(19) Deutsches Patent- und Markenamt

(10) **DE 11 2011 101 116 T5** 2013.01.10

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2011/121490**
in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **11 2011 101 116.4**
(86) PCT-Aktenzeichen: **PCT/IB2011/051219**
(86) PCT-Anmeldetag: **23.03.2011**
(87) PCT-Veröffentlichungstag: **06.10.2011**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **10.01.2013**

(51) Int Cl.: ***H03M 13/15*** (2013.01)
H03M 13/29 (2013.01)
G06F 11/10 (2013.01)

(30) Unionspriorität:
101584381 **30.03.2010** **EP**

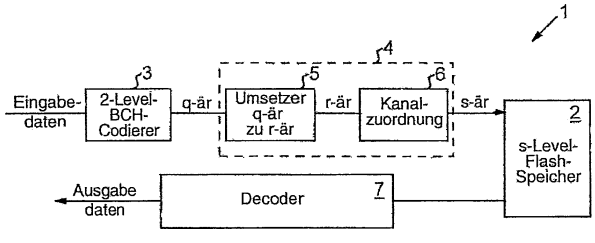
(71) Anmelder:
International Business Machines Corp., Armonk, N.Y., US

(74) Vertreter:
RICHARDT PATENTANWÄLTE GbR, 65185, Wiesbaden, DE

(72) Erfinder:
Eleftheriou, Evangelos S., Rüschlikon, CH; Cideciyan, Roy D., Rüschlikon, CH; Mittelholzer, Thomas, Rüschlikon, CH

(54) Bezeichnung: **Two-Level BCH-Codes für Solid-State-Speichereinheiten**

(57) Zusammenfassung: Es werden Verfahren und Vorrichtungen für das Codieren von Eingangsdaten zum Aufzeichnen im s-Level-Speicher (2) einer Solid-State-Speichereinheit (1) bereitgestellt, wobei s ≥ 2. Eingabedatenwörter werden in Gruppen mit M Eingabedatenwörtern in Übereinstimmung mit dem ersten und dem zweiten BCH-Code codiert, um für jede Gruppe einen Satz mit M ersten Codewörtern des ersten BCH-Codes zu erzeugen. Der Satz mit M ersten Codewörtern wird so erzeugt, dass zumindest eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt und wobei dieser zweite BCH-Code ein Subcode des ersten BCH-Codes ist. Die Sätze von M ersten Codewörtern werden dann im s-Level-Speicher aufgezeichnet (2). Wenn jedes der ersten und zweiten Codewörter N q-äre Symbole umfasst, bei denen $q = p^k$, k ist eine positive ganze Zahl und p ist eine Primzahl, kann das q-äre Codealphabet auf den s-Level-Speicher (2) abgestimmt werden, indem sichergestellt wird, dass q und s die u-te beziehungsweise v-te Potenz einer gemeinsamen Basis r sind, wobei u und v positive ganze Zahlen sind und k ≥ u, wobei $p^{(k/u)v} = s$.



Beschreibung

[0001] Die vorliegende Erfindung bezieht sich allgemein auf die Datencodierung in Solid-State-Speichereinheiten (SSDs). Es werden Verfahren und Vorrichtungen für die Codierung von Daten und die Aufzeichnung der codierten Daten in s-Level-Solid-State-Speichern bereitgestellt, bei denen s ≥ 2 die Anzahl verschiedener Werte (values) oder Zustände (levels) darstellt, die von der Speichergrundeinheit oder der ”Zelle” des Solid-State-Speichers angenommen werden können.

[0002] Eine Solid-State-Speichereinheit ist ein Speicher, der elektronische Schalttechnik, üblicherweise in integrierten Schaltkreisen, für die Datenspeicherung anstelle von herkömmlichen magnetischen oder optischen Medien wie Platten und Bänder benutzt. Durch Solid-State-Speichereinheiten wie zum Beispiel Flash-Speicher erfährt das Gebiet der Datenspeicherung derzeit eine wahre Revolution. Da diese Einheiten keine beweglichen Teile aufweisen, sind sie robuster als herkömmliche Speichereinheiten, und sie bieten eine außergewöhnliche Bandbreite, beträchtliche Einsparungen bei der Leistungsaufnahme und wahlfreie I/O-Leistung (Eingabe/Ausgabe), die um ganze Größenordnungen besser ist als diejenige von Festplattenlaufwerken (HDDs).

[0003] Herkömmliche Speichervorrichtungen wie zum Beispiel HDDs zeichnen Binärdaten auf. Das bedeutet, dass die Speichergrundeinheit in diesen Einheiten nur ein Bit der Daten speichert. Flüchtige Solid-State-Speichertechnologien wie zum Beispiel DRAMs (Dynamic Random Access Memories) zeichnen auch Binärdaten auf, und die Speichergrundzelle in solchen Speichern kann nur zwei Zustände annehmen (s = 2) und zeichnet daher nur Binärwerte auf. Bei anderen Typen von SSDs können die Speicherzellen jedoch mehr als zwei Zustände annehmen (s > 2). Zum Beispiel lassen Flash-Speicher und Phase Change Memory (PCM), zwei wichtige nichtflüchtige Speichertechnologien, Aufzeichnungen in mehreren Zuständen zu. Zum Beispiel können NOR-Flash-Speicher **4** Zustände, d. h. 2 Bits pro Zelle, aufzeichnen. Derzeit sind Multi-level Cell (MLC) NAND Flash-Speicherchips erhältlich, die 4 Datenbits pro einzelner Flash-Zelle unter Einsatz der 43-nm-Prozesstechnologie speichern können. Es wird erwartet, dass die PCM Technologie die Flash-Speichertechnologien ersetzt, wenn Prozesstechnologien unter 10 nm erforderlich sind. Wenn im Moment auch die handelsüblichen PCM-Chips nur ein Bit pro Zelle speichern, ist doch das Speichern von 4 Bits pro Zelle in PCM-Chips im Experiment vorgeführt worden.

[0004] Aufgrund des Erfolges von Solid-State-Speichern wie zum Beispiel Flash und PCM in Verbraucherprodukten wie Digitalkameras und Musikabspielgeräte werden diese Speichertechnologien jetzt auch für das Speichern im Unternehmensbereich ins Auge gefasst. Der Fehlerleistung, die bei Datenspeichereinheiten immer ein Hauptproblem ist, kommt immer mehr Bedeutung zu, wenn diese Technologien in den Unternehmensbereich vordringen. Während herkömmliche Speichervorrichtungen wie HDDs einen ECC-(error correction code = Fehlerkorrekturcode)Aufwand von ca. 10% bis 15% haben, beträgt er in handelsüblichen Flash-Speichern nur 2,5% bis 5%. In diesen Einheiten ist daher Effizienz kritisch für die EC-Codierung.

