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

(10) **DE 10 2008 009 851 A1** 2008.09.04

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2008 009 851.5**
(22) Anmeldetag: **11.02.2008**
(43) Offenlegungstag: **04.09.2008**

(51) Int Cl.[8]: ***G06F 12/14*** (2006.01)
***G06F 12/02*** (2006.01)

(30) Unionspriorität:

| | | |
|---|---|---|
| **10-2007-0014980** | **13.02.2007** | **KR** |
| **12/016,737** | **18.01.2008** | **US** |

(71) Anmelder:
**Samsung Electronics Co., Ltd., Suwon, Kyonggi, KR**

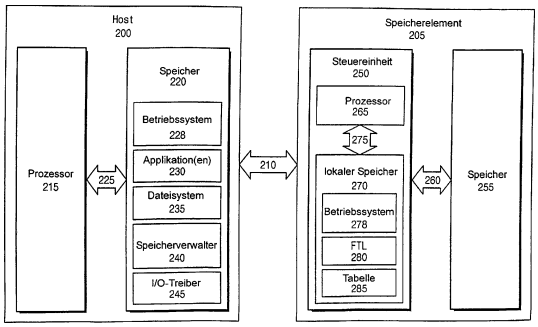
(74) Vertreter:
**Patentanwälte Ruff, Wilhelm, Beier, Dauster & Partner, 70174 Stuttgart**

(72) Erfinder:
**Park, Chan-Ik, Seoul, KR**

**Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen**

(54) Bezeichnung: **Hostdatenverarbeitungssystem, Speicherelement und korrespondierende Betriebsverfahren**

(57) Zusammenfassung: Ein Hostdatenverarbeitungssystem umfasst ein Speicherverwaltungsmodul (240), welches dazu konfiguriert ist, einen Dateilöschbefehl an ein externes Speicherelement (205), welches ein Löschen-vor-Schreiben-Speicherelement (255) umfasst, für wenigstens eine darin gespeicherte Datei zu senden.



## Beschreibung

**[0001]** Die vorliegende Erfindung bezieht sich auf ein Verfahren zum Betreiben eines Hostdatenverarbeitungssystems, ein Verfahren zum Betreiben eines Speicherelements, ein Verfahren zum Betreiben eines Datenverarbeitungssystems, ein Hostdatenverarbeitungssystem und ein Speicherelement.

**[0002]** Datenverarbeitungssysteme können ein Dateisystem verwenden, um Computerdateien zu speichern und zu organisieren, um einen Zugriff darauf einzurichten. Ein Dateisystem kann als ein Satz von abstrakten Datentypen angesehen werden, welche zur Speicherung, Organisation, Manipulation, Navigation, Zugriff und Wiederherstellung von Daten verwendet werden können. Dateisysteme können in drei Typen kategorisiert werden: Diskettendateisysteme, Netzwerkdateisysteme und Dateisysteme für spezielle Zwecke. Diskettendateisysteme werden allgemein zur Speicherung von Dateien auf einem Datenspeicherelement entwickelt. Netzwerkdateisysteme wirken allgemein als Client für ein Ferndatenzugriffsprotokoll. Dateisysteme für spezielle Zwecke beziehen sich allgemein auf ein beliebiges Dateisystem, welches kein Diskettendateisystem oder Netzwerkdateisystem ist. Ein Dateisystem für spezielle Zwecke kann beispielsweise ein System sein, in welchem Dateien dynamisch durch Software angeordnet werden und welches zur Kommunikation zwischen Computerprozessen und/oder temporären Speicherbereichen verwendet werden kann.

**[0003]** Wenn ein Benutzer eine Datei auf einem Computersystem löscht, führt das Dateisystem, welches auf dem Computer abläuft, den Löschbefehl aus und scheint aus Sicht des Benutzers die Datei aus dem Speicher zu entfernen. Tatsächlich lassen herkömmliche Dateisysteme die Dateidaten jedoch im physikalischen Speicher verbleiben. Dies ist unter Bezugnahme auf das File-Allocation-Table(FAT)-Dateisystem in Fig. 1 dargestellt. Ein Hostsystem umfasst ein Applikationsprogramm, welches mit einem Dateisystem **105** kommuniziert. Der Host weist ein angeschlossenes Speichersystem auf. Im Beispiel gemäß Fig. 1 ist das Speicherelement ein Flashspeicherelement, welches eine Flashübersetzungsschicht (FTL) **110** und ein Speicherelement **115**, wie z. B. ein Flashspeicherzellenfeld, aufweist. Die FTL **110** verfolgt die physikalischen Positionen der Speichereinheiten, welche Dateien im Speicherelement **115** zugeordnet sind, so dass das Dateisystem **105** nur auf logische Speichereinheiten zugreift.

**[0004]** Wie aus Fig. 1 hervorgeht, entfernt das Dateisystem **105**, wenn ein Applikationsprogramm **100** verwendet wird, um eine Datei File 1 zu entfernen, den Namen „File 1" aus dem Verzeichnis und platziert einen speziellen Code in das erste Zeichen des FAT-Eintrags der Datei File 1, um anzuzeigen, dass diese Einheit einer Speicherzuordnungen bzw. Speicherzuordnungseinheit für neue Dateien verfügbar ist. Im Speichersystem löscht die FTL **110** die Datei logisch, löscht jedoch nicht die Daten der Datei File 1 aus dem Speicherelement **115**. Stattdessen bleiben die Daten der Datei File 1 im Speicherelement **115** intakt und können sogar wiedergewonnen werden, beispielsweise durch Verwendung des DOS-Befehls „undelete", welcher vom Namen „File 1" gefolgt wird. Daher wird eine Datei durch das Dateisystem **105** und die FTL **110** nur logisch gelöscht, wenn ein Benutzer die Datei löscht. Defragmentieren oder Formatieren des Speicherelements können die durch eine gelöschte Datei besetzten Speichereinheiten im Speicherelement **115** verschieben, die Dateidaten bleiben jedoch im Speicherelement **115** intakt. Mit der Zeit kann das Sichern von neuen Dateien im Speicherelement **115** dazu führen, dass einige oder alle der gelöschten Dateispeichereinheiten überschrieben werden, bis dahin sind jedoch viele Anwendungen verfügbar, welche dazu verwendet werden können, die gelöschten Dateien wiederzugewinnen.

**[0005]** Da einige Dateien private oder sensible Informationen enthalten, kann es ein Benutzer vorziehen, dass diese Dateien im Speichersystem gelöscht werden, so dass sie nicht wieder herstellbar sind, bevor es einem Anderen erlaubt wird, auf das Speichersystem zuzugreifen oder das Speichersystem zu teilen.

**[0006]** Der Erfindung liegt das technische Problem zugrunde, ein Verfahren zum Betreiben eines Hostdatenverarbeitungssystems, ein Verfahren zum Betreiben eines Speicherelements, ein Verfahren zum Betreiben eines Datenverarbeitungssystems, ein Hostdatenverarbeitungssystem und ein Speicherelement bereitzustellen, welche einen unautorisierten Zugriff auf gelöschte Dateien verhindern.

**[0007]** Die Erfindung löst dieses Problem durch Bereitstellung von Verfahren mit den Merkmalen des Patentanspruchs 1, 5 oder 12, eines Hostdatenverarbeitungssystems mit den Merkmalen des Patentanspruchs 14 und eines Speicherelements mit den Merkmalen des Patentanspruchs 19.

**[0008]** Vorteilhafte Weiterbildungen der Erfindung sind in den Unteransprüchen angegeben, deren Wortlaut hiermit durch Bezugnahme in die Beschreibung aufgenommen wird, um unnötige Textwiederholungen zu vermeiden.

**[0009]** Es versteht sich, dass die vorliegende Erfindung auch als System und Computerprogrammprodukt ausgeführt sein kann.

