(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 442 490**

(51) Int. CI.:

G06F 11/10 (2006.01)
G11C 29/00 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **04.10.2006 E 06816218 (9)**

(97) Fecha y número de publicación de la concesión europea: **28.08.2013 EP 1941368**

(54) Título: **Métodos para la gestión y el almacenamiento de datos corregidos**

(30) Prioridad:

18.10.2005 US 253531

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
11.02.2014

(73) Titular/es:

INTELLIGENT INTELLECTUAL PROPERTY HOLDINGS LLC (100.0%)
1209 Orange Street
Wilmington, DE 19801, US

(72) Inventor/es:

GOROBETS, SERGEY ANATOLIEVICH;
ELHAMIAS, REUVEN;
GONZALEZ, CARLOS J. y
CONLEY, KEVIN M.

(74) Agente/Representante:

RIZZO, Sergio

DESCRIPCIÓN

Métodos para la gestión y el almacenamiento de datos corregidos

ANTECEDENTES

[0001] Esta invención hace referencia en general al funcionamiento de sistemas de memoria flash no volátil, y, más específicamente, a técnicas de renovación y corrección de datos almacenados en ella, particularmente en sistemas de memoria con bloques de celda de memoria muy grandes.

[0002] Existen muchos productos de memoria no volátil con éxito comercial utilizados hoy en día, particularmente en forma de pequeñas tarjetas de factor de forma, que emplean un conjunto de celdas flash EEPROM (en inglés Electrically Erasable Programmable Read Only Memory, memoria de solo lectura programable y borrable eléctricamente) formadas sobre uno o más chips de circuito integrado. Un controlador de memoria, normalmente pero no siempre en un chip del circuito integrado separado, conecta con un host al que se conecta la tarjeta de manera extraíble y controla el funcionamiento de la matriz de memoria dentro de la tarjeta. Dicho controlador típicamente incluye un microprocesador, alguna memoria de solo lectura (ROM) no volátil, una memoria de acceso aleatorio (RAM) y uno o más circuitos especiales como uno que calcula un código corrector de errores (ECC) a partir de datos mientras pasan a través del controlador durante la programación y lectura de datos. Alguna de las tarjetas comercialmente disponibles son las tarjetas CompactFlash™ (CF), las tarjetas Multimedia (MMC), tarjetas Secure Digital (SD), tarjetas Smart Media, personnel tags (P-Tag) y tarjetas Memory Stick. Los Hosts incluyen ordenadores personales, ordenadores portátiles, agendas digitales personales (PDAs), varios dispositivos de comunicación de datos, cámaras digitales, teléfonos móviles, reproductores de audio portátiles, sistemas de sonido para automóviles, y tipos de equipos similares. Además de la implementación de la tarjeta de memoria, este tipo de memoria puede estar integrada de manera alternativa en varios tipos de sistema host.

[0003] Dos arquitecturas generales de matriz de celdas de memoria han encontrado una aplicación comercial: NOR y NAND. En una puerta NOR típica, las celdas de memoria están conectadas entre una fuente de línea de bits adyacente y difusiones del drenador que se extienden en una dirección de columna con las puertas de control conectadas a líneas de palabra que se extienden a lo largo de filas de celdas. Una celda de memoria incluye al menos un elemento de almacenamiento situado sobre al menos una parte de la región del canal de celda entre la fuente y el drenaje. Un nivel programado de carga en los elementos de almacenamiento controla así una característica de funcionamiento de las celdas, que pueden ser leídas entonces al aplicar tensiones apropiadas sobre las celdas de memoria dirigidas. Los ejemplos de dichas celdas, sus usos en sistemas de memoria y métodos de fabricarlas se presentan en las patentes estadounidenses números 5.070.032, 5.095.344, 5.313.421, 5.315.541, 5.343.063, 5.661.053 y 6.222.762.

[0004] La puerta NAND utiliza cadenas de series de más de dos celdas de memoria, por ejemplo de 16 o 32, conectadas junto con uno o más transistores seleccionados entre líneas de bits individuales y una referencia potencial para formar columnas de celdas. Las líneas de palabra se extienden por las celdas dentro de un gran número de estas columnas. Una celda individual dentro de una columna se lee y verifica durante la programación provocando que las celdas restantes en la cadena sean difíciles de encender para que la corriente que fluye a través de una cadena dependa del nivel de carga almacenado en la celda dirigida. Los ejemplos de arquitectura de puertas NAND y su funcionamiento como parte del sistema de memoria se encuentran en las patentes estadounidenses números 5.570.315, 5.774.397, 6.046.935 y 6.522.580.

[0005] Los elementos de almacenamiento de carga de las actuales memorias flash EEPROM, según se debate en las patentes anteriormente mencionadas, son más comúnmente puertas flotantes de conductividad eléctrica, típicamente formadas a partir de un material conductor dopado de polisilicio. Un tipo de celda de memoria alternativa útil en los sistemas flash EEPROM utiliza un material dieléctrico no conductivo en lugar de la puerta flotante conductiva para almacenar carga de manera no volátil. Un material dieléctrico de triple capa formado con óxido de silicio, nitruro de silicio y óxido de silicio (ONO), se sitúa entre una puerta de control conductiva y una superficie de un sustrato semi-conductivo por encima del canal de celda de memoria. La celda se programa inyectando electrones desde el canal de celda hacia el nitruro, donde se retienen y almacenan en una región limitada, y se borran inyectando electrones calientes en el nitruro. Varias estructuras y conjuntos de celdas específicos empleando elementos dieléctricos de almacenamiento se describen en la solicitud de patente estadounidense publicada núm. 2003/0109093.

[0006] Como en la mayoría de las aplicaciones de circuito integrado, la presión para reducir el área del sustrato de silicona requerida para implementar alguna función del circuito integrado también existe con las matrices de celdas de memoria flash EEPROM. Constantemente se desea aumentar la cantidad de datos digitales que pueden almacenarse en un área dada de un sustrato de silicona, para aumentar la capacidad de almacenamiento de una tarjeta de memoria dada y otros tipos de paquetes, o para aumentar la capacidad y disminuir el tamaño. Una manera de aumentar la densidad de almacenamiento de datos consiste en almacenar más de un bit de datos por celda de

memoria y/o por unidad o elemento de almacenamiento. Esto se consigue dividiendo una ventana del intervalo de niveles de voltaje de la carga de un elemento de almacenamiento en más de dos estados. El uso de cuatro de estos estados permite a cada celda almacenar dos bits de datos, ocho estados almacenan tres bits de datos por elemento de almacenamiento, y así sucesivamente. Las estructuras de múltiples estados flash EEPROM utilizando puertas flotantes y su funcionamiento se describen en las patentes estadounidenses con números 5.043.940 y 5.172.338, y para las estructuras utilizando puertas flotantes dieléctricas en la solicitud de patente estadounidense núm. 10/280.352 mencionada con anterioridad. Las partes seleccionadas de un conjunto de celdas de memoria multi-estado también pueden funcionar en dos estados (binario) por varias raciones, de la manera descrita en las patentes estadounidenses con números 5.930.167 y 6.456.528.

[0007] Las celdas de memoria de un típico conjunto flash EEPROM se dividen en bloques discretos de celdas que se borran a la vez. Es decir, el bloque es la unidad de borrado, un número mínimo de celdas que pueden borrarse de manera simultánea. Cada bloque almacena típicamente una o más páginas de datos, la página siendo la unidad mínima de programación y lectura, pese a que más de una página se puede programar o leer en paralelo en diferentes sub-conjuntos o planos. Cada página almacena típicamente uno o más sectores de datos, el tamaño del sector estando definido por el sistema host. Un sector de ejemplo incluye 512 bytes de datos de usuario, siguiendo un estándar establecido con unidades de disco magnético, además de algunos conjuntos de bytes de información de cabecera sobre los datos de usuario y/o el bloque en el que se almacenan. Dichas memorias están típicamente configuradas con 16, 32 o más páginas dentro de cada bloque, y cada página almacena uno o solo unos pocos sectores de datos.

[0008] Con tal de aumentar el grado de paralelismo durante la programación de los datos de usuario en la matriz de memoria y la lectura de los datos de usuario a partir de ella, la matriz se divide típicamente en subconjuntos, comúnmente conocidos como planos, que contienen sus propios registros de datos y otros circuitos para permitir el funcionamiento paralelo con tal de que los sectores de datos puedan programarse o leerse desde cada uno de varios o todos los planos de manera simultánea. Un conjunto de un único circuito integrado puede estar físicamente dividido en planos, o cada plano puede formarse a partir de uno o más chips del circuito integrado. Los ejemplos de tal implementación de memoria se describen en las patentes estadounidenses con números 5.798.968 y 5.890.192.

[0009] Con tal de administrar de manera más eficiente la memoria, pueden enlazarse de manera lógica los bloques físicos para formar bloques virtuales o "metabloques". Es decir, cada metabloque está definido para incluir un bloque de cada plano. El uso del metabloque se describe en la publicación de la solicitud de patente internacional núm. WO 02/058074. El metabloque se identifica mediante una dirección del bloque lógico del host como destino para programar y leer datos. De manera similar, todos los bloques de un metabloque se borran a la vez. El controlador en un sistema de memoria gestionado con bloques tan grandes y/o metabloques realiza un número de funciones que incluyen la traducción entre las direcciones de bloque lógico (LBAs) recibidas desde un host, y los números de bloque físico (PBNs) dentro del conjunto de celda de memoria. Típicamente se identifican páginas individuales dentro de los bloques por desviaciones dentro de la dirección de bloque. La traducción de la dirección a menudo incluye el uso de términos intermedios de un número de bloque lógico (LBN) y una página lógica.

[0010] Los datos registrados en un metabloque se actualizan frecuentemente, la probabilidad de actualizaciones así como la capacidad de datos del metabloque aumenta. Los sectores actualizados de un metabloque lógico están normalmente escritos sobre otro metabloque físico. Los sectores que no cambian suelen copiarse desde el original al nuevo metabloque físico, como parte de la misma operación de programación, con tal de consolidar los datos. De manera alternativa, los datos sin cambiar pueden mantenerse en el metabloque original para su posterior consolidación con los datos actualizados en un único metabloque.

[0011] Es común gestionar sistemas de bloques grandes o metabloques con algunos bloques extra mantenidos en un fondo de bloques borrados. Cuando se actualizan una o más páginas de datos menos que la capacidad de un bloque, es común escribir las páginas actualizadas en un bloque borrado del fondo y después copiar los datos de las páginas no cambiadas desde el bloque original al bloque del fondo borrado. Las variaciones de esta técnica se describen en la publicación de solicitud internacional núm. WO 02/058074 mencionada con antelación. Con el tiempo, como resultado de la reescritura y actualización de los archivos de datos del host, muchos bloques pueden acabar con un relativamente pequeño número de sus páginas conteniendo datos válidos y las páginas restantes conteniendo datos que ya no son actuales. Para poder utilizar de manera eficiente la capacidad de almacenamiento de datos del conjunto, las páginas de datos válidos relacionados de manera lógica se recogen cada cierto tiempo a partir de fragmentos entre los múltiples bloques y se consolidan juntos en un número menor de bloques. Este proceso se denomina comúnmente "recolección de basura".

[0012] Las celdas flash individuales EEPROM almacenan una cantidad de carga en un elemento de almacenamiento de carga o una unidad que es representativa de uno o más bits de datos. El nivel de carga de un elemento de almacenamiento controla el umbral de voltaje (comúnmente conocida con la referencia VT) de su celda de memoria, que se utiliza como una base de lectura del estado de almacenamiento de la celda. Una ventana del umbral de voltaje se divide comúnmente en un número de márgenes, uno para cada dos o más estados de almacenamiento de

la celda de memoria. Estos márgenes se separan mediante bandas de seguridad que incluyen un nivel de determinación nominal que permite determinar los estados de almacenamiento de las celdas individuales. Estos niveles de almacenamiento cambian como resultado de la alteración de programación de carga, las operaciones de lectura o borrado realizadas cerca o en otras celdas, páginas o bloques de memoria relacionados. Por ejemplo, la programación de un conjunto de celdas de memoria compartiendo una línea o circuito con un segundo conjunto de celdas de memoria puede alterar los niveles de carga del segundo conjunto. El resultado final de esta alteración parásita si no se realiza una acción correctiva en la parte del controlador de sistema de almacenamiento, los datos en áreas expuestas que no están gestionados pueden estar corruptos, y en un caso extremo, más allá de las capacidades correctivas de cualquier código corrector de errores (ECCs) almacenado junto con los datos. Esta corrupción de los datos resultaría en una pérdida de los datos del usuario, presentando el sistema de almacenamiento como poco fiable. La extensión y naturaleza de tales alteraciones en un conjunto de celdas de memoria particular depende de su arquitectura, estructura y funcionamiento específicos.

[0013] Por lo tanto, es beneficioso restaurar los niveles de carga alterados de vuelta a los centros de sus márgenes de estado cada cierto tiempo, antes de que las operaciones de alteración provoque que cambien por completo sus márgenes definidos, en cuyo caso se leen los datos erróneos. Dicho proceso, llamado actualización o depuración de datos, se describe en las patentes estadounidenses con números 5.532.962 y 5.909.449. Como un aspecto adicional de esto, en los sistemas de memoria que utilizan los códigos de corrección de errores (ECCs), algunas cantidades leídas de bits de datos erróneos en la memoria pueden corregirse utilizando un ECC y los datos corregidos se reescriben entonces sobre una parte previamente borrada de la memoria. La reescritura de los datos provoca que cada uno de los niveles del umbral de la celda de memoria escrita se encuentre dentro del margen de estado designado ya que la programación de datos normalmente incluye un ajuste alternativo de la carga almacenada y una lectura-verificación del umbral de celda de memoria resultante hasta que alcanza el margen deseado que representa los datos almacenados en la celda. La patente estadounidense núm. US2003/0206460 describe una corrección de errores con ECCs en una memoria no volátil. Una lectura errónea de bloque se corrige utilizando ECCs y se vuelve a escribir en un bloque sustituto.

RESUMEN DE LA INVENCIÓN

[0014] La invención se define mediante las reivindicaciones adjuntas independientes.

[0015] Una operación de depuración completa se incluye en un sistema de memoria flash para compensar las alteraciones de los niveles de almacenamiento en un grupo de celdas de memoria provocadas por las operaciones de programación, lectura o borrado realizadas en otro grupo de celdas de memoria del mismo chip de circuito integrado. La probabilidad de tales alteraciones de datos aumenta a medida que los conjuntos de memoria disminuyen en tamaño. Existe una tendencia, para ahorrar espacio, de compartir varias líneas de señal entre grupos de celdas de memoria para que el grupo de celdas de memoria experimente una exposición potencialmente repetitiva a voltajes y/o corrientes durante la programación, lectura o borrado de las celdas de memoria de otro grupo. Al depurar los datos almacenados en todos los grupos de celdas de memoria de manera continua y organizada, la corrupción de datos almacenados a lo largo del tiempo se reduce significativamente. Además, el almacenamiento de un número aumentado de bits por celda de memoria con un número aumentado de márgenes más pequeños de estado del umbral de voltaje es práctico cuando los niveles de carga alterados se corrigen restaurándolos metódicamente con sus niveles deseados.