[0005] Bei den SSDs sind die Speicher in Speicherzonen oder Blöcken organisiert, die jeweils einen Satz Speicherplätze enthalten, in die Daten geschrieben werden können. Die EC-Codierung in SSDs wird durchgeführt, indem im Zustand der Schreibeinheit Redundanz hinzugefügt wird, d. h. in jeder Datenschreibadresse. Ein Flash-Speicher enthält zum Beispiel Datenschreibadressen, die als ”Seiten” bekannt sind. Jede Seite enthält eine Vielzahl von Sektoren, jedoch werden Schreibvorgänge normalerweise auf der Grundlage von Seiten durchgeführt. Ein EC-Code wird für die Eingabedaten berechnet, die auf jede Flash-Seite oder jeden Sektor in einer Seite geschrieben werden, und der EC-Code wird auf dieser Seite oder in diesem Sektor mit den Eingabedaten aufgezeichnet. Dieses Codieren ermöglicht das Beheben von Fehlern innerhalb der einzelnen Datenseiten. Lineare Codes wie RS-Codes (Reed-Solomon) und BCH-Codes (Bose-Chaudhuri-Hocquenghem) sind für dieses sogenannte ”One-level” EC-Codieren eingesetzt worden. „Lange” One-Level Codes, bei denen das Codieren auf der Grundlage einer Seite (d. h. ein Codewort pro Seite) durchgeführt wird, nutzen die vorhandene Seitenredundanz bestmöglich und zeigen daher die beste Fehlerleistung, die Implementierung dieser Schemas ist aber extrem komplex und geht bei großer Seitengröße und gewünschter starker Fehlerkorrekturfähigkeit mit einer hohen Leistungsaufnahme einher. Die Komplexität in Verbindung mit diesen Codes hängt von der Größe des endlichen Körpers ab (der Galoiskörper (GF)), in dem das arithmetische Codieren und Decodieren durchgeführt wird. (Im nachstehenden Text ist der Satz aller n-Tupel mit Elementen vom Galoiskörper mit q Elementen, z. B. mit GF (q) gekennzeichnet, ein Vektorraum, der mit $GF(q)^N$)gekennzeichnet wird. Zum Beispiel sind lange One-Level-Codes für zwei verschiedene Seitengrößen mit einer Belegung von 2 KiB und 4 KiB ausgelegt worden. Es wurde angenommen, dass der verfügbare ECC-Aufwand (Redundanz) der langen One-Level-Codes mit Belegungen von 2 KiB und 4 KiB 56 Byte beziehungsweise 120 Byte betrug. Der beste 2 KiB One-Level RS-Code (dabei sind 2 KiB die Seitengröße) kann bis zu 20 wahlfrei gewählte 11-Bit-Symbole pro Seite korrigieren, erfordert aber eine $GF(2^{11})$-Arithmetik sowohl zum Codieren als auch zum Decodieren.

Der beste 4 KiB One-Level RS-Code kann bis zu 40 nach dem Zufallsprinzip gewählte 12-Bit-Symbole in einer Seite korrigieren, erfordert aber eine GF(2^{12})-Arithmetik zum Codieren und zum Decodieren. Ferner erfordert der beste 2 KiB One-Level BCH-Code, der bis zu 29 nach dem Zufallsprinzip gewählte Bits pro Seite korrigieren kann, eine GF(2)-Arithmetik zum Codieren aber eine GF(2^{15})-Arithmetik zum Decodieren. Der beste 4 KiB One-Level BCH-Code, der bis zu 60 nach dem Zufallsprinzip gewählte Bits pro Seite korrigieren kann, erfordert eine GF(2)-Arithmetik zum Codieren aber eine GF(2^{16})-Arithmetik zum Decodieren. Aufgrund der übermäßigen Komplexität dieser langen Codes werden in der Praxis ”kurze” One-Level-Codes eingesetzt, die mehrere kürzere Codewörter in einer einzigen Seite enthalten, und damit wird die Fehlerleistung der Zweckmäßigkeit der Implementierung geopfert.

[0006] In der US-Patentschrift Nr. 7 047 478 B2 wird ein One-Level-Codiersystem für Multi-Level-Zellspeicherplatz offengelegt, bei dem der Betriebsmodus schaltbar ist zwischen einem Modus, in dem alle verfügbaren Speicherzustände benutzt werden und einem Modus, in dem nicht alle Zustände benutzt werden. Um dies zu bewerkstelligen, verwendet das Codierungsschema ein q-äres Alphabet (d. h. die Codewörter werden aus Symbolen gebildet, die q verschiedene Werte annehmen können) wobei q gleich der Anzahl der verfügbaren Zustände s der Multi-Level-Zellen ist.

[0007] Wie in ”Integrated Interleaving – A Novel ECC Architecture,” von M. Hassner, et al., IEEE Trans. an Magn., Bd. 37, Nr. 2, S. 773 bis 775, Mar. 2001, und in den US-Patentschriften Nr. 5 946 328, 5 942 005 und 7 231 578 B2 beschrieben ist, sind in HDDs Two-Level-Codierschemas mit Einsatz von verschachtelten RS-Codes verwendet worden. Diese Systeme beruhen auf den generellen verketteten RS-Codes, die zum Beispiel beschrieben sind in: E. L. Blokhund V. V. Zyablov, ”Generalized concatenated codes,” Plenum Publishing Corporation, S. 218 bis 222, 1976 (übersetzt von Problemy Peredachi Informatsii, Bd. 10, Nr. 3, S. 45–50, Juli-Sept., 1974); und J. Maucher et al., ”On the Equivalence of Generalized Concatenated Codes and Generalized Error Location Codes” IEEE Trans. an Information Theory, Bd. 46, Nr. 2, März 2000. Das oben angeführte Two-Level RS-Codierschema von Hassner et al. benutzt erste und zweite RS-Codes, mit C1 bzw. C2 gekennzeichnet, wobei der zweite RS-Code ein Subcode des ersten RS-Codes ist. Ein Satz von M ersten (C1) Codewörtern wird so erzeugt, dass eine lineare Kombination dieser M Codewörter ein zweites (C2) Codewort ist. Insbesondere sind eine Anzahl B ≥ 1 von gewichteten Summen der M C1 Codewörter im vorgenannten Satz jeweils C2 Codewörter des zweiten RS-Codes. Die Gewichtungskoeffizienten für diese gewichteten Summen werden von einer Vandermonde-Matrix definiert. Außerdem unterliegt das Codierschema der Begrenzung M < q, d. h., die Anzahl von C1 Codewörtern M im vorgenannten Satz ist geringer als die Gesamtanzahl der Elemente q des Galois-Körpers. Zum Beispiel gilt im Fall von binären linearen Codes über GF(q = 2) die Bedingung M < 2 für integrierte verschachtelte Codes, d. h. M = 1, und somit enthält die von Hassner et al. definierte Familie der integrierten verschachtelten Codes keine binären Two-Level-Codes. Die Benutzung von mehreren über einen Sektor verschachtelten RS-Codewörtern in diesem Schema erhöht die Robustheit gegen Burst-Fehler im HDD Kanal. RS-Codes sind dafür gut geeignet und werden im Allgemein bevorzugt. Dies kann teilweise auf die einfache Leistungsbewertung zurückzuführen sein, da die Gewichtsverteilungen von RS-Codes bekannt sind. Auch die Komplexität der Implementierung erweist sich als günstig im Vergleich mit BCH-Codes wie oben dargestellt, denn insbesondere das Decodieren ist bei RS-Codes erheblich weniger komplex.

[0008] Das Two-Level RS-Codieren ist auch für DRAMs eingesetzt worden, wie in ”Reliable Memories with Subline Accesses”, Junsheng Han et al., ISIT2007, Nizza, Frankreich, 24. Juni bis 29. Juni, S. 2531 bis 2535, und in der US-Patentanmeldung Nr. 2008/0168329 A1 erörtert wird. Das Two-Level RS-Codieren beruht auf dem oben angeführten integrierten Verschachtelungsschema von Hassner et al., obwohl Codeauslegungsdetails und Decodieralgorithmen nicht spezifiziert sind.

[0009] Bei einer Ausführungsform eines Aspekts der vorliegenden Erfindung wird ein Verfahren für die Aufzeichnung von Eingabedaten in den s-Level-Speicher einer Solid-State-Speichereinheit mit s ≥ 2 bereitgestellt. Das Verfahren umfasst:
Codieren von Eingabedatenwörtern in Gruppen mit M Eingabedatenwörtern in Übereinstimmung mit dem ersten und dem zweiten BCH-Code, um für jede Gruppe einen Satz mit M ersten Codewörtern des ersten BCH-Codes zu erzeugen, so dass eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt, wobei der zweite BCH-Code ein Subcode des ersten BCH-Codes ist; und
Aufzeichnung der Sätze der M ersten Codewörter in den s-Level-Speicher.

