtigen Speicher, der einen ersten Block und einen zweiten Block umfasst. Der erste Block umfasst eine erste Menge von Inhalten, die unter Verwendung eines ersten Algorithmus codiert sind, und der zweite Block umfasst eine zweite Menge von Inhalten, die unter Verwendung eines zweiten Algorithmus codiert sind. Der nichtflüchtige Speicher umfasst auch eine Datenstruktur, z. B. einen Löschzählstandblock, der so beschaffen ist, dass er angibt, dass die erste Menge von Inhalten unter Verwendung des ersten Algorithmus codiert ist und dass die zweite Menge von Inhalten unter Verwendung des zweiten Algorithmus codiert ist. Das Speichersystem umfasst ferner Codevorrichtungen zum Zugreifen auf die Datenstruktur. Solche Codevorrichtungen umfassen Codevorrichtungen zum Bestimmen, dass die erste Menge von Inhalten unter Verwendung des ersten Algorithmus codiert ist, und Codevorrichtungen zum Bestimmen, dass die zweite Menge von Inhalten unter Verwendung des zweiten Algorithmus codiert ist.

[0018] Ein weiterer Aspekt der Erfindung schafft ein Speichersystem mit:
einem nichtflüchtigen Speicher mit einer Vielzahl von Blöcken, wobei die Vielzahl von Blöcken einen ersten Block umfassen;
Codevorrichtungen zum Identifizieren des ersten Blocks, in dem Daten gespeichert werden sollen;
Codevorrichtungen zum Erhalten eines dem ersten Block zugeordneten Indikators;
Codevorrichtungen zum Bestimmen, wenn der Indikator angibt, dass die Daten unter Verwendung eines ersten Algorithmus codiert werden sollen;
Codevorrichtungen zum Codieren der Daten unter Verwendung des ersten Algorithmus, wenn festgestellt wird, dass die Daten unter Verwendung des ersten Algorithmus codiert werden sollen;
Codevorrichtungen zum Schreiben der unter Verwendung des ersten Algorithmus codierten Daten in den ersten Block; und
einem Speicherbereich, der die Codevorrichtungen speichert.

[0019] Vorzugsweise umfasst der Speicher ferner:
Codevorrichtungen zum Codieren der Daten unter Verwendung eines zweiten Algorithmus, wenn festgestellt wird, dass die Daten nicht unter Verwendung des ersten Algorithmus codiert werden sollen; und
Codevorrichtungen zum Schreiben der unter Verwendung des zweiten Algorithmus codierten Daten in den ersten Block.

[0020] Ein weiterer Aspekt der Erfindung schafft ein Speichersystem mit:
einem nichtflüchtigen Speicher mit einer Vielzahl von Blöcken, wobei die Vielzahl von Blöcken einen ersten Block umfassen, wobei der erste Block Daten umfasst;
Codevorrichtungen zum Identifizieren des ersten Blocks;
Codevorrichtungen zum Erhalten eines dem ersten Block zugeordneten Indikators;
Codevorrichtungen zum Bestimmen, wenn der Indikator angibt, dass die Daten unter Verwendung eines ersten Algorithmus codiert wurden;
Codevorrichtungen zum Decodieren der Daten unter Verwendung des ersten Algorithmus, wenn festgestellt wird, dass die Daten unter Verwendung des ersten Algorithmus codiert wurden; und
einem Speicherbereich, der die Codevorrichtungen speichert.

[0021] Ein weiterer Aspekt der Erfindung schafft ein Speichersystem mit:
einem nichtflüchtigen Speicher mit einer Vielzahl von Blöcken, wobei die Blöcke einen ersten Block und einen zweiten Block umfassen, wobei der erste Block eine erste Menge von Inhalten umfasst, die unter Verwendung eines ersten Algorithmus codiert sind, der zweite Block eine zweite Menge von Inhalten umfasst, die unter Verwendung eines zweiten Algorithmus codiert sind, wobei der nichtflüchtige Speicher ferner eine Datenstruktur umfasst, die so beschaffen ist, dass sie angibt, dass die erste Menge von Inhalten unter Verwendung des ersten Algorithmus codiert ist und dass die zweite Menge von Inhalten unter Verwendung des zweiten Algorithmus codiert ist;
Codevorrichtungen zum Zugreifen auf die Datenstruktur, wobei die Codevorrichtungen zum Zugreifen auf die Datenstruktur Codevorrichtungen zum Bestimmen, dass die erste Menge von Inhalten unter

Verwendung des ersten Algorithmus codiert ist, und Codevorrichtungen zum Bestimmen, dass die zweite Menge von Inhalten unter Verwendung des zweiten Algorithmus codiert ist, umfassen; und
einem Speicherbereich, der die Codevorrichtungen speichert.

[0022] Diese und weitere Vorteile der vorliegenden Erfindung werden beim Lesen der folgenden ausführlichen Beschreibungen und Studieren der verschiedenen Figuren der Zeichnungen ersichtlich.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0023] Bevorzugte Ausführungsformen der Erfindung werden nun mit Bezug auf die begleitenden Zeichnungen beschrieben, in denen:

[0024] Fig. 1a eine schematische Darstellung eines allgemeinen Hauptrechnersystems ist, das einen nichtflüchtigen Speicher umfasst.

[0025] Fig. 1b eine schematische Darstellung einer Speichervorrichtung, z. B. der Speichervorrichtung **120** von Fig. 1a, ist.

[0026] Fig. 1c eine schematische Darstellung eines Hauptrechnersystems ist, das einen eingebetteten nichtflüchtigen Speicher umfasst.

[0027] Fig. 2a ein Prozessablaufdiagramm ist, das ein Verfahren zum Schreiben von Anwenderdaten in einen Block unter Verwendung entweder eines 1-Bit- oder 2-Bit-ECC gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

[0028] Fig. 2b ein Prozessablaufdiagramm ist, das ein Verfahren zum Lesen von Inhalten aus einem Block innerhalb eines Systems, in dem die Inhalte entweder unter Verwendung eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert sein können, gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

[0029] Fig. 3 ein Prozessablaufdiagramm ist, das ein Verfahren zum Initialisieren eines Speichersystems, in dem Inhalte von Blöcken unter Verwendung einer Hybrid-ECC-Implementierung codiert werden können, gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

[0030] Fig. 4a eine schematische Blockdiagrammdarstellung eines Prozesses zum Zurückgewinnen von unbrauchbaren Blöcken gemäß einer Ausführungsform der vorliegenden Erfindung ist.

[0031] Fig. 4b eine schematische Darstellung eines Abschnitts eines Löschzählstandblocks gemäß einer Ausführungsform der vorliegenden Erfindung ist.

[0032] Fig. 5a ein Prozessablaufdiagramm ist, das ein Verfahren zum Schreiben in einen Block innerhalb eines Systems, in dem sich zurückgewonnene Blöcke befinden können und Daten unter Verwendung entweder eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert werden können, gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

[0033] Fig. 5b ein Prozessablaufdiagramm ist, das ein Verfahren zum Lesen von Inhalten von einem Block, der ein zurückgewonnener Block sein kann und Inhalte aufweisen kann, die entweder unter Verwendung eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert sind, gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

[0034] Fig. 6 ein Prozessablaufdiagramm ist, das ein Verfahren zum Initialisieren eines Speichersystems mit einer Hybrid-ECC-Implementierung, das zurückgewonnene Blöcke umfasst, gemäß einer Ausführungsform der vorliegenden Erfindung darstellt.

[0035] Fig. 7 eine schematische Blockdiagrammdarstellung einer Systemarchitektur gemäß einer Ausführungsform der vorliegenden Erfindung ist.

AUSFÜHRLICHE BESCHREIBUNG DER AUSFÜHRUNGSFORMEN

[0036] Ein Fehlerkorrekturcode-Algorithmus (ECC-Algorithmus), wie z. B. entweder ein 1-Bit-ECC-Algorithmus oder ein 2-Bit-ECC-Algorithmus, wird häufig verwendet, um in einem physikalischen Block eines nichtflüchtigen Speichers zu speichernde Daten zu codieren und gespeicherte Daten zu decodieren. Die Verwendung von ECC-Algorithmen ermöglicht im Allgemeinen, dass die Genauigkeit von Daten, die innerhalb eines physikalischen Blocks gespeichert sind, verbessert wird. Die Verwendung eines rechenintensiveren 2-Bit-ECC-Algorithmus kann gegenüber einem weniger rechenintensiven 1-Bit-ECC-Algorithmus aufgrund der Fähigkeit eines 2-Bit-ECC-Algorithmus, mehr fehlerhafte Bits zu korrigieren, als unter Verwendung eines 1-Bit-ECC-Algorithmus korrigiert werden können, bevorzugt sein. Die Implementierung eines 2-Bit-ECC-Algorithmus ist jedoch, obwohl sie erhöhte Fehlerkorrekturfähigkeiten bereitstellt, hinsichtlich einer Anzahl von Berechnungen und Leistungsanforderungen aufwändiger als ein 1-Bit-ECC-Algorithmus.

[0037] In vielen Fällen, in denen ein Block, in dem Daten gespeichert sind, relativ jung ist und daher nicht einer relativ hohen Anzahl von Löschzyklen unterzogen wurde, kann ein 1-Bit-ECC-Algorithmus ausreichen, um die Integrität von vielen der Daten sicherzustellen. An sich kann die Implementierung ei-

nes 2-Bit-ECC-Algorithmus nicht erforderlich sein. Wenn jedoch ein Block älter wird und einer relativ hohen Anzahl von Löschzyklen unterzogen wurde, kann ein 1-Bit-ECC-Algorithmus nicht ausreichen, um ein gewünschtes Niveau an Datenintegrität sicherzustellen, und die Verwendung eines 2-Bit-ECC-Algorithmus kann die Integrität der Daten signifikant verbessern.

[0038] Eine Hybrid-ECC-Implementierung ermöglicht, dass ECC-Algorithmen, die zum Codieren und Decodieren von Daten verwendet werden, dynamisch umgeschaltet werden. Insbesondere können in einer Ausführungsform Daten, die in Blöcken gespeichert werden, die einer relativ niedrigen Anzahl von Löschzyklen unterzogen wurden, unter Verwendung eines weniger rechenintensiven und weniger genauen Algorithmus, z. B. eines 1-Bit-ECC-Algorithmus, codiert werden, während Blöcke, die einer relativ hohen Anzahl von Löschzyklen unterzogen wurden, unter Verwendung eines rechenintensiveren und genaueren Algorithmus, wie z. B. eines 2-Bit-ECC-Algorithmus, codiert werden können. Durch dynamisches Feststellen, wenn Daten unter Verwendung eines genaueren Algorithmus wie z. B. eines 2-Bit-ECC-Algorithmus anstatt eines "Standard"-Algorithmus, z. B. eines weniger genauen Algorithmus wie z. B. eines 1-Bit-ECC-Algorithmus, codiert werden sollen, kann der Algorithmus, der zum Codieren von in einem speziellen Block zu speichernden Daten gewählt wird, in Abhängigkeit von den Eigenschaften des speziellen Blocks ausgewählt werden, und der genauere Algorithmus kann wirksam verwendet werden, wenn er einen beträchtlichen Nutzen schaffen würde. Wenn sich beispielsweise ein Block dem Ende seiner hochgerechneten Nutzlebensdauer nähert, kann die Verwendung eines 2-Bit-ECC-Algorithmus zum Codieren von im Block zu speichernden Daten die Genauigkeit oder Integrität der im Block gespeicherten Daten verbessern und kann auch ermöglichen, dass die Nutzlebensdauer des Blocks potentiell verlängert wird. Unter Verwendung eines weniger rechenintensiven Algorithmus zum Codieren von Daten, die in Blöcken gespeichert werden sollen, die sich nicht dem Ende ihrer Nutzlebensdauern nähern, können ferner die Leistungsanforderungen eines gesamten Speichersystems verringert werden, wodurch die Haltbarkeit des gesamten Speichersystems verbessert wird.

[0039] In einer Ausführungsform kann ein Schwellenlöschzählstand oder eine Schwellenanzahl von Löschzyklen als Indikator dessen verwendet werden, ob ein weniger rechenintensiver ECC-Algorithmus mit geringerer Genauigkeit oder ein rechenintensiverer ECC-Algorithmus mit höherer Genauigkeit verwendet werden soll, um in einen Block zu schreibende Daten zu codieren. Wenn ein Vergleich der Anzahl von Löschzyklen, denen der Block unterzogen wurde, mit der Schwellenanzahl angibt, dass der Block mehr Löschzyklen unterzogen wurde als die Schwellenanzahl, dann kann der ECC-Algorithmus mit höherer Genauigkeit verwendet werden, da der Block als nahe dem Ende seiner Nutzlebensdauer betrachtet werden kann.

[0040] Flash-Speichersysteme oder allgemeiner nichtflüchtige Speichervorrichtungen, die eine Hybrid-ECC-Implementierung verwenden können, die ermöglicht, dass Blöcke innerhalb eines Systems unter Verwendung von verschiedenen ECC-Algorithmen codiert werden, umfassen im Allgemeinen einen Flash-Speicher, z. B. NAND- oder MLC-NAND, Karten und Chipsätze. Typischerweise werden Flash-Speichersysteme in Verbindung mit einem Hauptrechnersystem verwendet, so dass das Hauptrechnersystem Daten in die Flash-Speichersysteme schreiben oder Daten aus diesen lesen kann. Einige Flash-Speichersysteme umfassen jedoch einen eingebetteten Flash-Speicher und eine Software, die auf einem Hauptrechner ausgeführt wird, so dass sie im Wesentlichen als Steuereinheit für den eingebetteten Flash-Speicher wirkt, wie nachstehend mit Bezug auf Fig. 1c erörtert wird. Mit Bezug auf Fig. 1a wird ein allgemeines Hauptrechnersystem, das eine nichtflüchtige Speichervorrichtung, z. B. eine CompactFlash-Speicherkarte, umfasst, beschrieben. Ein Hauptrechner- oder Computersystem **100** umfasst im Allgemeinen einen Systembus **104**, der ermöglicht, dass ein Mikroprozessor **108**, ein Direktzugriffsspeicher (RAM) **112** und Eingabe/Ausgabe-Schaltungen **116** kommunizieren. Es sollte erkannt werden, dass das Hauptrechnersystem **100** im Allgemeinen andere Komponenten umfassen kann, z. B. Anzeigevorrichtungen und eine Vernetzungsvorrichtung, die für Erläuterungszwecke nicht gezeigt sind.

[0041] Im Allgemeinen kann das Hauptrechnersystem **100** in der Lage sein, Informationen zu erfassen, einschließlich, jedoch nicht begrenzt auf Standbildinformationen, Audioinformationen und Videobildinformationen. Solche Informationen können in Echtzeit erfasst werden und können in einer drahtlosen Weise zum Hauptrechnersystem **100** übertragen werden. Obwohl das Hauptrechnersystem **100** im Wesentlichen ein beliebiges System sein kann, ist das Hauptrechnersystem **100** typischerweise ein System wie z. B. eine Digitalkamera, eine Videokamera, eine zelluläre Kommunikationsvorrichtung, ein Audioabspielgerät oder ein Videoabspielgerät. Es sollte jedoch erkannt werden, dass das Hauptrechnersystem **100** im Allgemeinen im Wesentlichen ein beliebiges System sein kann, das Daten oder Informationen speichert und Daten oder Informationen abruft.