[0016] Una operación de depuración implica la lectura de datos en áreas que han recibido una exposición a potenciales señales de alteración, y la realización de alguna acción correctiva si se determina que estos datos han sido alterados. Las alteraciones pueden detectarse, por ejemplo, al comprobar la integridad de los datos, ya sea leyendo los datos u obteniendo los resultados de un comprobación ECC de los datos. La acción correctiva puede implicar la reescritura de datos en la misma ubicación, o en una ubicación diferente, y puede implicar operación de gestión de datos o defectos de mayor nivel.

[0017] La operación de depuración puede adaptarse a la integridad de datos leídos. Por ejemplo, si se encuentra un nivel de umbral de errores de datos en una región de la matriz de celdas de memoria, la velocidad de depuración en esa región deberá aumentar. Por el contrario, si no se encuentran errores o sólo unos pocos errores de datos menos que el umbral en una región dada, esto permite disminuir la velocidad de depuración de la región dada. La frecuencia y ubicación de la depuración también pueden adaptarse a niveles de uso y otros parámetros del sistema. Estas y otras características de depuración se calculan para proporcionar un equilibrio entre la necesidad de mantener la integridad de los datos y la necesidad de mantener un alto nivel de rendimiento del sistema. Se evitan particularmente las operaciones de depuración que no tienen probabilidad de mejorar la integridad de datos.

[0018] Las operaciones de depuración se llevan a cabo preferiblemente en segundo plano, cuando el sistema de memoria no está leyendo o escribiendo datos. El sistema de memoria puede fijarse en el host para indicar cuándo no requiere el host que la memoria almacene o recupere datos, para llevar a cabo durante ese tiempo la operación de depuración.

[0019] Incluso si se detectan uno más errores de datos en una lectura de depuración en particular, puede determinarse no llevar a cabo una acción correctiva si el error (o errores) puede(n) corregirse con el ECC tras la lectura de los datos, con tal de mantener el rendimiento del sistema. Cuando los datos de usuario y los datos de cabecera de los sectores individuales tienen su propio ECC, es menos probable que continúen sin corregirse los errores de datos en los datos de cabecera que los errores en los datos de usuario.

[0020] Cuando existen demasiados errores de bit para una lectura de depuración de datos bajo condiciones normales a corregir mediante el ECC, los datos pueden volver a leerse con condiciones de referencia más relajadas con tal de leer las celdas cuyos niveles de carga se han desplazado fuera del margen normal. Una vez que se han verificado correctamente por el ECC, los datos de lectura pueden volver a escribir entonces dentro de niveles de carga normales. En cambio, cuando una lectura de depuración de los datos no revela errores, esos datos pueden volver a leerse bajo condiciones de referencia más restrictivas con tal de determinar la calidad de datos almacenados. Es decir, los niveles de carga que se han desplazado fuera de sus márgenes óptimos se detectan para que puedan volver a escribirse dentro de tales márgenes como parte de la acción correctiva depuradora.

[0021] En los sistemas de memoria que utilizan grandes bloques o metabloques de celdas de memoria que almacenan de manera individual un gran número de sectores de datos host, un sector (o varios sectores) de datos de un primer bloque físico que se corrige durante una operación de depuración puede reescribirse en un segundo bloque en el que se copian los sectores de datos restantes del primer bloque, siendo compatible con el método de borrado de fondo de la gestión de memoria arriba descrita. De manera alternativa, un bloque o metabloque pueden dedicarse al almacenamiento temporal de los sectores de datos corregidos por depuración hasta que otros sectores de datos de los mismos bloques o metabloques que el sector (o los sectores) corregidos necesiten desplazarse por alguna otra razón, como por ejemplo para la recolección de basura, cuando los sectores de datos corregidos por depuración pueden reocombinarse con otros sectores de datos del mismo bloque o metabloque. Esto mejora el rendimiento del sistema.

[0022] Una vez que se determina la necesidad de llevar a cabo la acción correctiva con datos específicos, esa acción puede posponerse en caso de que llevarla a cabo en ese momento pueda afectar de manera negativa el rendimiento del sistema, y si los datos pueden leerse sin la acción correctiva, si fuera necesario, antes de que la posterior acción correctiva se lleve a cabo. Todos los datos, direcciones y diferentes parámetros corregidos como se determina en el momento del desplazamiento están temporalmente almacenados y recuperados posteriormente al ejecutar la acción correctiva pospuesta. En sistemas organizados en grandes bloques o metabloques de celdas de memoria, las acciones de depuración correctiva pueden posponerse hasta que se programa una acción correctiva para una cantidad de datos dada de un bloque o un metabloque dado, en cuyo caso todos los sectores de datos pospuestos del bloque o metabloque dado se corrigen a la vez. Esta acción puede reducir la cantidad de copias y reescrituras de datos que ocurren cuando los sectores de datos del bloque o metabloque dado se consolidan juntos de nuevo.

[0023] Finalmente, si se descubre que una celda de memoria dada, columna de celdas, bloque de celdas, u otra unidad de celdas expuesta necesita una depuración frecuente, la unidad puede descartarse del sistema antes de que se degrade a donde los datos almacenados ya no pueden leerse o corregirse.

[0024] Además de corregir datos durante las operaciones de depuración, algunos datos se corrigen cuando se descubre que los datos contienen errores graves durante una operación de lectura normal. Los errores graves son aquellos que requieren una corrección inmediata y copiarse en una nueva ubicación. Las operaciones de lectura normalmente tienen limitaciones temporales, para que los datos deban volver al host dentro de un límite temporal. Este límite temporal puede evitar la copia de un bloque de datos cuando se descubre un error grave durante una operación de lectura. Sin embargo, una parte del bloque puede copiarse dentro del límite temporal. Típicamente, solo la parte de datos (sector, página u otra parte que sea menor a un bloque) que contiene el error grave se copia y la parte restante del bloque se queda sin copiar. La parte de datos corregida puede copiarse en un bloque de corrección dedicado. Un bloque de corrección dedicado puede contener partes de datos de diferentes bloques de la matriz de memoria. Pueden utilizarse técnicas similares en cualquier situación en la que los datos deban corregirse y copiarse dentro de un límite temporal. Al copiar todos los datos se excedería el límite temporal, sólo partes que contienen errores pueden copiarse y esto puede realizarse dentro del límite temporal. Incluso cuando no se aplica un límite temporal, dichas técnicas pueden utilizarse para aumentar la velocidad de corrección.

[0025] Una parte de los datos de un bloque de corrección y los datos no corregidos del bloque original correspondiente se copian ambos en un bloque de consolidación como parte de una operación de consolidación. Esta acción vuelve obsoleto al bloque original y vuelve obsoletos a los sectores corregidos en el bloque de corrección. El bloque de corrección puede compactarse cuando se llena demasiado. La compactación implica copiar sólo los sectores válidos del bloque de corrección en un bloque de corrección compacto, volviendo obsoleto al primer bloque de corrección. Esta acción deja disponible el primer espacio en el bloque de corrección compactado.

[0026] Donde las partes de datos están actualizadas y se mantiene más de una versión de una parte de los datos en una memoria, el controlador tiene un sistema para determinar qué versión se corrigió y por lo tanto qué parte de los

datos es válida. Por ejemplo, donde un bloque de actualización y un bloque de corrección se utilizan, ambos contienen copias de un sector con la misma dirección lógica. El bloque válido es el escrito más recientemente. El controlador determina cual fue escrito antes mediante un indicador de fecha y hora o manteniendo un índice o cualquier otro medio adecuado para determinar el orden en el que se escribieron los sectores.

[0027] La unidad de datos utilizados para el ECC es generalmente un sector. Cada sector tiene un ECC de datos generados de manera individual para él y después, los errores se descubren en función de los sectores. Cuando se descubre un error grave en un sector, el sector se corrige y se envía una copia del sector corregido al bloque de corrección. Sin embargo, en algunos ejemplos, otros sectores también pueden almacenarse en el bloque de corrección que no tiene errores graves. Debido a que una página es la unidad de escritura en la memoria, la copia página por página puede ser más conveniente. Así, los sectores restantes en una página que tiene un sector con un error grave se copian con el sector corregido si contienen errores o si no. También pueden utilizarse otras unidades de corrección y copia.

[0028] Las características anteriores pueden implementarse de manera individual o juntas en varias combinaciones, dependiendo de la aplicación específica. Aspectos, ventajas y características adicionales del sistema de depuración se incluyen aquí en la siguiente descripción de los ejemplos ilustrativos, cuya descripción debería tenerse en cuenta junto con los dibujos adjuntos.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[0029] Las Figuras 1A y 1B son diagramas de bloque de una memoria no volátil y un sistema host, respectivamente, que funcionan juntos;

[0030] La Figura 2 ilustra un primer ejemplo de organización de la matriz de memoria de la Figura1A;

[0031] La Figura 3, muestra un ejemplo del sector de datos del host con datos de cabecera mientras se almacenan en la matriz de memoria de la Figura 1A;

[0032] La Figura 4 ilustra un segundo ejemplo de organización de la matriz de memoria de la Figura 1A;

[0033] La Figura 5 ilustra un tercer ejemplo de organización de la matriz de memoria de la Figura 1A;

[0034] La Figura 6 muestra una extensión del tercer ejemplo de organización de la matriz de memoria en la Figura 1A;

[0035] La Figura 7 es un diagrama de circuito del grupo de celdas de memoria del conjunto de la Figura 1A con una configuración particular;

[0036] La Figura 8 es un diagrama de flujo que ilustra los principales pasos en la depuración de datos;

[0037] Las Figuras 9A y 9B son un diagrama de flujo de un ejemplo específico de una operación de depuración; y

[0038] La Figura 10 muestra distribuciones de niveles del umbral de tensión de un grupo programado de celdas de memoria.

[0039] La figura 11 muestra un diagrama de flujo para el reemplazo parcial de un bloque de datos que contiene un sector con un error grave durante una operación de lectura.

[0040] La Figura 12 muestra una operación de lectura durante la cual dos sectores con errores graves que se encuentran en un bloque original se corrigen y copian en un bloque de corrección.

[0041] La Figura 13 muestra la consolidación del bloque original y del bloque de corrección de la Figura 12.

[0042] La Figura 14 muestra la compactación de un bloque de corrección creando un espacio libre.

[0043] La Figura 15 muestra múltiples versiones de un sector de datos en bloques diferentes de memoria como resultado de la corrección y actualización del sector.

[0044] La Figura 16 muestra la corrección de un sector con un error grave y la copia de la página que contiene el sector en la página de un bloque de corrección.

DESCRIPCIÓN DE LOS MODOS DE REALIZACIÓN EJEMPLARES

Arquitecturas de memoria y su funcionamiento

[0045] Con referencia inicialmente a la Figura 1A, una memoria flash incluye un conjunto de celdas de memoria y un controlador. En el ejemplo mostrado, dos dispositivos de circuito integrado (chips) 11 y 13 incluyen un conjunto 15 de celdas de memoria y varios circuitos lógicos 17. Los circuitos lógicos 17 conectan con un controlador 19 sobre un chip separado a través de circuitos de datos, órdenes y estados, y también proporcionan dirección, transferencia de datos y detección, y otros apoyos del conjunto 13. El número de chips de la matriz de memoria pueden estar entre uno y muchos, dependiendo de la capacidad de almacenamiento proporcionando. El controlador y parte del conjunto completo puede combinarse de manera alternativa en un único chip de circuito integrado pero ésta no es actualmente una alternativa económica.

[0046] Un controlador típico 19 incluye un microprocesador 21, una memoria de programa, una memoria de solo lectura (ROM) 23 principalmente para almacenar firmware y una memoria de acceso aleatorio (RAM) 25 principalmente para el almacenamiento temporal de los datos de usuario ya sea estando escritos o leídos en los chips de memoria 11 y 13. Mientras se utiliza la ROM como memoria de programa para almacenar firmware en este ejemplo, en otros ejemplos puede utilizarse la EEPROM o RAM. Las combinaciones de EEPROM, ROM y RAM también pueden utilizarse como memoria de programa. Los circuitos 27 se conectan con los chips de la matriz de memoria y los circuitos 29 conectan con un host mediante conexiones 31. La integridad de los datos se determina en este ejemplo calculando un ECC con circuitos 33 dedicados a calcular el código. A medida que los datos se transfieren del host a la memoria flash para su almacenamiento, el circuito calcula un ECC a partir de los datos y el código se almacena en la memoria. Cuando esos datos de usuario se leen más tarde desde la memoria, vuelven a pasar a través del circuito 33 que calcula el ECC mediante el mismo algoritmo y compara ese código con aquel calculado y almacenado con los datos. Si son comparables, se confirma la integridad de los datos. Si difieren, dependiendo del algoritmo ECC específico utilizado, aquellos bits erróneos, hasta un número que pueda soportar el algoritmo, pueden identificarse y corregirse.

[0047] Las conexiones 31 de la memoria de la Figura 1A se unen con las conexiones 31' de un sistema host, del que se muestra un ejemplo en la Figura 1B. Los datos se transfieren entre el host y la memoria de la Figura 1A a través de la conexión de circuitos 35. Un host típico también incluye un microprocesador 37, una ROM 39 (u otra forma de memoria de programa) para almacenar códigos de firmware y RAM 41. Otros circuitos y subsistemas 43 a menudo incluyen un disco duro magnético de almacenamiento de alta capacidad, circuitos de conexión para un teclado, un monitor y similares, dependiendo del sistema host particular. Algunos ejemplos de tales hosts incluyen ordenadores personales, ordenadores portátiles, ordenadores de mano, agendas electrónicas, agendas digitales personales (PDAs), reproductores de audio y MP3, cámaras digitales, cámaras de vídeo, máquinas electrónicas de juego, dispositivos de telefonía con cable y sin cable, contestadores automáticos, grabadoras de voz, routers de red y otros.

[0048] La memoria de la Figura 1A puede implementarse como una pequeña tarjeta adjunta que contiene el controlador y todos sus dispositivos de circuito de la matriz de memoria de forma que se conecta de manera extraíble con el host de la Figura1B. Es decir, las conexiones de unión 31 y 31' permiten que una tarjeta pueda desconectarse y desplazarse a otro host, o reemplazarse al conectar otra tarjeta al host. De manera alternativa, los dispositivos de la matriz de memoria pueden incluirse en una tarjeta separada que se conecta de manera eléctrica y mecánica con una tarjeta que contiene el controlador y las conexiones 31. Como alternativa adicional, la memoria de la Figura 1A puede incorporarse dentro del host de la Figura 1B, donde las conexiones 31 y 31' se hacen permanentemente. En este caso, la memoria suele contenerse dentro de una estructura del host junto con otros componentes.

[0049] Ciertos términos utilizados en esta descripción pueden necesitar una explicación. Un "sector" hace referencia a unidades independientemente dirigibles de datos a los que se accede durante la lectura del host y las operaciones de escritura. Un sector de datos tiene normalmente un tamaño de 512 bytes.