[0010] Anders als bei der üblichen Verwendung von RS-Codes wie oben besprochen verwenden Verfahren des Ausführungsbeispiels der vorliegenden Erfindung BCH-Codes zum Two-Level-Codieren in s-Level-Speichereinheiten. Ausführungsbeispiele der vorliegenden Erfindung beruhen teilweise auf der Erkenntnis, dass

BCH-Codes besonders gut für die Korrektur der Fehlermuster in SSD Speicherkanälen geeignet sind. Außerdem kann mit der Verwendung eines Two-Level-BCH-Codierschemas der Vorteil der Fehlerleistung genutzt und gleichzeitig die Auswirkung der Komplexität des Decoders minimiert werden. Eine gute Fehlerratenleistung kann in der Tat erreicht werden, ohne dass die Komplexität der Implementierung wesentlich vergrößert wird. Mit der Verwendung von BCH-Codes können trotz der bestehenden Nachteile beim Two-Level-Codieren in SSDs hoch effiziente Fehlerkorrektursysteme mit niedriger Komplexität bei der Implementierung und geringer Leistungsaufnahme erzielt werden. Ferner können, wie weiter unten besprochen wird, die Merkmale des BCH-Codes leicht darauf abgestimmt werden, damit sie der Anzahl der s-Levels der SSD Zellen und der Fehlerstruktur des SSD Kanals entsprechen. Insgesamt gesehen können daher Aufzeichnungsverfahren, die die Erfindung ausmachen, praktische und leistungsfähige Systeme für das zuverlässige Speichern von Daten in s-Level Solid-State-Speichern bereitstellen.

[0011] Im Allgemeinen können Eingabedatenwörter, die codiert werden sollen, eine Vielzahl von Datensymbolen umfassen, und bei den Datensymbolen kann es sich um Symbole mit einem Bit (d. h. binäre) oder mit mehreren Bits handeln. In bevorzugten Ausführungsformen der Erfindung umfasst sowohl das erste als auch das zweite Codewort eine Vielzahl (mit N gekennzeichnet) von q-ären Symbolen, wobei $q = p^k$, k ist eine positive ganze Zahl, und p ist eine Primzahl. Außerdem ist in besonders bevorzugten Ausführungsformen das q-äre Alphabet des Two-Level-BCH-Codes für den besonders leistungsfähigen Betrieb auf die Anzahl s-Level des Solid-State-Speichers abgestimmt. Dies wird mit der Benutzung eines Codes erreicht, bei dem q und s u-te bzw. v-te Potenzen einer gemeinsamen Basis r sind, und dabei sind u und v positive ganze Zahlen und $k \geq u$. Dies gewährleistet, dass das q-äre Codealphabet zum s-ären Speicher gemäß der Bedingung $p^{(k/u)v} = s$ ”passt”. Wenn sichergestellt wird, dass der Code diese Auflage des „passenden Alphabets” erfüllt, können die q-ären Codewortsymbole einfach und effizient für die Aufzeichnung im s-Level-Speicher in ein s-äres Alphabet umgesetzt werden und stellen einen hocheffizienten Betrieb sowie eine Flexibilität bei der Wahl des Codes bereit. Da insbesondere q und s in manchen Fällen gleich sein könnten, werden weiter unten bevorzugte Ausführungsformen ausführlich dargestellt, in denen $q \neq s$. Da die die Erfindung verkörpernden Verfahren darüber hinaus für binäre ($s = 2$) Solid-State-Speicher eingesetzt werden können, stellen bevorzugte Ausführungsformen einen flexiblen und effizienten Betrieb von Multi-Level-Solid-State-Speichern ($s > 2$) bereit.

[0012] In besonders bevorzugten Ausführungsbeispielen wird jeder Satz von M ersten Codewörtern an einer jeweiligen Schreibadresse des Solid-State-Speichers aufgezeichnet. In einem Flash-Speicher zum Beispiel enthält jede Seite einen einzigen Satz von M ersten Codewörtern. Es kann jedoch bei manchen Ausführungsformen wünschenswert sein, eine Vielzahl von Sätzen von M ersten Codewörtern in jeder Schreibadresse aufzuzeichnen, zum Beispiel ein Satz pro Sektor einer Flash-Seite. Es sind auch Ausführungsformenvorstellbar, bei denen jeder Satz von M ersten Codewörtern in einer Vielzahl von Schreibadressen des Solid-State-Speichers aufgezeichnet wird.

[0013] Gemäß einer Ausführungsform eines anderen Aspekts der Erfindung wird ein Computerprogramm bereitgestellt, das Programmcodemittel dazu umfasst, dass ein Computer ein Verfahren gemäß dem ersten Aspekt der Erfindung durchführt. Es versteht sich, dass der Begriff „Computer” im allgemeinsten Sinn verwendet wird und jegliche Einheit, Komponente oder ein System einschließt, die fähig zur Datenverarbeitung sind, um ein Computerprogramm zu implementieren. Außerdem kann ein Computerprogramm, das die Erfindung verkörpert, ein unabhängiges Programm darstellen, oder es kann ein Element eines größeren Programms sein, und es kann zum Beispiel auf einem computerlesbaren Medium wie zum Beispiel einer Platte oder einer elektronische Übermittlung zum Laden in einen Computer dargeboten werden. Das Programmcodemittel des Computerprogramms kann jeglichen Ausdruck in jeglicher Sprache, beliebigem Code oder Notation eines Satzes von Anweisungen umfassen, die dazu beabsichtigt sind, dass ein Computer das betreffende Verfahren durchführt, und zwar entweder direkt oder nach allein der (a) Umsetzung in eine andere Sprache, Code oder Notation oder/und (b) Wiedergabe in einer anderen materiellen Form.

[0014] Eine Ausführungsform eines dritten Aspekts der Erfindung stellt eine Solid-State-Speichereinheit bereit, die Folgendes umfasst:
Einen s-Level-Solid-State-Speicher, bei dem $s \geq 2$; und
einen Two-Level-BCH-Codierer für das Codieren von Eingabedatenwörtern in Gruppen mit M Eingabedatenwörtern in Übereinstimmung mit dem ersten und dem zweiten zu erzeugenden BCH-Code, für jede Gruppe einen Satz von M ersten Codewörtern des ersten BCH-Codes, so dass eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt, wobei der zweite BCH-Code ein Subcode des ersten BCH-Codes ist;
wobei die Einheit so eingerichtet ist, dass sie die Sätze von M ersten Codewörtern im s-Level-Speicher aufzeichnet.

[0015] Ein Ausführungsform eines vierten Aspekts der Erfindung stellt eine Codiervorrichtung für das Codieren von Eingabedaten bereit, die in der s-Level-Solid-State-Speichereinheit aufgezeichnet werden sollen, bei der $s \geq 2$. Die Codiervorrichtung umfasst:
Einen Two-Level-BCH-Codierer für das Codieren von Eingabedatenwörtern in Gruppen von M Eingabedatenwörtern in Übereinstimmung mit dem ersten und dem zweiten zu erzeugenden BCH-Code, für jede Gruppe einen Satz von M ersten Codewörtern des ersten BCH-Codes, so dass eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt, wobei der zweite BCH-Code ein Subcode des ersten BCH-Codes ist, wobei sowohl das erste als auch das zweite der Codewörter N q-äre Symbole umfasst, bei denen $q \neq s$, $q = p^k$, k ist eine positive-ganze Zahl, und p ist eine Primzahl, bei denen q und s u-te bzw. v-te Potenzen einer gemeinsamen Basis r sind, und dabei sind u und v positive ganze Zahlen und $k \geq u$, dabei ist $p^{(k/u)v} = s$; und
einen Symbolumsetzer für die Umsetzung der q-ären Symbole jedes ersten Codeworts in ein s-äres Alphabet für die Aufzeichnung im s-Level-Speicher.