[0042] Das Hauptrechnersystem **100** kann auch ein System sein, das entweder nur Daten erfasst oder nur Daten abruft. Das heißt, das Hauptrechnersystem **100** kann in einer Ausführungsform ein zweckgebundenes System sein, das Daten speichert, oder

das Hauptrechnersystem **100** kann ein zweckgebundenes System sein, das Daten liest. Als Beispiel kann das Hauptrechnersystem **100** eine Speicherschreibvorrichtung sein, die so beschaffen ist, dass sie nur Daten schreibt oder speichert. Alternativ kann das Hauptrechnersystem **100** eine Vorrichtung wie z. B. ein MP3-Spieler sein, der typischerweise dazu beschaffen ist, Daten zu lesen oder abzurufen und nicht Daten zu erfassen.

[0043] Eine nichtflüchtige Speichervorrichtung **120**, die in einer Ausführungsform eine entnehmbare nichtflüchtige Speichervorrichtung ist, ist so beschaffen, dass sie mit dem Bus **104** koppelt, um Informationen zu speichern. Ein wahlweiser Schnittstellenblock **130** kann ermöglichen, dass die nichtflüchtige Speichervorrichtung **120** indirekt mit dem Bus **104** koppelt. Wenn er vorhanden ist, dient der Eingabe/Ausgabe-Block **116** zum Verringern der Belastung am Bus **104**, wie für Fachleute verständlich ist. Die nichtflüchtige Speichervorrichtung **120** umfasst einen nichtflüchtigen Speicher **124** und ein wahlweises Speichersteuersystem **128**. In einer Ausführungsform kann die nichtflüchtige Speichervorrichtung **120** auf einem einzelnen Chip oder einem Chipschaltkreis implementiert sein. Alternativ kann die nichtflüchtige Speichervorrichtung **120** auf einem Mehrchipmodul oder auf mehreren diskreten Komponenten, die einen Chipsatz bilden können und zusammen als nichtflüchtige Speichervorrichtung **120** verwendet werden können, implementiert sein. Eine Ausführungsform der nichtflüchtigen Speichervorrichtung **120** wird nachstehend mit Bezug auf Fig. 1b genauer beschrieben.

[0044] Der nichtflüchtige Speicher **124**, z. B. ein Flash-Speicher wie z. B. ein NAND-Flash-Speicher oder ein MLC-NAND-Flash-Speicher, ist dazu beschaffen, Daten derart zu speichern, dass auf die Daten zugegriffen und diese gelesen werden können, wie erforderlich. Die im nichtflüchtigen Speicher **124** gespeicherten Daten können auch gelöscht werden, wie geeignet, obwohl es selbstverständlich sein sollte, dass einige Daten im nichtflüchtigen Speicher **124** nicht löschbar sein können. Die Prozesse des Speicherns von Daten, Lesens von Daten und Löschens von Daten werden im Allgemeinen durch das Speichersteuersystem **128** oder, wenn das Speichersteuersystem **128** nicht vorhanden ist, durch die vom Mikroprozessor **108** ausgeführte Software gesteuert. Die Operation des nichtflüchtigen Speichers **124** kann derart gemanagt werden, dass die Lebensdauer des nichtflüchtigen Speichers **124** im Wesentlichen maximiert wird, indem im Wesentlichen bewirkt wird, dass Abschnitte des nichtflüchtigen Speichers **124** im Wesentlichen gleich abgenutzt werden.

[0045] Die nichtflüchtige Speichervorrichtung **120** wurde im Allgemeinen als Vorrichtung mit einem wahlweisen Speichersteuersystem **128**, d. h. mit einer Steuereinheit, beschrieben. Häufig kann die nichtflüchtige Speichervorrichtung **120** separate Chips für die Funktionen des nichtflüchtigen Speichers **124** und des Speichersteuersystems **128**, d. h. der Steuereinheit, umfassen. Obwohl beispielsweise nichtflüchtige Speichervorrichtungen, einschließlich, jedoch nicht begrenzt auf PC-Karten, Compact-Flash-Karten, Multimediakarten und sichere digitale Karten, Steuereinheiten umfassen, die auf einem separaten Chip implementiert sein können, können andere nichtflüchtige Speichervorrichtungen keine Steuereinheiten umfassen, die auf einem separaten Chip implementiert sind. In einer Ausführungsform, in der die nichtflüchtige Speichervorrichtung **120** keine separaten Speicher- und Steuereinheitschips umfasst, können die Speicher- und Steuereinheitsfunktionen in einem einzelnen Chip integriert sein, wie für Fachleute zu erkennen ist. Alternativ kann die Funktionalität des Speichersteuersystems **128** vom Mikroprozessor **108** bereitgestellt werden, wie beispielsweise in einer Ausführungsform, in der die nichtflüchtige Speichervorrichtung **120** keine Speichersteuereinheit **128** umfasst, wie vorstehend erörtert.

[0046] Mit Bezug auf Fig. 1b wird die nichtflüchtige Speichervorrichtung **120** gemäß einer Ausführungsform der vorliegenden Erfindung genauer beschrieben. Wie vorstehend beschrieben, umfasst die nichtflüchtige Speichervorrichtung **120** einen nichtflüchtigen Speicher **124** und kann ein Speichersteuersystem **128** umfassen. Der Speicher **124** und das Steuersystem oder die Steuereinheit können primäre Komponenten der nichtflüchtigen Speichervorrichtung **120** sein, obwohl, wenn der Speicher **124** beispielsweise eine eingebettete NAND-Vorrichtung wie z. B. ein eingebetteter MLC-NAND-Speicher ist, die nichtflüchtige Speichervorrichtung **120** kein Steuersystem **128** umfassen kann. Der Speicher **124** kann eine Matrix von Speicherzellen sein, die auf einem Halbleitersubstrat ausgebildet sind, wobei ein oder mehrere Bits von Daten in den einzelnen Speicherzellen gespeichert werden, indem ein oder zwei oder mehr Pegel von Ladung in einzelnen Speicherelementen der Speicherzellen gespeichert werden. Ein nichtflüchtiger, elektrisch löschbarer, programmierbarer Flash-Festwertspeicher (EEPROM) ist ein Beispiel einer üblichen Art von Speicher für solche Systeme.

[0047] Wenn es vorhanden ist, kommuniziert das Steuersystem **128** über einen Bus **15** mit einem Hauptrechner oder einem anderen System, das das Speichersystem verwendet, um Daten zu speichern. Der Bus **15** ist im Allgemeinen ein Teil des Busses **140** von Fig. 1a. Das Steuersystem **128** steuert auch die Operation des Speichers **124**, der eine Speicherzellenmatrix **11** umfassen kann, um Daten zu schreiben, die vom Hauptrechner geliefert werden, Daten zu lesen, die vom Hauptrechner angefordert werden, und verschiedene organisatorische Funktionen beim

Betreiben des Speichers **124** durchzuführen. Das Steuersystem **128** umfasst im Allgemeinen einen Universal-Mikroprozessor, dem ein nichtflüchtiger Softwarespeicher, verschiedene Logikschaltungen und dergleichen zugeordnet sind. Eine oder mehrere Zustandsmaschinen sind häufig auch enthalten, um die Durchführung von speziellen Routinen zu steuern.

[0048] Die Speicherzellenmatrix **11** wird typischerweise vom Steuersystem **128** oder Mikroprozessor **108** über Adressendecodierer **17** adressiert. Die Decodierer **17** legen die korrekten Spannungen an Gate- und Bitleitungen der Matrix **11** an, um Daten in eine Gruppe von Speicherzellen, die vom Steuersystem **128** adressiert werden, zu programmieren, Daten aus dieser zu lesen oder diese zu löschen. Zusätzliche Schaltungen **19** umfassen Programmiertreiber, die Spannungen steuern, die an Elemente der Matrix angelegt werden und die von den in eine adressierte Gruppe von Zellen programmierten Daten abhängen. Die Schaltungen **19** umfassen auch Leseverstärker und andere Schaltungen, die zum Lesen von Daten aus einer adressierten Gruppe von Speicherzellen erforderlich sind. In die Matrix **11** zu programmierende Daten oder Daten, die vor kurzem aus der Matrix **11** gelesen wurden, werden typischerweise in einem Pufferspeicher **21** innerhalb des Steuersystems **128** gespeichert. Das Steuersystem **128** enthält gewöhnlich auch verschiedene Register zum vorübergehenden Speichern von Befehls- und Zustandsdaten und dergleichen.

[0049] Die Matrix **11** ist in eine große Anzahl von BLÖCKEN 0-N Speicherzellen unterteilt. Wie es für Flash-EEPROM-Systeme üblich ist, ist der Block typischerweise die kleinste Löscheinheit. Das heißt, jeder Block enthält die minimale Anzahl von Speicherzellen, die zusammen gelöscht werden. Jeder Block ist typischerweise in eine Anzahl von Seiten unterteilt. Wie für Fachleute zu erkennen ist, kann eine Seite die kleinste Programmiereinheit sein. Das heißt, eine grundlegende Programmieroperation schreibt Daten in ein Minimum von einer Seite von Speicherzellen oder liest sie daraus. Einer oder mehrere Sektoren von Daten sind typischerweise innerhalb jeder Seite gespeichert. Wie in Fig. 1b gezeigt, umfasst ein Sektor Anwenderdaten und Overhead-Daten. Overhead-Daten umfassen typischerweise einen ECC, der aus den Anwenderdaten des Sektors berechnet wurde. Ein Abschnitt **23** des Steuersystems **128** berechnet den ECC, wenn Daten in die Matrix **11** programmiert werden, und prüft auch den ECC, wenn Daten aus der Matrix **11** gelesen werden. Alternativ werden die ECCs in anderen Seiten oder anderen Blöcken gespeichert als die Anwenderdaten, die sie betreffen.

[0050] Ein Sektor von Anwenderdaten ist typischerweise 512 Bytes entsprechend der Größe eines Sektors in Magnetplattenlaufwerken. Overhead-Daten oder redundante Daten sind typischerweise zusätzliche 16 Bytes. Ein Sektor von Daten ist am üblichsten in jeder Seite enthalten, aber zwei oder mehr Sektoren können statt dessen eine Seite bilden. Eine beliebige Anzahl von Seiten kann im Allgemeinen einen Block bilden. Beispielsweise kann ein Block aus acht Seiten bis zu 512, 1024 oder mehr Seiten gebildet sein. Die Anzahl von Blöcken wird so gewählt, dass sie eine gewünschte Datenspeicherkapazität für das Speichersystem bereitstellt. Die Matrix **11** ist typischerweise in einige Untermatrizes (nicht dargestellt) unterteilt, von denen jede einen Anteil der Blöcke enthält, die etwas unabhängig voneinander arbeiten, um den Grad an Parallelität bei der Ausführung von verschiedenen Speicheroperationen zu erhöhen. Ein Beispiel der Verwendung von mehreren Untermatrizes ist im US-Patent Nr. 5 890 192 beschrieben.

[0051] In einer Ausführungsform ist ein nichtflüchtiger Speicher wie z. B. ein MLC-NAND-Speicher in ein System, z. B. ein Hauptrechnersystem, eingebettet. Fig. 1c ist eine schematische Darstellung eines Hauptrechnersystems, das einen eingebetteten nichtflüchtigen Speicher umfasst. Ein Hauptrechner- oder Computersystem **150** umfasst im Allgemeinen einen Systembus **154**, der ermöglicht, dass ein Mikroprozessor **158**, ein RAM **162** und Eingabe/Ausgabe-Schaltungen **166** unter anderen Komponenten (nicht dargestellt) des Hauptrechnersystems **150** kommunizieren. Ein nichtflüchtiger Speicher **174**, z. B. ein Flash-Speicher, ermöglicht, dass Informationen innerhalb des Hauptrechnersystems **150** gespeichert werden. Eine Schnittstelle **180** kann zwischen dem nichtflüchtigen Speicher **174** und dem Bus **154** vorgesehen sein, um zu ermöglichen, dass Informationen aus dem nichtflüchtigen Speicher **174** gelesen und in diesen geschrieben werden.

[0052] Der nichtflüchtige Speicher **174** kann vom Mikroprozessor **158** gemanagt werden, der effektiv eine oder beide von Software und Firmware ausführt, die dazu beschaffen sind, den nichtflüchtigen Speicher **174** zu steuern. Das heißt, der Mikroprozessor **158** kann als Steuereinheit dienen, die Codevorrichtungen, d. h. Software-Codevorrichtungen oder Firmware-Codevorrichtungen, abarbeitet, die ermöglichen, dass der nichtflüchtige Speicher **174** gesteuert wird. Solche Codevorrichtungen, die nachstehend beschrieben werden, können ermöglichen, dass physikalische Blöcke im nichtflüchtigen Speicher **174** adressiert werden, und können ermöglichen, dass Informationen in den physikalischen Blöcken gespeichert, aus diesen gelesen und aus diesen gelöscht werden.

[0053] Innerhalb eines Speichersystems, das eine Hybrid-ECC-Implementierung verwendet, werden, bevor Daten in einen Block geschrieben werden können, dem Block zugeordnete Informationen typi-

scherweise erhalten und untersucht, um einen geeigneten ECC-Algorithmus zu bestimmen, der zum Codieren der Daten zu verwenden ist, die in den Block geschrieben werden sollen. Mit Bezug auf Fig. 2a werden die Schritte zum Schreiben von Anwenderdaten in einen Block unter Verwendung entweder eines 1-Bit- oder 2-Bit-ECC gemäß einer Ausführungsform der vorliegenden Erfindung beschrieben. Ein Prozess **200** zum Schreiben von Daten beginnt in Schritt **204**, in dem ein Block, in den geschrieben werden soll, identifiziert wird. Der Block kann ein ungenutzter Block sein, z. B. ein Block, der von einer übrigen Blockgruppe erhalten wird, oder der Block kann ein Block sein, der gerade in Gebrauch ist und dazu beschaffen ist, zusätzliche Inhalte anzunehmen. Sobald der Block identifiziert ist, wird der Löschzählstand für den Block aus dem Löschzählstandblock in Schritt **208** gelesen. Ein Löschzählstandblock, wie in der gleichzeitig anhängigen US-Patentanmeldung Nr. 10/281 626 beschrieben, ist eine Datenstruktur, die im nichtflüchtigen Speicher gespeichert wird und Löschzählstände enthält, die im Wesentlichen allen nutzbaren Blöcken innerhalb eines nichtflüchtigen Speichers zugeordnet sind. Ein Löschzählstand gibt im Allgemeinen die Anzahl von Malen an, die ein gegebener Block gelöscht wurde, und sieht daher eine Angabe der Lebensdauer des Blocks vor.