[0050] El "sistema de memoria" como se usa aquí es un sistema que consiste en uno o más dispositivos de memoria no volátil y el hardware y el software necesarios para almacenar y recuperar datos en y desde la memoria. Las partes variables de la funcionalidad completa del sistema de memoria puede implementarse tanto en un subsistema completamente dedicado al almacenamiento de datos, como en el mismo sistema host. El sistema de memoria puede integrarse en un sistema host o puede ser extraíble, como en la forma de una tarjeta muy pequeña. Las partes de un sistema de memoria extraíble pueden extraerse por sí mismas, por ejemplo si el medio de almacenamiento es extraíble desde la parte del controlador. Cualquier parte de un sistema host dedicado específicamente para almacenar datos en un sistema de memoria también se considera una parte del sistema de memoria. Dicha funcionalidad host puede incluir librerías, drivers, o aplicaciones de software especializado además de cualquier hardware que resida en el sistema host.

[0051] Para los fines aquí previstos, un "sistema host" es un sistema que generalmente tiene una funcionalidad diferente al almacenamiento de datos, pero que también se conecta al sistema de memoria, o tiene un sistema de memoria integrado en él. Pueden existir sistemas host cuyo único propósito es el almacenamiento de datos.

[0052] Varias de las técnicas de actualización y depuración de datos almacenados en la memoria flash aquí descritas pueden implementarse en sistemas con varias configuraciones específicas, ejemplos de lo cual se dan en las Figuras 2 - 6. La Figura 2 ilustra una parte de la matriz de memoria en la que las celdas de memoria se agrupan en bloques, las celdas en cada bloque pueden borrarse a la vez como parte de una única operación de borrado, normalmente de manera simultánea. El bloque físico es la unidad mínima de borrado.

[0053] El tamaño de los bloques de celda de memoria individuales de la Figura 2 puede variar pero una forma practicada con fines comerciales incluye un sector único de datos en un bloque individual. Los contenidos de dicho sector de datos se ilustra en la Figura 3. Los datos de usuario 51 son normalmente de 512 bytes. Además de los datos de usuario 51 existen datos de cabecera que incluyen un ECC 53 calculado a partir de los datos de usuario, parámetros 55 relacionados con los datos de sector y/o el bloque en el que el sector está programado y un ECC 57 calculado a partir de parámetros 55 y cualquier otro dato de cabecera que puede incluirse. Los parámetros 55 pueden incluir una cantidad relacionada con el número de ciclos de programa/borrado experimentados por el bloque ("hot counts"), siendo actualizada esta cantidad después de cada ciclo o de un número de ciclos preestablecido. Un uso de esta cantidad de experiencia consiste en remapear regularmente las direcciones de bloque lógico en diferentes bloques físicos con tal de equilibrar el uso (nivelación del desgaste) de todos los bloques. Otro uso de la cantidad de experiencia consiste en cambiar los voltajes y otros parámetros de programación, lectura y/o borrado como una función del número de ciclos experimentados en diferentes sectores. Los usos adicionales de las cantidades de experiencia en el proceso de identificación de bloques a depurar se describen abajo.

[0054] Los parámetros 55 también pueden incluir una indicación de los valores de bit asignados a cada estado de almacenamiento de las celdas de memoria, comúnmente conocido como su "rotación". Es decir, los estados lógicos de los datos se mapean en diferentes estados de almacenamiento físico. Esta acción también tiene un efecto beneficioso en la nivelación del desgaste. Una o más banderas también pueden incluirse en los parámetros 55 que indican estados o condiciones. Las indicaciones de niveles de voltaje a utilizar para programar y/o borrar el bloque también pueden almacenarse dentro de los parámetros 55, estos voltajes pueden actualizarse como el número de ciclos experimentados por el bloque y otros cambios de factores. Otros ejemplos de parámetros de cabecera 55 incluyen la identificación de cualquier celda defectuosa dentro del bloque, la dirección lógica del bloque de datos que se mapea en este bloque físico y la dirección de cualquier bloque físico sustituto en caso de que el primer bloque sea defectuoso. La combinación particular de parámetros 55 que se utiliza en cualquier sistema de memoria variará de acuerdo con el diseño. Además, algunos o todos los datos de cabecera pueden almacenarse en bloques físicos dedicados a dicha función, en lugar de en el bloque que contiene los datos de usuario o en el que se relacionan los datos de cabecera.

[0055] El bloque físico multi-sector de la Figura 4es diferente al bloque del sector de datos únicos de la Figura 2. Un bloque de ejemplo 59, la unidad mínima de borrado, contiene cuatro páginas 0-3, cada una de las cuales es la unidad mínima de programación. Uno o más sectores host de datos se almacenan en cada página, normalmente junto con los datos de cabecera incluyendo al menos el ECC calculado a partir de los datos de sector y puede tener la forma del sector de datos de la Figura 3. Cuando los datos de menos de todas las páginas se actualizan, los datos actualizados se almacenan típicamente en una página de un bloque borrado a partir de un fondo de bloque borrado y los datos en las páginas restantes sin cambiar se copian a partir del bloque original en el nuevo bloque. Entonces, se borra el bloque original. Las variaciones de esta técnica de gestión de un bloque grande incluye la escritura de datos actualizados en una página de otro bloque sin mover los datos del bloque original ni borrarlos. El resultado son múltiples páginas con la misma dirección lógica. La página más reciente de datos se identifica mediante alguna técnica conveniente como con el tiempo de programación que se almacena como un campo en el sector o página de datos de cabecera.

[0056] Un conjunto adicional de bloque físico multi-sector se ilustra en la Figura 5. Aquí, el conjunto total de celdas de memoria está físicamente dividido en dos o más planos, estando ilustrados cuatro planos 0-3. Cada plano es un sub-conjunto de celdas de memoria que tiene sus propios registros de datos, amplificadores de detección, decodificadores de dirección y similares para ser capaces de operar de manera muy independiente al resto de planos. Todos los planos pueden proporcionarse en un único dispositivo de circuito integrado o en múltiples dispositivos, siendo un ejemplo el formar cada plano a partir de uno o más dispositivos de circuito integrado. Cada bloque en el sistema de la Figura 5 contiene 16 páginas P0 - P15, cada página con la capacidad de uno, dos o más sectores de datos host y algún dato de cabecera.

[0057] En la Figura 6 se ilustra otra disposición de celda de memoria. Cada plano físico contiene un gran número de bloques de celdas. Para aumentar el grado de paralelismo de la operación, los bloques dentro de diferentes planos están conectados de manera lógica para formar metabloques. Dicho metabloque se ilustra en la Figura 6 estando formado por un bloque 3 de plano 0, bloque 1 de plano 1, bloque 1 de plano 2 y bloque 2 de plano 3. Cada metabloque es dirigible de manera lógica y el controlador de memoria asigna y mantiene seguimiento de los bloques que forman los metabloques individuales. El sistema host preferiblemente conecta con el sistema de memoria en unidades de datos que equivalen a la capacidad de los metabloques individuales. Dicho bloque de datos lógico 61 de la Figura 6, por ejemplo, se identifica mediante un bloque lógico de direcciones (LBA) que se mapea con el

controlador en el bloque físico de números (PBNs) de los bloques que conforman el metabloque. Todos los bloques del metabloque se borran a la vez, y las páginas de cada bloque están preferiblemente programadas y se leen de manera simultánea.

[0058] Existen muchas arquitecturas de matriz de memoria diferentes, configuraciones y estructuras de celda específicas que pueden emplearse para implementar las memorias arriba descritas con respecto a las Figuras 2 - 6. Un bloque de la matriz de memoria del tipo NAND se muestra en la Figura 7 con el objetivo de ilustrar unos cuantos mecanismos de interferencia. Un gran número de cadenas de series de celdas de memoria conectadas orientadas en columna están conectadas entre una fuente común 65 de voltaje VSS y una de las líneas de bit BL0 - BLN que se conectan a su vez con circuitos 67 que contienen decodificadores de dirección, drivers, amplificadores de detección de lectura y similares. Específicamente, una de esas cadenas contiene transistores de almacenamiento de carga 70, 71 .. 72 y 74 conectados en serie entre los transistores seleccionados 77 y 79 en extremos opuestos de las cadenas. En este ejemplo, cada cadena contiene 16 transistores de almacenamiento pero también son posibles otros números. Las líneas de palabra WL0 - WL15 se extienden a lo largo de un transistor de almacenamiento de cada cadena y se conectan a circuitos 81 que contienen decodificadores de dirección y drivers de la fuente de voltaje de las líneas de palabra. Los voltajes en las líneas 83 y 84 controlan la conexión de todas las cadenas en el bloque junto con la fuente de voltaje 65 y/o las líneas de bit BL0 - BLN a través de sus transistores seleccionados. Los datos y direcciones vienen del controlador de memoria.

[0059] Cada fila de transistores de almacenamiento de carga (celdas de memoria) del bloque forma una página que se programa y lee en conjunto. Un voltaje apropiado se aplica a la línea de palabra (WL) de tal página para programar o leer sus datos mientras los voltajes aplicados a las líneas de palabra restantes se seleccionan para volver conductivos sus respectivos transistores de almacenamiento. En el transcurso de la programación o lectura de una fila (página) de transistores de almacenamiento, pueden alterarse niveles de carga anteriormente almacenados en filas no seleccionadas debido a los voltajes aplicados a lo largo de todas las cadenas y a sus líneas de palabra.

Varios aspectos del proceso de depuración

[0060] Existen dos fases principales de depuración, la fase de lectura y la fase de acción correctiva. La lectura de depuración se distingue de otras lecturas de sistema en que generalmente implica la selección y lectura de datos en áreas del sistema de memoria no directamente relacionadas con la finalización de una operación de host particular, ni con cualquier número de otras operaciones de sistema, como la nivelación del desgaste. Otra característica diferenciadora de la lectura de depuración es que no se reúne información útil en el sistema a partir de la lectura de datos, sino que, el resultado de la verificación de la integridad de los datos es el objetivo de la operación. La acción consecuente en la parte del sistema se guía por el resultado de la verificación de integridad, y no por el propio dato en particular. El sistema puede requerir por lo tanto el uso de alguna información de la lectura de datos, como los datos de cabecera, si los datos no pasan la verificación de integridad y se requiere una acción correctiva. Estas características de no completar una operación host particular y no obtener ningún dato útil de la memoria son diferencias fundamentales entre las lecturas de depuración y otras lecturas de datos llevadas a cabo por el sistema.

[0061] La selección de áreas particulares para la lectura de depuración suele guiarse por la localización y el número de operaciones de lectura, escritura y borrado llevadas a cabo en el curso normal de la operación de sistema en el contexto de las características físicas del dispositivo de memoria. Generalmente, las lecturas de depuración se llevarán a cabo en áreas de la matriz de memoria que han sido expuestas a voltajes, corrientes o diafonías como resultado de operaciones en otras áreas. De manera alternativa, las localizaciones de lectura de depuración pueden separarse de otras operaciones de la memoria, y se llevan a cabo para seguir una secuencia determinista o aleatoria. Sin embargo, esto puede resultar en una pérdida de rendimiento del sistema, ya que deberían realizarse más lecturas para obtener la misma cantidad de cobertura de más áreas alteradas.

[0062] Un aspecto adicional de las lecturas de depuración es la selección de cuándo se debe llevar a cabo la operación de lectura de depuración. En general, la operación de depuración puede iniciarse como respuesta a cualquier número de factores, como un número de operaciones host, un número de lectura física, operaciones de lectura, escritura y/o borrado, un periodo de tiempo, características de uso del host, o alguna secuencia aleatoria o pseudo-aleatoria, la generación y verificación de los cuales puede estar unida a cualquiera de los anteriores.

[0063] La escritura de depuración se distingue de otras escrituras de sistema en que se lleva a cabo generalmente como resultado de una verificación de integridad fallida. Una escritura de depuración solo es única en el contexto de la lectura de depuración. Pueden llevarse a cabo otras operaciones de escritura con mecanismos similares a las escrituras de depuración las cuales no se llevan a cabo con ese fin específico. Como ejemplo, las operaciones de escritura pueden ser resultado de las verificaciones de integridad fallidas tras las operaciones de lectura o escritura llevadas a cabo en el transcurso de la operación normal del sistema de memoria. En otro ejemplo, los datos pueden leerse o reescribirse con el fin de actualizarse en ausencia de lecturas de depuración, no basando la decisión de escribir en los datos de la verificación de integridad sino en otro factor. Dicho factor puede ser la existencia de un área del conjunto con un uso o exposición alto, en cuyo caso los datos dentro del área pueden reescribirse o

moverse. Un continuo movimiento o actualización de datos puede llevarse a cabo de una manera determinista o aleatoria. Los datos pueden leerse y reescribirse con el propósito de nivelar el desgaste, pero tiene el beneficio no intencionado de actualizar los datos de una manera que vence los problemas de alteración.

[0064] De hecho, la relativa actualización de una unidad de datos puede utilizarse para determinar si se inicia la depuración de esa unidad de datos cuando se cumplen otros criterios para llevarlo a cabo. Es decir, si una unidad de datos ha sido reprogramada recientemente como parte de la nivelación de desgaste o consolidación de datos (recolección de basura), anterior a la operación de depuración u otras, la depuración actual puede evitarse ya que esos datos han sido actualizados recientemente. La relativa actualización de varias unidades de datos puede mantenerse, por ejemplo, con cuentas de experiencia ("hot counts") o sellos de fecha y hora almacenados con las unidades de datos, como parte de los datos de cabecera de los bloques. De manera alternativa, los bloques físicos pueden agruparse de acuerdo con la actualización de los datos almacenados en ellos, con el grupo al que un bloque pertenece estando almacenado como datos de cabecera del bloque. La relativa actualización de los bloques que se convierten de otro modo en candidatos de la depuración puede utilizarse entonces como un factor para seleccionar aquellos que realmente se depuran. El rendimiento del sistema mejora entonces limitando las operaciones de depuración a aquellas unidades de datos que han sido almacenadas el tiempo suficiente para que los niveles de carga almacenados hayan sido lo suficientemente alterados para requerir atención.

[0065] Con tal de controlar la actualización relativa de los datos almacenados, tanto los bloques lógicos como los físicos pueden agruparse de manera efectiva en conjuntos basados en cuando ha sido reprogramada la última actualización. A todos los bloques dentro de la matriz de memoria completa, o, de manera alternativa a los bloques dentro de un plano, zona u otra parte del conjunto, se les puede proporcionar un valor inicial relativo "hot count", y cada vez que el bloque se reprograma, el valor "hot count" relativo puede actualizarse al valor del conjunto o grupo desplazado más recientemente. Una vez que un cierto número de bloques están en el grupo más recientemente reprogramado, el valor del grupo más recientemente reprogramado, y cualquier bloque reprogramado posteriormente puede ser actualizado al valor del grupo nuevo. Como resultado, los grupos diferentes pueden crearse con una distinción clara relativamente entre los bloques más recientemente reprogramados y los menos recientemente reprogramados. En general, se permite al valor "hot count" de un bloque reconducirse para permitir el uso de un número de campos relativamente pequeño.