[0016] Wenn hier im Allgemeinen Merkmale mit Bezugnahme auf eine Ausführungsform eines Aspekts der Erfindung beschrieben werden, können entsprechende Merkmale in Ausführungsformen eines anderen Aspekts der Erfindung bereitgestellt werden.

[0017] Nachfolgend wird beispielhaft eine bevorzugtes Ausführungsform der Erfindung unter Bezugnahme auf die begleitenden Figuren beschrieben, die Folgendes zeigen:

[0018] Fig. 1 ist ein schematisches Blockschaubild einer s-Level Solid-State-Speichereinheit, die die Erfindung verkörpert;

[0019] Fig. 2 zeigt ein erstes Two-Level-Codierschema für den Einsatz in der Einheit der Fig. 1;

[0020] Fig. 3 zeigt ein zweites Two-Level-Codierschema für den Einsatz in der Einheit der Fig. 1;

[0021] Fig. 1 ist eine vereinfachte schematische Darstellung einer Solid-State-Speichereinheit, hier eine Flash-Speichereinheit **1**, mit den Hauptelementen, die zu dem zu beschreibenden Codiersystem dazugehören. Die Einheit **1** beinhaltet einen s-Level-Flash-Speicher **2** zum Speichern in integrierten Arrays von Flash-Speicherzellen, wobei jede Zelle $s \geq 2$ verschiedene Speicherzustände annehmen kann. Der Flash-Speicher **2** wird hier in dieser Figur zwar nur als ein einziger Block gezeigt, er kann aber generell jede gewünschte Konfiguration von Flash-Speichereinheiten umfassen, die zum Beispiel von einem einzigen Chip oder Blockschnitt bis zu einer Vielzahl von Speicherbänken reichen, die jeweils mehrere Packs mit Speicherchips enthalten. Die Einheit **1** weist eine Codiervorrichtung auf, die in verallgemeinerter Form so dargestellt wird, dass sie einen Two-Level-BCH-Codierer **3** enthält und einen Symbolumsetzer **4**, der von der gestrichelten Linie in der Figur umgeben ist. Der Symbolumsetzer **4** umfasst einen Umsetzer **5** für q-är zu r-är und eine Channel-Mapping-Einheit **6**. Die entsprechende Decodiervorrichtung der Einheit **1** wird in der Figur allgemein als Decoder **7** angegeben.

[0022] Im Allgemeinen könnte die Funktionalität des BCH-Codierers **3**, des Symbolumsetzers **4** und des Decoders **7** in Hardware, Software oder in einer Kombination davon implementiert werden. Zum Beispiel könnte der Codiervorgang im BCH-Codierer **3** vollständig oder teilweise von Software durchgeführt werden, die einen Prozessor der Codiervorrichtung so konfiguriert, dass das unten ausführlich dargestellte Codierschema implementiert wird. Dem Fachmann wird nach der Beschreibung eine solche geeignete Software bekannt sein. Jedoch wird aus Gründen der Betriebsgeschwindigkeit der Einsatz von verdrahteten Logikschaltkreisen allgemein für die möglichst weitgehende Implementierung der Funktionalität bevorzugt. Außerdem könnte der Codierprozess im Allgemeinen durch systematisches Codieren implementiert werden (wenn die Eingabedaten in den Codierprozess durch das Codieren unverändert bleiben, ein Paritätscode zu den nichtcodierten Symbolen jedoch hinzugefügt wird, um das Ausgabe-Codewort zu erhalten), oder durch nichtsystematisches Codieren (wenn die Eingabedaten in den Codierprozess durch das Codieren in das Ausgabecodewort eingebettet werden). Zwecks Einfachheit der Implementierung des Codierers (und des entsprechenden Decoders) wird jedoch das systematische Codieren bevorzugt. Geeignete Implementierungen der Codier- und der Decodiervorrichtung der Einheit **1** werden dem Fachmann nach dieser Beschreibung ohne Weiteres geläufig sein.

[0023] Beim Betrieb der Einheit **1** werden die im Flash-Speicher **2** zu speichernden Eingabedaten für den BCH-Codierer **3** bereitgestellt. Die Eingabedaten umfassen eine Aufeinanderfolge von Eingabedatenwörtern, diese werden vom BCH-Codierer **3** mit einem weiter unten beschriebenen Two-Level-BCH Codierschema in Gruppen mit M Eingabedatenwörtern codiert. Für jede Gruppe mit M Eingabedatenwörtern (die Wörter mit verschiedenen Anzahlen von Datensymbolen umfassen können, wie weiter unten beschrieben wird), gibt der

BCH-Codierer **3** einen Satz von M q-ären Symbolcodewörtern aus (d. h. ein einziges Symbol dieser Codewörter kann q mögliche Werte haben), wobei $q = p^k$, k ist eine positive ganze Zahl, und p ist eine Primzahl. Je nach der Beziehung zwischen q und s funktioniert der Symbolumsetzer **4** wie weiter unten beschrieben, um die q-ären Symbolcodewörter in ein s-äres Alphabet für die Aufzeichnung umzusetzen. Die s-ären Symbolcodewörter werden dann im Flash-Speicher **2** aufgezeichnet, wobei jede Flash-Zelle ein s-äres Symbol eines Codeworts speichert. Bei dieser Ausführungsform stellt jede Seite des Flash-Speichers ausreichend Speicherplatz für einen einzelnen Satz von M s-ären Codewörtern bereit, wobei jeder Satz von M s-ären Codewörtern in einer jeweiligen Flash-Seite gespeichert wird. Wenn dann später eine Seite aus dem Flash-Speicher **2** gelesen wird, führt der Decoder **7** den zur Codiervorrichtung **3**, **4** umgekehrten Vorgang durch, um die s-ären Codewörter zu decodieren, und er implementiert die Fehlerentdeckung und die Wiederherstellung auf allgemein bekannte Weise. Damit werden die ursprünglichen Eingabedatenwörter wiederhergestellt und vom Decoder **7** als Ausgabedaten bereitgestellt.