[0054] Nachdem der Löschzählstand für den Block gelesen ist, wird in Schritt **212** festgestellt, ob der Löschzählstand geringer ist als ein Schwellenlöschzählstand. Ein Schwellenlöschzählstand kann ein Löschzählstand sein, der von einem Anwender eines nichtflüchtigen Speichersystems festgelegt wird, oder der Schwellenlöschzählstand kann ein Systemparameter sein. Der Wert eines Schwellenlöschzählstandes kann umfangreich variieren. In einer Ausführungsform, in der ein Block wahrscheinlich nicht mehr nutzbar ist, nachdem der Block ungefähr 10000 mal gelöscht wurde, kann ein Schwellenlöschzählstand beispielsweise auf einen Wert zwischen ungefähr 9500 und ungefähr 9800 gesetzt werden. Allgemeiner kann ein Schwellenlöschzählstand auf einen Wert gesetzt werden, der zwischen ungefähr zwei Prozent und ungefähr fünf Prozent weniger als der Löschzählstand ist, der als die Lebensdauer eines Blocks bestimmend betrachtet wird.

[0055] Wenn in Schritt **212** festgestellt wird, dass der Löschzählstand des Blocks geringer ist als der Schwellenlöschzählstand, dann ist die Angabe, dass die Inhalte, z. B. Anwenderdaten, die im Block gespeichert werden sollen, unter Verwendung eines 1-Bit-ECC-Algorithmus codiert werden sollen. Folglich werden in Schritt **224** die Inhalte, die im Block gespeichert werden sollen, unter Verwendung eines 1-Bit-ECC-Algorithmus codiert. Sobald die Inhalte codiert sind, werden die codierten Inhalte in Schritt **228** in den Block geschrieben und der Prozess des Schreibens von Inhalten in einen Block wird beendet.

[0056] Bei Rückkehr zu Schritt **212** ist, wenn die Feststellung ist, dass der Löschzählstand nicht geringer ist als der Schwellenlöschzählstand, dann die Implikation, dass der Block sich dem Ende seiner Nutzlebensdauer nähert. An sich werden die im Block gespeicherten Inhalte in Schritt **216** unter Verwendung eines 2-Bit-ECC-Algorithmus codiert. Nachdem die Blockinhalte codiert sind, werden die codierten Inhalte dann in Schritt **220** in den Block geschrieben und der Prozess des Schreibens von Inhalten in einen Block wird beendet.

[0057] Wenn Inhalte von Blöcken, z. B. physikalischen Blöcken, innerhalb eines Speichersystems unter Verwendung von entweder 1-Bit- oder 2-Bit-ECC-Algorithmen codiert werden können, wird, bevor die Inhalte genau decodiert werden können, eine Feststellung hinsichtlich dessen durchgeführt, welcher ECC-Algorithmus verwendet wurde, um die Inhalte zu codieren. Mit anderen Worten, ein Prozess zum Lesen von Inhalten von Blöcken umfasst im Allgemeinen das Identifizieren, ob die Inhalte unter Verwendung eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert wurden. Fig. 2b ist ein Prozessablaufdiagramm, das die Schritte darstellt, die einem Verfahren zum Lesen von Inhalten aus einem Block innerhalb eines Systems, in dem die Inhalte entweder unter Verwendung eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert werden können, gemäß einer Ausführungsform der vorliegenden Erfindung zugeordnet sind. Ein Prozess **240** zum Lesen der Inhalte eines Blocks beginnt in Schritt **244**, in dem ein zu lesender Block identifiziert wird. Der Block kann im Allgemeinen von einer Gruppe von Blöcken erhalten werden, die gerade in Gebrauch sind, d. h. die gerade verwendet werden, um Informationen zu speichern. Nachdem der Block identifiziert ist, wird ein dem Block entsprechender Eintrag in einem Löschzählstandblock in Schritt **248** identifiziert. Typischerweise umfasst der Eintrag für den Block im Löschzählstandblock einen Löschzählstand für den Block.

[0058] In Schritt **252** wird festgestellt, ob der Löschzählstand des Blocks unter einem Löschzählstandschwellenwert oder dem Schwellenwert liegt, der im Wesentlichen bestimmt, wann die Inhalte eines Blocks unter Verwendung eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert werden sollen. Wenn festgestellt wird, dass der Löschzählstand des Blocks unter dem Schwellenwert liegt, dann werden die Daten im Block in Schritt **256** unter Verwendung eines 1-Bit-ECC-Algorithmus decodiert. Sobald die Daten unter Verwendung des 1-Bit-ECC-Algorithmus decodiert sind, wird der Prozess zum Lesen von Inhalten aus einem Block beendet. Wenn in Schritt **252** alternativ festgestellt wird, dass der Löschzählstand nicht unterhalb des Schwellenwerts liegt, dann ist die Angabe, dass die Daten unter Verwendung eines 2-Bit-ECC-Algorithmus co-

diert wurden. Folglich werden die Daten im Block in Schritt **260** unter Verwendung eines 2-Bit-ECC-Algorithmus decodiert. Nachdem die Daten decodiert sind, wird der Prozess zum Lesen von Daten aus einem Block beendet.

[0059] Während des Verlaufs der Operation eines Speichersystems können zum Verlängern der Lebensdauer eines nichtflüchtigen Speichers des Speichersystems statische Blöcke oder Blöcke, die selten aktualisiert oder gelöscht werden, identifiziert werden. Solche statischen Blöcke können als unwahrscheinlich aktualisiert identifiziert werden und daher können die Inhalte von solchen statischen Blöcken in Blöcke mit relativ hohen Löschzählständen kopiert werden. Durch Kopieren der Inhalte von statischen Blöcken in Blöcke mit relativ hohen Löschzählständen können die statischen Blöcke effektiv zur Verwendung zurückgeführt werden. Ferner kann sich das Speichern der Inhalte von statischen Blöcken in Blöcken mit relativ hohen Löschzählständen als effiziente Verwendung der Blöcke mit relativ hohen Löschzählständen erweisen, da die Inhalte unwahrscheinlich aktualisiert und gelöscht werden, wodurch die Lebensdauer der Blöcke mit relativ hohen Löschzählständen verlängert wird. Das Vertauschen von statischen Blöcken mit Blöcken, die relativ hohe Löschzählstände aufweisen, ist in der gleichzeitig anhängigen US-Patentanmeldung Nr. 10/281 739 erörtert.

[0060] Häufig kann das Vertauschen von statischen Blöcken mit Blöcken, die relativ hohe Löschzählstände aufweisen, durchgeführt werden, wenn ein Speichersystem initialisiert wird, z. B. wenn ein Speichersystem eingeschaltet wird, nachdem es ausgeschaltet war. Mit Bezug auf Fig. 3 wird ein Initialisierungsprozess für ein Speichersystem, in dem Inhalte von Blöcken unter Verwendung entweder eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert werden können, gemäß einer Ausführungsform der vorliegenden Erfindung beschrieben. Ein Prozess **300** zum Initialisieren eines Speichersystems umfasst das Identifizieren von statischen Blöcken in Schritt **304**. Das Identifizieren von statischen Blöcken kann im Allgemeinen das Identifizieren von Blöcken umfassen, die effektiv nicht aktiv verwendet werden. Sobald statische Blöcke identifiziert sind, werden ungenutzte Blöcke mit relativ hohen Löschzählständen in Schritt **308** identifiziert. Typischerweise umfasst das Identifizieren der ungenutzten Blöcke mit relativ hohen Löschzählständen das Identifizieren einer Menge von ungenutzten Blöcken, die die höchsten Löschzählstände aller ungenutzten Blöcke besitzen, die zu einem nichtflüchtigen Speicher des Speichersystems gehören.

[0061] Nachdem ungenutzte Blöcke mit relativ hohen Löschzählständen identifiziert sind, wird in Schritt **312** festgestellt, ob der mittlere Löschzählstand, der den Blöcken des nichtflüchtigen Speichers zugeordnet ist, größer ist als ein Löschzählstandschwellenwert. Ein mittlerer Löschzählstand gibt eine mittlere Anzahl von Malen an, die ungenutzte Blöcke innerhalb des nichtflüchtigen Speichers gelöscht wurden, und ist in der gleichzeitig anhängigen US-Patentanmeldung Nr. 10/281823 erörtert. Wenn festgestellt wird, dass der mittlere Löschzählstand nicht größer ist als der Löschzählstandschwellenwert, dann ist die Angabe, dass die Inhalte des statischen Blocks zur Speicherung in den ungenutzten Blöcken mit relativ hohen Löschzählständen unter Verwendung eines 1-Bit-ECC-Algorithmus codiert werden können. An sich geht der Prozessablauf zu Schritt **328** weiter, in dem die Inhalte der statischen Blöcke decodiert werden. Es sollte erkannt werden, dass die zum Decodieren oder Lesen aus den statischen Blöcken gehörenden Schritte die vorstehend mit Bezug auf Fig. 2b beschriebenen Schritte sein können.

[0062] Beim Decodieren der Inhalte der statischen Blöcke werden die decodierten Inhalte in Schritt **332** unter Verwendung eines 1-Bit-ECC-Algorithmus codiert. Die codierten Inhalte werden dann in die ungenutzten Blöcke mit den relativ hohen Löschzählständen in Schritt **336** kopiert oder anderweitig in diesen gespeichert und der Prozess zum Initialisieren eines Speichersystems wird beendet.

[0063] Bei Rückkehr zu Schritt **312** ist, wenn festgestellt wird, dass der mittlere Löschzählstand größer ist als der Löschzählstandschwellenwert, dann die Angabe, dass die ungenutzten Blöcke mit den relativ hohen Löschzählständen wahrscheinlich alle Löschzählstände aufweisen, die höher sind als der Löschzählstandschwellenwert. An sich geht der Prozessablauf von Schritt **312** zu Schritt **316** weiter, in dem die Inhalte der statischen Blöcke entweder unter Verwendung eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus, wie geeignet, decodiert werden. Sobald die Inhalte der statischen Blöcke decodiert sind, werden die Inhalte in Schritt **320** unter Verwendung eines 2-Bit-ECC-Algorithmus codiert. Nachdem die Inhalte codiert sind, werden die Inhalte in die Blöcke mit den relativ hohen Löschzählständen in Schritt **324** kopiert oder in diesen gespeichert und der Prozess zum Initialisieren eines Speichersystems wird beendet.

[0064] In einer Ausführungsform der vorliegenden Erfindung können zurückgewonnene Blöcke in Gebrauch sein. Das heißt, Blöcke, die vorher als unbrauchbare Blöcke betrachtet wurden, können nach einem gründlichen Testprozess zur Verwendung zurückgewonnen werden, insbesondere im Fall, dass eine Gruppe von übrigen, nutzbaren Blöcken unzureichend ist. Das Managen von Blöcken, die als unbrauchbar identifiziert werden, und das Zurückgewinnen von Blöcken, die als unbrauchbar identifiziert

sind, sind in der gleichzeitig anhängigen provisorischen US-Patentanmeldung Nr. 60/421965 erörtert. Gründliche Testprozesse können das Testen von Blöcken umfassen, die als zunehmende Defekte aufweisend identifiziert wurden, z. B. Blöcke, die vorher nutzbar waren, aber nicht mehr als nutzbar betrachtet werden, indem Testinhalte in die Blöcke unter Verwendung eines 2-Bit-ECC-Algorithmus geschrieben werden und Testinhalte aus den Blöcken unter Verwendung eines 2-Bit-ECC-Algorithmus gelesen werden. Wenn ein Block, der als einen zunehmenden Defekt aufweisend identifiziert wurde, einen gründlichen Testprozess besteht, dann kann der Block wiederum als nutzbar betrachtet werden. Wenn sie verwendet werden, wird in solche Blöcke jedoch im Allgemeinen unter Verwendung eines 2-Bit-ECC-Algorithmus geschrieben und aus diesen gelesen, um die Integrität der in den Blöcken gespeicherten Inhalte zu verbessern.

[0065] Fig. 4a ist eine schematische Blockdiagrammdarstellung eines Prozesses zum Zurückgewinnen von unbrauchbaren Blöcken gemäß einer Ausführungsform der vorliegenden Erfindung. Eine Gruppe oder eine Sammlung **402** von Blöcken **406**, die als unbrauchbar identifiziert sind, werden unter Verwendung eines Testers **410** getestet, wenn ein Versuch unternommen wird, zumindest einige der unbrauchbaren Blöcke **406** zurückzugewinnen. Die unbrauchbaren Blöcke **406** können in einem Löschzählstandblock als unbrauchbar identifiziert werden. Typischerweise werden Blöcke mit Werksdefekten wie z. B. der Block **406c** nicht auf eine mögliche Zurückgewinnung getestet, da solche Blöcke im Allgemeinen nicht zurückgewinnbar sind. Solche Blöcke, z. B. der Block **406c**, werden häufig vom Hersteller als unbrauchbar identifiziert und können auch Blöcke umfassen, die vorher in einem Zurückgewinnungsprozess versagt haben.

[0066] Blöcke mit zunehmenden Defekten wie z. B. die Blöcke **406a**, **406b** können einem Testen durch einen Tester **410** unterzogen werden, um festzustellen, ob zumindest einige solche Blöcke brauchbar sein können. Wie gezeigt, wird der Block **406a** als Block **406a'** zurückgewonnen. Durch Zurückgewinnen des Blocks **406a** als Block **406a'** ist die Angabe in der beschriebenen Ausführungsform, dass der Block **406a'** einen Codier- und Decodiertest unter Verwendung eines 2-Bit-ECC-Algorithmus bestanden hat.

[0067] Sobald ein Block zurückgewonnen wurde, kann der Block in einem Löschzählstandblock als zurückgewonnen identifiziert werden. Fig. 4b ist eine schematische Darstellung eines Abschnitts eines Löschzählstandblocks gemäß einer Ausführungsform der vorliegenden Erfindung. Ein Löschzählstandblock **440** umfasst Einträge **442**, **444**, **446**, die verschiedenen physikalischen Blöcken innerhalb eines nichtflüchtigen Speichers entsprechen. Obwohl die Anzahl von Bytes in jedem Eintrag **442**, **444**, **446** ungefähr drei ist, kann die Anzahl von Bytes, die jedem Eintrag **442**, **444**, **446** zugeordnet sind, in der beschriebenen Ausführungsform variieren.

[0068] Wie gezeigt, ist ein höchstwertiges Bit, das dem Eintrag **442** zugeordnet ist, auf "0" gesetzt, um anzugeben, dass der Block **442** brauchbar ist. Das niedrigstwertige Bit oder die niedrigstwertigen Bits, die dem Eintrag **442** zugeordnet sind, sind der Löschzählstand für den dem Eintrag **442** zugeordneten Block. Daher ist der dem Eintrag **442** zugeordnete Block als brauchbar identifiziert und weist einen Löschzählstand von 200 auf. Typischerweise kann ein HEX-Wert verwendet werden, um den Löschzählstand anzugeben, obwohl für eine leichte Erörterung der Löschzählstand als alphanumerisch gezeigt ist. Es sollte selbstverständlich sein, dass die Bits im Eintrag **442** im Allgemeinen in einer Vielfalt von verschiedenen Weisen organisiert sein können, d. h. ein anderes Bit als das höchstwertige Bit kann gesetzt werden, um anzugeben, dass der dem Eintrag **442** zugeordnete Block brauchbar ist, und andere Bits als die niedrigstwertigen Bits können auf den Löschzählstand des Blocks gesetzt werden.

[0069] Ein höchstwertiges Bit im Eintrag **444** identifiziert einen Block, der dem Eintrag **444** entspricht, als zurückgewonnen. Das niedrigstwertige Bit oder die niedrigstwertigen Bits des Eintrages **444** identifizieren den entsprechenden Block als einen Löschzählstand von 9870 aufweisend.