[0066] Cuando los "hot counts" relativos se utilizan, sustancialmente todos los bloques pueden empezar con un valor base '0' cuando existen ocho posibles valores, en un ejemplo específico, p.ej., valores de entre '0' y '7'. Siete de cada ocho valores pueden utilizarse mientras que un valor se reserva para proporcionar un espacio entre el valor que representa los bloques más recientemente reprogramados del valor que identifica los bloques que contienen los datos más antiguos. En este ejemplo, los bloques que están escritos reciben un nuevo valor de '1' para indicar que son los más recientemente programados. Una vez que un cierto número de bloques han sido actualizados al nuevo valor de '1', los bloques posteriormente programados pueden recibir un nuevo valor de '2'. Un valor de '3' puede asignarse eventualmente a los bloques recientemente reprogramados una vez que se haya asignado a cierto número de bloques el valor '2'. En algún punto, la cuenta se reconducirá para que los bloques menos usados recientemente tengan un valor de '2', los bloques más recientemente programados tengan un valor de '0' y el valor '1' proporcione un espacio entre los dos para que los valores de los bloques con los datos más antiguos y los más nuevos estén claramente definidos. Finalmente, todos los bloques en el cajón más antiguo se reecribirán, ya sea mediante escritura host, depuración, nivelando el desgaste o mediante otro mecanismo. En el ejemplo anterior, el cajón '2' estará por lo tanto vacío, y ese valor puede servir como espacio, mientras que el cajón '1' puede utilizarse para identificar los bloques escritos más recientemente. Cuando un bloque se convierte en candidato para una operación de depuración en base a otros criterios, puede evitarse su depuración si su valor "hot count" relativa lo sitúa en uno de los grupos de bloques más recientemente reprogramados.

[0067] De manera alternativa, los valores "hot count" absolutos pueden mantenerse para el bloque lógico, el físico o ambos, en cuyo caso el sistema puede utilizar preferiblemente dichos valores "hot count" para tomar la decisión de depurar. Es decir, cuando se reprograma un bloque, su valor "hot count" absoluto se aumenta, disminuye o mantiene de otra forma para proporcionar una indicación del número total de veces que el bloque se ha reprogramado. Los bloques con valores "hot count" absolutos indicando un gran número de operaciones de reprogramación han sido típicamente reprogramados más recientemente que los bloques con valores "hot count" absolutos indicando un número bajo de operaciones de reprogramación. Por lo tanto, la depuración de los datos almacenados en bloques con un número relativamente alto de operaciones de reprogramación puede evitarse, ya que no es probable que los datos hayan sido alterados de manera significativa.

[0068] Existen muchos algoritmos de depuración específicos y operaciones relacionadas con la memoria que pueden llevarse a cabo de manera alterna. La depuración puede controlarse mediante el controlador de sistema de memoria o, de manera alternativa, en cada uno de los dispositivos de circuito integrado de celda de memoria (chips), o incluso en parte o completamente mediante el host. El sistema de memoria puede estar conectado al host de manera extraíble o, de manera alternativa, puede integrarse dentro del host.

[0069] La fase de lectura de la operación de depuración puede llevarse a cabo en un número de formas diferentes. La velocidad de depuración puede ajustarse para optimizar el rendimiento a la vez que se mantiene un nivel específico de integridad de datos. Tanto las velocidades a las que se realizan las operaciones de depuración como el número de sectores de datos leídos en un momento pueden ajustarse, por ejemplo. Dicho ajuste puede realizarse de manera automática como parte del algoritmo de depuración. Por ejemplo, las velocidades y ubicaciones de depuración pueden combinarse con las velocidades de exposición desiguales alteradas en diferentes áreas de la memoria. Las lecturas de depuración también pueden posponerse para optimizar el rendimiento del sistema, o para cubrir necesidades específicas a tiempo real.

[0070] También existen varias alternativas en la manera en que se realiza la lectura de depuración. Por ejemplo, un conjunto de datos puede leerse tanto con niveles de referencia nominales como con niveles de referencia con margen. La cantidad de margen puede dirigirse a mecanismos de alteración específicos existentes. Si los datos leídos no pueden corregirse por el ECC, por ejemplo, entonces una lectura con márgenes más amplios puede ser capaz de recuperar los datos. Si los datos se leen normalmente sin errores, entonces una lectura con márgenes más estrechos puede proporcionar información de la calidad de los datos.

[0071] Tras una lectura de depuración, se toma la decisión de comenzar la acción correctiva. Entre los factores existentes dicha decisión puede basarse en incluir un nivel de actividad basado en un número y/o un modelo de bits erróneos detectados.

[0072] La acción correctiva más común incluye la reescritura de los datos leídos en la misma ubicación o en una diferente en la memoria. Dicha escritura de depuración puede posponerse para optimizar el rendimiento del sistema, o para cubrir necesidades específicas a tiempo real. La acción correctiva puede incluir adicionalmente la rotación de los datos durante una operación de escritura de depuración; es decir, los estados de la celda de memoria que representan los datos almacenados específicos cambian su estado original. La acción correctiva también puede incluir celdas de mapeo, columnas u otras estructuras que se consideran susceptibles a la alteración fuera del sistema.

[0073] Una gran cantidad de protocolos host tienen una interfaz abstracta del sistema de memoria para que la memoria se dirija generalmente por el host mediante un número de dirección de bloque lógico (LBA). Existen modos alternativos equivalentes de direccionamiento, como el cilindro-cabezal-sector (CHS) en el protocolo host ATA, pero el concepto fundamental es que el host no tiene conocimiento de la ubicación física en la que el sistema de memoria ha almacenado un sector de datos del host dado. El sistema host tiene disponible un conjunto adyacente de forma lineal de direcciones de bloque lógico independientes en las que almacenar y recuperar sectores de datos. Estos protocolos host abstractos generalmente requieren la presencia de un controlador en el sistema de memoria con tal de controlar las operaciones de memoria, realizar la funcionalidad de mapeado; la gestión de datos, la recuperación de errores, etc. Los sistemas de memoria que operan con estos protocolos host abstractos dependen preferiblemente del controlador en el sistema de memoria para elaborar operaciones de depuración, ya que generalmente el host no tiene conocimiento de los aspectos físicos del Sistema de Memoria.

[0074] Por otra parte, algunos protocolos host tienen una interfaz en la que el propio host lleva a cabo las funciones de gestión de memoria. Los sistemas de memoria que se ajustan a estos protocolos tienen típicamente una funcionalidad de controlador mínima, si es que la tienen. Pueden existir varias partes de funcionalidad, como por ejemplo, sin carácter limitativo, la generación ECC, verificación ECC, o corrección ECC, que pueden llevarse a cabo por el Sistema de Memoria en lugar de por el Sistema Host. Los sistemas de memoria que operan en estos protocolos gestionados por la memoria host dependen comúnmente del host para realizar las operaciones de depuración, ya que el sistema de memoria normalmente tiene una lógica insuficiente para realizar las operaciones de depuración.

[0075] Algunos sistemas de memoria tienen un controlador dedicado cuyo objetivo consiste en activar el dispositivo de memoria y realizar la funcionalidad de gestión de la memoria. Otros sistemas de memoria no tienen controladores de memoria dedicados, sino que dependen de partes de la infraestructura host para llevar a cabo las operaciones de gestión de memoria. Como ejemplo, un dispositivo de memoria no volátil puede conectarse directamente a un microprocesador de uso general en el sistema host, con la funcionalidad de gestión de la memoria llevada a cabo por el software. En dichos sistemas de memoria sin controladores, el mismo subsistema responsable de las operaciones de gestión de memoria también lleva a cabo preferiblemente la funcionalidad de depuración.

[0076] En los sistemas de memoria que tienen controlador, es posible que la funcionalidad del controlador se integre en el propio dispositivo de memoria no volátil (chip de circuito integrado). En un ejemplo extremo, todo el controlador puede integrarse en un dispositivo de memoria.

[0077] El sistema de memoria puede estar integrado en el sistema host y tener integrada una funcionalidad a diferentes niveles en subsistemas host de uso general o que tienen otra funcionalidad. En dichos sistemas de memoria integrados, se siguen los mismos protocolos host generalmente, pese a que este puede no ser el caso

necesariamente. Como regla general, sin embargo, el mismo conjunto de funcionalidad se requiere para operar el sistema de memoria.

[0078] Mientras que el sistema de memoria típicamente lleva a cabo las operaciones de depuración, en el caso de un protocolo host abstracto es posible que el sistema host pueda iniciar operaciones de depuración en dichos sistemas mediante el uso de una orden especial u otra transacción de interfaz predefinida. Una razón para implementar esta funcionalidad puede ser que el sistema host esté más alerta en los periodos de tiempo en los que el sistema de memoria no es accesible para almacenar o recuperar datos, y el sistema host puede aprovechar la oportunidad para iniciar la operación de depuración durante esos periodos de tiempo. De esta manera, la fiabilidad total del sistema puede aumentarse con un impacto mínimo en el rendimiento. El mecanismo utilizado para iniciar una operación de depuración puede dedicarse específicamente para la depuración, o puede ser un mecanismo de uso general para notificar al sistema de memoria que hay tiempo disponible para las operaciones de preparación. En el último caso, las operaciones de pulverización pueden ser una o varias operaciones llevadas a cabo por el sistema de memoria durante dichos periodos de tiempo.

[0079] La región de exposición debido a la operación de memoria dada es generalmente extensa, por lo que no resulta práctico depurar toda la región expuesta cada vez que se lleva a cabo una operación. Generalmente, sólo una parte del área expuesta puede seleccionarse para depuración, y la velocidad de depuración debe establecerse para que la operación de depuración detecte las áreas más alteradas antes de que el número de bits erróneos y el nivel de celdas desplazadas exceda cualquier plan de recuperación disponible en el sistema de memoria.

[0080] La velocidad a la que se llevan a cabo las operaciones de depuración es un parámetro importante que afecta tanto la integridad de los datos como el rendimiento del sistema de memoria. Cuanto mayor es la velocidad de depuración, mayor es la probabilidad de detectar celdas alteradas en un bloque de datos antes de que el número de bits erróneos y el nivel de celdas desplazadas exceda cualquier plan de recuperación disponible en el sistema de memoria. Sin embargo, cuanto mayor es la velocidad de depuración, mayor es la degradación en el rendimiento del sistema de memoria ya que para esta operación se dedican cantidades crecientes de tiempo de la memoria y del controlador. Con el fin de garantizar un nivel deseado de integridad de los datos sacrificando la menor cantidad de rendimiento posible, la velocidad de depuración se optimiza al mínimo estrictamente requerido, con la banda de seguridad que se considere apropiada. La velocidad a la que pueden llevarse a cabo las operaciones pueden variar de dos maneras: 1) selección del momento en el que se lleva a cabo la operación de depuración, y 2) selección de un número de sectores para realizar la lectura de depuración en un momento.

[0081] Es posible que la velocidad de depuración requerida para mantener la integridad de los datos cambie a lo largo de la vida útil del producto. Por ejemplo, a medida que las celdas pasan por más ciclos, la tasa de alteración puede aumentar o disminuir. Si se establece una velocidad fija de depuración en el momento de producción, con tal de mantener la integridad del sistema de memoria, debería utilizarse la velocidad más alta requerida a lo largo de la vida útil del sistema de memoria. Esto daría como resultado una velocidad más alta de depuración cuando una más baja fuera suficiente, resultando en un mayor sacrificio del rendimiento del sistema de memoria que el necesitado en ciertos momentos de la vida útil del sistema de memoria. Existen varios enfoques para ajustar las velocidades a lo largo de la vida útil de un sistema de memoria.

[0082] Es posible establecer una velocidad variable de depuración en el momento de elaboración del sistema de memoria. Un método para realizar esto consiste en proporcionar una tabla de valores de velocidad que establece diferentes velocidades de depuración con cualquier métrica que afecte la tasa de alteración. Un ejemplo es una tabla que contiene velocidades de depuración para diferentes cuentas de ciclos de programación/borrado experimentadas por varias partes de la matriz de memoria. Si el sistema de memoria mantiene cuentas de ciclos, entonces el parámetro de velocidad de depuración se seleccionaría de la tabla basada en el peor de los casos o en la media de cuentas de ciclo de una región dada en el conjunto.

[0083] Otro enfoque consiste en permitir al sistema de memoria autoajustar la velocidad de depuración basada en los resultados de las operaciones de depuración previas. Por ejemplo, si el sistema de memoria mantiene un registro de operaciones de depuración y determina que un porcentaje muy bajo de operaciones de depuración requiere una acción correctiva, se puede ajustar ralentizando la velocidad a la que realiza las lecturas de depuración. Por otra parte, si se determina que un porcentaje muy alto de operaciones de depuración requieren acciones correctivas, se puede ajustar ralentizando la velocidad a la que se realizan las lecturas de depuración. Otra métrica mediante la que el sistema de memoria puede ajustar la velocidad de depuración es la cantidad de desplazamiento detectado en los elementos de almacenamiento individuales durante las operaciones de depuración previas, o el número de bits erróneos. En cualquiera de los casos anteriores, el sistema puede ajustarse a los parámetros de velocidad de depuración de manera adaptable con cada nuevo resultado, o puede registrar la información para un ajuste periódico.

[0084] Existen varias técnicas que pueden utilizarse para seleccionar cuándo realizar la operación de depuración, algunos ejemplos de las cuales son: 1) aleatoria o pseudo-aleatoria, 2) basada en un número de operaciones host, 3)

basada en un número de operaciones del dispositivo de memoria, 4) basad en un intervalo de tiempo. Es posible utilizar cualquiera de los métodos arriba descritos para ajustar la velocidad a la que se realizan las operaciones de depuración. La hipótesis siguiente consiste en que cualquiera de los parámetros requeridos para tomar la decisión de depuración se proporcione con el momento de producción. Existen diferentes parámetros de velocidad para las operaciones de lectura, escritura y borrado, ya que cualquiera de estas puede alterar la memoria en diferentes proporciones.

[0085] Es posible utilizar números aleatorios (RN) o números pseudoaleatorios (PRN) para determinar la frecuencia de las operaciones de depuración si están disponibles en el sistema, o generadas por el sistema. Abajo se describe un número de métodos para utilizar una secuencia RN o PRN para decidir si se realiza la operación de depuración o no. Todos los métodos siguientes asumen que el RN se verifica con un intervalo regular, y la decisión de realizar la depuración no está basada en un ensayo del valor RN.

[0086] Un método para utilizar un número aleatorio para determinar la velocidad de depuración consiste en realizar una operación lógica AND de un RN o de un PRN con un parámetro de máscara conteniendo algún número de bits establecidos con el valor uno, y el resto establecidos en cero. La decisión de realizar una depuración o no se basa en si la operación AND da como resultado un valor cero o un valor que no es cero. Un parámetro de máscara con más unos da como resultado un valor cero con menos frecuencia que un valor de máscara con más ceros. La tabla abajo muestra las velocidades aproximadas de depuración con diferentes valores de máscara de ocho bits, asumiendo que un resultado AND cero provoca una operación de depuración. Nótese que con una secuencia realmente aleatoria, sólo el número de bits con valor de uno afecta a la velocidad, y no al propio valor, así que los valores en la tabla son solo valores ejemplares.