[0024] Nun wird die Funktion des BCH-Codierers **3** ausführlicher dargestellt. Jede für den Codierer **3** bereitgestellte Gruppe von M Eingabedatenwörtern besteht aus $(M - P)K_1$-Symbol Datenwörtern aus dem Vektorraum $GF(q)^K{}_1$ und PK_2-Symbol Datenwörtern aus dem Vektorraum $GF(q)^K{}_2$. Der Codierer **3** ordnet die $(M - P)K_1$-Symbol Datenwörter den jeweiligen N-Symbol ersten Codewörtern in Übereinstimmung mit einem ersten BCH-Code zu. Dieser erste BCH-Code besteht aus einem vordefinierten Satz von N-Tupeln aus dem Vektorraum $GF(q)^N$. Somit ordnet der Code die eingegebenen q-är-Symbol Datenwörter den q-är-Symbol ersten Codewörtern auf einer Eins-zu-Eins-Basis zu, wobei $N - K_1$ Paritätssymbole in jedem der $(M - P)$ ersten Codewörter einen ersten Teil der Redundanz bereitstellen, die für den ECC-Prozess notwendig ist. Ferner ordnet der Codierer **3** die verbleibenden PK_2-Symbol Datenwörter den jeweiligen N-Symbol zweiten Codewörtern in Übereinstimmung mit einem zweiten BCH-Code zu. Dieser zweite BCH-Code besteht aus einem weiteren vordefinierten Satz von N-Tupeln aus dem Vektorraum $GF(q)^N$ und ist ein Subcode des ersten BCH-Codes, d. h., der Satz aller zweiten Codewörter ist ein Teilsatz des Satzes aller ersten Codewörter. Somit ordnet der zweite Code die eingegebenen q-är-Symbol Datenwörter dem q-är-Symbol zweiten Codewörtern (die auch erste Codewörter sind) auf einer Eins-zu-Eins-Grundlage zu, wobei $N - K_2$ Paritätssymbole in jedem der P Codewörter einen zweiten Teil der Redundanz bereitstellen, die für den ECC-Prozess notwendig ist. Somit erzeugt der Codierer einen ursprünglichen Satz von M Codewörtern. Dann wird ein letzter Satz von M Codewörtern erhalten, indem die P zweiten Codewörter im ursprünglichen Satz durch P veränderte (erste) Codewörter ersetzt werden, in dem das i-te Symbol (wobei i = 1 bis N) jedes veränderten Codeworts eine bestimmte lineare Kombination der i-ten Symbole der M Codewörter im ursprünglichen Satz ist. Diese veränderten Codewörter werden so erzeugt, dass der letzte Satz von M ersten Codewörtern durch den zweiten BCH-Code zwingend belegt wird. Spezifisch ist zumindest eine vorbestimmte lineare Kombination des Satzes von M ersten Codewörtern ein Codewort des zweiten BCH-Codes. Somit erzeugt der resultierende Satz von M ersten Codewörtern, wenn er nach einer oder mehreren vorbestimmten linearen Kombinationen kombiniert wird, ein oder mehrere zweite Codewörter des zweiten BCH-Codes.

[0025] Vorstehend wird ein Beispiel einer Implementierung für das Two-Level-Codierschema im Codierer **3** beschrieben, es sind aber verschiedene Two-Level-Codeaufbauten im Codierer **3** vorstellbar. Gemäß den Ausführungsformen der Erfindung wird jede Gruppe von M Eingabedatenwörtern in Übereinstimmung mit den ersten und den zweiten BCH-Codes codiert, um einen Satz von M ersten. Codewörtern des ersten BCH-Codes so zu erzeugen, dass eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt und dieser zweite Code ein Subcode des ersten BCH-Codes ist. Ein einfaches Beispiel ist schematisch in **Fig. 2** dargestellt. Die ersten und die zweiten Codewörter sind durch die beiden unterschiedlichen rechteckigen Blöcke oben in der Figur dargestellt. Die zweiten Codewörter wie auch die ersten Codewörter bestehen jeweils aus N q-ären Symbolen wie angegeben. Bei einer gegebenen Gruppe von M Eingabedatenwörtern ordnet der resultierende Satz von M ersten Codewörtern bei Verkettung in einer definierten Reihenfolge effektiv ein Two-Level-Codewort des gesamten Two-Level-BCH-Codes zu, wie in der Mitte der Figur gezeigt ist. Dieses Two-Level-Codewort erfüllt die unten in der Figur dargestellte Bedingung. Diese besteht darin, dass die Summe (Symbol-für-Symbol) der M ersten Codewörter ein Codewort des zweiten BCH-Codes erzeugt.

[0026] Ein weiteres Beispiel eines Two-Level-Codierschemas ist in **Fig. 3** dargestellt. In diesem Fall erzeugt jede einer Vielzahl von linearen Kombinationen der M ersten Codewörter ein Codewort des zweiten BCH-Codes. Insbesondere ist bei einem definierten Verkettungsbefehl jede einer Vielzahl von gewichteten Summen der M ersten Codewörter ein jeweiliges zweites Codewort. Dies wird in verallgemeinerter Form von der Bedingung unten in der Figur dargestellt, wobei "*" die Multiplikation kennzeichnet. Insbesondere werden zweite Codewörter von P gewichteten Summen der M Codewörter erhalten, wobei die Gewichtungskoeffizienten für

diese gewichteten Summen durch ein Paritätsprüfmuster definiert werden, das einem dritten linearen Code entspricht, wie nachstehend beschrieben wird.

[0027] Diese Klasse von Two-Level-BCH-Codes C über dem endlichen Körper GF(q) beruht auf der Definition von drei Codes C_1 und C_2 und C* über dem endlichen Körper GF(q), wobei q eine Potenz einer Primzahl ist. Der Code C_1 ist der erste BCH-Code [N, K_1, d(C_1)] mit minimaler Entfernung d(C_1), die vom (N – K_1) × N Paritätsprüfmuster H_1 definiert wird. Der zweite Code C2 ist der zweite BCH-Code [N, K_2, d(C_2)] mit minimaler Entfernung d(C_2). Dies ist ein Subcode des ersten Codes C_1 und wird vom (N – K_2) × N Paritätsprüfmuster

$$H_2 = \begin{bmatrix} H_1 \\ H_a \end{bmatrix}$$

definiert, wobei H_a ein (K_1 – K_2) × N Paritätsprüfmuster ist, das die Auflagen der Paritätsprüfung darstellt, die der C_2 Code zusätzlich zu den Auflagen H_1 der Paritätsprüfung erfüllen muss. Es sei H* = [$h^*_{i,j}$] ein vollständiges Paritätsprüfmuster über GF(q) der Größe P × M, das einen linearen [M, K*, d(C*)] Code der Dimension K* = M – P und mit minimaler Entfernung d(C*) ist. Der gesamte Two-Level-Code C der Länge MN, der aus dem linearen Code C* mit dem Paritätsprüfmuster H* = [$h^*_{i,j}$] und den Codes C_1 und C_2 erzielt wird, wobei $C_2 \subset C_1$, wird vom nachstehenden Paritätsprüfmuster definiert,

$$H = \begin{bmatrix} H_1 & 0 & \cdots & 0 \\ 0 & H_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & H_1 \\ & H_b & & \end{bmatrix} \text{ wobei } H_b = \begin{bmatrix} h^*_{0,0}H_a & h^*_{0,1}H_a & \cdots & h^*_{0,M-1}H_a \\ h^*_{1,0}H_a & h^*_{1,1}H_a & \cdots & h^*_{1,M-1}H_a \\ \vdots & \vdots & \vdots & \vdots \\ h^*_{P-1,0}H_a & h^*_{P-1,1}H_a & \cdots & h^*_{P-1,M-1}H_a \end{bmatrix} \text{ ein}$$

Blockmuster der Größe [(N – K_1)M + (K_1 – K_2)P] × [MN] ist, das auch durch H_b = [$h^*_{i,j}H_a$] auf der Grundlage dessen charakterisiert werden kann, dass eine einfache Darstellung des (i, j)-ten Subblocks von H_b benutzt wird. Die Musterelemente $h^*_{i,j}$ geben hier die Gewichtungskoeffizienten für die gewichteten Summen in **Fig. 3** an. Es kann gezeigt werden, dass der resultierende Two-Level-Code C der Länge MN die Dimension K*K_1 + PK_2 und minimale Entfernung hat:

d(C) = min{d(C*)d(C_1), d(C_2)}.