**[0010]** Vorteilhafte, nachfolgend im Detail beschriebene Ausführungsformen der Erfindung sowie die zu deren besserem Verständnis oben erläuterten, her-

kömmlichen Ausführungsbeispiele sind in den Zeichnungen dargestellt. Es zeigt/zeigen:

**[0011]** Fig. 1 ein Blockdiagramm eines Datenverarbeitungssystems zur Darstellung von Operationen, welche eine herkömmliche Dateilöschung betreffen,

**[0012]** Fig. 2 ein Blockdiagramm eines Datenverarbeitungssystems in Übereinstimmung mit beispielhaften Ausführungsformen der vorliegenden Erfindung,

**[0013]** Fig. 3 ein Blockdiagramm zur Darstellung einer Datenstruktur zum Zuordnen von Einheiten einer Speicherzuordnungen in einem Speicherelement zu Indikatoren, die anzeigen, ob die Einheiten einer Speicherzuordnungen gültige oder ungültige Daten enthalten in Übereinstimmung mit einigen Ausführungsformen der vorliegenden Erfindung, und

**[0014]** Fig. 4 bis Fig. 8 Flussdiagramme zur Darstellung der Funktionsweise des Datenverarbeitungssystems gemäß Fig. 2 in Übereinstimmung mit einigen Ausführungsformen der vorliegenden Erfindung.

**[0015]** Die vorliegende Erfindung kann als Verfahren, System und/oder als Computerprogrammprodukt realisiert sein. Entsprechend kann die vorliegende Erfindung vollständig als Hardware und/oder als Software einschließlich Firmware, systemeigene Software, Mikrocode usw. ausgeführt sein. Zudem kann die vorliegende Erfindung in Form eines Computerprogrammprodukts auf einem computernutzbaren oder computerlesbaren Speichermedium ausgeführt sein, welches in dem Medium eingebetteten computernutzbaren oder computerlesbaren Programmcode aufweist, welcher von oder in Verbindung mit einem Befehlsausführungssystem benutzt wird. Im Zusammenhang mit der vorliegenden Beschreibung kann ein computernutzbares oder computerlesbares Medium ein beliebiges Medium sein, welches das Programm zur Verwendung durch oder in Verbindung mit dem Befehlsausführungssystem, der Befehlsausführungsvorrichtung oder dem Befehlsausführungsbauelement enthalten, speichern, kommunizieren, verbreiten oder transportieren kann.

**[0016]** Die computernutzbaren oder computerlesbaren Medien können beispielsweise elektronische, magnetische, optische, elektromagnetische, Infrarot- oder Halbleitersysteme, Vorrichtungen, Geräte oder Verbreitungsmedien umfassen, sind jedoch nicht darauf beschränkt. Speziellere Beispiele einer nicht vollständigen Liste von computerlesbaren Medien können das Folgende umfassen: eine elektrische Verbindung mit einer oder mehreren Leitungen, eine tragbare Computerdiskette, einen Speicher mit direktem Zugriff (RAM), einen Nur-Lese-Speicher (ROM), einen löschbaren und programmierbaren Nur-Lese-Speicher (EPROM oder Flashspeicher), eine optische Leitung und eine CD-ROM. Es versteht sich, dass das computernutzbare oder das computerlesbare Medium sogar Papier oder ein anderes geeignetes Medium sein kann, auf welchem das Programm geschrieben ist und von welchem es, beispielsweise durch einen optischen Abtastvorgang des Papiers oder des anderen Mediums, elektrisch erfassbar ist, dann kompiliert, interpretiert oder falls erforderlich auf andere Weise verarbeitet und dann im Computerspeicher gespeichert werden kann.

**[0017]** Aus Gründen der Darstellung werden hier verschiedene Ausführungsformen der vorliegenden Erfindung unter Bezugnahme auf einen Flashspeicher beschrieben. Es versteht sich, dass das Speicherelement nicht auf einen Flashspeicher begrenzt ist, sondern allgemein als Löschen-vor-Schreiben-Speicherelement (erase before write memory device) implementierbar ist. Daher kann das Datenspeicherelement eine Speicherkarte, ein Solid-State-Drive(SSD)-Element, ein ATA-Buselement, ein serielles ATA(SATA)-Buselement, eine Multi-Media-Card (MMC), ein secure-Digital(SD)-Element, einen Speicherstick, eine Hard-Disk-Drive (HDD), eine Hybrid-Hard-Drive (HHD) und/oder ein Universal-Serial-Bus(USB)-Flashlaufwerk umfassen.

**[0018]** Flashspeicher sind häufig blockweise und seitenweise organisiert. Ein typischer Block kann 32 Seiten, wobei jede Seite 512 Bytes umfasst, oder 64 Seiten aufweisen, wobei jede Seite 2048 Bytes umfasst. Jede Seite weist typischerweise einige zugehörige Bytes auf, welche für eine Fehlerdetektierung und/oder eine Fehlerkorrektur verwendet werden können. Während ein Flashspeicher auf eine direkte Weise gelesen oder programmiert werden kann, muss er blockweise gelöscht werden. Flashspeicher können eine Seitengröße als Speichereinheitsgröße verwenden, um eine Lese- und/oder Schreiboperation auszuführen. Einige Ausführungsformen der vorliegenden Erfindung werden hier im Zusammenhang mit einer Verwendung einer Seite als Lese-/Schreiboperationseinheit eines Speichers und eines Blocks als eine Löschvorgangseinheit eines Speichers beschrieben. Es versteht sich, dass Ausführungsformen der vorliegenden Erfindung nicht auf Seiten und Blöcke als Speicheroperationseinheiten beschränkt sind. Vielmehr kann die Einheit des Speichers zum Ausführen einer Lese- oder einer Schreiboperation allgemein als Lese-/Schreiboperationseinheit spezifiziert werden und die Einheit eines Speichers zum Ausführen eines Löschvorgangs kann allgemein als Löschoperationseinheit spezifiziert werden.

**[0019]** Gemäß einigen Ausführungsformen der vorliegenden Erfindung kann ein Datenverarbeitungssystem, welches ein Hostsystem und ein externes Datenspeicherelement mit einem Löschen-vor-Schreiben-Speicherelement umfasst,

durch Senden eines Dateilöschbefehls vom Host zum Datenspeicherelement für eine oder mehrere darin gespeicherte Dateien betrieben werden. Der Dateilöschbefehl kann eine logische Adresse und ungültig zu machende Daten spezifizieren, welche den der gelöschten Datei zugeordnet sind. Das Datenspeicherelement kann basierend auf der spezifizierten logischen Adresse und den ungültig zu machenden Daten eine oder mehrere Einheiten einer Speicherzuordnungen im Löschen-vor-Schreiben-Speicherelement als ungültige Daten enthaltend identifizieren. Bei einigen Ausführungsformen kann das Datenspeicherelement eine Datenstruktur pflegen bzw. unterhalten, welche physikalischen Adressen für Einheiten einer Speicherzuordnungen im Löschen-vor-Schreiben-Speicherelement Indikatoren zuordnet, die anzeigen, ob die Einheiten einer Speicherzuordnungen ungültige Daten enthalten. Die Datenstruktur kann dazu verwendet werden, Einheiten einer Speicherzuordnung zu markieren, welche den gelöschten Dateien zugeordnet sind, welche ungültige Daten enthalten.

**[0020]** Nun bezugnehmend auf Fig. 2 umfasst ein Datenverarbeitungssystem einen Host **200** und ein Speicherelement **205**, welche über eine Schnittstelle **210** gekoppelt sind. Die Schnittstelle **210** kann eine standardisierte Schnittstelle, wie beispielsweise eine ATA-, SATA-, PATA-, USB-, SCSI-, ESDI-, IEEE-1394-, IDE- und/oder eine Kartenschnittstelle sein. Der Host **200** umfasst einen Prozessor **215**, welcher über einen Adressen-/Datenbus **225** mit einem Speicher **220** kommuniziert. Der Prozessor **215** kann beispielsweise ein kommerziell verfügbarer oder ein kundenspezifischer Mikroprozessor sein. Der Speicher **220** ist für ein oder mehrere Speicherelemente exemplarisch, welche die Software und Daten enthalten, welche verwendet werden, um das Datenverarbeitungssystem in Übereinstimmung mit einigen Ausführungsformen der Erfindung zu betreiben. Der Speicher **220** kann die folgenden Bauelementtypen Cache, ROM, PROM, EPROM, EEPROM, Flash, SRAM und DRAM umfassen, ist aber nicht darauf beschränkt.