Valor de máscara (hexadecimal)	Valor de máscara (binario)	Velocidad de depuración
0x00	00000000	1/1
0x01	00000001	1/2
0x03	00000011	1/4
0x07	00000111	1/8
0x0F	00001111	1/16
0x1F	00011111	1/32
0x3F	00111111	1/64
0x7F	01111111	1/128
0xFF	11111111	1/256

[0087] Otro método relacionado consiste en contar el número de bits en un RN de valor 1 para determinar si se lleva a cabo o no la operación de depuración. De nuevo, el número de bits en el RN puede compararse con una máscara o valor indicando un umbral. Otro método adicional implica comparar el valor de un RN con el del parámetro de velocidad de depuración. La decisión de realizar una depuración o no, se basaría en si el RN excedía el valor numérico del umbral. Por ejemplo, para un valor máximo de 5, el RN sería 5/256 la mayoría de las veces.

[0088] De manera alternativa, la velocidad de depuración puede estar ligada al número de operaciones host realizadas. Por ejemplo, una operación de depuración puede llevarse a cabo con cada operación de lectura, escritura y borrado host N, donde N es un parámetro de velocidad que establece la velocidad de depuración. Además, la velocidad de depuración puede estar ligada al número de operaciones de memoria realizadas. Por ejemplo, una operación de depuración puede realizarse con cada lectura NR, escritura NW y/o operaciones de borrado NE, donde NR, NW y NE son parámetros de velocidad que establecen la velocidad de depuración. Como alternativa adicional, donde el sistema de memoria incluye un método para medir intervalos de tiempo, las depuraciones se realizan entonces en un intervalo de tiempo regular, donde T es un parámetro proporcionado para establecer la velocidad de depuración.

[0089] Con tal de mantener el rendimiento del sistema de memoria puede ser deseable posponer una lectura de depuración incluso después de tomar la decisión de realizar una lectura de depuración. Las razones para realizar esto pueden incluir consideraciones a tiempo real. Por ejemplo, un host puede requerir cierta transferencia de datos, y la dedicación de recursos para depurar en ciertos momentos puede afectar la habilidad del sistema de memoria para alcanzar una velocidad de datos garantizados Con esa finalidad, el sistema de memoria puede poner en cola los parámetros de operación de depuración para su procesamiento posterior, en un momento en el que la depuración no afecte al rendimiento del host. Las operaciones de depuración pueden posponerse hasta un tiempo después en el procesamiento de la orden del host, un tiempo después del procesamiento de la orden, o hasta una orden posterior del host. En tal caso, los parámetros de operación de depuración sobre los que se ha decidido se almacenan y procesan más tarde cuando sea más conveniente para el host.

[0090] Ya que sólo una parte de la región expuesta de la matriz de memoria puede depurarse en un momento dado,

un aspecto importante para conseguir una cobertura adecuada es la selección de dónde depurar una vez que se ha tomado la decisión de realizar la depuración. La selección de donde depurar suele estar relacionada con las operaciones de lectura, escritura y/o borrado del dispositivo de memoria. Integrado en la selección está el conocimiento de la región sobre la que una operación de memoria dada alterará otros elementos de almacenamiento. Relacionar el área a depurar con el área operada permite que las operaciones de depuración sean más eficaces, ya que las operaciones de depuración no se realizan en áreas de la memoria que no tienen probabilidad de alterarse.

[0091] Un método para seleccionar la ubicación de la depuración implica el uso de un RN o PRN para seleccionar una ubicación desde dentro del área de exposición de una operación dada. Como media, las áreas que experimentan la mayor exposición tendrán más oportunidades de seleccionarse para depurar. La velocidad de depuración se ajustará para considerar el peor caso posible de cobertura de número aleatorio, dado que algunas áreas se seleccionarán más a menudo que otras, y sólo como promedio se realizará una cobertura uniforme.

[0092] Otro método para seleccionar la ubicación de depuración implica moverse de manera determinista a través de la región expuesta. Sin embargo, este método requeriría registrar tanto las operaciones de memoria como las operaciones de depuración para asegurar una cobertura total.

[0093] Si la actividad de la memoria se registra, es posible entonces obtener una medición de la cantidad de exposición a potenciales operaciones de alteración que un área dada sufre. De acuerdo con tal información, el sistema de memoria puede buscar a través del registro las áreas que han recibido la mayor exposición, y limitar la operación de depuración a esas áreas. Este método puede utilizarse para guiar cualquiera de los métodos anteriores para determinar la velocidad y las ubicaciones de depuración. Generalmente, los ciclos de programación/borrado pueden rastrearse mediante el mantenimiento de la información de cuenta de ciclo. Sin embargo, generalmente no resulta práctico registrar la actividad de lectura con lo que no es probable que este método se utilice para rastrear la exposición de lectura.

[0094] De manera alternativa, las ubicaciones de las acciones correctivas pueden registrarse y utilizarse como una guía de las áreas que pueden ser más susceptibles a las alteraciones que otras, o que pueden recibir una mayor exposición que las otras. El sistema de memoria puede utilizar esta información para pesar la elección de las áreas a depurar, particularmente para las áreas que han requerido una acción correctiva con más frecuencia que otras.

[0095] En sistemas donde no se mantiene el conocimiento de las regiones potenciales de exposición a la alteración, tanto los métodos aleatorios como determinísticos pueden utilizarse independientes a tal conocimiento. En este caso, la depuración se realiza a lo largo del sistema de memoria sin tener en cuenta dónde provoca la actividad host que ocurran operaciones en la memoria de sistema. Sin embargo, la velocidad de depuración se ajusta lo suficientemente alta como para asegurar una cobertura suficiente. Generalmente esto conlleva un gran gasto de recursos ya que algunas áreas del sistema de memoria experimentarán mucha más exposición que otras, y son preferiblemente las principales áreas de enfoque de las operaciones de depuración.

[0096] El objetivo de la operación de depuración consiste en detectar los elementos de almacenamiento alterados antes de que el número de bits erróneos y el nivel de celdas cambiadas exceda cualquier plan de recuperación disponible en el sistema de memoria. Con este objetivo, es generalmente deseable detectar la alteración tan pronto como sea posible y antes de perder una gran parte de la banda de seguridad para un nivel máximo de voltaje debido a la alteración.

[0097] Según se describe arriba en los antecedentes, las memorias flash normalmente almacenan datos en estados discretos, o márgenes de niveles de almacenamiento de carga, cada uno de los cuales se separa de otros estados por la banda de seguridad. Existe un nivel de detección generalmente nominal que discrimina entre cada estado sobre el cual se considera que un elemento de almacenamiento se encuentra en un estado, y por debajo del cual se considera que está en otro estado. Cuando se altera un elemento de almacenamiento dado, el nivel al que se ha programado o borrado puede comenzar a cambiar. Si el nivel del elemento de almacenamiento se acerca al nivel de detección de discriminación, o lo cruza, produce datos en un estado diferente al que se programaron o borraron. El error se manifestará generalmente como uno o más bits erróneos en los datos, generalmente puede detectarse mediante el uso de un ECC cubriendo el campo de datos.

[0098] Estableciendo márgenes o adecuando las condiciones de lectura para que el nivel de discriminación se desplace más hacia un estado u otro provocará que los elementos de almacenamiento alterados se detecten en el estado erróneo incluso si la cantidad de desplazamiento no provoca un error bajo condiciones de lectura nominales. Esto permite al sistema detectar el desplazamiento antes de que se acerque al punto en el que provocaría errores durante una operación normal del sistema de memoria.

[0099] Si los mecanismos de alteración son conocidos por afectar a los niveles de almacenamiento de datos de una manera específica, es posible dirigir la detección de esos mecanismos específicos de alteración estableciendo

márgenes a las condiciones de lectura que afrontan los cambios de nivel esperados. Mientras que la situación ideal consistiría en dirigir los mecanismos de alteración esperados con una única operación de lectura bajo un único conjunto de condiciones de margen, esto podría no suceder con frecuencia. Podría ser necesario realizar múltiples operaciones de lectura bajo condiciones diferentes. Por ejemplo, es posible que diferentes mecanismos de alteración presentes en una memoria provoquen que los elementos de almacenamiento se vuelvan más programados o más borrados. Los elementos de almacenamiento por encima y debajo de un nivel de discriminación pueden desplazarse hacia él, en cuyo caso sería necesario verificar primero un cambio en los niveles de almacenamiento hacia un nivel de discriminación desde un estado, y después desde otro.

[0100] Existen dos medidas individuales de la calidad de los datos que pueden utilizarse como umbrales para determinar si debería llevarse a cabo una acción correctiva: 1) la detección de errores de datos utilizando el ECC, y 2) pese a que solo unos pocos o ningún error de datos se detecten, un cambio en los niveles de almacenamiento de carga puede detectarse antes de provocar errores de lectura de datos.

[0101] Según se indicó anteriormente, las condiciones de lectura de depuración pueden tener márgenes con tal de dirigir ciertos mecanismos de alteración esperados, o simplemente para verificar un margen suficiente en los niveles almacenados. Si un dato se lee bajo condiciones nominales o de margen, la decisión sobre si se toma una acción correctiva o no puede basarse en el número de bits erróneos detectados durante la operación de lectura de depuración. Por ejemplo, si el número de bits erróneos se encuentra por debajo de las capacidades correctivas del ECC del sistema, el sistema puede decidir posponer la acción correctiva, o ignorar el error completamente.

[0102] Además de utilizar el número de bits erróneos como un umbral para iniciar la acción correctiva, el sistema puede tomar la decisión de corregir basándose en otros factores como el modelo de bits erróneos. Por ejemplo, las capacidades de corrección del ECC pueden ser sensibles al modelo de bit erróneo, o los modelos de bit erróneo pueden ser indicativos de un mecanismo particular de alteración conocido en la memoria no volátil. Existen otras razones para basar el umbral en modelos de bit erróneo. El modelo de bit erróneo se revela generalmente durante una operación de corrección ECC.

[0103] Puede ser deseable para el rendimiento posponer una acción correctiva de depuración incluso si se ha determinado que la acción correctiva es necesaria. Las razones para realizar esto pueden incluir consideraciones a tiempo real. Por ejemplo un host puede requerir cierta transferencia de datos, y la dedicación de recursos para depurar en ciertos momentos puede afectar la habilidad del sistema de memoria para alcanzar la velocidad de datos garantizada. Con tales fines, el sistema de memoria puede poner en cola los parámetros de funcionamiento de la acción correctiva de depuración para un procesamiento posterior, en un momento en el que el rendimiento de la acción correctiva de depuración no afecte el funcionamiento del host. Las operaciones de acción correctiva de depuración pueden posponerse hasta un tiempo después en el procesamiento de la orden host, un tiempo después del procesamiento de la orden, o hasta una orden posterior del host. El punto principal es que los parámetros de operación de depuración se almacenarían y procesarían más tarde cuando sea más conveniente para el host.

[0104] Existen dos extremos significativamente distintos de arquitecturas de memoria, cada uno de los cuales lleva a diferentes métodos de mapeado de datos host en ubicaciones físicas del dispositivo de memoria y de gestión y acceso de datos una vez programados. La diferencia fundamental entre las dos arquitecturas hace referencia al tamaño de la unidad mínima de borrado y su relación con el tamaño del sector de datos de protocolo host.

[0105] Si el tamaño de la unidad mínima de borrado se aproxima al del sector de datos host, entonces es posible implementar un esquema de direccionamiento directo, en el que un sector de datos host está generalmente mapeado en la misma ubicación física que el escrito por el host. En tal esquema, los contenidos anteriores están generalmente borrados de la ubicación física antes de escribir los datos de reemplazo en su lugar. Esta reescritura puede implicar una lectura-modificación-escritura si el tamaño de la unidad mínima de borrado es mayor al del sector host. Es posible en este esquema de direccionamiento que los sectores host se vuelvan a mapear para alternar ubicaciones dentro del sistema de almacenamiento, pero generalmente esto sólo ocurre durante las redistribuciones del sector como parte de una operación de gestión defectuosa, o para algún otro fin de la fiabilidad del sistema. Después de dicho mapeo, el sector host permanecerá generalmente mapeado a la nueva ubicación física hasta que ocurra otra reasignación. En dicho esquema de mapeado de direccionamiento directo, la acción correctiva debida a una alteración generalmente implica borrar los contenidos de la ubicación física determinada como alterada, y reescribirlos en la misma ubicación. Cualquier error durante la reescritura podría gestionarse de manera consistente con la excepción general de gestión empleada en el sistema, incluyendo, sin carácter limitativo, volver a borrar y escribir, reasignar, alterar los parámetros de borrado o escritura, etc.

[0106] Por otra parte, si el tamaño de la unidad mínima de borrado es mucho mayor que el de los sectores host, entonces es común implementar un esquema de direccionamiento de borrado de fondo. Dicho esquema de direccionamiento también puede implementarse con memorias que también tienen una pequeña unidad mínima de borrado. En un esquema de borrado de fondo, los sectores host están normalmente agrupados en bloques lógicos que se mapean en unidades físicas de borrado. Un sector de dato host dado se rodea con otros sectores de datos

host dentro de una unidad de borrado. Cuando el host escribe un sector de datos dado, sería impracticable leer todos los sectores que rodean el sector objetivo, borrar el bloque, y después escribir todos los sectores de nuevo en la unidad de borrado con un sector host conteniendo los datos nuevos. Para evitar excesivas operaciones de cabecera, un esquema de borrado de concentraciones generalmente mantiene algún número de unidades de borrado en el estado borrado con el fin de mantener nuevas versiones de datos a medida que se escriben. Los sectores están generalmente agrupados juntos en grupos lógicos con el fin de gestionar datos, y en general el sistema intentará mantener una proximidad física de los sectores en el mismo grupo lógico. Varias unidades de borrado pueden combinarse para formar una construcción lógica llamada metabloque, o bloque virtual, normalmente con el fin de permitir un alto paralelismo de escritura. Generalmente, los nuevos datos del host se escribirán en una ubicación borrada. En algún punto, otros sectores del mismo grupo lógico que no se han reemplazado con nuevos datos se copian en el mismo metabloque, o se copia junto con los datos escritos más recientemente en otro metabloque. Después, una vez que todos los datos en un metabloque dado o en una unidad de borrado han sido reemplazados o copiados en otro lugar, el metabloque o la unidad de borrado se borrará y se considerará parte del fondo de borrado de unidades borradas. En tal esquema de direccionamiento de borrado de concentraciones, la acción correctiva debida a una alteración generalmente implica copiar los datos afectados en otra ubicación del sistema de memoria, junto con cualquier consolidación de datos requerida por el sistema para un comportamiento normal del sistema de memoria.