[0028] Es ist klar, dass zahlreiche Two-Level-BCH-Codes nach den Codierarchitekturen der **Fig. 2** und **Fig. 3** aufgebaut werden können, und insbesondere die Codes C_1 und C_2 und C* können nach den Parametern von Einzelsystemen gewählt werden. Zum Beispiel können spezifische Codes gewählt werden, und dies auf der Grundlage der Anzahl von s-Levels des s-ären Speichers und der Leistungserfordernisse eines gegebenen Systems. Zur Verbesserung der Leistungsfähigkeit und Vereinfachung der Implementierung wird jedoch in bevorzugten Ausführungsbeispielen der Einheit **1** der BCH-Code so gewählt, dass:

$q = p^k = r^u$, wobei $k \geq u$; und (1)

$s = r^v$. (2)

wobei u und v positive ganze Zahlen sind. In anderen Worten, q und s sind die u-te beziehungsweise v-te Potenz einer gemeinsamen Basis r. Dies führt zu der Bedingung:

$p^{(k/u)V} = s$,

die hier als Auflage des ”übereinstimmenden Alphabets” gilt. Die Erfüllung dieser Auflage stellt sicher, dass das q-äre Alphabet des Two-Level-BCH-Codes mit dem s-ären Speicher zwecks effizientestem Betrieb der Einheit übereinstimmt. Insbesondere wird Flexibilität bei der Wahl der Codierung bereitgestellt und gleichzeitig sichergestellt, dass der q-äre Code im Symbolumsetzer **4** einfach und effizient zu einem s-ären Alphabet umgesetzt werden kann. Somit beschränken sich Codierschemas nicht auf diejenigen, bei denen q = s, sondern sie können so ausgewählt werden, dass sie zu besten Ergebnissen für die besonderen s-Level-Speichermerkmale führen. Beim Betrieb der Vorrichtung **1**, wobei s ≠ q, werden die q-är Symbol Codewörter zu r-är Symbol Codewörtern umgesetzt, und zwar durch den q-är zu r-är Umsetzer **5**, wobei r^v = s ist, wie oben defi-

niert. Wenn hier v > 1, (d. h. r ≠ s), können die r-är Symbole den s Speicherkanälen des Speichers **2** in der Kanalzuordnungseinheit **6** zugeordnet werden. Insbesondere werden aufeinanderfolgende Sätze von v r-är Symbolen jedes r-är Eingabecodeworts den jeweiligen entsprechenden Zuständen des s-Level-Speichers in Übereinstimmung mit einem vorbestimmten Zuordnungsschema zugeordnet, das alle möglichen Werte von v r-är Symbolen den anderen der s-Levels zuordnet. Die Kanalzuordnung kann hier auf jede beliebige Weise durchgeführt werden, indem zum Beispiel ein bekanntes Gray Zuordnungsschema benutzt wird, das zu den Fehlermerkmalen des s-ären Speichers passt, wie sich für den Fachmann zeigt. Jeder Satz von v r-ären Eingabesymbolen entspricht somit einem s-ären Symbol am Ausgang der Kanalzuordnungseinheit **6** und wird als der entsprechende Zustand in einer jeweiligen s-ären Zelle des Speichers **2** aufgezeichnet.

[0029] In Ausführungsbeispielen, bei denen u = 1 wobei q = r, ist eine Umsetzung von q-är zu r-är natürlich nicht notwendig, und der Umsetzer **5** kann weggelassen oder übergangen werden. Wenn jedoch r ≠ s angenommen wird, kann die Kanalzuordnung immer noch wie oben durchgeführt werden, hier für Sätze von v q-ären Symbolen. Ebenso kann in der Ausführungsform, bei der r = s ist, die Kanalzuordnungseinheit **6** weggelassen oder übergangen werden. Im Allgemeinen kann daher der Symbolumsetzer **4** eine oder beide Umsetzungen von q-är zu r-är und die Kanalzuordnung nach den besonderen Parametern des Speichersystems vornehmen. Es wird darauf hingewiesen, dass die beschriebene Two-Level-Codiertechnik auch für Systeme eingesetzt werden kann, in denen q = s, und in diesem Fall kann der Symbolumsetzer **4** insgesamt weggelassen oder umgangen werden.

[0030] Das obige System bietet zwar außergewöhnliche Flexibilität bei der Wahl der Codierung, es werden aber nachstehend zwei besonders interessante Familien von Codierschemas für den Einsatz in der Speichereinheit **1** beschrieben.

[0031] Die erste Familie Codierschemas zeichnet sich durch r = s > 2 aus und ist besonders nützlich, wenn der Speicherkanal als ein s-äres Kanalmodell mit unabhängigen s-ären Symbolfehlern beschrieben werden kann. Die zweite Familie Codierschemas zeichnet sich durch r = 2 und k = u aus und ist besonders nützlich, wenn die Verkettung der Kanalzuordnungseinheit, des Speicherkanals, des Rückleseprozesses und der umgekehrten Kanalzuordnung im Decoder **7** mit einem binären Kanal mit unabhängigen Bitfehlern beschrieben werden kann. Der unabhängige Bitfehlerkanal ist ein gutes Modell für nichtflüchtige s-Level-Speicher, wenn Gray-Codierung als Kanalzuordnung benutzt wird und Fehler im nichtflüchtigen Speicherkanal dazu führen, dass ein Kanalzustand irrtümlicherweise als ein anschließender Kanalzustand gelesen wird. In diesem Fall ist $q = 2^k$, r = 2 und $s = 2^v$. Für einen. unabhängigen Bitfehlerkanal wäre es gut, ein Two-Level binäres BCH-Codierschema zu wählen, bei dem q = 2, d. h., p = 2 und k = 1 ist. Es kann jedoch vorteilhaft sein, aus Gründen der Implementierung und/oder Fehlerratenleistung einen Code über GF(q), q > 2 zu nutzen, wobei die Arithmetik des endlichen Körpers in $GF(q = 2^k)$, k > 1 durchgeführt wird.

[0032] Nun werden zwei spezifische Beispiele von Two-Level-Codes für Flash-Speicher detailliert beschrieben. Derzeit enthält eine typische Seite im Flash-Speicher 4 KiB Benutzerdaten. Ferner wird angenommen, dass die verfügbare ECC Redundanz pro Seite 224 Byte beträgt. Wenn ein Binärcode benutzt wird, muss die Codelänge MN 34560 (in Bit) betragen, und die Dimension des Codes muss zumindest 32768 (in Bit) betragen. Es werden zwei binäre Two-Level-Codes mit Länge 34560 = M × N = 9 × 3840 konstruiert auf der Grundlage von Länge-3840 binäre BCH-Codes mit Decodieralgorithmen über $GF(2^{12})$.

[0033] Im ersten Beispiel wird der binäre Two-Level-Code C von einem ersten binären BCH-Code C_1 mit Parametern [N = 3840, K_1 = 3660, d_1 = 31] konstruiert, ein zweiter binärer BCH-Code C2 mit den Parametern [N = 3840, K_2 = 3492, d_2 = 59], und der Einzelparitätsprüfcode C* mit Länge M = 9, Dimension K* = 8 und d(C*) = 2. Der resultierende Two-Level-Code hat die Dimension 32772 und kann daher 4 KiB Nutzerdaten aufnehmen. Ferner kann d(C) = 59, d. h. der Code, bis zu 29 Fehlern korrigieren.

[0034] Im zweiten Beispiel wird der binäre Two-Level-Code C von einem ersten binären BCH-Code C_1 mit Parametern [N = 3840, K_1 = 3732, d_1 = 19] konstruiert, ein zweiter binärer BCH-Code C_2 mit den Parametern [N = 3840, K_2 = 3528, d_2 = 53], und der gekürzte Hamming Code C* mit Länge M = 9, Dimension K* = 5 und d(C*) = 3. Der resultierende Two-Level-Code hat die Dimension 32772 und kann daher 4 KiB Nutzerdaten aufnehmen. Ferner kann d(C) = 53, d. h. der Code, bis zu 26 Fehler korrigieren.

[0035] Diese binären Two-Level-Code-Beispiele über GF(2) sollten mit dem binären BCH-Code mit den Parametern [34560, 32768, 225] verglichen werden. Dieser lange BCH-Code kann bis zu 112 Fehler korrigieren, aber sein Decodieralgorithmus funktioniert im breiten $GF(2^{16})$ Körper, der derzeit für SSD Speicheranwendungen unpraktisch ist.