[0070] Wie vorher erwähnt, werden, wenn ein zurückgewonnener Block verwendet wird, um Daten zu speichern, die im zurückgewonnenen Block gespeicherten Daten im Allgemeinen unter Verwendung eines 2-Bit-ECC-Algorithmus codiert, um die Integrität der Daten ungeachtet dessen, ob der Löschzählstand des zurückgewonnenen Blocks über einem Löschzählstandschwellenwert liegt, sicherzustellen. Wenn in einen Block innerhalb eines Systems, in dem sich zurückgewonnene Blöcke befinden können, geschrieben werden soll oder aus diesem gelesen werden soll, werden daher die zurückgewonnenen Blöcke identifiziert. Mit Bezug als nächstes auf Fig. 5a wird ein Verfahren zum Schreiben in einen Block innerhalb eines Systems, in dem sich zurückgewonnene Blöcke befinden können und Daten unter Verwendung entweder eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert werden können, gemäß einer Ausführungsform der vorliegenden Erfindung beschrieben. Ein Prozess **500** zum Schreiben von Daten beginnt in Schritt **504**, in dem ein Block, in den geschrieben werden soll, identifiziert wird. Ein Eintrag, der dem Block entspricht, wird von einem Löschzählstandblock in Schritt **508** gelesen. Typischerweise umfasst der Eintrag einen Löschzählstand für den Block und eine Angabe dessen, ob

der Block zurückgewonnen wurde.

[0071] Sobald der Eintrag, der dem Block entspricht, in Schritt **508** gelesen ist, wird in Schritt **512** festgestellt, ob der Block ein zurückgewonnener Block ist. Wenn festgestellt wird, dass der Block ein zurückgewonnener Block ist, dann werden die in den Block zu schreibenden oder in diesem zu speichernden Inhalte in Schritt **516** unter Verwendung eines 2-Bit-ECC-Algorithmus codiert. Nachdem die Inhalte codiert sind, werden die codierten Inhalte in Schritt **520** in den Block geschrieben und der Prozess zum Schreiben von Inhalten in einen Block wird beendet.

[0072] Bei Rückkehr zu Schritt **512** begibt sich der Prozessablauf, wenn festgestellt wird, dass der Block kein zurückgewonnener Block ist, dann zu Schritt **524**, in dem festgestellt wird, ob der Löschzählstand für den Block, der aus dem Löschzählstandblock in Schritt **508** gelesen werden kann, geringer ist als ein Schwellenlöschzählstand. Wenn festgestellt wird, dass der Löschzählstand für den Block nicht geringer ist als Schwellenlöschzählstand, dann ist die Angabe, dass ein 2-Bit-ECC-Algorithmus verwendet werden soll, um die Inhalte zu codieren, die in dem in Schritt **504** identifizierten Block gespeichert werden sollen. An sich begibt sich der Prozessablauf von Schritt **524** zu Schritt **516**, in dem die Inhalte unter Verwendung eines 2-Bit-ECC-Algorithmus codiert werden.

[0073] Wenn alternativ in Schritt **524** festgestellt wird, dass der Löschzählstand nicht geringer ist als der Schwellenlöschzählstand, dann werden die in den identifizierten Block zu schreibenden Inhalte in Schritt **528** unter Verwendung eines 1-Bit-ECC-Algorithmus codiert. Die codierten Inhalte werden dann in Schritt **532** im Block gespeichert oder in diesen geschrieben und der Prozess zum Schreiben von Inhalten in einen Block wird beendet.

[0074] Fig. 5b ist ein Prozessablaufdiagramm, das ein Verfahren zum Lesen von Inhalten aus einem Block, der ein zurückgewonnener Block sein kann und Inhalte aufweisen kann, die entweder unter Verwendung eines 1-Bit-ECC-Algorithmus oder eines 2-Bit-ECC-Algorithmus codiert sind, gemäß einer Ausführungsform der vorliegenden Erfindung darstellt. Ein Leseprozess **550** beginnt in Schritt **554**, in dem ein Block mit zu lesenden Inhalten identifiziert wird. Ein Eintrag, der dem Block entspricht, wird dann in Schritt **558** in einem Löschzählstandblock identifiziert. Der Eintrag umfasst im Allgemeinen eine Angabe dessen, ob der Block ein zurückgewonnener Block ist, sowie einen Löschzählstand für den Block.

[0075] Nachdem der dem Block entsprechende Eintrag identifiziert ist, wird in Schritt **562** eine Feststellung hinsichtlich dessen durchgeführt, ob der Block ein zurückgewonnener Block ist. Typischerweise kann eine solche Feststellung durch Untersuchen des in Schritt **558** identifizierten oder aus dem Löschzählstandblock gelesenen Eintrages durchgeführt werden. Wenn festgestellt wird, dass der Block ein zurückgewonnener Block ist, dann werden die im Block gespeicherten Inhalte unter Verwendung eines 2-Bit-ECC-Algorithmus in Schritt **574** decodiert. Sobald die Inhalte decodiert sind, wird der Prozess zum Lesen von Daten oder Inhalten aus einem Block beendet.

[0076] Wenn alternativ in Schritt **562** festgestellt wird, dass der Block kein zurückgewonnener Block ist, dann wird in Schritt **566** festgestellt, ob der Löschzählstand des Blocks unter einem Schwellenlöschzustand liegt. Wenn festgestellt wird, dass der Löschzählstand des Blocks nicht unter dem Schwellenwert liegt, dann ist die Angabe, dass die Inhalte unter Verwendung eines 2-Bit-ECC-Algorithmus codiert wurden. An sich werden die Daten in Schritt **574** unter Verwendung eines 2-Bit-ECC-Algorithmus decodiert. Wenn andererseits festgestellt wird, dass der Löschzählstand des Blocks unter dem Schwellenwert liegt, dann werden die Daten in Schritt **570** unter Verwendung eines 1-Bit-ECC-Algorithmus decodiert. Nachdem die Daten decodiert sind, wird der Prozess des Lesens von Daten oder Inhalten aus einem Block beendet.

[0077] Im Allgemeinen ist ein Systeminitialisierungsprozess für ein System, das ein Vertauschen von statischen Blöcken ermöglicht und Blöcke umfasst, die zurückgewonnen werden können, von einem Systeminitialisierungsprozess für ein System, das das Vertauschen von statischen Blöcken ermöglicht, aber zurückgewonnene Blöcke nicht berücksichtigt, verschieden. Mit Bezug auf Fig. 6 wird ein Verfahren zum Initialisieren eines Speichersystems mit einer Hybrid-ECC-Implementierung, das zurückgewonnene Blöcke umfasst, gemäß einer Ausführungsform der vorliegenden Erfindung beschrieben. Ein Initialisierungsprozess beginnt in Schritt **604**, in dem statische Blöcke identifiziert werden. Sobald die statischen Blöcke identifiziert sind, werden ungenutzte Blöcke mit relativ hohen Löschzählständen in Schritt **608** identifiziert. Die ungenutzten Blöcke mit relativ hohen Löschzählständen sind im Allgemeinen so beschaffen, dass sie mit den statischen Blöcken vertauscht werden, so dass die statischen Blöcke zurückgeführt oder anderweitig wieder verwendet werden können, während die Inhalte der statischen Blöcke in den ungenutzten Blöcken mit relativ hohen Löschzählständen gespeichert werden.

[0078] In Schritt **612** wird eine Feststellung hinsichtlich dessen durchgeführt, ob der mittlere Löschzählstand, der dem Speichersystem zugeordnet ist und der aus einem Löschzählstandblock erhalten werden kann, größer ist als ein Löschzählstandschwellenwert. Wenn festgestellt wird, dass der mittlere Lösch-

zählstand größer ist als der Schwellenlöschzählstand, dann werden in Schritt **616** irgendwelche statischen Blöcke, die zurückgewonnene Blöcke sind, identifiziert. Eine solche Identifikation kann entweder durch Untersuchen von redundanten Bereichen, die den statischen Blöcken zugeordnet sind, oder durch Untersuchen von zugehörigen Einträgen in einem Löschzählstandblock druchgeführt werden.

[0079] Nachdem statische Blöcke, die zurückgewonnen wurden, identifiziert sind, werden die Inhalte der zurückgewonnenen statischen Blöcke in Schritt **620** unter Verwendung eines 2-Bit-ECC-Algorithmus decodiert. In Schritt **624** werden dann die Inhalte der restlichen statischen Blöcke decodiert, wie geeignet. Das heißt, die Inhalte der restlichen statischen Blöcke, die Löschzählstände aufweisen, die geringer sind als der Schwellenlöschzählstand, werden unter Verwendung eines 1-Bit-ECC-Algorithmus decodiert, während die Inhalte der restlichen statischen Blöcke, die Löschzählstände aufweisen, die größer sind als der Schwellenlöschzählstand, unter Verwendung eines 2-Bit-ECC-Algorithmus decodiert werden. Sobald alle Inhalte der statischen Blöcke decodiert sind, werden die Inhalte unter Verwendung eines 2-Bit-ECC-Algorithmus in Schritt **628** codiert und die codierten Inhalte werden in den Blöcken mit den relativ hohen Löschzählständen in Schritt **632** gespeichert oder in diese kopiert. Nach dem Kopieren der codierten Inhalte in Blöcke wird der Systeminitialisierungsprozess beendet.

[0080] Bei Rückkehr zu Schritt **612** und zur Feststellung dessen, ob ein mittlerer Löschzählstand größer als ein Schwellenlöschzählstand ist, ist, wenn festgestellt wird, dass der mittlere Löschzählstand nicht größer ist als der Schwellenlöschzählstand, dann die Angabe, dass die ungenutzten Blöcke mit relativ hohen Löschzählständen wahrscheinlich Löschzählstände aufweisen, die geringer sind als der Schwellenlöschzählstand. An sich begibt sich der Prozessablauf von Schritt **612** zu Schritt **636**, in dem ungenutzte Blöcke, die zurückgewonnene Blöcke sind, identifiziert werden. Das heißt, die ungenutzten Blöcke mit relativ hohen Löschzählständen, die vorher zurückgewonnen wurden, werden identifiziert. Sobald die ungenutzten Blöcke mit relativ hohen Löschzählständen, die vorher zurückgewonnen wurden, identifiziert sind, werden irgendwelche statischen Blöcke, die zurückgewonnene Blöcke sind, in Schritt **640** identifiziert.

[0081] Von Schritt **640** geht der Prozessablauf zu Schritt **644** weiter, in dem die Inhalte von zurückgewonnenen statischen Blöcken unter Verwendung eines 2-Bit-ECC-Algorithmus decodiert werden. Nachdem die Inhalte der zurückgewonnenen statischen Blöcke decodiert sind, werden die Inhalte der restlichen statischen Blöcke, wie geeignet, in Schritt **648** decodiert. Typischerweise können einige der statischen Blöcke unter Verwendung eines 1-Bit-ECC-Algorithmus decodiert werden, während andere statische Blöcke unter Verwendung eines 2-Bit-ECC-Algorithmus decodiert werden können. Beim Decodieren der Inhalte der restlichen statischen Blöcke werden irgendwelche Inhalte, die in zurückgewonnenen ungenutzten Blöcken gespeichert werden sollen, in Schritt **652** unter Verwendung eines 2-Bit-ECC-Algorithmus codiert. In Schritt **656** werden die Inhalte, die in ungenutzten Blöcken gespeichert werden sollen, die vorher nicht zurückgewonnen wurden, unter Verwendung eines 1-Bit-ECC-Algorithmus codiert. Sobald im Wesentlichen alle Inhalte codiert wurden, werden die codierten Inhalte in ihre vorgesehenen ungenutzten Blöcke mit relativ hohen Löschzählständen in Schritt **660** kopiert und der Initialisierungsprozess wird beendet.

[0082] Im Allgemeinen ist die Funktionalität, die dem Implementieren einer Hybrid-ECC-Implementierung zugeordnet ist, in der Software, z. B. als Programmcodevorrichtungen, oder als Firmware für ein Hauptrechnersystem, das einen nichtflüchtigen Speicher oder eine nichtflüchtige Speicherkomponente umfasst, vorgesehen. Eine Ausführungsform einer geeigneten Systemarchitektur, die der Software oder Firmware zugeordnet ist, die für ein Hauptrechnersystem vorgesehen ist, ist in Fig. 7 gezeigt. Eine Systemarchitektur **700** umfasst im Allgemeinen eine Vielzahl von Modulen, die umfassen können, jedoch nicht begrenzt sind auf ein Anwendungsschnittstellenmodul **704**, ein Systemmanagermodul **708**, ein Datenmanagermodul **712**, einen Datenintegritätsmanager **716** und einen Vorrichtungsmanager und ein Schnittstellenmodul **720**. Im Allgemeinen kann die Systemarchitektur **700** unter Verwendung von Softwarecodevorrichtungen oder Firmware implementiert werden, auf die durch einen Prozessor, z. B. den Prozessor **108** von Fig. 1a, zugegriffen werden kann.

[0083] Im Allgemeinen kann das Anwendungsschnittstellenmodul **704** so beschaffen sein, dass es mit einem nichtflüchtigen Speicher wie z. B. einem Flash-Speicher (nicht dargestellt) oder allgemeiner Medien kommuniziert, um die Medien während des Verlaufs einer Initialisierungs- oder Systemformatierungsanforderung zu initialisieren. Das Anwendungsschnittstellenmodul **704** kann auch aus einem Sektor, Cluster oder einer Seite, die den Medien zugeordnet sind, lesen sowie in diese schreiben. Zusätzlich zur Kommunikation mit Medien steht das Anwendungsschnittstellenmodul **704** typischerweise auch mit dem Systemmanagermodul **708** und dem Datenmanagermodul **712** in Kommunikation.

[0084] Das Systemmanagermodul **708** umfasst ein Systeminitialisierungsuntermodul **724**, ein Löschzählstandblock-Managementuntermodul **726** und ein Leistungsmanagementblock-Untermodul **730**. Das Systeminitialisierungsmodul **724** ist im Allgemeinen

so beschaffen, dass es ermöglicht, dass eine Initialisierungsanforderung verarbeitet wird, und kommuniziert typischerweise mit dem Löschzählstandblock-Managementuntermodul **726**. Das Systeminitialisierungsmodul **724** ist auch so beschaffen, dass es eine logisch-physikalische Blockzuweisung von einem zu vielen auflöst.

[0085] Das Löschzählstandblock-Managementuntermodul **726** umfasst eine Funktionalität, um zu bewirken, dass Löschzählstände von Blöcken gespeichert werden, und eine Funktionalität, um zu bewirken, dass ein mittlerer Löschzählstand unter Verwendung von einzelnen Löschzählständen berechnet sowie aktualisiert wird. Mit anderen Worten, das Löschzählstandblock-Managementuntermodul **726** ermöglicht effektiv, dass Löschzählstände katalogisiert werden, und ermöglicht, dass ein mittlerer Löschzählstand gewartet wird. In einer Ausführungsform synchronisiert das Löschzählstandblock-Managementuntermodul **726** ferner auch im Wesentlichen den Löschzählstand von im Wesentlichen allen Blöcken in einem Löschzählstandblock während einer Initialisierungsanforderung eines Gesamtsystems. Obwohl das Löschzählstandblock-Managementuntermodul **726** so beschaffen sein kann, dass es bewirkt, dass ein mittlerer Löschzählstand in einem Löschzählstandblock gespeichert wird, sollte erkannt werden, dass das Leistungsmanagementblock-Untermodul **730** statt dessen verwendet werden kann, um zu ermöglichen, dass der mittlere Löschzählstand gespeichert wird.