[0107] Sin embargo, es posible que los datos que requieren una acción correctiva durante una operación de depuración se traten de manera diferente a las operaciones de escritura debido a la actividad host o a la gestión general de datos. Es posible escribir los datos en una ubicación especial reservada para mantener dichos datos durante un corto periodo de tiempo hasta que sea conveniente para el sistema consolidarlos con datos contiguos de manera lógica, dicha área mantenida en reserva, o asignada a petición. También es posible bajo algunas circunstancias que el sistema simplemente borre y reescriba los datos en la misma ubicación. Por ejemplo, si pocos o ningún dato reside en el bloque físico, los otros datos podrían moverse a otro lugar de manera permanente, o almacenarse temporalmente en otro lugar durante el borrado, posiblemente en la memoria volátil, y volverse a escribir junto con los datos que requieren una acción correctiva. Cualquier error durante la escritura de datos se gestionaría de una manera consistente con la gestión de excepción general utilizada por el sistema, incluyendo, sin carácter limitativo, la reescritura, reasignación, escritura en una nueva ubicación, alteración de los parámetros de borrado y escritura, inversión forzada de la polaridad de los datos, etc.

[0108] Algunos elementos de almacenamiento individuales, grupos de elementos de almacenamiento, u otras estructuras pueden ser más susceptibles que otras a la alteración. Como ejemplo de estas posibles estructuras o agrupaciones, los elementos de almacenamiento de memoria flash pueden organizarse para compartir un drenaje, fuente, o línea de puerta común, y debido a la variación en el proceso es posible que uno de esos conjuntos de elementos de almacenamiento conectados experimente una exposición potencialmente más fuerte a las señales de variación que el promedio. De manera similar, los elementos de almacenamiento individual pueden ser más susceptibles que otros debido a la variación en el proceso utilizado para formar la matriz. El sistema de almacenamiento puede diseñarse para mapear permanentemente o reasignar estas áreas consideradas susceptibles a la variación. En un caso extremo, cualquier elemento de almacenamiento u otras estructuras pueden volver a mapearse tras el primer incidente de la acción correctiva de depuración. Por ejemplo, esos elementos de almacenamiento que contienen los bits que requieren una corrección ECC pueden mapearse si los elementos de almacenamiento redundante están disponibles. En otros modos de realización, las ubicaciones de acciones correctivas pueden registrarse y analizarse después con el fin de volver a mapear los elementos de almacenamiento o agrupaciones posteriormente. El análisis puede incluir un número de veces máximo que una agrupación dada ha requerido una acción correctiva. En algunos casos los datos registrados pueden necesitar analizarse con conocimiento de los aspectos físicos del dispositivo de memoria con tal de detectar la presencia de estructuras físicas esperadas, así como elementos de almacenamiento conectados. La manera en la que los elementos de almacenamiento o agrupaciones de elementos de almacenamiento se vuelven a mapear depende de la implementación del sistema de memoria. Por ejemplo, un área física que contiene un sector lógico puede contener elementos de almacenamiento redundante con el fin de reemplazar los elementos de almacenamiento que se mapean. En otro ejemplo, un sector entero o grupo de sectores puede mapearse si ha requerido acción correctiva.

[0109] Las operaciones de depuración pueden llevarse a cabo en un segundo plano de las operaciones host, o entre operaciones host. Puede ser deseable cambiar el comportamiento de la operación de depuración dependiendo del tipo de operación que el host haya requerido. Por ejemplo, puede no ser conveniente que la acción correctiva de depuración tenga lugar durante la ejecución de una orden de lectura host, en cuyo caso la acción correctiva puede posponerse a un momento más conveniente para el sistema de memoria.

[0110] Un enfoque alternativo para evitar la alteración consiste en mover los datos que potencialmente han recibido una exposición a la depuración una primera verificación para determinar si los datos son erróneos. Existen muchos enfoques para seleccionar la frecuencia y ubicación de los datos a mover, la mayoría de los cuales se han descrito arriba.

Modo de realización general de la depuración

[0111] Un diagrama de flujo de la Figura 8 delinea los pasos principales en un ejemplo de algoritmo de depuración de la memoria flash. Un primer paso 91 monitoriza la actividad del sistema de memoria para determinar cuando ha ocurrido un evento que desencadena la operación de depuración. Dicho evento desencadenante puede ser uno o más de los siguientes:

1. Cuando ocurre una lectura de datos, escritura de datos u operación de borrado dentro de un bloque dado u otra unidad del conjunto que puede alterar los niveles de carga de otras unidades. La intensidad y/o duración de la operación puede ser importante para determinar si se desencadena la operación de depuración, así como la susceptibilidad del conjunto a las alteraciones (como cuando la memoria funciona en multi-estado con márgenes de nivel de carga estrechos que definen los estados individuales).
2. Cuando una operación de lectura del sistema normal de una unidad dada lee datos con al menos uno o un número pre-establecido de bits erróneos.
3. Cuando el margen lee (con niveles de referencia establecidos para leer una distribución programada más estrecha que la lectura normal) muestra que los niveles del umbral de las celdas programadas, pese a que no existen bits erróneos, no son adecuados.
4. Tras pasar un intervalo de tiempo predefinido desde la última operación de depuración.
5. Cuando el host inicia una operación de depuración.

[0112] El objetivo total consiste en establecer una operación de depuración lo suficientemente a menudo para detectar datos alterados antes de volverse tan numerosos como para exceder cualquier esquema de recuperación (ECC y similares) que pueden estar disponibles en el sistema de memoria.

[0113] Una vez que el evento desencadenante se detecta, el siguiente paso 93 determina ubicaciones dentro de la matriz de memoria para realizar una operación de depuración. Los candidatos de ejemplo pueden determinarse de la siguiente manera:

1. Los bloques u otra(s) unidad(es) fuera de la unidad dada que comparte líneas de bits, líneas de palabras u otras líneas de señal con la unidad dada, sobre la que se aplican voltajes o corrientes durante la operación de lectura, escritura o borrado.
2. Otra(s) unidad(es) fuera de la unidad dada con un campo de líneas de señal conectadas con aquellas de la unidad dada para que exista una diafonía entre ellos.
3. Si otra(s) unidad(es) se programa(n) o borra(n) puede determinar si está(n) potencialmente afectada(s) por la operación de la unidad dada.
4. La(s) unidad(es) potencialmente afectada(s) puede depender de si la operación es de lectura, escritura o borrado.
5. Si se desencadena por un sistema de lectura normal de la unidad dada que descubre errores en los datos de lectura, se identifica la unidad dada para someterla a una operación de depuración.
6. La relativa actualización de los datos en bloques candidatos puede utilizarse para eliminar aquellos que se han reescrito más recientemente.

[0114] A la identificación de los bloques de memoria u otras unidades candidatas a la depuración no les sigue necesariamente la depuración. Según se indica en el siguiente paso 95, la iniciación de la operación de depuración puede posponerse bajo ciertas circunstancias, ejemplos de esto son:

1. Las lecturas de depuración pueden posponerse para optimizar el rendimiento del sistema, o para cubrir necesidades específicas de funcionamiento del sistema de memoria a tiempo real.
2. La depuración de optimación automática: La velocidad a la que se realizan las operaciones de depuración puede ajustarse de manera dinámica al rendimiento para equilibrar de manera óptima y mantener el rendimiento y la integridad de los datos.
3. La urgencia para realizar una operación de depuración puede determinarse en parte a partir de características de las unidades que se identifican como potencialmente afectadas, como su nivel de experiencia (el número de ciclos de programación/borrado), y el tipo de evento que desencadenó la operación de borrado.
4. Un evento desencadenante de depuración también puede limitarse a ocurrir en una manera determinística, aleatoria o pseudoaleatoria:
   (a) Tras un número específico de operaciones host; (b) Tras un número específico de operaciones de lectura, escritura y/o borrado física/o;
   (c) Tras un periodo de tiempo específico;
   (d) Basado en características de uso del host; o
   (e) Una secuencia aleatoria o pesudoaleatoria, cuya generación y verificación puede estar ligada a

cualquiera de las anteriores.

[0115] En el siguiente paso 97, todos o solo algunos bloques u otras unidades de la memoria identificados/as en el paso 93 como candidatos/as para la depuración se seleccionan para la operación de depuración. La selección de criterios incluye:

1. El número de unidades determinadas que se identificarán para depurar en cualquier operación puede seleccionarse para compensar el efecto del tiempo que tarda la depuración en cuanto a la actuación del sistema y la necesidad de mantener la integridad de los datos.

2. La manera en que se mantiene este equilibrio puede utilizarse para identificar un número de unidades para la operación de depuración actual que depende de la edad de la memoria, el número de ciclos de programa/borrado experimentados por las unidades identificadas, el tipo de evento que ha desencadenado la operación de depuración y un historial de acción correctiva necesariamente tomado sobre los datos en las unidades identificadas.

3. El número de unidades incluidas en una operación de depuración puede depender de la edad de la memoria, el número de ciclos de programa/borrado experimentados por las unidades dadas, y el tipo de evento que ha desencadenado la operación de depuración.

4. De manera alternativa, pueden seleccionarse menos unidades que todas las identificadas para depuración siguiendo una secuencia determinísica, aleatoria o pseudoaleatoria de una manera no relacionada con el uso normal.

[0116] Después, según se indica en el paso 99, los datos se leen a partir de las unidades seleccionadas (lectura de depuración) de acuerdo con lo siguiente:

1. Una lectura de depuración no está directamente relacionada con la finalización de ninguna operación host en particular, o con otras operaciones de sistema, como la nivelación de desgaste, pero normalmente se realiza de la misma manera a medida que se leen los datos normales.

2. Las lecturas de margen (lecturas con niveles de referencia diferentes a aquellos de las lecturas normales de datos) pueden llevarse a cabo en circunstancias específicas:

(a) Como una segunda lectura de datos que tienen demasiados errores a corregir por el ECC, donde los niveles de referencia en la segunda lectura se relajan para leer una distribución programada más amplia que la lectura normal (o puede llevarse a cabo de manera alternativa en el paso 107 de corrección de datos abajo mencionado); o

(b) Cuando se desencadena el mecanismo de alteración específico previsto provocado por el evento desencadenante probablemente desplazando de manera significativa los niveles programados, la lectura inicial podría ser una lectura de margen con niveles de referencia más amplios; o

(c) Como segunda lectura de datos que no tienen ningún o pocos bits erróneos corregibles por el ECC para identificar los datos almacenados como niveles de carga que se han desplazado de los niveles óptimos, donde los niveles de referencia de esta segunda lectura se ajustan para leer una distribución más estrecha que la de una lectura normal. (las lecturas de depuración de margen se analizan más adelante con respecto a la Figura 10).

3. Como parte de la lectura de un bloque, también se leen los datos de cabecera. Si la relativa actualización de los datos se almacena en el encabezamiento, entonces puede utilizarse para identificar los bloques reescritos recientemente que no necesitan depurarse.

[0117] El siguiente paso 101 evalúa la integridad de los datos leídos. Como parte de la lectura de depuración, un ECC puede calcularse a partir de la lectura de datos de usuario de la(s) unidad(es) de memoria y comparado con el ECC que se calculó anteriormente y almacenó junto con los datos de usuario durante la programación. Mientras no haya bits erróneos o haya un pequeño número de errores que puedan corregirse por el ECC, la segunda lectura de margen con los niveles de referencia relajados, como se describe arriba, no necesita ejecutarse.

[0118] Una lectura adicional que puede realizarse opcionalmente es de los niveles de celdas de seguimiento, si se proporcionan en la matriz de memoria, para ver si sus valores almacenados han cambiado. Las celdas de memoria de seguimiento suelen incluirse para establecer los niveles de referencia utilizados para leer las celdas de memoria.

[0119] Una vez que los datos se leen y el número de errores (si existen) de los datos de lectura se conocen, un siguiente paso 103 determina si la acción correctiva es tanto necesaria como deseable. Algunas consideraciones son las siguientes:

1. Un enfoque consiste en corregir los datos con cualquier número de bits erróneos, provocando así que el depurado afecte de la manera más significativa el rendimiento del sistema de memoria.

2. Otro enfoque consiste en renunciar a la corrección de datos a no ser que el número de bits de datos

erróneos sean superiores a algún umbral N por unidad leída, reduciendo así el impacto de la depuración en el rendimiento.

3. El modelo de bits de datos erróneos, según se determina mediante la comparación de los ECCs, también puede utilizarse para determinar si la corrección de los datos es deseable.

4. Los datos que se reescribieron más recientemente (un grado relativamente alto de actualización) preferiblemente no se depuran.

Generalmente, el efecto de completar la operación de depuración en funcionamiento y la gravedad de los errores de la integridad de datos presente y futura se equilibran preferiblemente cuando se determina si se corrigen los datos erróneos.

[0120] En el paso siguiente 105, se determina si se realizar una acción correctiva de depuración o si se pospone un tiempo. Una de las consideraciones para tomar esta decisión es la misma que en el paso 95, es decir, el aplazamiento con tal de cubrir necesidades específicas de funcionamiento a tiempo real en el sistema de memoria en ese momento. Generalmente, la finalización de la operación de depuración se pospone preferiblemente si su finalización posterior causa un impacto menor sobre el rendimiento del sistema que si se realizara en ese momento. Normalmente no es deseable interrumpir un procesamiento normal por el controlador para realizar una depuración, especialmente cuando la operación de depuración tarda más de lo normal debido a que deben corregirse un gran número de errores por el ECC, se necesitan consolidar datos, etc. Cuando se pospone, la lectura de datos por la lectura de depuración, sus direcciones, cualquier error determinado por el análisis ECC y otros parámetros de la operación de depuración desarrollados en los pasos anteriores se almacenan temporalmente. Estos datos y parámetros se leen después y la operación de depuración se completa en un momento en el que el impacto sobre el rendimiento del sistema es menor.

[0121] Cuando la operación de depuración debe completarse, los datos se corrigen en el siguiente paso 107. Los datos se corrigen utilizando el ECC en este caso. Si esto no se lleva a cabo como parte del paso 99 arriba, una o más lecturas de margen (donde los niveles de referencia se relajan para leer una distribución programada más amplia que la utilizada durante la lectura anterior) pueden requerir recuperar datos si una lectura normal o de depuración produce más bits erróneos que los corregibles por el ECC en uso. Entonces, en el paso 109, se toma una acción correctiva apropiada. Esto puede incluir la reescritura de los datos corregidos en la misma ubicación en la que fueron leídos. Esto, por supuesto, requiere primero un borrado del bloque en el que se reescriben que los datos, que puede ser práctico en la organización de la memoria de la Figura 2 donde cada sector de datos corregido rellena un bloque de celda de memoria. Pero no es eficiente hacerlo en las organizaciones de memoria de bloques grandes de las Figuras 5 y 6 a no ser que existan suficientes sectores de datos corregidos que hayan sido corregidos y necesiten reescribirse con direcciones consecutivas de manera lógica para rellenar o casi rellenar un bloque.

[0122] De manera alternativa, los datos corregidos pueden reescribirse en una ubicación diferente a la que se leyó anteriormente en la operación de depuración. Cuando la disposición de memoria utiliza grandes bloques de almacenamiento como los mostrados en las Figuras 5 y 6, el borrado de fondo u otras técnicas de organización del bloque grande anteriormente descritas pueden emplearse cuando los datos a reescribir son menores que la capacidad de un bloque. No resulta extraño, por ejemplo, que los datos a corregir se encuentren sólo en una o en dos páginas de un bloque que contiene 16, 32 o más páginas. Para tales casos, un bloque puede estar dedicado en uno o más planos del conjunto para el almacenamiento temporal de la página o páginas de datos corregidos. Los datos reescritos se combinan entonces con los datos sin corregir almacenados en páginas del bloque original cuando es conveniente hacerlo, por ejemplo durante la consolidación de datos (colección de basura) que se inicia por otros factores durante el funcionamiento normal de la memoria. De manera alternativa, dicha consolidación puede comenzar al llenarse este bloque o después de que una cantidad de tiempo pre-establecida pase después de haber escrito páginas de datos en el bloque dedicado, o según sea conveniente para la operación del sistema.