[0036] Ein anderer Vergleich des oben beschriebenen Two-Level-Codes wird mit einem 9-Wege-Verschachtelungsungsschema eines binären One-Level-BCH-Codes mit den Parametern [3840, 3648, 33] angestellt, der bis zu 16 Fehler korrigieren kann. Der Decoder für jeden der 9 verschachtelten BCH-Codes funktioniert über GF (2^{12}). Obwohl die Two-Level-Codes Decoder haben, die über den gleichen GF(2^{12}) Körper funktionieren, können sie im Vergleich zum einfachen 9-Wege-Verschachtelungsschema substanziell mehr Fehler korrigieren.

[0037] Der Einsatz von Two-Level-BCH-Codes mit übereinstimmendem Alphabet für die Seitencodierung im s-Level Flash wie oben beschrieben kann sehr leistungsfähiges Codieren für Fehlerkorrektur bereitstellen. BCH-Codes sind besonders für die Fehlermerkmale von s-Level-Solid-State-Speicher geeignet wie nichtflüchtige Flash-Speicher und PCM-Speicher, wobei beidem Speichermedium unabhängige Symbolfehlern auftreten können, bei denen Symbole in aktuellen nichtflüchtigen Speicherprodukten die Länge von 1 Bit bis 4 Bit haben. Außerdem kann die verbesserte Fehlerleistung mit minimaler Auswirkung auf die Komplexität der Implementierung und die Leistungsaufnahme haben. Daher kann mit Einsatz des beschriebenen Systems eine außergewöhnliche Gesamtleistung erzielt werden.

[0038] Es ist natürlich klar, dass an den beschriebenen beispielhaften Ausführungsformen viele Änderungen und Veränderungen vorgenommen werden können, ohne den Rahmen der Erfindung zu verlassen.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

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

Zitierte Patentliteratur

- US 7047478 B2 [**0006**]
- US 5946328 [**0007**]
- US 5942005 [**0007**]
- US 7231578 B2 [**0007**]

Zitierte Nicht-Patentliteratur

- ”Integrated Interleaving – A Novel ECC Architecture,” von M. Hassner, et al., IEEE Trans. an Magn., Bd. 37, Nr. 2, S. 773 bis 775, Mar. 2001 [**0007**]
- E. L. Blokhund V. V. Zyablov, ”Generalized concatenated codes,” Plenum Publishing Corporation, S. 218 bis 222, 1976 [**0007**]
- Problemy Peredachi Informatsii, Bd. 10, Nr. 3, S. 45–50, Juli-Sept., 1974 [**0007**]
- J. Maucher et al., ”On the Equivalence of Generalized Concatenated Codes and Generalized Error Location Codes” IEEE Trans. an Information Theory, Bd. 46, Nr. 2, März 2000 [**0007**]
- ”Reliable Memories with Subline Accesses”, Junsheng Han et al., ISIT2007, Nizza, Frankreich, 24. Juni bis 29. Juni, S. 2531 bis 2535 [**0008**]
- Hassner et al. [**0008**]

Patentansprüche

1. Verfahren zum Aufzeichnen von Eingabedaten in den s-Level-Speicher (**2**) einer Solid-State-Speichereinheit (**1**), wobei $s \geq 2$ ist und das Verfahren Folgendes umfasst:
Codieren von Eingabedatenwörtern in Gruppen mit M Eingabedatenwörtern in Übereinstimmung mit einem ersten und einem zweiten BCH-Code, um für jede Gruppe einen Satz mit M ersten Codewörtern des ersten BCH-Codes zu erzeugen, so dass eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt, wobei der zweite BCH-Code ein Subcode des ersten BCH-Codes ist; und
Aufzeichnen der Sätze der M ersten Codewörter im s-Level-Speicher (**2**).

2. Verfahren nach Anspruch 1, wobei die Summe der M ersten Codewörter in dem Satz das zweite Codewort ist.

3. Verfahren nach Anspruch 1 oder Anspruch 2, wobei jede einer Vielzahl von gewichteten Summen der M ersten Codewörter in dem Satz ein jeweiliges zweites Codewort ist, und wobei Gewichtungskoeffizienten für die Vielzahl von gewichteten Summen durch ein Paritätsprüfmuster definiert werden, das einem dritten linearen Code entspricht.

4. Verfahren nach einem beliebigen der vorstehenden Ansprüche, wobei:
jedes der ersten und der zweiten Codewörter N q-äre Symbole umfasst, wobei $q = p^k$, k ist eine positive ganze Zahl, und p ist eine Primzahl; und
q und s sind u-te beziehungsweise v-te Potenzen einer gemeinsamen Basis r, wobei u und v positive ganze Zahlen sind und $k \geq u$, und $p^{(k/u)v} = s$.

5. Verfahren nach Anspruch 4, wobei $s \neq q$.

6. Verfahren nach Anspruch 5, wobei das Verfahren das Umsetzen jedes q-ären ersten Codeworts in ein r-äres erstes Codewort beinhaltet, bevor sie im s-Level-Speicher (**2**) aufgezeichnet werden.

7. Verfahren nach Anspruch 6, wobei $r \neq s$, wobei das Verfahren die Zuordnung von aufeinanderfolgenden Sätzen von v r-ären Symbolen von jedem r-ären ersten Codewort zu jeweiligen entsprechenden Zuständen des s-Level-Speichers (**2**) in Übereinstimmung mit einem vorbestimmten Zuordnungsschema beinhaltet sowie das Aufzeichnen jedes Satzes von v r-ären Symbolen als der entsprechende Zustand im s-Level-Speicher (2).

8. Verfahren nach Anspruch 5, wobei $q = r \neq s$, wobei das Verfahren die Zuordnung von aufeinanderfolgenden Sätzen von v q-ären Symbolen von jedem q-ären ersten Codewort zu jeweiligen entsprechenden Zuständen des s-Level-Speichers (2) in Übereinstimmung mit einem vorbestimmten Zuordnungsschema beinhaltet sowie das Aufzeichnen jedes Satzes von v q-ären Symbolen als der entsprechende Zustand im s-Level-Speicher (2).

9. Verfahren nach einem beliebigen der vorstehenden Ansprüche 1 bis 6, wobei $r = s > 2$.

10. Verfahren nach einem beliebigen der vorstehenden Ansprüche 1 bis 8, wobei $r = 2$ und $k = u$.

11. Verfahren nach einem beliebigen der vorstehenden Ansprüche, wobei $s > 2$.

12. Verfahren nach einem beliebigen der vorstehenden Ansprüche, das die Aufzeichnung jedes der Sätze von M ersten Codewörtern in einer jeweiligen Schreibadresse des Solid-State-Speichers (**2**) beinhaltet.

13. Computerprogramm, das Programmcodemittel umfasst, um einen Computer zu veranlassen, ein Verfahren nach einem beliebigen der vorstehenden Ansprüche durchführt.