**[0021]** Wie aus Fig. 2 hervorgeht, kann der Speicher **220** fünf oder mehr Kategorien von Software und/oder Daten enthalten, beispielsweise ein Betriebssystem **228**, Applikation(en) **230**, ein Dateisystem **235**, einen Speicherverwalter **240** und I/O-Treiber **245**. Das Betriebssystem **228** steuert allgemein den Betrieb des Hosts **200**. Insbesondere kann das Betriebssystem **228** die Software- und/oder Hardwareressourcen des Hosts **200** verwalten und kann die Ausführung von Programmen durch den Prozessor **215** koordinieren. Die Applikation(en) **230** repräsentieren verschiedene Applikationsprogramme, die im Host **200** ablaufen können. Das Dateisystem **235** ist das System, welches zum Speichern und Organisieren von Computerdateien und/oder Daten im Speicher **220** und/oder in Speicherpositionen, wie dem Speicherelement **205**, verwendet wird. Das verwendete Dateisystem **235** kann insbesondere auf dem Betriebssystem **228** basieren, welches im Host **200** läuft. Der Speicherverwalter **240** kann Speicherzugriffsoperationen auf den Speicher **220** und/oder Operationen verwalten, welche ein externes Bauelement, wie beispielsweise das Speicherelement **205**, betreffen. Die I/O-Treiber **245** können zum Übertragen von Informationen zwischen dem Host **200** und einem anderen Gerät, wie z. B. dem Speicherelement **205**, einem Computersystem oder einem Netzwerk, wie z. B. dem Internet, verwendet werden.

**[0022]** In Übereinstimmung mit verschiedenen Ausführungsformen der vorliegenden Erfindung kann der Host **200** ein persönlicher digitaler Assistent (PDA), ein Computer, ein digitaler Audioplayer, eine digitale Kamera und/oder ein mobiles Endgerät sein.

**[0023]** Das Speicherelement **205** umfasst eine Steuereinheit **250**, welche über einen Adressen-/Datenbus **260** mit einem Speicher **255** kommuniziert. Der Speicher **255** kann eine Vielzahl von verschiedenen Speichertypen umfassen und kann allgemein als Löschen-vor-Schreiben-Speicher beschrieben werden. Daher kann das Speicherelement **405** in Übereinstimmung mit verschiedenen Ausführungsformen der vorliegenden Erfindung ein Solid-State-Drive(SSD)-Element, ein ATA-Buselement, ein serielles ATA(SATA)-Buselement, eine Multi-Media-Card (MMC), ein secure-Digital(SD)-Element, ein Speicherstick, eine Hard-Disk-Drive (HDD), eine Hybrid-Hard-Drive (HHD) und/oder ein Universal-Serial-Bus(USB)-Flashlaufwerk sein. Die Steuereinheit **250** umfasst einen Prozessor **265**, welcher über einen Adressen-/Datenbus **275** mit einem lokalen Speicher **270** kommuniziert. Der Prozessor **265** kann beispielsweise ein kommerziell verfügbarer oder ein kundenspezifischer Mikroprozessor sein. Der lokale Speicher **270** ist exemplarisch für ein oder mehrere Speicherelemente, welche die Software und Daten enthalten, welche für den Betrieb des Speicherelements **205** in Übereinstimmung mit einigen Ausführungsformen der vorliegenden Erfindung verwendet werden. Der lokale Speicher **270** kann die folgenden Bauelementtypen Cache, ROM, PROM, EPROM, EEPROM, Flash, SRAM und DRAM umfassen, ist aber nicht darauf beschränkt.

**[0024]** Wie aus Fig. 2 hervorgeht, kann der lokale Speicher **270** drei oder mehr Kategorien von Software und/oder Daten umfassen, beispielsweise ein Betriebssystem **278**, ein Flashübersetzungsschicht(FTL)-Modul **280** und eine Tabelle **285**. Das Betriebssystem **278** steuert allgemein die Operation des Speicherelements **205**. Insbesondere kann das Betriebssystem **278** die Software- und/oder Hardwareressourcen des Speicherelements **205** verwalten und kann die Ausführung von Programmen durch

den Prozessor **265** koordinieren. Das FTL-Modul **280** kann in Flashspeicherelementen verwendet werden. Wie oben beschrieben ist, wird ein Flashchip blockweise gelöscht. Die typische Lebensdauer eines Flashspeichers beträgt ungefähr 100,000 Löschoperationen je Block. Um zu vermeiden, dass ein Teil des Flashspeichers früher als ein anderer Teil abgenutzt wird, werden Flashbauelemente allgemein dazu entworfen, Löschperioden über den Speicher zu verteilen, was auch als „Abnutzungsverteilung" bezeichnet werden kann. Das FTL-Modul **280** kann als Schnittstelle zwischen dem Dateisystem **235** und der Position von Dateien/Daten im Speicher **255** verwendet werden, so dass das Dateisystem **235** die aktuelle aufgrund der Abnutzungsverteilung eingenommene Position der Dateien/Daten im Speicher **255** nicht verfolgen muss. Die Tabelle **285** kann durch das FTL-Modul **280** gepflegt werden und kann dazu verwendet werden, physikalischen Adressen für Einheiten einer Speicherzuordnung im Speicher **255** Indikatoren zuzuordnen, die anzeigen, ob die Einheiten einer Speicherzuordnungen ungültige Daten enthalten.

**[0025]** Ein Beispiel für die Tabelle **285** ist in Fig. 3 für einen Flash-Typ-Speicher dargestellt, in welchem eine Seite als Einheit einer Speicherzuordnung verwendet wird und ein Block vier Seiten umfasst. Wie aus Fig. 3 hervorgeht, ordnet die Tabelle **285** die physikalischen Adressen der Seiten des Flashspeichers **255** den logischen Adressen zu, welche im Dateisystem **235** verwendet werden. Des Weiteren umfasst die Tabelle **285** eine Spalte, welche anzeigt, ob jede bestimmte Seite im Flashspeicher **255** ungültige oder gültige Daten aufweist. Im dargestellten Beispiel enthält der Block von Seiten, welche die logischen Adressen 0 bis 3 aufweisen, ungültige Daten und kann daher gelöscht werden. Die Tabelle **285** kann verwendet werden, um eine Löschoperation zu triggern, wenn alle Seiten in einem Block als ungültige Daten enthaltend bestimmt werden. Herkömmlicherweise kann beispielsweise, wenn eine zweite Schreiboperation mit einer logischen Adresse Seitenadresse 0 versucht wurde, darauf geschlossen werden, dass die logische Seitenadresse 0 ungültige Daten enthält. Es ist jedoch nicht klar, ob die logischen Seitenadressen 1 bis 3 ebenfalls ungültige Daten enthalten. Daher werden die Daten in den logischen Seitenadressen 1 bis 3 an eine andere Stelle kopiert, um die logische Seitenadresse 0 freizugeben, so dass der gesamte Block, welcher die logischen Seitenadressen 0 bis 3 enthält, gelöscht werden kann. Diese Kopieroperation kann unnötig sein, wenn die logischen Seitenadressen 1 bis 3 ungültige Daten enthalten. Die Tabelle **285** kann einen Indikator bereitstellen, der anzeigt, welche der Seiten ungültige Daten enthalten, um die unnötigen Kopieroperationen zu reduzieren, wie oben beschrieben ist. Obwohl hier als Tabelle dargestellt, versteht es sich, dass die Tabelle **285** in Übereinstimmung mit verschiedenen Ausführungsformen der Erfindung auch als andere Datenstrukturtypen implementiert werden kann.