[0123] Puede resultar conveniente utilizar uno o varios bloques físicos comunes designados para almacenar temporalmente las reescrituras de depuración de datos desde páginas de muchos otros bloques, y después consolidar datos de las páginas reescritas con los datos de otras páginas de sus bloques originales. Por ejemplo, las reescrituras de depuración de las páginas de los bloques 1 y 2 están temporalmente almacenadas en un bloque temporal 3. Después, las reescrituras de depuración del bloque 1 se copian a partir del bloque 3 en un bloque borrado 4, y las páginas que se mantienen sin cambios del bloque 1 también se copian en el bloque 3 para consolidar los datos mapeados en un bloque físico 1. Ocurre lo mismo en otro bloque borrado 5, en el que la reescritura de depuración del bloque 2 se almacena en el bloque 3 y las páginas de datos sin cambiar del bloque 2 se consolidan.

[0124] En el sistema de la Figura 6 que utiliza metabloques, puede existir uno de estos bloques dedicados en cada plano para almacenar datos de otros bloques en ese plano, o un único bloque dedicado en todo el sistema en el que se escriben los datos corregidos de todas las páginas en cualquier plano. La operación específica que se elige depende del número de páginas en los bloques individuales, el número de páginas en toda la matriz de memoria y la previsión del número esperado y frecuencia de las páginas cuyos datos se corregirán por el proceso de depuración.

Una matriz de memoria con una estructura y/o una operación que la vuelve susceptible a las alteraciones necesitará más bloques dedicados que si fuera de otra manera. Si la lectura de depuración no revela errores de datos pero la operación de depuración se elabora con tal de desplazar los niveles de carga almacenados de vuelta a la mitad de sus márgenes de estado, los datos leídos simplemente se reescriben ya que obviamente las correcciones no son necesarias.

[0125] La manera en la que algunos sistemas de memoria se utilizan provoca algunas alteraciones. El proceso de depuración se lleva a cabo entonces pocas veces ya que existen pocos eventos desencadenantes para reconocer. Además, es práctico almacenar permanentemente una cantidad relativamente pequeña de datos reescritos juntos en uno o más bloques físicos designados, sin contar con los bloques físicos en los que los datos depurados residían originalmente. En este caso, las páginas de datos depurados no se recombinan en bloques físicos con páginas de otros datos lógicamente contiguos como un paso en el proceso de depuración. Otro elemento de la acción correctiva consiste en reescribir los datos corregidos con una rotación diferente a como fueron escritos originalmente. Por ejemplo, los cuatro márgenes de estado de almacenamiento análogos de un sistema de cuatro estados pueden designarse originalmente para representar 00, 01, 10 y 11, respectivamente, y tras la reescritura designada para representar 11, 10, 01 y 00. La conversión se realiza mediante el controlador de memoria cuando los datos se leen, utilizando datos de la rotación específica que se mantiene como parte de los datos de cabecera o de otra manera. La rotación de datos se aplica de manera beneficiosa a las reescrituras tanto de los datos corregidos como de los datos que no necesitan corrección.

[0126] Un elemento adicional de la acción correctiva que puede incluirse consiste en mapear páginas o bloques de la memoria que acumulan un historial por haber sufrido más alteraciones que la media. Esto se detecta monitorizando el número de datos erróneos en varias páginas, bloques, planos y/u otras regiones definidas del conjunto. Por supuesto, se debe tener cuidado para evitar el mapeado de páginas o bloques que experimentan errores de datos por razones diferentes a una alta susceptibilidad a alteraciones, como por ejemplo que un bloque se utilice más o de manera diferente a la media.

[0127] El algoritmo de depuración suele controlarse con el controlador de la memoria pero podría, de manera alternativa, estar controlado por el host. Cuando se controla con el controlador de la memoria, el algoritmo se codifica en el firmware del sistema de memoria que se ejecuta normalmente con el microprocesador 21 (Figura 1A) a partir del controlador RAM 25 durante el funcionamiento de la memoria.

[0128] El algoritmo de depuración que se utiliza depende de ciertas características del sistema de memoria, en particular de los dispositivos de memoria 11 y 13 (Figura 1A). Los ejemplos de características son los siguientes:

(a) El número de sectores de datos host almacenados en la unidad de memoria de programación (normalmente una página);

(b) El número de estados almacenados en las celdas de memoria y la extensión del margen de umbral designado para cada estado de almacenamiento;

(c) Si la unidad de programación (normalmente una página) incluye un sector de datos host único o múltiples;

(d) El número de unidades de programación (normalmente una página) en una unidad de borrado (normalmente un bloque); y

(e) La densidad de las celdas de memoria y la extensión y tendencia específicas para que una operación en una unidad provoque alteraciones de datos en otra unidad.

Modo de realización específico de la depuración

[0129] Un modo de realización más específico de un algoritmo de depuración se ilustra en el diagrama de flujo de la Figura 9. Generalmente, se utilizan los mismos pasos de procesamiento previamente descritos pero la implementación se muestra con más detalle, incluyendo lecturas de margen de depuración que se describen con respecto a la Figura 10 para un ejemplo de sistema de cuatro estados (2 bits almacenados en cada unidad de almacenamiento de carga física). Dos puntos de entrada al procesamiento se muestran en la Figura 9, el 115 cuando un evento desencadenante de depuración se detecta y el 117 cuando se recibe una orden para ejecutar depuraciones pospuestas.

[0130] En el paso 115, la memoria se monitoriza para un evento desencadenante de depuración, según se menciona arriba para el paso 91 de la Figura 8. De manera similar, en un paso 119, la unidad o unidades de celda de memoria candidatas, se determinan para su depuración, igual que se describe en el paso 93 de la Figura 8. Entonces, similar al paso 95, el paso 121 de la Figura 9 determina si debería posponerse la depuración en cualquiera de las unidades candidatas debido a que hay otras operaciones de sistema que necesitan realizarse en el mismo momento. En tal caso, según indica el paso 123, las direcciones y otros datos necesarios de las unidades de celda candidatas se almacenan temporalmente y el proceso espera a completar la operación de prioridad más alta del sistema para reiniciarse. De manera alternativa, la finalización de la operación de depuración puede posponerse hasta que una

orden especial posterior se recibe desde el host o se genera por el controlador de la memoria en el paso 117, como es el caso cuando se pospone más tarde en el proceso. Después de que los datos para la depuración pospuesta se almacenen en el paso 123, la operación de depuración particular siendo ejecutada se finaliza a no ser que existan más unidades para depurar, según determina el paso 124. Si existen más unidades para depurar, el proceso regresa al paso 119.

[0131] El siguiente paso 125 realiza la misma función que el paso 97 de la Figura 8, es decir la selección de las unidades de celda de memoria candidatas a la depuración. Los pasos siguientes de la Figura 9 se llevan a cabo en una de las unidades seleccionadas cada vez. El siguiente paso 126 causa que los niveles de intervalo de interrupción de lectura y otras condiciones de lectura se establezcan inicialmente en niveles de lectura normales. El paso 127 realiza entonces una lectura de depuración bajo las condiciones establecidas en la primera de las unidades de celda de memoria, que corresponde con el paso 99 de la Figura 8. Los pasos posteriores se realizan en esta misma unidad con un bucle 129 volviendo al paso 127 para las siguientes unidades seleccionadas en orden hasta que se depuran todas. Por supuesto, si sólo se selecciona una unidad para depurar en una operación particular, no existe un bucle de vuelta 129.

[0132] Tras la lectura de depuración del paso 127 con los márgenes de lectura establecidos en el paso 126, un paso 131 determina si existen errores de dato, similar al paso 101 de la Figura 8. Si existen errores, un siguiente paso 133 determina si los errores son corregibles. Cuando se utiliza un ECC para verificar la integridad de los datos, entonces este paso 133 determina si el número de bits erróneos en la unidad de datos leída están dentro de las capacidades correctivas del algoritmo ECC particular que está siendo usado. Si los errores pueden corregirse, el siguiente paso 135 determina si la corrección es tanto necesaria como deseable, correspondiendo con el paso 103 de la Figura 8. El paso 135 es tan simple como determinar si el número de bits erróneos pero corregibles es menor al umbral de uno o unos pocos, en cuyo caso puede determinarse no tomar una acción correctiva.

[0133] La decisión en el paso 135 también puede verse afectada en cuanto a si los datos y sus ECC son datos de usuario o de cabecera. Según se describe con respecto a la Figura 3, los datos de cabecera pueden tener su propio ECC. Cuando este sea el caso, la integridad del usuario y los datos de cabecera pueden verificarse de manera separada en los pasos 131 y 133, e incluso procesarse uno por uno a través de todo el bucle volviendo con la vuelta 129 considerando los datos de usuario y sus ECC como una unidad y los datos de cabecera y sus ECC como otra unidad de datos, pese a que pueden almacenarse en la misma página. A pesar del uso de los respectivos ECC de datos de cabecera y de usuario serán normalmente los mismos, el proceso puede funcionar para mantener más firmemente la integridad de los datos de cabecera. Este es un ejemplo de equilibrio entre el mantenimiento del rendimiento del sistema, cuyas operaciones de depuración excesivas se rebajan, y el mantenimiento de la integridad de los datos almacenados a largo plazo, cuya depuración se desea conseguir.

[0134] Si se toma la decisión en el paso 135 de llevar a cabo la corrección de los errores corregibles en la unidad de datos, el siguiente paso 137 pregunta si la corrección de datos debería posponerse. La depuración se pospone preferiblemente, por ejemplo, si existen demasiados bits erróneos en los datos leídos como para que su corrección dure más tiempo del disponible en esta etapa. Si no se pospone, los errores se corrigen en el paso 139, similar al paso 107 de la Figura 8. El siguiente paso 140 pregunta entonces si debería posponerse la acción correctiva. La depuración suele posponerse si la reescritura de los datos también incluye el movimiento de otros datos debido a una consolidación necesaria que tardará más tiempo del disponible. Si no existe aplazamiento, los datos corregidos se reescriben, en el paso 141, de acuerdo con una de las opciones de reescritura arriba descritas cuando el modo de realización general de la Figura 8 se describe. Tras una reescritura exitosa de la unidad de datos corregida, se determina en el paso 143 si existen otras unidades de celda de memoria que se seleccionaron en el paso 125 para depurar. Si es así, el paso 145 incrementa la siguiente unidad en orden y vuelve a realizarse el procesamiento para esta nueva unidad a través del bucle 129 desde el paso 127.

[0135] Hasta ahora, se ha descrito el camino principal del diagrama de flujo de la Figura 9. Pero existen muchas ramas diferentes que pueden desviarse a lo largo del camino cuando la resolución de algunas de las preguntas es diferente a aquellas arriba descritas. La primera de dichas ramas a describir se toma desde el paso 133 cuando existen tantos datos de bits erróneos en la unidad sujeto que no pueden corregirse por el ECC. Según se indica en el paso 151, los niveles del umbral de lectura se establecen diferentes a aquellos niveles usados para una lectura de depuración en el paso 127, y entonces, en el paso 153, los datos de la unidad se leen de nuevo con estos niveles de umbral diferentes. Dicha lectura con márgenes se ilustra en la Figura 10.

[0136] Se utiliza un ejemplo de operación de los elementos de almacenamiento individual en una matriz de memoria flash en cuatro estados en la Figura 10. Se muestra una distribución del número de elementos de almacenamiento dentro de una unidad de celdas de memoria, sin alteraciones, en cada uno de los cuatro márgenes de nivel de voltaje del umbral. Las bandas de seguridad 155, 156 y 157 de los umbrales de voltaje se mantienen entre los márgenes de nivel del umbral de estado de almacenamiento sin contener datos de ninguna celda dentro. Esta es la condición programada deseable que existe inmediatamente después de programar y verificar los estados de una unidad de celdas. Pero ya que otras unidades se programan, leen y/o borran entonces, estos datos pueden

alterarse. Las alteraciones se muestran en niveles de umbral que se desplazan de un lado a otro en las bandas de seguridad adyacentes. Esto puede ocurrir para una pequeña proporción de celdas dentro de cada distribución de estado o toda la distribución puede desplazarse o extenderse hacia las bandas de seguridad adyacentes, dependiendo de la naturaleza de las alteraciones.

[0137] Para una lectura normal, se utilizan los niveles de punto de interrupción 159, 160 y 161 aproximadamente en medio de las respectivas bandas de seguridad 155, 156 y 157. Es decir, los niveles del umbral de las celdas siendo leídas se comparan con estos niveles de punto de interrupción para determinar sus estados de almacenamiento. Los errores ocurren cuando los niveles del umbral de las celdas dentro de un estado se desplazan a través de una banda de seguridad atravesando un nivel de punto de interrupción. Por ejemplo, cuando los niveles del umbral de las celdas en el estado 1 disminuyen hasta que se encuentran por debajo del nivel de punto de interrupción 159, aquellas celdas se leen entonces como en el estado 0. De manera similar, si los niveles de umbral de las celdas en estado 1 aumentan hasta por encima del nivel de punto de interrupción 160, una lectura normal dirá que aquellas celdas se encuentran en el estado 2. Dicha datos erróneos leídos se identifican entonces con el proceso ECC. Pero cuando existen demasiados errores para que el ECC los corrija, se realiza una segunda lectura con diferentes niveles de punto de interrupción entre estados mediante los pasos 151 y 153 (Figura 9). Los niveles de punto de interrupción 159, 160 y 161 se desplazan dentro de sus respectivas bandas de seguridad en la dirección del desplazamiento esperado de los niveles de umbral de almacenamiento desde las alteraciones, para que los niveles desplazados vuelvan al mismo lado de las bandas de seguridad en el que estaban antes de ser alterados.

[0138] Después de volver a leer los datos, se determina en el paso 165 mediante el uso del ECC si permanece algún dato erróneo. Si no, el procesamiento entonces procede al paso 135 del camino principal de procesamiento, incluyendo determinar si la corrección de datos y la reescritura deberían posponerse. Si en el paso 165 se descubre que existen datos erróneos, entonces el siguiente paso 167 determina si son corregibles por el ECC. Si lo son, el proceso salta al paso 135 del camino de proceso principal.