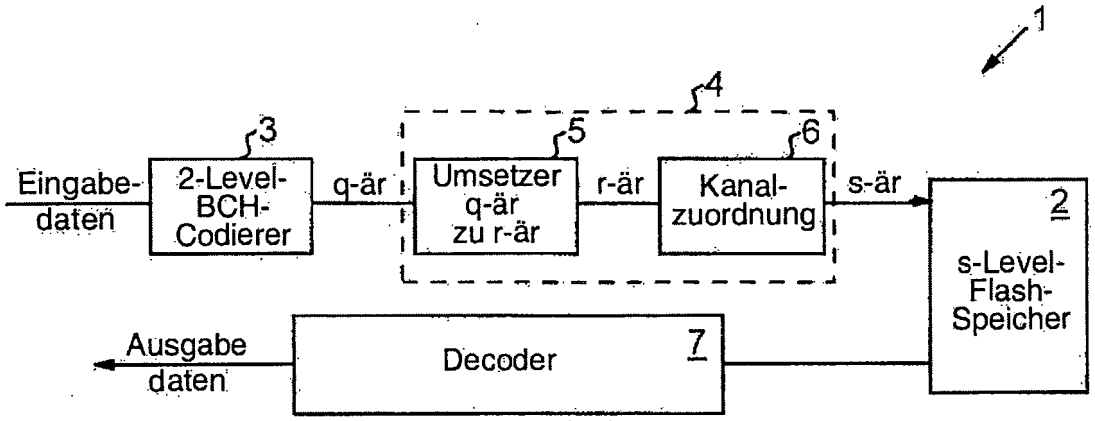
14. Codiervorrichtung zum Codieren von Eingabedaten für die Aufzeichnung im s-Level-Speicher (**2**) einer Solid-State-Speichervorrichtung (**1**), wobei $s \geq 2$, und die Codiervorrichtung Folgendes umfasst:
einen Two-Level-BCH-Codierer (**3**) zum Codieren von Eingabedatenwörtern in Gruppen mit M Eingabedatenwörtern in Übereinstimmung mit dem ersten und dem zweiten BCH-Code, um für jede Gruppe einen Satz mit M ersten Codewörtern des ersten BCH-Codes zu erzeugen, so dass eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt, wobei der zweite BCH-Code ein Subcode des ersten BCH-Codes ist, wobei jedes der ersten und der zweiten Codewörter N q-äre Symbole umfasst, wobei $q \neq s$, $q = p^k$, k ist eine positive ganze Zahl, und p ist eine Primzahl, und wobei q und s u-te

beziehungsweise v-te Potenzen einer gemeinsamen Basis r sind, wobei u und v positive ganze Zahlen sind und $k \geq u$, wobei $p^{(k/u)v} = s$; und
einen Symbolumsetzer (**4**) für das Umsetzen der q-ären Symbole von jedem ersten Codewort zu einem s-ären Alphabet zum Aufzeichnen im s-Level-Speicher (**2**).

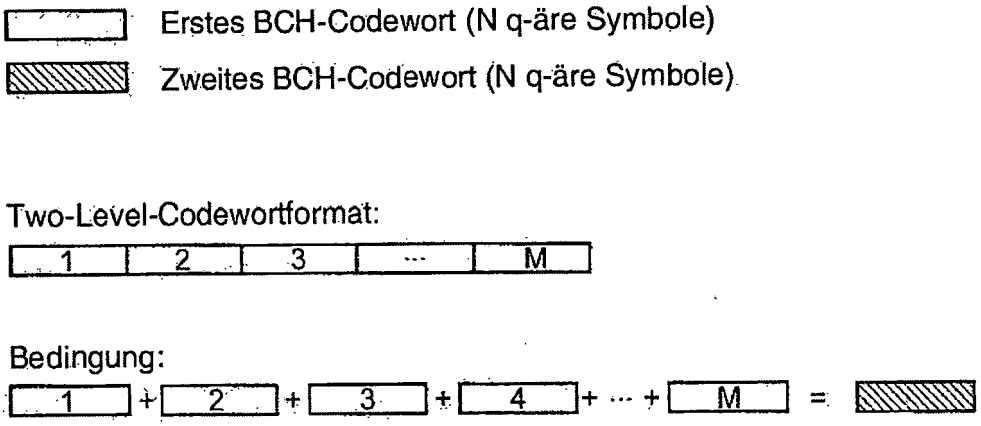
15. Solid-State-Speichereinheit (**1**), die Folgendes umfasst:
s-Level-Solid-State-Speicher (**2**), wobei $s \geq 2$; und
einen Two-Level-BCH-Codierer(**3**) zum Codieren von Eingabedatenwörtern in Gruppen mit M Eingabedatenwörtern in Übereinstimmung mit dem ersten und dem zweiten BCH-Code, um für jede Gruppe einen Satz mit M ersten Codewörtern des ersten BCH-Codes zu erzeugen, so dass eine vorbestimmte lineare Kombination der M ersten Codewörter ein zweites Codewort des zweiten BCH-Codes erzeugt, wobei der zweite BCH-Code ein Subcode des ersten BCH-Codes ist;
wobei die Einheit (**1**) geeignet ist, die Sätze von M ersten Codewörtern im s-Level-Speicher (**2**) aufzuzeichnen.

Es folgen 2 Blatt Zeichnungen

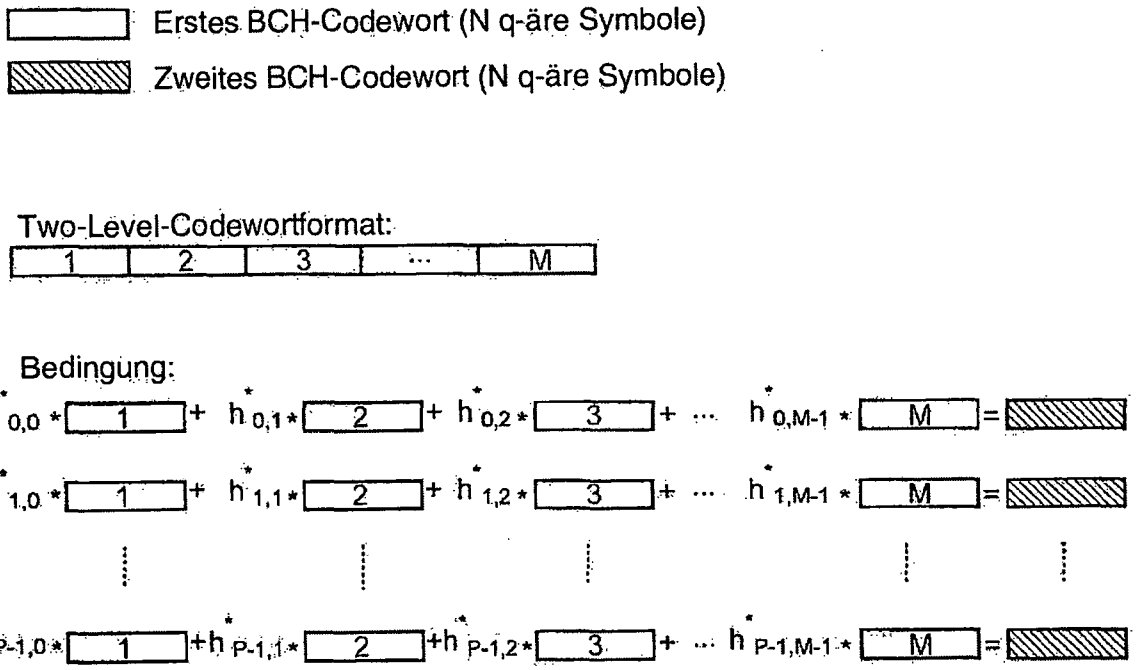
Anhängende Zeichnungen



Figur 1



Figur 2


Erstes BCH-Codewort (N q-äre Symbole)
Zweites BCH-Codewort (N q-äre Symbole)
Two-Level-Codewortformat:
1
2
3
...
M
Bedingung:
h*0,0 * 1 + h*0,1 * 2 + h*0,2 * 3 + ... h*0,M-1 * M =
h*1,0 * 1 + h*1,1 * 2 + h*1,2 * 3 + ... h*1,M-1 * M =
h*P-1,0 * 1 + h*P-1,1 * 2 + h*P-1,2 * 3 + ... h*P-1,M-1 * M =

Figur 3