**[0026]** Obwohl Fig. 2 eine Datenverarbeitungssystemsoftwarearchitektur in Übereinstimmung mit einigen Ausführungsformen der vorliegenden Erfindung zeigt, versteht es sich, dass die vorliegende Erfindung nicht auf eine solche Konfiguration beschränkt ist, sondern es ermöglicht, eine beliebige Konfiguration zu umfassen, welche dazu in der Lage ist, die hier beschriebenen Operationen auszuführen.

**[0027]** Der Computerprogrammcode zum Ausführen der oben unter Bezugnahme auf Fig. 2 beschriebenen Operationen der Bauelemente und/oder Systeme kann zur Vereinfachung der Abwicklung in einer höheren Programmiersprache wie JAVA, C und/oder C++ geschrieben sein. Zusätzlich kann der Computerprogrammcode zum Ausführen der Operationen von Ausführungsformen der vorliegenden Erfindung auch in anderen Programmiersprachen wie beispielsweise übersetzten Sprachen geschrieben werden, ist aber nicht darauf beschränkt. Einige Module oder Routinen können in Assemblersprache oder sogar als Mikrocode geschrieben werden, um die Leistungsfähigkeit und/oder die Speicherausnutzung zu verbessern. Es versteht sich, dass die Funktionalität eines beliebigen oder von allen Programmmodulen auch unter Verwendung von diskreten Hardwarekomponenten, einem oder mehreren applikationsspezifischen integrierten Schaltkreisen (ASICs) oder einem programmierten digitalen Signalprozessor oder Mikrocontroller implementiert werden kann.

**[0028]** Die vorliegende Erfindung wird nachfolgend unter Bezugnahme auf Nachrichtenfluss-, Flussdiagramm- und/oder Blockdiagrammdarstellungen von Verfahren, Systemen, Bauelementen und/oder Computerprogrammprodukten in Übereinstimmung mit einigen Ausführungsformen der Erfindung beschrieben. Diese Nachrichtenfluss-, Flussdiagramm- und/oder Blockdiagrammdarstellungen zeigen weiter Operationen zum Betreiben eines Datenverarbeitungssystems, welches ein externes Datenspeicherelement umfasst. Selbstverständlich kann jede(r) Nachricht/Block der Nachrichtenfluss-, Flussdiagramm- und/oder Blockdiagrammdarstellungen und Kombinationen von Nachrichten/Blöcken in den Nachrichtenfluss-, Flussdiagramm- und/oder Blockdiagrammdarstellungen als Computerprogrammanweisungen und/oder Hardwareoperationen implementiert werden. Diese Computerprogrammanweisungen können einem Prozessor eines allgemeinen Rechners, eines speziellen Rechners oder anderen programmierbaren Datenverarbeitungsvorrichtungen zur Verfügung gestellt werden, um eine Maschine zu implementieren, so dass die Anweisungen, welche über den Prozessor des Rechners oder den anderen programmierbaren Datenverarbeitungsvorrichtungen ausgeführt werden, Mittel zum Implementieren der Funktionen bilden, welche durch den oder die Nach-

richtenfluss-, Flussdiagramm- und/oder Blockdiagrammblock oder -blöcke spezifiziert werden.

**[0029]** Diese Computerprogrammanweisungen können auch in einem computerverwendbaren oder computerlesbaren Speicher gespeichert werden, welcher einen Rechner oder eine andere programmierbare Datenverarbeitungsvorrichtung veranlasst, auf eine vorgegebene Weise zu funktionieren, so dass im computerverwendbaren oder computerlesbaren Speicher gespeicherten Anweisungen einen Gegenstand erzeugen, welcher Anweisungen umfasst, welche die Funktionen implementieren, welche durch den oder die Nachrichtenfluss-, Flussdiagramm- und/oder Blockdiagrammblock oder -blöcke spezifiziert werden.

**[0030]** Die Computerprogrammanweisungen können in den Computer oder andere programmierbare Datenverarbeitungsvorrichtungen geladen werden, um eine Reihe von Betriebsschritten zu bewirken, welche in einem Computer oder in einer anderen programmierbaren Vorrichtung ausgeführt werden können, um einen computerimplementierten Prozess zu erzeugen, so dass die Anweisungen, welche auf dem Computer oder auf der anderen programmierbaren Vorrichtung ausgeführt werden, Schritte zum Implementieren der Funktionen zur Verfügung stellen, welche durch den oder die Nachrichtenfluss-, Flussdiagramm- und/oder Blockdiagrammblock oder -blöcke spezifiziert werden.

**[0031]** Bezugnehmend auf Fig. 4 beginnen die Operationen mit einem Block **400**, in welchem das Hostdatenverarbeitungssystem **200** einen Dateilöschbefehl für eine oder mehrere Dateien an das externe Speicherelement **205** sendet, welches ein Löschen-vor-Schreiben-Speicherelement umfasst, wie beispielsweise ein Flashspeicherelement. in Übereinstimmung mit in Fig. 5 dargestellten verschiedenen Ausführungsformen der vorliegenden Erfindung kann im Block **500** eine Dateilöschoperation im Host **200** detektiert werden. Dies kann beispielsweise dadurch erfolgen, dass detektiert wird, dass Metadaten, welche dem Dateisystem zugeordnet sind, mit einem Löschcode für eine gelöschte Datei aktualisiert worden sind. In Reaktion auf das Detektieren der Dateilöschoperation im Host **500** kann der Dateilöschbefehl an das externe Speicherelement **205** gesendet werden. Bei einigen Ausführungsformen kann der Dateilöschbefehl eine logische Adresse und ungültig zu machende Daten spezifizieren, welche der gelöschten Datei zugeordnet sind.

**[0032]** Bezugnehmend auf Fig. 6 beginnt eine beispielhafte Dateilöschoperation im externen Speicherelement **205** in einem Block **600**, in welchem der Dateilöschbefehl, welcher die logische Adresse und ungültig zu machende Daten für eine oder mehr Dateien spezifiziert, vom Host **200** empfangen wird. Das Speicherelement **205** identifiziert basierend auf der spezifizierten logischen Adresse und den ungültig zu machenden Daten eine oder mehrere Einheiten einer Speicherzuordnungen im Speicher **255** als ungültige Daten enthaltend. In einigen in Fig. 7 dargestellten Ausführungsformen kann das FTL-Modul **280** eine Datenstruktur enthalten, wie beispielsweise die in Fig. 3 dargestellte Tabelle **285**, welche im Block **700** logischen Adressen physikalische Adressen der Einheiten einer Speicherzuordnung im Speicher **255** zuordnet. Die Datenstruktur kann auch einen Indikator umfassen, der anzeigt, ob die verschiedenen Einheiten einer Speicherzuordnung ungültige Daten enthalten. Wenn eine physikalische Adresse einer Einheit einer Speicherzuordnung als einer gelöschten Datei zugeordnet identifiziert wird, kann das FTL-Modul **280** die Datenstruktur aktualisieren, um im Block **705** anzuzeigen, dass die identifizierte Einheit einer Speicherzuordnung ungültige Daten enthält.