[0086] Zusätzlich dazu, dass es mit dem Anwendungsschnittstellenmodul **704** in Kommunikation steht, steht das Systemmanagermodul **708** auch mit dem Datenmanagermodul **712** sowie dem Vorrichtungsmanager- und Schnittstellenmodul **720** in Kommunikation. Das Datenmanagermodul **712**, das mit sowohl dem Systemmanagermodul **708** als auch dem Anwendungsschnittstellenmodul **704** kommuniziert, kann eine Funktionalität umfassen, um eine Sektorabbildung bereitzustellen, die effektiv logische Sektoren in physikalische Sektoren umsetzt. Das heißt, das Datenmanagermodul **712** ist dazu beschaffen, logische Blöcke in physikalische Blöcke abzubilden. Das Datenmanagermodul **712** kann auch eine Funktionalität umfassen, die den Betriebssystem- und Dateisystem-Schnittstellenschichten zugeordnet ist, und ermöglicht, dass Gruppen innerhalb Blöcken gemanagt werden, wie in der gleichzeitig anhängigen US-Patentanmeldung Nr. 10/281 855 beschrieben. In einer Ausführungsform kann das Datenmanagermodul **712** dazu beschaffen sein zu ermöglichen, dass ein im Wesentlichen sequenzexterner Schreibprozess geschieht.

[0087] Das Vorrichtungsmanager- und Schnittstellenmodul **720**, das mit dem Systemmanagermodul **708**, dem Datenmanager **712** und dem Datenintegritätsmanager **716** in Kommunikation steht, stellt typischerweise eine Flash-Speicher-Schnittstelle bereit und umfasst eine Funktionalität, die Hardwareabstraktionen zugeordnet ist, z. B. eine E/A-Schnittstelle. Das Datenintegritätsmanagermodul **716** sieht unter anderen Funktionen eine ECC-Bearbeitung vor.

[0088] Obwohl nur einige Ausführungsformen der vorliegenden Erfindung beschrieben wurden, sollte es selbstverständlich sein, dass die vorliegende Erfindung in vielen anderen speziellen Formen verkörpert werden kann, ohne vom Gedanken oder Schutzbereich der vorliegenden Erfindung abzuweichen. Beispielsweise wurde ein 1-Bit-ECC-Algorithmus als geeignet zur Verwendung zum Codieren von Inhalten eines Blocks mit einem Löschzählstand, der niedriger ist als ein Löschzählstandschwellenwert, beschrieben, während ein 2-Bit-ECC-Algorithmus als geeignet zur Verwendung zum Codieren von Inhalten eines Blocks mit einem Löschzählstand, der höher ist als ein Löschzählstandschwellenwert, beschrieben wurde. Es sollte jedoch erkannt werden, dass in einigen Ausführungsformen ein 2-Bit-ECC-Algorithmus verwendet werden kann, um Inhalte eines Blocks mit einem Löschzählstand zu codieren, der niedriger ist als ein Löschzählstandschwellenwert, und ein Algorithmus mit noch höherer Genauigkeit als ein 2-Bit-ECC-Algorithmus, z. B. ein 3-Bit-ECC-Algorithmus, verwendet werden kann, um Inhalte eines Blocks mit einem Löschzählstand, der höher ist als der Löschzählstandschwellenwert, zu codieren.

[0089] Obwohl ECC-Algorithmen im Allgemeinen als 1-Bit-ECC-Algorithmen oder 2-Bit-ECC-Algorithmen beschrieben wurden, können ECC-Algorithmen außerdem statt dessen 1-Symbol-ECC-Algorithmen bzw. 2-Symbol-ECC-Algorithmen sein. Ferner können die tatsächlichen verwendeten ECC-Algorithmen umfangreich variieren. Geeignete ECC-Algorithmen können umfassen, sind jedoch nicht begrenzt auf Reed-Solomon-Algorithmen, Hamming-Code-Algorithmen und binäre Hamming-Code-Algorithmen. In einer Ausführungsform kann beispielsweise ein geeigneter 1-Bit-ECC-Algorithmus ein Hamming-Code-Algorithmus sein, während ein geeigneter 2-Bit-ECC-Algorithmus ein Reed-Solomon-Algorithmus sein kann.

[0090] Die zu den verschiedenen Verfahren der vorliegenden Erfindung gehörenden Schritte können umfangreich verändert werden. Im Allgemeinen können Schritte hinzugefügt, entfernt, umgeordnet und geändert werden, ohne vom Gedanken oder Schutzbereich der vorliegenden Erfindung abzuweichen. Die Prozesse der Initialisierung eines Speichersystems, das eine Hybrid-ECC-Implementierung umfasst, können beispielsweise das Löschen von statischen Blöcken umfassen. Für Ausführungsformen, in denen zurückgewonnene Blöcke verwendet werden können, um Daten zu speichern, kann außerdem, be-

vor festgestellt wird, ob ein Block zurückgewonnen wurde, zuerst festgestellt werden, ob der Löschzählstand für den fraglichen Block einen Löschzählstandschwellenwert überschreitet. Daher sollen die vorliegenden Beispiele als erläuternd und nicht einschränkend betrachtet werden und die Erfindung soll nicht auf die hierin angegebenen Details begrenzt sein, sondern kann innerhalb des Schutzbereichs der beigefügten Ansprüche modifiziert werden.

[0091] Jedes in dieser Patentbeschreibung (wobei der Begriff die Ansprüche einschießt) offenbarte und/oder in den Zeichnungen gezeigte Merkmal kann in die Erfindung unabhängig von anderen offenbarten und/oder dargestellten Merkmalen integriert werden.

[0092] Der Text der hiermit eingereichten Zusammenfassung wird hier als Teil der Patentbeschreibung wiederholt.

[0093] Verfahren und Vorrichtungen zur Verwendung von verschiedenen Fehlerkorrekturcode-Algorithmen zum Codieren und Decodieren von Inhalten von Blöcken innerhalb eines nichtflüchtigen Speichers werden offenbart. Gemäß einem Aspekt der vorliegenden Erfindung umfasst ein Verfahren zum Speichern von Daten innerhalb eines nichtflüchtigen Speichers das Identifizieren eines ersten Blocks, in dem die Daten gespeichert werden sollen, und das Erhalten eines Indikators, der angibt, dass die Daten unter Verwendung eines ersten Algorithmus codiert werden sollen. Die Daten werden unter Verwendung des ersten Algorithmus codiert, wenn die Verwendung des ersten Algorithmus festgestellt wird, nach welchem Punkt die unter Verwendung des ersten Algorithmus codierten Daten in den ersten Block geschrieben werden.

QUERVERWEIS AUF VERWANDTE ANMELDUNGEN

[0094] Die vorliegende Erfindung betrifft die gleichzeitig anhängigen US-Patentanmeldungen Nrn. 10/281 739, 10/281 823, 10 281 670, 10/281 824, 10/281 631, 10/281 855, 10/281 762, 10/281 696, 10/281 626 und 10/281 804 sowie die gleichzeitig anhängigen vorläufigen US-Patentanmeldungen Nrn. 60/421 910, 60/421 725, 60/421 965, 60/422 166, 60/421 746 und 60/421 911, jeweils eingereicht am 28. Oktober 2002.

Patentansprüche

1. Verfahren zum Speichern von Daten in einem nichtflüchtigen Speicher (**124**; **174**) eines Speichersystems, wobei das Verfahren umfasst:
Identifizieren (**204**) eines ersten Blocks, in dem die Daten gespeichert werden sollen;
und
dadurch gekennzeichnet, dass das Verfahren ferner umfasst:
Erhalten (**208**) eines dem ersten Block zugeordneten Indikators, wobei der Indikator einen Wert besitzt, der die Zuverlässigkeit des ersten Blocks angibt;
in Reaktion (**212**) darauf, dass der dem ersten Block zugeordnete Indikator ein Kriterium erfüllt, Codieren (**224**) der zu speichernden Daten unter Verwendung eines ersten Fehlerdetektionsalgorithmus und Schreiben (**228**) der codierten Daten unter Verwendung des ersten Fehlerdetektionsalgorithmus in den ersten Block; und
in Reaktion (**212**) darauf, dass der dem ersten Block zugeordnete Indikator das Kriterium nicht erfüllt, Codieren (**216**) der zu speichernden Daten unter Verwendung eines zweiten Fehlerdetektionsalgorithmus, wobei der zweite Fehlerdetektionsalgorithmus eine höhere Fehlerdetektionsfähigkeit als der erste Fehlerdetektionsalgorithmus hat, und Schreiben (**220**) der codierten Daten unter Verwendung des zweiten Fehlerdetektionsalgorithmus in den ersten Block.

2. Verfahren nach Anspruch 1, ferner dadurch gekennzeichnet, dass der Indikator einen Zählstand enthält, der der Anzahl entspricht, in der der Block gelöscht worden ist; wobei das Kriterium den Indikator, der einen Zählstand besitzt, der kleiner als ein Schwellenwert ist, umfasst.

3. Verfahren nach Anspruch 1 oder Anspruch 2; ferner dadurch gekennzeichnet, dass der Indikator angibt, ob der Block ein zurückgewonnener Block ist, wobei das Kriterium den Indikator, der angibt, dass der Block kein zurückgewonnener Block ist, umfasst.

4. Verfahren nach Anspruch 1, das ferner umfasst:
Identifizieren (**244**) des ersten Blocks als einen Block, aus dem Daten gelesen werden sollen;
Erhalten des dem ersten Block zugeordneten Indikators;
in Reaktion (**252**) auf die Tatsache, dass der Indikator das Kriterium erfüllt, Decodieren (**256**) der Daten unter Verwendung des ersten Fehlerdetektionsalgorithmus; und
in Reaktion (**252**) auf die Tatsache, dass der Indikator das Kriterium nicht erfüllt, Decodieren (**260**) der Daten unter Verwendung des zweiten Fehlerdetektionsalgorithmus.

5. Verfahren nach Anspruch 1 oder Anspruch 4, bei dem der erste Fehlerdetektionsalgorithmus ein 1-Bit-ECC-Algorithmus ist und der zweite Fehlerdetektionsalgorithmus ein 2-Bit-ECC-Algorithmus ist.

6. Verfahren nach Anspruch 4, ferner dadurch gekennzeichnet, dass der Indikator angibt, ob der erste Block ein zurückgewonnener Block ist,
wobei das Kriterium den Indikator, der angibt, dass der erste Block kein zurückgewonnener Block ist,

umfasst; und
wobei der Indikator dann, wenn der erste Block ein zurückgewonnener Block ist, ferner so beschaffen ist, dass er angibt, dass die Daten unter Verwendung des zweiten Fehlerdetektionsalgorithmus codiert worden sind.

7. Verfahren nach Anspruch 4, ferner dadurch gekennzeichnet, dass der Indikator einen Zählstand enthält, der der Anzahl entspricht, in der der Block gelöscht worden ist; wobei das Kriterium den Indikator, der einen Zählstand besitzt, der kleiner als ein Schwellenwert ist, umfasst.

8. Verfahren nach Anspruch 2 oder Anspruch 7, bei dem der Indikator ferner einen Wert umfasst, der der durchschnittlichen Anzahl entspricht, in der physikalische Blöcke des nichtflüchtigen Speichers gelöscht worden sind.

9. Verfahren nach Anspruch 1 oder Anspruch 4, bei dem der Indikator in einer von dem ersten Block getrennten Datenstruktur gespeichert wird und bei dem das Erhalten des dem Block zugeordneten Indikators das Erhalten des Indikators aus der Datenstruktur umfasst.

10. Verfahren nach Anspruch 1 oder Anspruch 4, bei dem der nichtflüchtige Speicher ein Flash-Speicher ist.

11. Verfahren nach Anspruch 1 oder Anspruch 4, bei dem der nichtflüchtige Speicher entweder ein NAND-Flash-Speicher oder ein MLC-NAND-Flash-Speicher ist.

12. Verfahren nach Anspruch 1 oder Anspruch 4, bei dem sowohl der erste als auch der zweite Fehlerdetektionsalgorithmus einen Fehlerdetektions- und einen Fehlerkorrekturalgorithmus enthält.

13. Speichersystem (**128**; **700**), das umfasst:
einen nichtflüchtigen Speicher (**124**; **174**), der einen ersten Block enthält, in dem Daten gespeichert werden sollen; und
Mittel, die den genannten Block identifizieren;
wobei das Speichersystem gekennzeichnet ist durch:
Mittel (**724**), die einen dem ersten Block zugeordneten Indikator enthalten, wobei der Indikator einen Wert hat, der die Zuverlässigkeit des ersten Blocks angibt;
Mittel, die in Reaktion darauf, dass der dem ersten Block zugeordnete Indikator ein Kriterium erfüllt, die zu speichernden Daten unter Verwendung eines ersten Fehlerdetektionsalgorithmus codieren;
Mittel, die in Reaktion darauf, dass der Indikator das Kriterium nicht erfüllt, die zu speichernden Daten unter Verwendung eines zweiten Fehlerdetektionsalgorithmus, wobei der zweite Fehlerdetektionsalgorithmus eine höhere Fehlerdetektionsfähigkeit als der erste Fehlerdetektionsalgorithmus besitzt, codieren; und
Mittel, die die codierten Daten in den ersten Block schreiben.

14. Speichersystem nach Anspruch 13, bei dem die Identifizierungsmittel, die Erhaltungsmittel, die Codierungsmittel und die Schreibmittel jeweils Codevorrichtungen sind,
wobei der nichtflüchtige Speicher ferner mehrere Blöcke einschließlich des ersten Blocks enthält;
und ferner gekennzeichnet durch:
einen Speicherbereich, der die Codevorrichtungen speichert.

15. Speichersystem nach Anspruch 13, bei dem der Indikator einen Zählstand enthält, der einer Anzahl entspricht, in der der Block gelöscht worden ist; und wobei das Kriterium den Indikator, der einen Zählstand besitzt, der kleiner als ein Schwellenwert ist, umfasst.

16. Speichersystem nach Anspruch 13 oder Anspruch 15, bei dem der Indikator angibt, ob der Block ein zurückgewonnener Block ist, wobei das Kriterium den Indikator, der angibt, dass der Block kein zurückgewonnener Block ist, umfasst.

17. Speichersystem nach Anspruch 15, das umfasst:
Codevorrichtungen, um zu bestimmen, ob der Indikator einen Zählstand besitzt, der kleiner als der Schwellenwert ist.