**[0033]** Wenn verschiedene Speicheroperationen mit dem Speicherelement ausgeführt werden, kann es wünschenswert sein, eine „Abfallsammeloperation" auszuführen, um größere Blöcke von freiem zusammenhängendem Speicher zu bilden. In Übereinstimmung mit einigen Ausführungsformen der vorliegenden Erfindung kann das FTL-Modul **280** die Tabelle **285** dazu verwenden, zu bestimmen, wann Speichereinheiten, die ungültige Daten enthalten, zu sammeln sind, anstatt zu warten, bis das Betriebssystem **228** des Hosts **200** oder das Betriebssystem **278** des Speicherelements **205** eine periodische Abfallsammeroperation triggert. Bezugnehmend auf Fig. 8 kann das FTL-Modul **280** im Block **800** durch Auswerten des ungültigen Datenfelds für jede der Lese-/Schreiboperationseinheiten, siehe Fig. 3, bestimmen, ob alle der Lese-/Schreiboperationseinheiten, wie z. B. Seiten für ein Flashspeicherelement, in einer Löschoperationseinheit, wie z. B. einem Block für ein Flashspeicherelement, ungültige Daten enthalten. Im Block **805** kann eine Löschoperation für die Löschoperationseinheit ausgeführt werden, wenn alle Lese-/Schreiboperationseinheiten als ungültige Daten enthaltend markiert sind. Auf diese Weise können die aktuellen physikalischen Dateidaten gelöscht werden, sobald eine Löschoperationseinheit zum Löschen bereit ist. Dies kann für Applikationen wünschenswert sein, welche persönliche oder sensible Daten betreffen, da das physikalische Löschen einer Datei aus einem Speicher eines Speicherelements schneller durchgeführt wird, als wenn darauf gewartet wird, bis die Datei durch die Möglichkeit von mehrfachen Dateischreiboperationen gelöscht wird, welche im Speicherelement ausgeführt werden.

**[0034]** Die Flussdiagramme gemäß Fig. 4 bis Fig. 8 zeigen Architektur, Funktionsweise und Operationen von einigen Ausführungsformen von Verfahren, Systemen und Computerprogrammprodukten zum Betreiben eines Datenverarbeitungssystems, welches

ein externes Datenspeicherelement umfasst. In diesem Zusammenhang repräsentiert jeder Block ein Modul, Segment oder Teil von Codes, welche eine oder mehrere ausführbare Anweisungen zum Implementieren der spezifizierten logischen Funktion(en) enthalten. Es sei angemerkt, dass in anderen Implementierungen die in den Blöcken angegebenen Funktionen außerhalb der in Fig. 4 bis Fig. 8 angegebenen Reihenfolgen ausgeführt werden können. Zwei in Reihenfolge dargestellte Blöcke können beispielsweise tatsächlich im Wesentlichen gleichzeitig ausgeführt werden oder die Blöcke können in Abhängigkeit von der betroffenen Funktionalität auch in umgekehrter Reihenfolge ausgeführt werden.

**Patentansprüche**

1. Verfahren zum Betreiben eines Hostdatenverarbeitungssystems mit den Schritten:
– Senden eines Dateilöschbefehls an ein externes Speicherelement (**205**), welches ein Löschen-vor-Schreiben-Speicherelement (**255**) umfasst, für wenigstens eine darin gespeicherte Datei.

2. Verfahren nach Anspruch 1, weiter umfassend:
– Detektieren einer Dateilöschoperation im Host und
– Senden des Dateilöschbefehls an das externe Speicherelement in Reaktion auf das Detektieren der Dateilöschoperation.

3. Verfahren nach Anspruch 2, wobei das Detektieren der Dateilöschoperation umfasst, dass Metadaten, welche einem Dateisystem zugeordnet sind, mit einem Löschcode für eine gelöschte Datei aktualisiert worden sind.

4. Verfahren nach einem der Ansprüche 1 bis 3, wobei der Dateilöschbefehl eine logische Adresse und ungültig zu machende Daten spezifiziert, welche der gelöschten Datei zugeordnet sind.

5. Verfahren zum Betreiben eines Speicherelements (**205**), welches ein Löschen-vor-Schreiben-Speicherelement (**255**) aufweist, mit den Schritten:
– Empfangen eines Dateilöschbefehls von einem Host (**200**), welcher eine logische Adresse und ungültig zu machende Daten für wenigstens eine im Löschen-vor-Schreiben-Speicherelement (**255**) gespeicherte Datei spezifiziert, und
– Identifizieren von wenigstens einer Einheit einer Speicherzuordnung im Löschen-vor-Schreiben-Speicherelement (**255**) als ungültige Daten enthaltend basierend auf der spezifizierten logischen Adresse und den ungültig zu machenden Daten.

6. Verfahren nach Anspruch 5, wobei das Identifizieren der wenigstens einen Einheit einer Speicherzuordnung im Löschen-vor-Schreiben-Speicherelement als ungültige Daten enthaltend umfasst:
– Pflegen einer Datenstruktur, welche physikalischen Adressen für Einheiten einer Speicherzuordnungen im Löschen-vor-Schreiben-Speicherelement Indikatoren zuordnet, die anzeigen, ob die Einheiten einer Speicherzuordnungen ungültige Daten enthalten, und
– Identifizieren der wenigstens einen Einheit einer Speicherzuordnung als ungültige Daten enthaltend in der Datenstruktur.

7. Verfahren nach Anspruch 5 oder 6, wobei das Speicherelement ein Flashübersetzungsschicht(FTL)-Modul umfasst, welches dazu konfiguriert ist, die Datenstruktur zu verwalten.

8. Verfahren nach einem der Ansprüche 5 bis 7, wobei das Löschen-vor-Schreiben-Speicherelement mittels Lese-/Schreiboperations-einheiten und Löschoperationseinheiten organisiert ist, wobei das Verfahren weiter umfasst:
– Bestimmen, ob alle Lese-/Schreiboperationseinheiten in einer der Löschoperationseinheiten ungültige Daten enthalten, und
– Ausführen einer Löschoperation mit der einen der Löschoperationseinheiten, wenn alle Lese-/Schreiboperationseinheiten in der einen Löschoperationseinheit ungültige Daten enthalten.

9. Verfahren nach Anspruch 8, wobei das Bestimmen, ob alle Lese-/Schreiboperationseinheiten in der einen der Löschoperationseinheiten ungültige Daten enthalten, umfasst:
– Bestimmen, dass alle Lese-/Schreiboperationseinheiten in der einen der Löschoperationseinheiten in der Datenstruktur als ungültige Daten enthaltend angezeigt werden.

10. Verfahren nach Anspruch 8 oder 9, wobei die Löschoperation mit der einen der Löschoperationseinheiten getrennt von einer periodischen Abfallsammeloperation ausgeführt wird, welche in dem Löschen-vor-Schreiben-Speicherelement ausgeführt wird.

11. Verfahren nach einem der Ansprüche 8 bis 10, wobei die Datenstruktur logischen Adressen, welche im Host verwendet werden, physikalische Adressen für Lese-/Schreiboperationseinheiten im Löschen-vor-Schreiben-Speicherelement zuordnet.

12. Verfahren zum Betreiben eines Datenverarbeitungssystems mit den Schritten:
– Senden eines Dateilöschbefehls an ein externes Speicherelement (**205**), welches ein Löschen-vor-Schreiben-Speicherelement (**255**) umfasst, wobei der Dateilöschbefehl eine logische Adresse und ungültig zu machende Daten für wenigstens eine im Löschen-vor-Schreiben-Speicherelement gespeicherte Datei spezifiziert, und
– Identifizieren von wenigstens einer Einheit einer

Speicherzuordnung im Löschen-vor-Schreiben-Speicherelement (**255**) als ungültige Daten enthaltend basierend auf der spezifizierten logischen Adresse und den ungültig zu machenden Daten.

13. Verfahren nach Anspruch 12, wobei das Identifizieren der wenigstens einen Einheit einer Speicherzuordnung im Löschen-vor-Schreiben-Speicherelement als ungültige Daten enthaltend umfasst:
– Pflegen einer Datenstruktur, welche physikalischen Adressen für Einheiten einer Speicherzuordnungen im Löschen-vor-Schreiben-Speicherelement Indikatoren zuordnet, die anzeigen, ob die Einheiten einer Speicherzuordnungen ungültige Daten enthalten, und
– Identifizieren der wenigstens einen Einheit einer Speicherzuordnung als ungültige Daten enthaltend in der Datenstruktur.

14. Hostdatenverarbeitungssystem, umfassend:
– ein Speicherverwaltungsmodul (**240**), welches dazu konfiguriert ist, einen Dateilöschbefehl an ein externes Speicherelement (**205**), welches ein Löschen-vor-Schreiben-Speicherelement (**255**) umfasst, für wenigstens eine darin gespeicherte Datei zu senden.

15. System nach Anspruch 14, wobei das Speicherverwaltungsmodul weiter dazu konfiguriert ist, eine Dateilöschoperation im Host zu detektieren und den Dateilöschbefehl in Reaktion auf das Detektieren der Dateilöschoperation an das externe Speicherelement zu senden.

16. System nach Anspruch 15, wobei das System weiter ein Dateisystem umfasst, wobei das Speicherverwaltungsmodul weiter dazu konfiguriert ist, zu detektieren, dass Metadaten, welche dem Dateisystem zugeordnet sind, mit einem Löschcode für eine gelöschte Datei aktualisiert sind.

17. System nach Anspruch 15 oder 16, wobei der Dateilöschbefehl eine logische Adresse und ungültig zu machende Daten spezifiziert, welche der gelöschten Datei zugeordnet sind.

18. System nach einem der Ansprüche 14 bis 17, wobei das Speicherelement eine Speicherkarte und/oder ein Universal-Serial-Bus(USB)-Flashlaufwerk ist und/oder das Hostdatenverarbeitungssystem ein persönlicher digitaler Assistent, ein Computer, ein digitaler Audioplayer, eine Digitalkamera und/oder ein mobiles Endgerät ist.

19. Speicherelement (**205**), umfassend:
– ein Löschen-vor-Schreiben-Speicherelement (**255**),
– ein Speicherverwaltungsmodul (**250**), welches dazu konfiguriert ist, einen Dateilöschbefehl von einem Host (**200**) zu empfangen, welcher eine logische Adresse und ungültig zu machende Daten für wenigstens eine im Löschen-vor-Schreiben-Speicherelement (**255**) gespeicherte Datei spezifiziert, und wenigstens eine Einheit einer Speicherzuordnung im Löschen-vor-Schreiben-Speicherelement als ungültige Daten enthaltend basierend auf der spezifizierten logischen Adresse und den ungültig zu machenden Daten zu identifizieren.

20. Speicherelement nach Anspruch 19, weiter umfassend:
– eine Datenstruktur, welche physikalischen Adressen für Einheiten einer Speicherzuordnungen im Löschen-vor-Schreiben-Speicherelement Indikatoren zuordnet, die anzeigen, ob die Einheiten einer Speicherzuordnungen ungültige Daten enthalten, und
– ein Übersetzungsmodul, das dazu konfiguriert ist, die wenigstens eine Einheit einer Speicherzuordnung als ungültige Daten enthaltend in der Datenstruktur zu identifizieren.

21. Speicherelement nach Anspruch 20, wobei das Löschen-vor-Schreiben-Speicherelement mittels Lese-/Schreiboperationseinheiten und Löschoperationseinheiten organisiert ist, wobei das Speicherverwaltungsmodul weiter dazu konfiguriert ist, zu bestimmen, ob alle Lese-/Schreiboperationseinheiten in einer der Löschoperationseinheiten ungültige Daten enthalten, und eine Löschoperation mit der einen der Löschoperationseinheiten auszuführen, wenn alle Lese-/Schreiboperationseinheiten in der einen Löschoperationseinheit ungültige Daten enthalten.

22. Speicherelement nach Anspruch 21, wobei das Speicherverwaltungsmodul weiter dazu konfiguriert ist, zu bestimmen, dass alle Lese-/Schreiboperationseinheiten in der einen der Löschoperationseinheiten in der Datenstruktur als ungültige Daten enthaltend angezeigt werden.

23. Speicherelement nach Anspruch 21 oder 22, wobei das Speicherverwaltungsmodul weiter dazu konfiguriert ist, die Löschoperation mit der einen der Löschoperationseinheiten getrennt von einer periodischen Abfallsammeloperation auszuführen, welche in dem Löschen-vor-Schreiben-Speicherelement ausgeführt wird.

24. Speicherelement nach einem der Ansprüche 21 bis 23, wobei die Datenstruktur logischen Adressen, welche im Host verwendet werden, den physikalischen Adressen für Lese-/Schreiboperationseinheiten im Löschen-vor-Schreiben-Speicherelement zuordnet.

25. Datenverarbeitungssystem nach einem der Ansprüche 19 bis 24, wobei das Speicherelement eine Speicherkarte, ein Solid-State-Drive(SSD)-Element, ein ATA-Buselement, ein serielles ATA(SATA)-Buselement, eine Multi-Media-Card (MMC), ein Secure-Digital(SD)-Element, ein Speicherstick, eine

Hard-Disk-Drive (HDD), eine Hybrid-Hard-Drive (HHD) und/oder ein Universal-Serial-Bus(USB)-Flashlaufwerk ist.

Es folgen 5 Blatt Zeichnungen

**Anhängende Zeichnungen**

Fig. 1
(Stand der Technik)