18. Speichersystem nach Anspruch 13, bei dem der nichtflüchtige Speicher mehrere Blöcke enthält, wobei die Blöcke den ersten Block und einen zweiten Block umfassen, wobei der erste Block eine erste Menge von Inhalten enthält, die unter Verwendung des ersten Fehlerdetektionsalgorithmus codiert worden sind, und der zweite Block eine zweite Menge von Inhalten enthält, die unter Verwendung des zweiten Fehlerdetektionsalgorithmus codiert worden sind,
wobei der nichtflüchtige Speicher ferner eine Datenstruktur enthält, die so beschaffen ist, dass sie angibt, dass die erste Menge von Inhalten unter Verwendung des ersten Fehlerdetektionsalgorithmus codiert worden ist und dass die zweite Menge von Inhalten unter Verwendung des zweiten Fehlerdetektionsalgorithmus codiert worden ist;
und wobei das System ferner umfasst:
Codevorrichtungen, um auf die Datenstruktur zuzugreifen, wobei die Codevorrichtungen zum Zugreifen auf die Datenstruktur Codevorrichtungen, die bestimmen, dass die erste Menge von Inhalten unter Verwendung des ersten Fehlerdetektionsalgorithmus codiert worden ist, sowie Codevorrichtungen, die bestimmen, dass die zweite Menge von Inhalten unter Verwendung des zweiten Fehlerdetektionsalgorithmus codiert worden ist, umfassen; und einen

Speicherbereich, der die Codevorrichtungen speichert.

19. Speichersystem nach Anspruch 13, das umfasst:
Mittel, die den dem ersten Block zugeordneten Indikator erhalten;
Mittel, die in Reaktion auf die Tatsache, dass der Indikator das Kriterium erfüllt, die Daten von dem ersten Block unter Verwendung des ersten Fehlerdetektionsalgorithmus decodieren; und
Mittel, die in Reaktion auf die Tatsache, dass der Indikator das Kriterium nicht erfüllt, die Daten von dem ersten Block unter Verwendung des zweiten Fehlerdetektionsalgorithmus decodieren.

20. Speichersystem nach Anspruch 19, das umfasst:
den nichtflüchtigen Speicher, der mehrere Blöcke enthält, wobei die mehreren Blöcke den ersten Block enthalten, wobei der erste Block Daten enthält;
wobei die Erhaltungsmittel und die Decodierungsmittel Codevorrichtungen sind; und
das System ferner einen Speicherbereich aufweist, der die Codevorrichtungen speichert.

21. Speichersystem nach Anspruch 19 oder Anspruch 20, bei dem der Indikator angibt, ob der Block ein zurückgewonnener Block ist; wobei das Kriterium den Indikator, der angibt, dass der Block kein zurückgewonnener Block ist, umfasst.

22. Speichersystem nach Anspruch 19, bei dem der Indikator einen Zählstand enthält, der einer Anzahl entspricht, in der der Block gelöscht worden ist, wobei das Kriterium den Indikator, der einen Zählstand besitzt, der kleiner als ein Schwellenwert ist, umfasst.

23. Speichersystem nach Anspruch 22, das umfasst:
Codevorrichtungen, um zu bestimmen, ob der Indikator einen Zählstand besitzt, der kleiner als der Schwellenwert ist.

24. Speichersystem nach Anspruch 13 oder Anspruch 19, bei dem der erste Fehlerdetektionsalgorithmus ein 1-Bit-ECC-Algorithmus ist und der zweite Fehlerdetektionsalgorithmus ein 2-Bit-ECC-Algorithmus ist.

25. Speichersystem nach Anspruch 13 oder Anspruch 19, bei dem der nichtflüchtige Speicher entweder ein NAND-Flash-Speicher oder ein MLC-NAND-Flash-Speicher ist.

Es folgen 10 Blatt Zeichnungen

Anhängende Zeichnungen

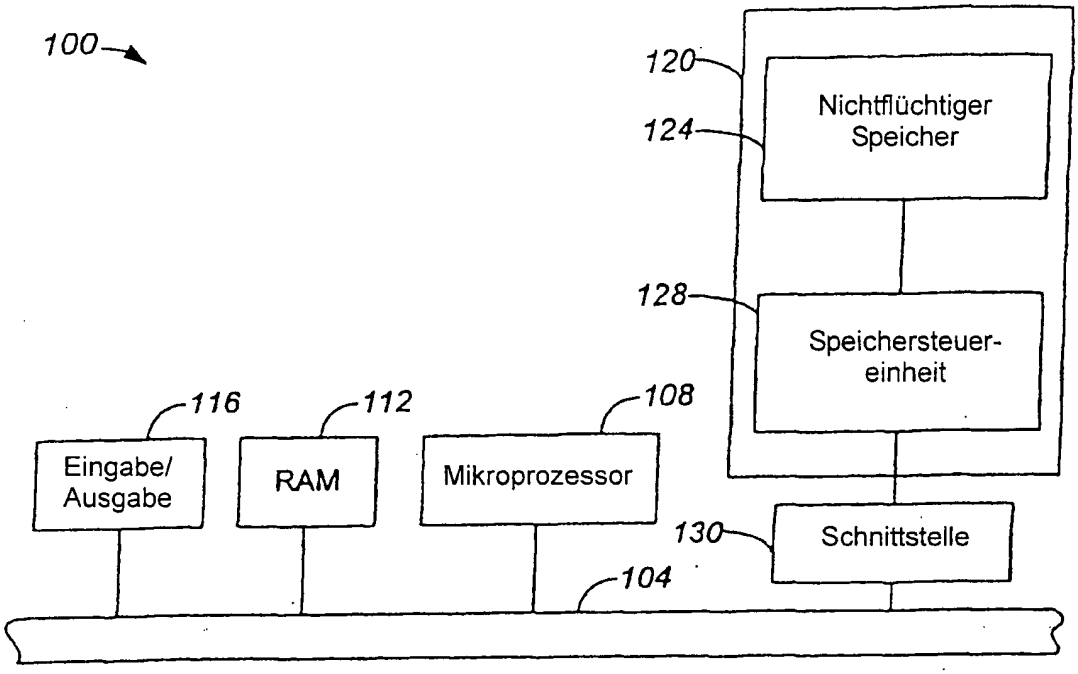

Fig. 1a

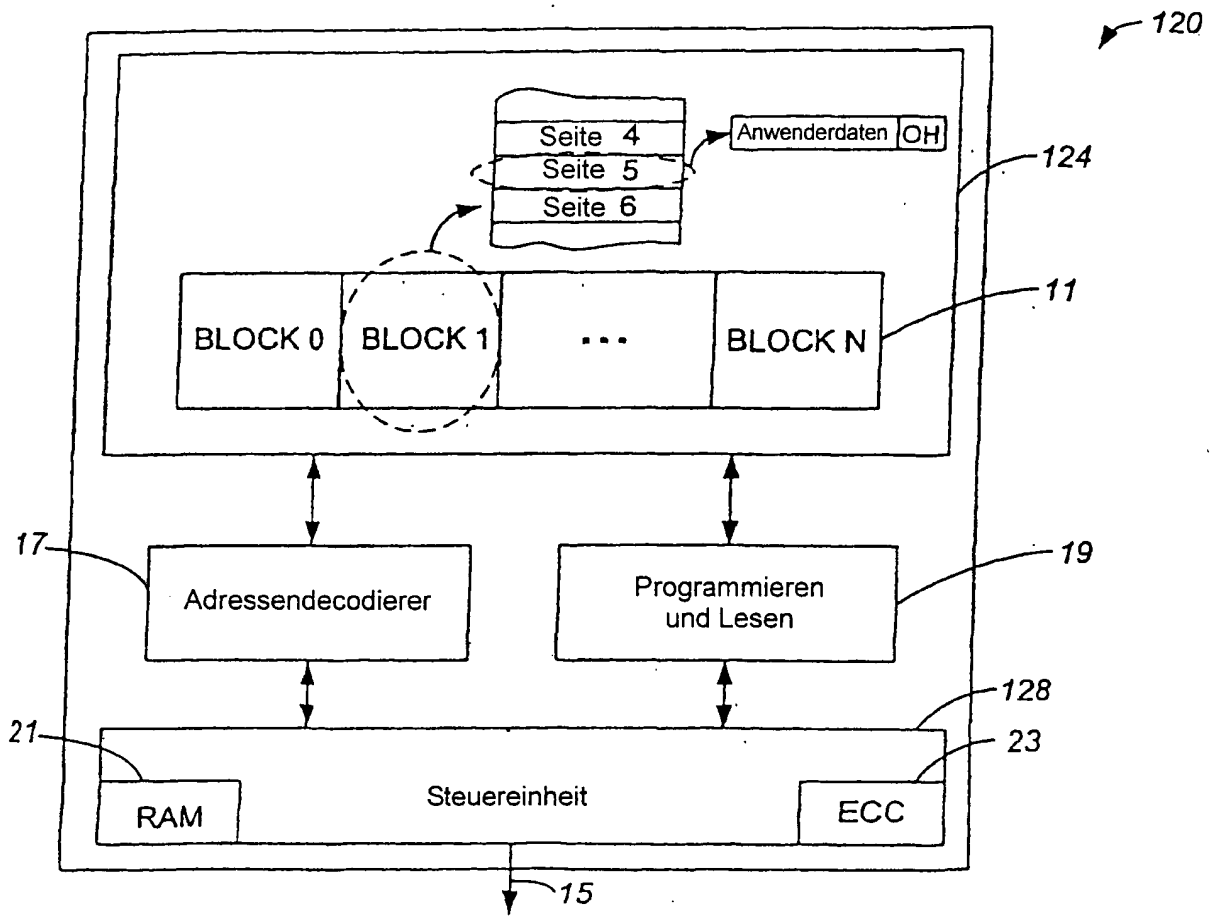

Fig. 1b

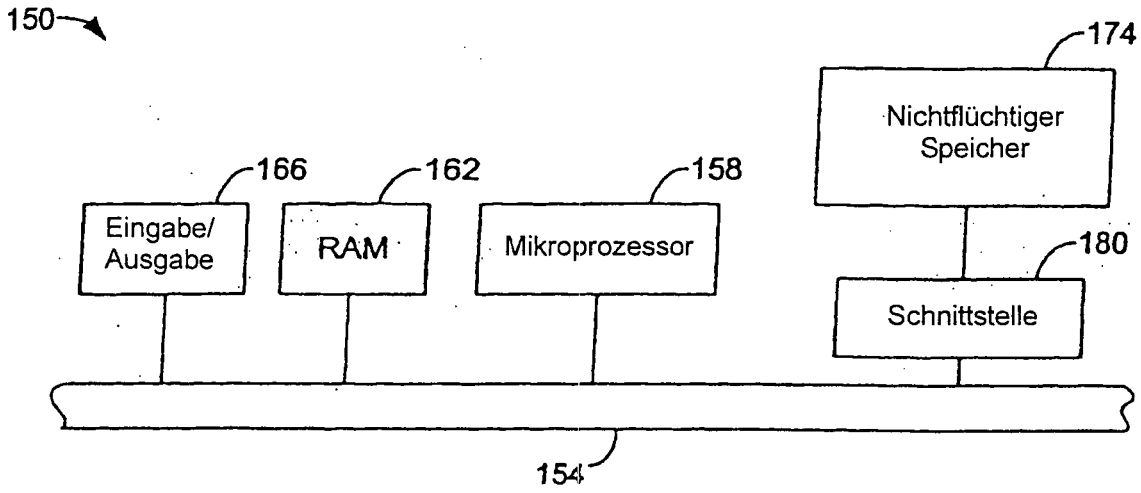
150
174
Nichtflüchtiger Speicher
166
162
158
Eingabe/ Ausgabe
RAM
Mikroprozessor
180
Schnittstelle
154

Fig. 1c

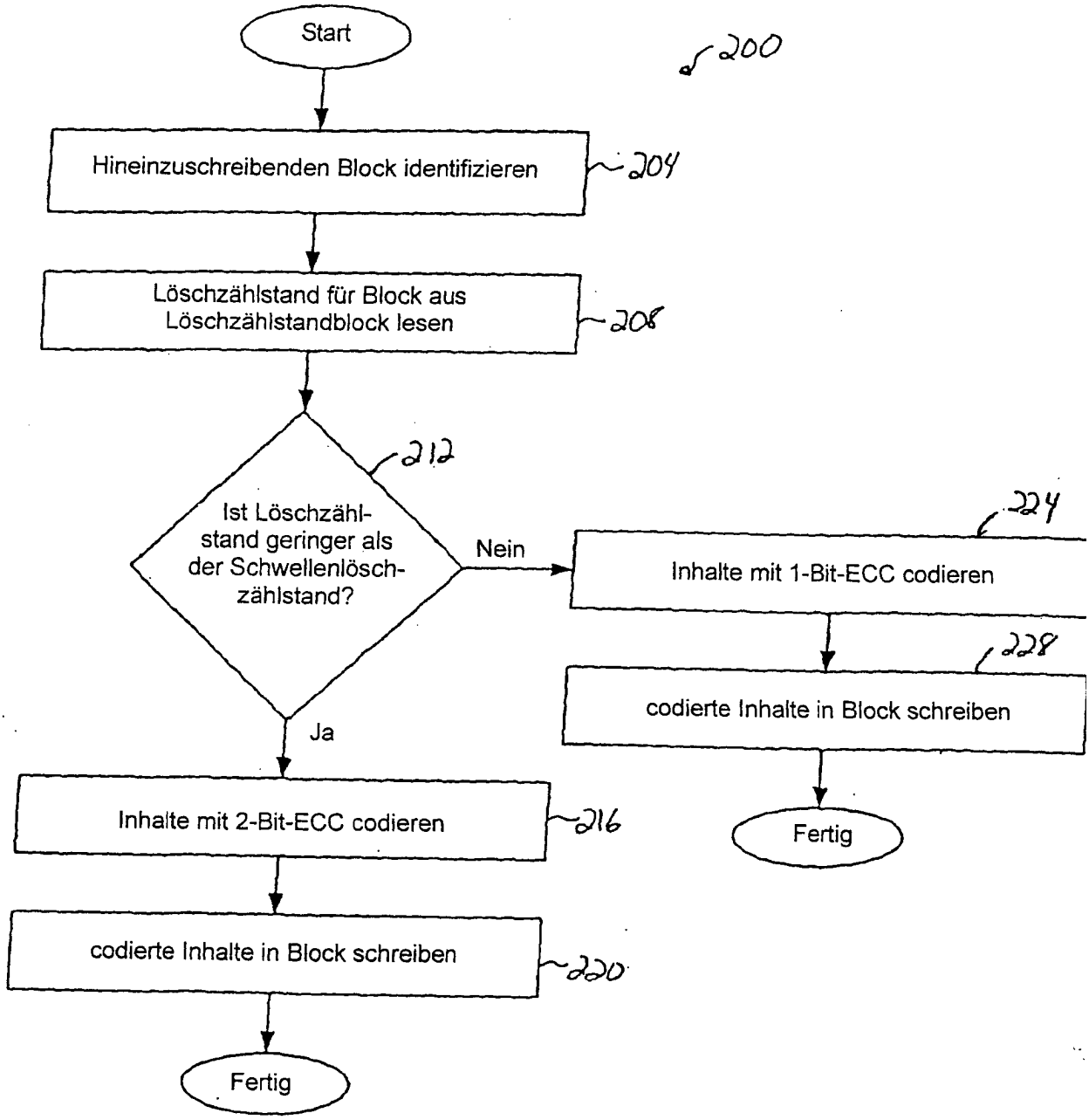
Start
200
Hineinzuschreibenden Block identifizieren
204
Löschzählstand für Block aus Löschzählstandblock lesen
208
212
Ist Löschzähl-stand geringer als der Schwellenlösch-zählstand?
Nein
224
Inhalte mit 1-Bit-ECC codieren
228
codierte Inhalte in Block schreiben
Fertig
Ja
Inhalte mit 2-Bit-ECC codieren
216
codierte Inhalte in Block schreiben
220
Fertig