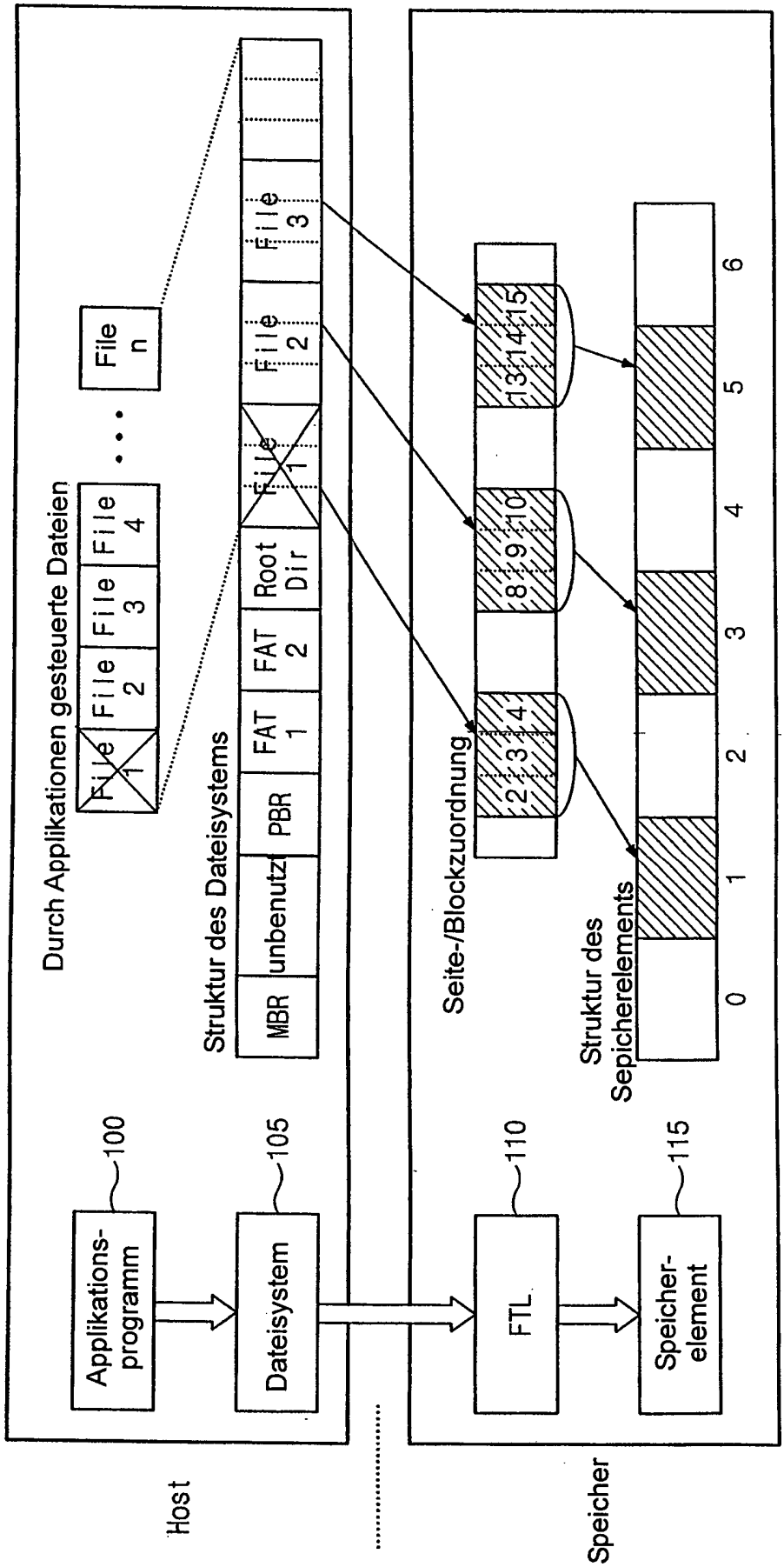


Fig. 2

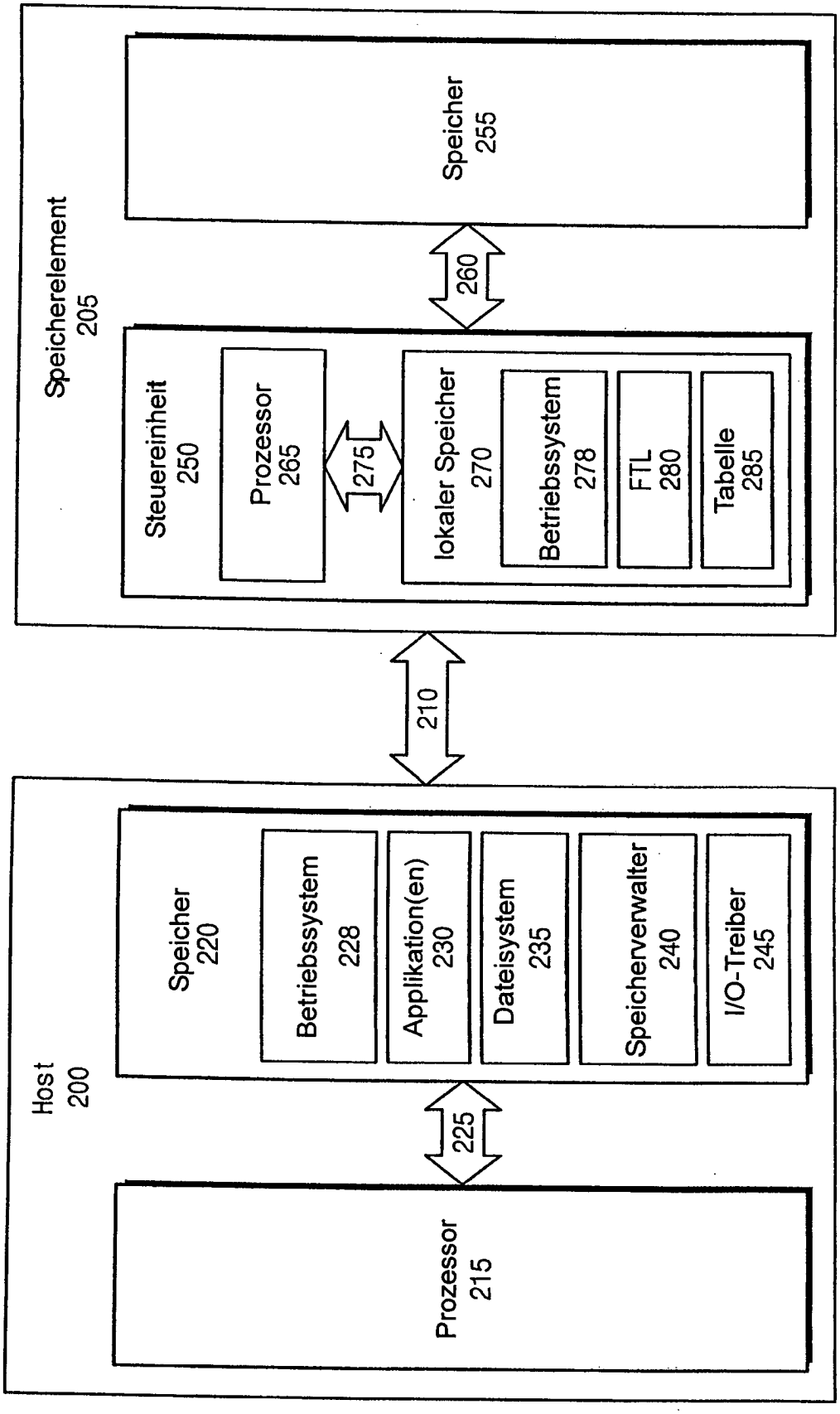

Host
200
Prozessor
215
225
Speicher
220
Betriebssystem
228
Applikation(en)
230
Dateisystem
235
Speicherverwalter
240
I/O-Treiber
245
210
Speicherelement
205
Steuereinheit
250
Prozessor
265
275
lokaler Speicher
270
Betriebssystem
278
FTL
280
Tabelle
285
260
Speicher
255

Fig. 3

Block

| logische Adresse | physikalische Adresse | ungültig Y/N |
|---|---|---|
| 0 | 1 | Y |
| 1 | 2 | Y |
| 2 | 3 | Y |
| 3 | 4 | Y |
| 4 | 5 | N |
| 5 | 6 | N |
| 6 | 7 | N |

Fig. 4

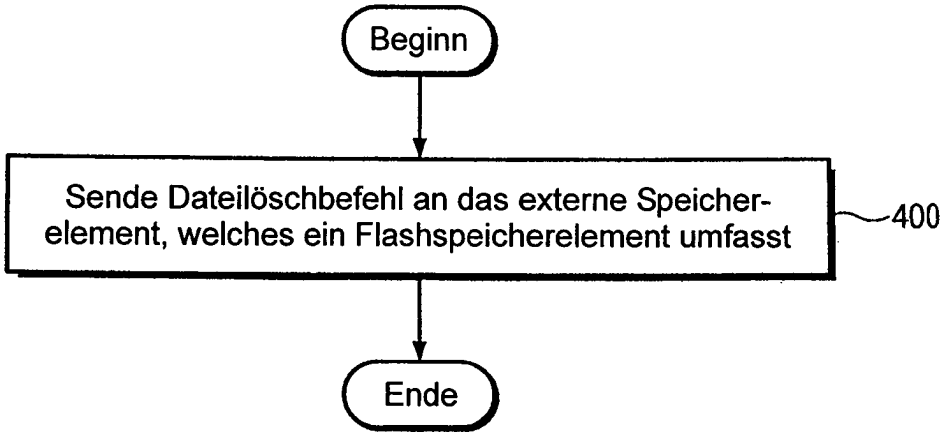


Fig. 5

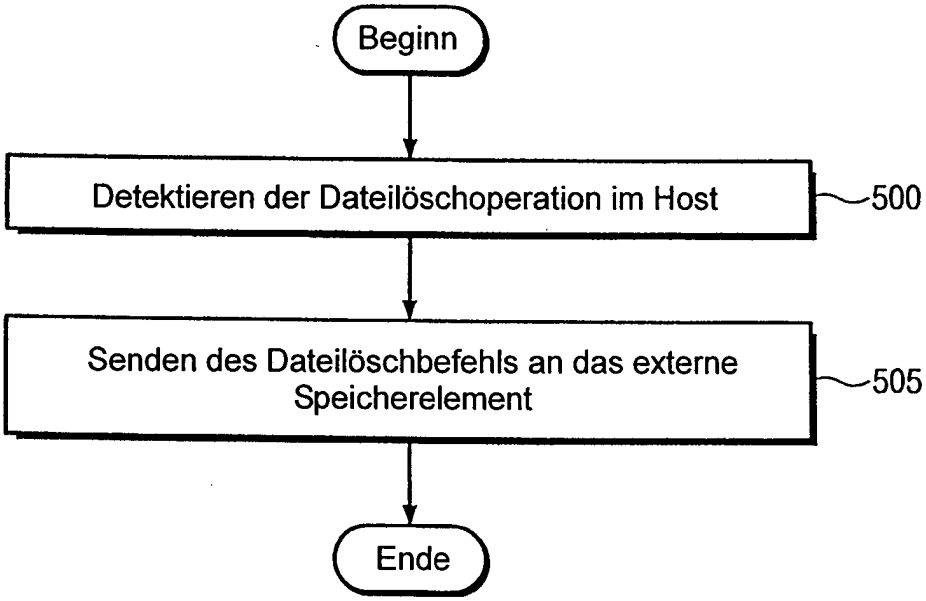

Beginn
Detektieren der Dateilöschoperation im Host
500
Senden des Dateilöschbefehls an das externe Speicherelement
505
Ende

Fig. 6

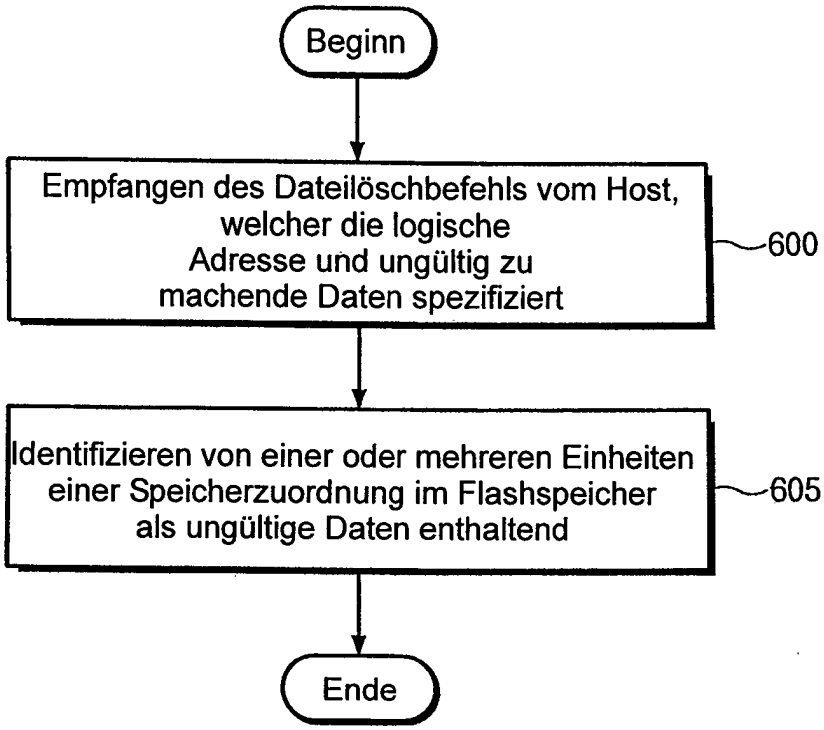

Beginn
Empfangen des Dateilöschbefehls vom Host, welcher die logische Adresse und ungültig zu machende Daten spezifiziert
600
Identifizieren von einer oder mehreren Einheiten einer Speicherzuordnung im Flashspeicher als ungültige Daten enthaltend
605
Ende

Fig. 7

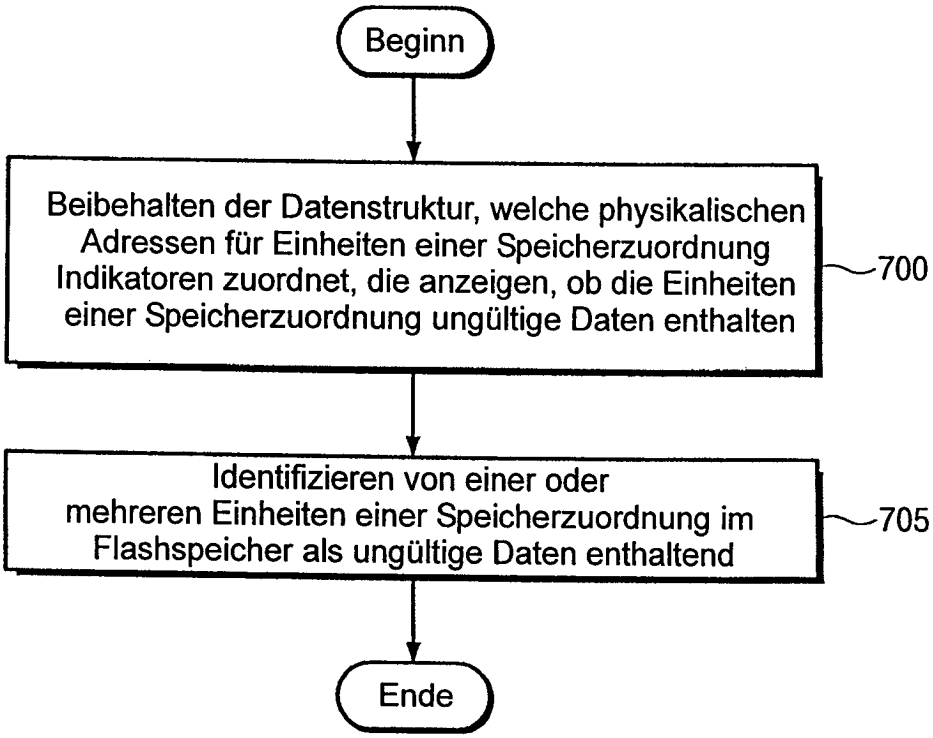
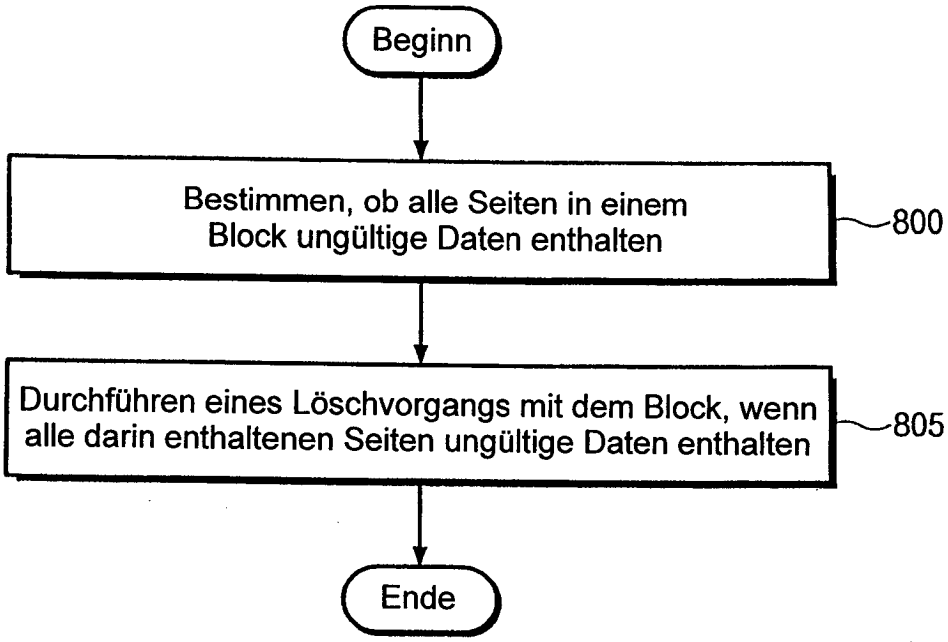

Beginn
Beibehalten der Datenstruktur, welche physikalischen Adressen für Einheiten einer Speicherzuordnung Indikatoren zuordnet, die anzeigen, ob die Einheiten einer Speicherzuordnung ungültige Daten enthalten
700
Identifizieren von einer oder mehreren Einheiten einer Speicherzuordnung im Flashspeicher als ungültige Daten enthaltend
705
Ende

Fig. 8


Beginn
Bestimmen, ob alle Seiten in einem Block ungültige Daten enthalten
800
Durchführen eines Löschvorgangs mit dem Block, wenn alle darin enthaltenen Seiten ungültige Daten enthalten
805
Ende