Fig. 2a

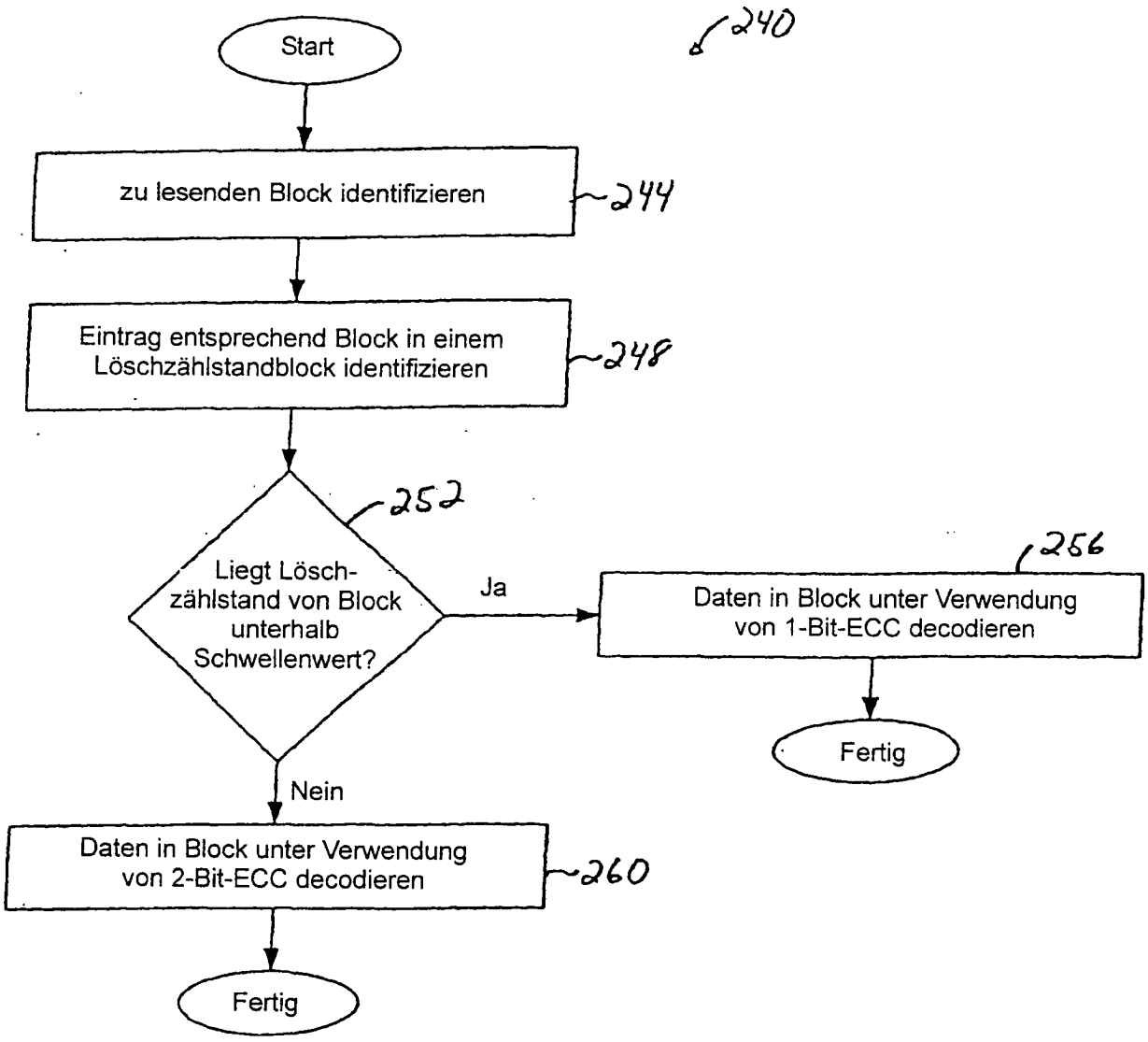
240
Start
zu lesenden Block identifizieren
244
Eintrag entsprechend Block in einem Löschzählstandblock identifizieren
248
252
Liegt Lösch-zählstand von Block unterhalb Schwellenwert?
Ja
256
Daten in Block unter Verwendung von 1-Bit-ECC decodieren
Fertig
Nein
Daten in Block unter Verwendung von 2-Bit-ECC decodieren
260
Fertig

Fig. 2b

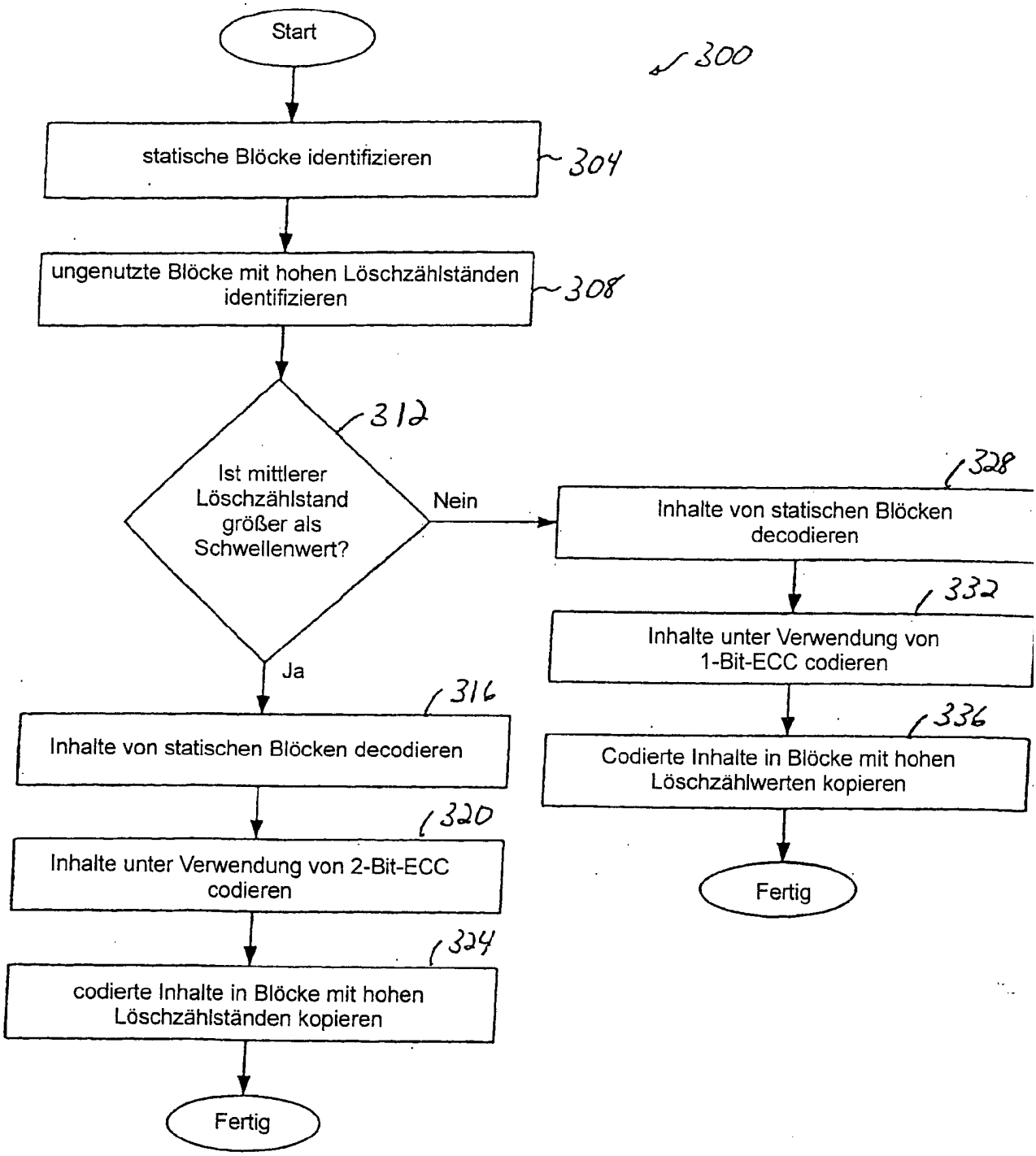
Start
300
statische Blöcke identifizieren
304
ungenutzte Blöcke mit hohen Löschzählständen identifizieren
308
312
Ist mittlerer Löschzählstand größer als Schwellenwert?
Nein
328
Inhalte von statischen Blöcken decodieren
332
Inhalte unter Verwendung von 1-Bit-ECC codieren
336
Codierte Inhalte in Blöcke mit hohen Löschzählwerten kopieren
Fertig
Ja
316
Inhalte von statischen Blöcken decodieren
320
Inhalte unter Verwendung von 2-Bit-ECC codieren
324
codierte Inhalte in Blöcke mit hohen Löschzählständen kopieren
Fertig

Fig. 3

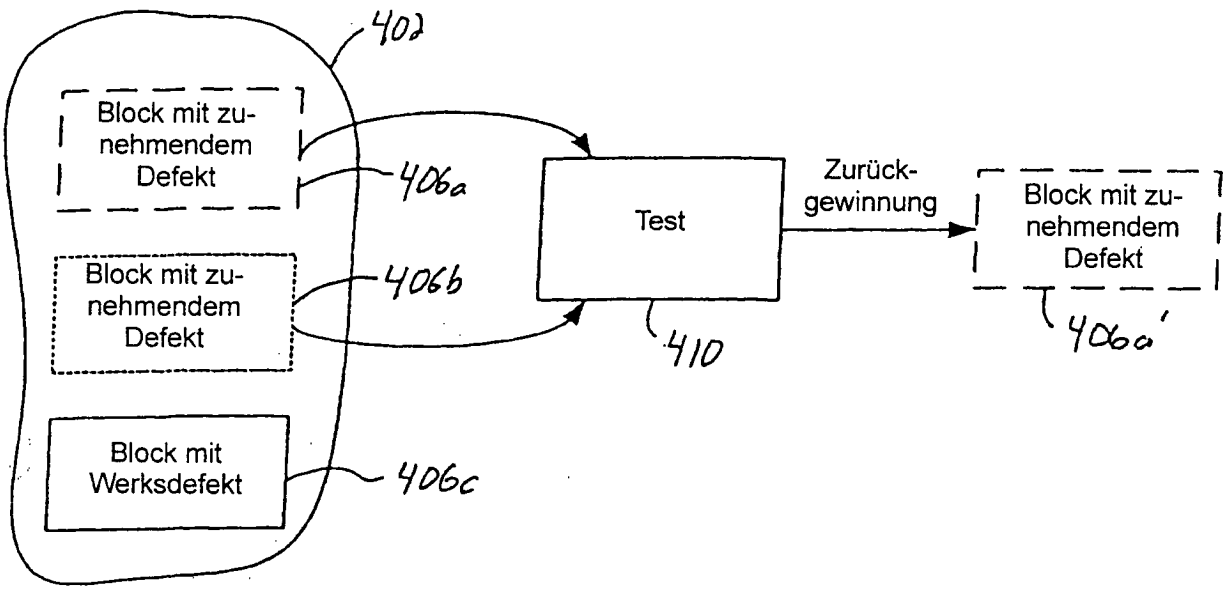
402
Block mit zu-nehmendem Defekt
406a
Block mit zu-nehmendem Defekt
406b
Block mit Werksdefekt
406c
Test
410
Zurück-gewinnung
Block mit zu-nehmendem Defekt
406a'

Fig. 4a

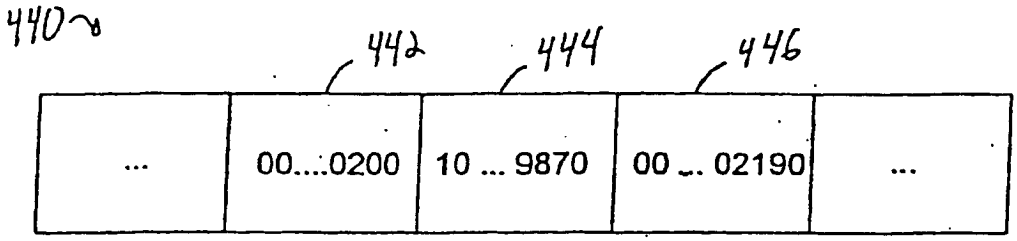
440
442
444
446
...
00...0200
10 ... 9870
00 ... 02190
...

Fig. 4b

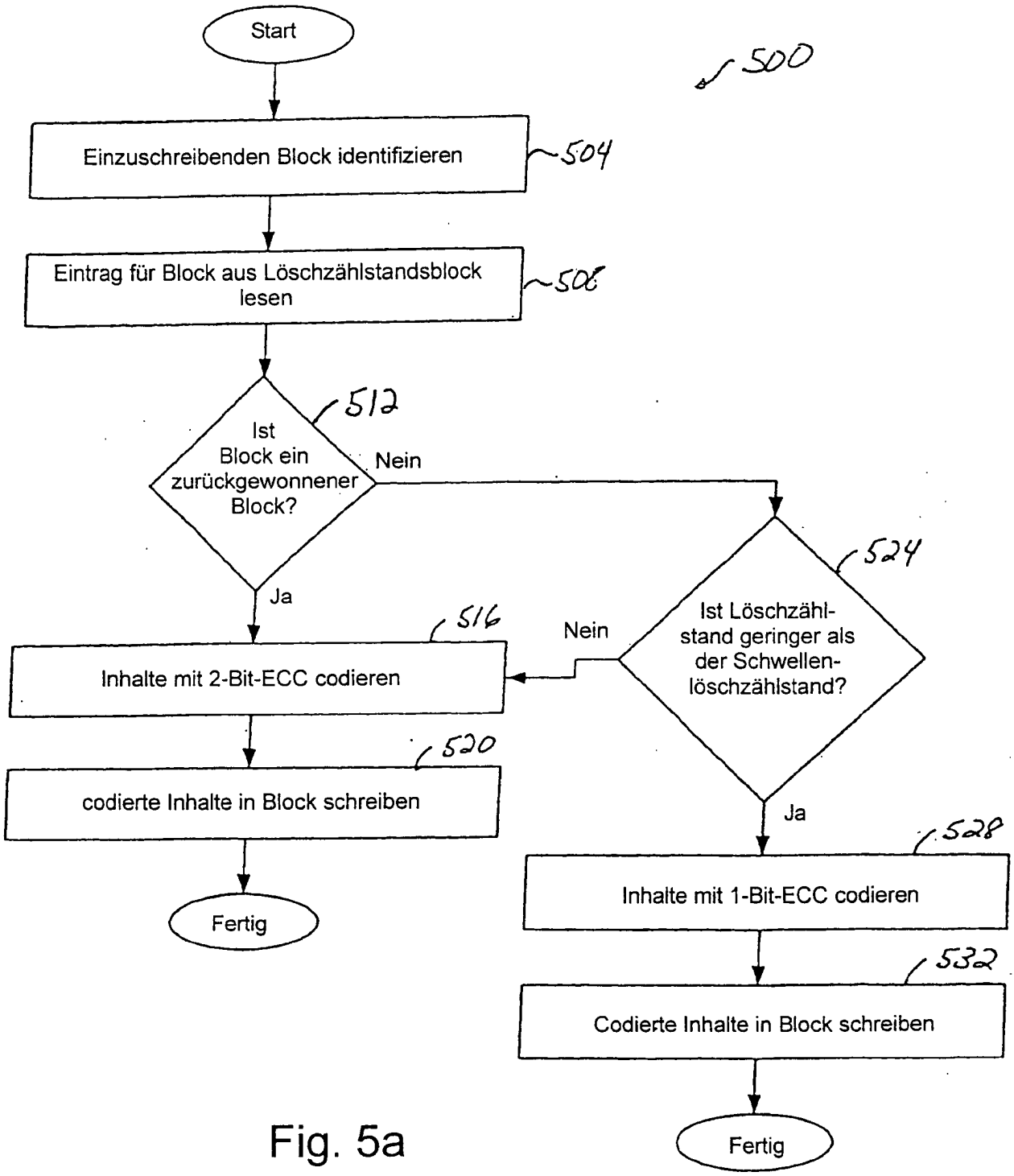
Start
500
Einzuschreibenden Block identifizieren
504
Eintrag für Block aus Löschzählstandsblock lesen
506
512
Ist Block ein zurückgewonnener Block?
Nein
Ja
524
Ist Löschzähl-stand geringer als der Schwellen-löschzählstand?
Nein
516
Inhalte mit 2-Bit-ECC codieren
520
codierte Inhalte in Block schreiben
Fertig
Ja
528
Inhalte mit 1-Bit-ECC codieren
532
Codierte Inhalte in Block schreiben
Fertig

Fig. 5a

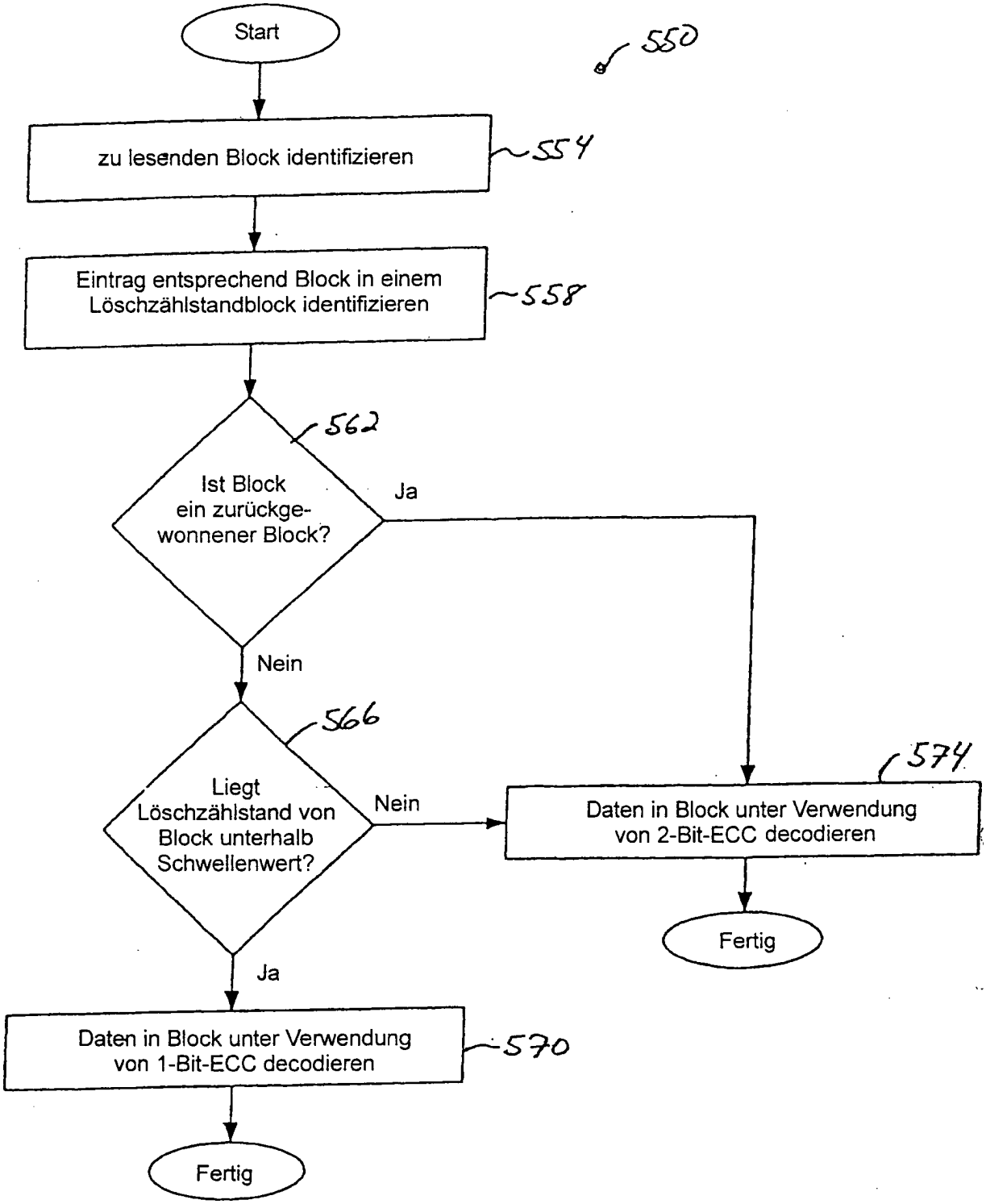
Start
550
zu lesenden Block identifizieren
554
Eintrag entsprechend Block in einem Löschzählstandblock identifizieren
558
562
Ist Block ein zurückgewonnener Block?
Ja
Nein
566
Liegt Löschzählstand von Block unterhalb Schwellenwert?
Nein
574
Daten in Block unter Verwendung von 2-Bit-ECC decodieren
Fertig
Ja
Daten in Block unter Verwendung von 1-Bit-ECC decodieren
570
Fertig

Fig. 5b

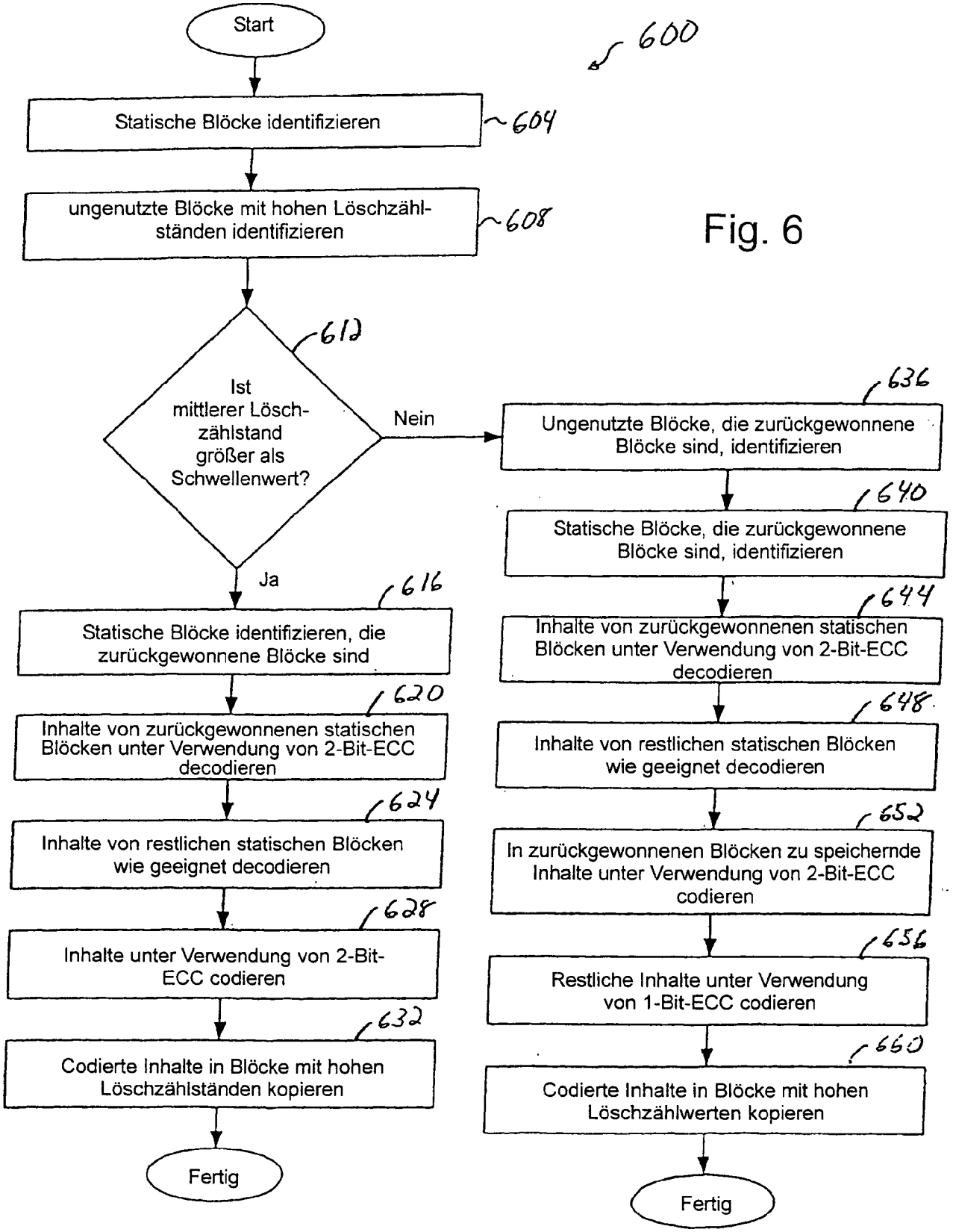
Start
600
Statische Blöcke identifizieren
604
ungenutzte Blöcke mit hohen Löschzählständen identifizieren
608
612
Ist mittlerer Löschzählstand größer als Schwellenwert?
Nein
Ja
636
Ungenutzte Blöcke, die zurückgewonnene Blöcke sind, identifizieren
640
Statische Blöcke, die zurückgewonnene Blöcke sind, identifizieren
644
Inhalte von zurückgewonnenen statischen Blöcken unter Verwendung von 2-Bit-ECC decodieren
648
Inhalte von restlichen statischen Blöcken wie geeignet decodieren
652
In zurückgewonnenen Blöcken zu speichernde Inhalte unter Verwendung von 2-Bit-ECC codieren
656
Restliche Inhalte unter Verwendung von 1-Bit-ECC codieren
660
Codierte Inhalte in Blöcke mit hohen Löschzählwerten kopieren
Fertig
616
Statische Blöcke identifizieren, die zurückgewonnene Blöcke sind
620
Inhalte von zurückgewonnenen statischen Blöcken unter Verwendung von 2-Bit-ECC decodieren
624
Inhalte von restlichen statischen Blöcken wie geeignet decodieren
628
Inhalte unter Verwendung von 2-Bit-ECC codieren
632
Codierte Inhalte in Blöcke mit hohen Löschzählständen kopieren
Fertig

Fig. 6

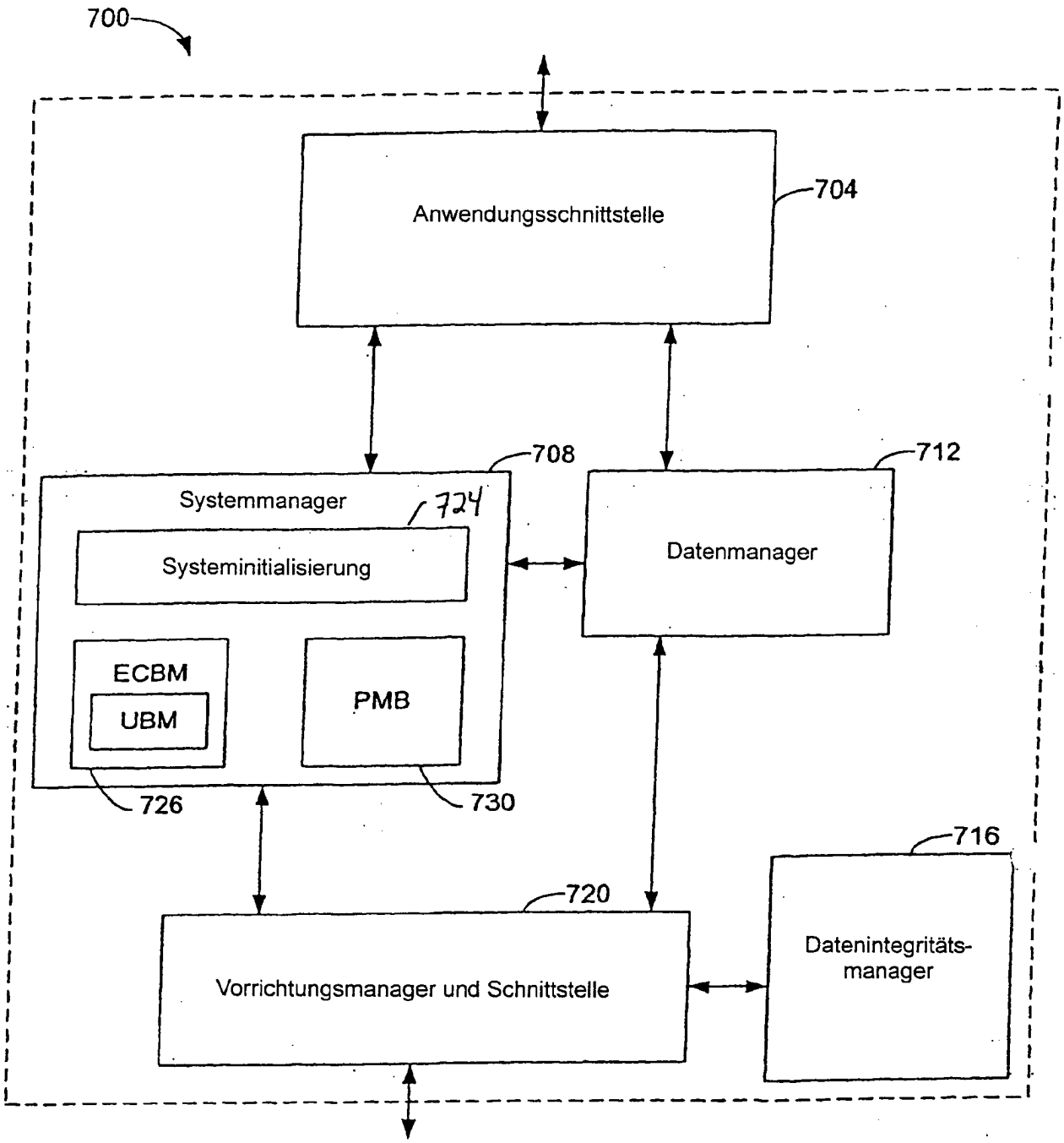
700
Anwendungsschnittstelle
704
708
712
Systemmanager
724
Systeminitialisierung
Datenmanager
ECBM
UBM
PMB
726
730
716
720
Vorrichtungsmanager und Schnittstelle
Datenintegritäts-
manager

Fig. 7