[0139] Pero si en el paso 167 se descubre que los datos erróneos no son corregibles, entonces puede incluirse una lectura adicional incluso con diferentes niveles de punto de interrupción. En el paso 169, se determina si todavía existe alguna condición relajada de lectura sin probar. Si es el caso, las condiciones de reintento se incrementan en el paso 170 y el procesamiento vuelve a los pasos 151 y 153 para leer los datos con estas nuevas condiciones. Pero si no existen más condiciones de lectura diferentes proporcionadas, entonces los datos en la unidad del conjunto objeto deben permanecer con sus errores incorregibles. Se determina entonces que la operación de depuración de esa unidad ha fallado. Un solución a esto podría ser copiar todos los datos válidos en un nuevo sistema de memoria y después descartar la memoria presente. Otra solución consiste en marcar simplemente esta unidad como mala, mediante una bandera almacenada en un dato de cabecera o en otro lugar, y proceder al paso 143 para continuar con la depuración de otras unidades del conjunto. Los datos en esa unidad se pierden. Ya que estos datos se verifican como parte de la programación original para estar inicialmente en su margen de nivel de umbral correcto, este posible resultado de una acumulación de operaciones de alteración a lo largo del tiempo muestra el deseo de depurar con la suficiente frecuencia para evitar esta consecuencia desfavorable.

[0140] Volviendo al paso 135, si se determina en él que la corrección de datos no es ni necesaria ni deseable, el procesamiento salta al paso 143 para continuar la depuración con cualquier añadido de las unidades identificadas. Además, volviendo a los pasos 137 y 140, si se determina que la corrección o reescritura de datos debería posponerse, entonces los datos de lectura, sus direcciones, el ECC y cualquier identificación de bits erróneos y otros datos anteriormente determinados se almacenan en el paso 171 y el proceso entonces salta al paso 143. Estos datos se leen entonces cuando las acciones pospuestas se alcanzan para su finalización, según se describe abajo.

[0141] Volviendo al paso 131, la siguiente acción tras determinar que no existen datos erróneos podría ser no hacer nada más con los datos de la unidad actual saltando al paso 143. Pero se puede desear una verificación adicional y un ajuste de los niveles de almacenamiento de las celdas a partir de las cuales se leen incluso los datos válidos. Esto incluye la reescritura de datos con niveles de punto de interrupción diferentes a los usados en la lectura inicial, para identificar cualquier celda donde la carga almacenada se ha movido hacia la banda de seguridad entre los niveles definidos para los varios estados (ver Figura 10), incluso pese a que aquellos niveles no han cruzado el nivel del punto de interrupción para provocar una lectura normal de los datos erróneos. Mediante el paso 172, se determina si existe cualquier condición de lectura que no haya sido probada ya. Si es así, el paso 173 provoca la selección de nuevos niveles de punto de interrupción y/u otras condiciones de lectura de depuración. El procesamiento entonces vuelve al paso 126 donde se establecen esas condiciones de lectura y los datos entonces se leen en el paso 127 con esas condiciones. Los niveles de punto de interrupción utilizados durante esta lectura, por ejemplo, se desplazan a los bordes de las bandas de seguridad 155, 156 y 157 de la Figura 10. Si, en el paso 131, se determina mediante el uso del ECC que existen datos erróneos, los bits de datos erróneos indican el desplazamiento de los niveles de carga dentro de las bandas de seguridad. Por lo que es deseable corregir y reescribir estos datos, después de determinar si tal acción debería posponerse, para que los niveles de carga almacenados se muevan fuera de las bandas de seguridad hacia donde se pretendía en las distribuciones de

estado de la Figura 10.

[0142] Si se determina en el paso 131 que no existe ningún error en los datos leídos con los márgenes más restrictivos, finaliza el procesamiento sobre la unidad de datos actual. Se determina entonces mediante el paso 172 si existen más condiciones de lectura de depuración definidas que no han sido probadas todavía con esta unidad de datos. Si es así, los datos pueden leerse de nuevo con niveles de punto de interrupción más ajustados. Es decir, los datos pueden leerse por segunda vez con un primer conjunto de condiciones de lectura alternativas disponibles en el paso 172 con niveles de punto de interrupción desplazados sólo en una parte del camino a lo largo de sus respectivas bandas de seguridad y se repite de nuevo esta tercera vez con los niveles del punto de interrupción del segundo conjunto de condiciones de lectura alternativas seleccionadas por el paso 172 movidos incluso más allá de los bordes de las bandas de seguridad para un ajuste más preciso de los niveles de almacenamiento de carga, si es necesario. Se pueden proporcionar tantas condiciones de lectura de depuración adicionales como sea práctico proporcionar.

[0143] Volviendo al paso 135, donde se determina si se acepta algún nivel de datos erróneos sin corrección, debe reconocerse que este no es consistente con la precisión de los ajustes realizados por la rama de pasos 172 y 172 de los datos almacenados descritos que no contienen errores. Por lo tanto, si esta rama de proceso se utiliza, el paso 135 seguramente no permitirá la aceptación de datos erróneos sin corrección.

[0144] Tras determinar en el paso 143 que todas las unidades de datos identificadas actualmente para someterse a una depuración han sido de hecho depuradas, cualquiera de estas unidades de datos que han mostrado una tendencia insólita a la alteración de sus datos puede descartarse opcionalmente del mapeo del sistema. Un paso 185 determina si existen estas unidades de la memoria física que deberían extraerse del sistema, y, en tal caso, se descartan del mapeo mediante el paso 187. La operación de depuración se completa entonces.

[0145] Sin embargo, si las correcciones y/o reescrituras de datos han sido pospuestas, el controlador del sistema de memoria o del sistema host tendrán esto en cuenta. En un momento apropiado, cuando no existan operaciones de memoria pendientes con mayor prioridad, la depuración de las unidades parcialmente depuradas puede completarse, comenzando con el paso 117. Los datos, el ECC y otra información almacenada para una unidad en el paso 171 se lee en el paso 191. Si los datos de esa unidad necesitan corrección, entonces esta se lleva a cabo en el paso 193. En el paso 195, los datos correctos se reescriben. Si no existen otras operaciones de depuración parcialmente completas que necesiten completarse, entonces esto se determina en el paso 197 y el paso 185 se ejecuta o se acaba el proceso. Si existen otras unidades a completar, el procesamiento incrementa a la siguiente unidad, en el paso 199, y los pasos 191, 193 y 195 y se repiten para los datos de esas unidades.

[0146] En caso de que las direcciones de las unidades de celda de memoria candidatas se almacenen en el paso 123 para una depuración pospuesta, el procesamiento puede continuar automáticamente con la operación de depuración en una de las unidades pospuestas en el momento en el que el controlador no tenga que implementar acciones de mayor prioridad. Ya que el paso de aplazamiento 121 es temprano en el proceso de la Figura 9, la vuelta a la unidad pospuesta provoca la reanudación del procesamiento empezando por el paso 125 y continuando a lo largo del camino, según se describe arriba. Es posible la corrección y/o reescritura de tal unidad para posponerse de nuevo en uno de los pasos 137 o 140.

[0147] En contraste con las operaciones de depuración regulares, las limitaciones de tiempo pueden afectar a la forma de manejar los datos cuando el ECC detecta uno o más errores durante una operación de lectura en respuesta a una orden de lectura host o en otros momentos. Mientras que las operaciones de depuración regulares se llevan a cabo generalmente en el momento en el que se encuentran disponibles los recursos y el tiempo no es crítico, a veces los datos necesitan corregirse cuando hay poco tiempo. Por ejemplo, cuando el dato se lee desde la memoria en respuesta a una orden de lectura host, los datos pueden necesitar corregirse antes de enviarlos al host. Además, una copia corregida de los datos puede almacenarse en la matriz de memoria si los errores son graves.

[0148] En general, los datos que contienen errores pueden categorizarse en tres tipos. Un primer tipo de datos contienen errores corregibles que están a un nivel aceptable y no requieren una corrección y reemplazo inmediatos. Estos datos se corrigen fácilmente antes de enviarlos al host pero se dejan sin corregir en la memoria no volátil. Un segundo tipo de datos tienen errores corregibles que pueden corregirse pero están a un nivel que requiere un almacenamiento inmediato de los datos corregidos. Un tercer tipo de datos contienen errores que no son corregibles. El sujeto principal de los próximos ejemplos es el segundo tipo de datos, datos con errores que pueden corregirse pero son lo suficientemente graves como para requerir el almacenamiento inmediato de una copia de los datos corregidos. Los métodos principales de detección y corrección de errores mencionados en esta aplicación utilizan los ECCs que se generan para cada sector y se almacenan con el sector. Sin embargo, se entenderá que son igualmente aplicables otros métodos alternativos de detección y corrección de errores. Por ejemplo, puede descubrirse la existencia de un error utilizando un código de redundancia, un código de redundancia cíclica o uno o más bits de paridad. La ubicación de un error podría determinarse mediante un ensayo físico de una parte de una matriz de memoria. Una corriente de fuga o una característica física similar podría utilizarse para indicar la

ubicación de una celda defectuosa que contiene datos corruptos. La corrección de un error podría llevarse a cabo de acuerdo con una variedad de métodos. Muchos de estos métodos utilizan datos redundantes para recrear una parte de los datos corruptos. El uso del tipo de redundancia ECC o RAID (del inglés Redundant Array of Independent Disk, es decir conjunto redundante de discos independientes) permite la corrección de una parte de datos (como un sector) incluso si la localización exacta del error es desconocida. Otros métodos, que pueden utilizar menos bits de datos redundantes, permiten la corrección donde se conoce la ubicación del error. Por ejemplo, si un error se indica con un bit de paridad y la ubicación del error es conocida (a partir de un ensayo físico u otros medios) el bit puede "voltearse" o invertirse para proporcionar los datos corregidos.

[0149] El primer tipo de datos contiene errores a tan bajo nivel que no merecen utilizar los recursos para corregir los errores durante una operación normal. Los errores se corrigen fácilmente utilizando un ECC cuando se lean los datos. Cuando los recursos del sistema están disponibles, los datos pueden corregirse como parte de la operación de depuración. El segundo tipo de datos contiene errores a un nivel más alto que el primer tipo de datos por lo que los datos necesitan reemplazarse con los datos corregidos. Típicamente, esta situación surge cuando el número de errores está cercano al límite que puede corregirse de manera fiable utilizando el ECC. Si los datos no se reemplazan, existe el riesgo de que los datos se vuelvan incorregibles. Por lo tanto, resulta mejor reemplazarlos para evitar perder los datos. La determinación de en qué categoría incluir los datos puede basarse en un umbral predeterminado. Si los datos tienen un número de errores por debajo del umbral, no se necesita un reemplazo inmediato. Si los datos tienen un número de errores por encima del umbral, se requiere un reemplazo inmediato. Donde el ECC permite la corrección de 6 bits por sector con certeza, el nivel de umbral puede ser de 4 bits. Por lo tanto, si existen entre 1 y 4 bits erróneos en un sector simplemente se mantiene como está y se corrige utilizando el ECC cuando se lea. Si existen más de 4 bits erróneos, los datos se reemplazan por la copia corregida de los datos cuando la corrección sea posible. Los datos también pueden corregirse y reemplazarse de esta manera donde los errores son demasiados para corregirse con certeza, pero todavía son corregibles con una razonable alta probabilidad. Los datos incorregibles pueden volver a situarse para que los datos no se vuelvan más corruptos. Los nuevos datos del ECC pueden generarse para los datos incorregibles con tal de que los datos incorregibles se restauren si se vuelven más corruptos. Por lo tanto, pese a que los datos son incorregibles utilizando un ECC (o cualquier otra corrección de datos utilizada en el sistema de memoria) los datos se mantienen y se evita un daño mayor. Esto permite al host acceder a dichos datos corruptos. En algunos casos, el host puede contener sistemas de corrección de errores adicionales que utilizan dichos datos corruptos para generar datos originales. Por ejemplo, archivar el software que funciona en el host puede almacenar datos adicionales redundantes que pueden utilizarse para obtener los datos originales utilizando los datos corruptos, o el sistema de memoria puede ser una unidad en un conjunto del tipo RAID. Si los datos incorregibles se volvieran más corruptos, existe el riesgo de una mala corrección, es decir que una operación de corrección devolvería datos incorrectos. Por lo tanto, previniendo una corrupción mayor, se evita dicha mala corrección.

[0150] En un modo de realización, los datos que requieren una corrección y reemplazo se leen durante una operación de lectura normal en respuesta a la orden de lectura host. Sin embargo, los datos se almacenan en bloques tan grandes que copiar un bloque entero tardaría demasiado y excedería el tiempo límite para la operación de lectura. Por el contrario, sólo un pequeño número de sectores se reemplaza cuando el ECC detecta errores en un bloque, los otros sectores se mantienen sin corregir porque o no contienen errores o contienen errores en un nivel aceptable. Los sectores corregidos pueden almacenarse en un bloque dedicado que almacena sectores corregidos de datos desde múltiples bloques. Los sectores sin corregir en el bloque original pueden consolidarse más tarde con los sectores corregidos de datos a partir del bloque dedicado para formar un bloque válido completo con sectores almacenados de manera secuencial. Esta técnica también puede utilizarse en otras situaciones donde es deseable almacenar fácilmente los datos corregidos.

[0151] Un ejemplo de arquitectura de memoria utilizando bloques grandes, o metabloques, se describe en la patente estadounidense núm. 6.763.424. Los metabloques se elaboran generalmente con múltiples bloques de borrado que se enlazan juntos para escribirse y borrarse como una única unidad. Por lo tanto, un metabloque se trata como un bloque grande y puede considerarse un único bloque para los fines del ejemplo presente. En algunos ejemplos, una operación de lectura debe completarse en 100 milisegundos, pero pueden requerirse 200 milisegundos para escribir un metabloque completo. Otros ejemplos de metabloques y sus usos se describen en la solicitud de patente estadounidense 10/749.189, titulada "Adaptive Metablocks".

[0152] La figura 11 muestra un diagrama de flujo para el reemplazo parcial de un bloque de datos en respuesta a una orden de lectura del host. Una orden de lectura se recibe desde el host que requiere datos con un margen particular de direcciones lógicas. El diagrama de flujo muestra la implementación de una rutina de corrección y reemplazo sector por sector. El controlador de memoria identifica la ubicación física en la matriz de memoria de un sector con una dirección lógica dada por el host y lee los datos almacenados en esa ubicación 201. El análisis ECC se lleva a cabo en el sector según se lee desde la matriz de memoria para determinar si existe cualquier error en el sector 203. Si hay errores presentes, se corrigen entonces 205 y los datos corregidos se envían al host 207. Si un sector de datos tiene un error de nivel grave (es decir, por encima de un umbral predeterminado para un error de nivel aceptable) 209, entonces una copia del sector corregido se almacena en un bloque dedicado en otro lugar en la

matriz de memoria no volátil 211. El sector se almacena de manera normal (con datos ECC y cualquier dato de cabecera que pueda utilizarse). Esta rutina se repite 213 hasta que todos los sectores identificados por la orden de lectura del host se han enviado al host. En un ejemplo común, la mayoría de los sectores tendrán errores que están a un nivel aceptable, ya sea cero o no cero pero por debajo del umbral. Sólo unos pocos sectores tienen errores de niveles que requieren almacenar los sectores corregidos. Únicamente los sectores corregidos se almacenan para reemplazar sólo sectores con errores severos.Otros sectores en el mismo bloque como un sector con errores graves permanecen donde están sin reemplazarse. Dicho reemplazo sector por sector implica que en lugar de intentar reemplazar todo un bloque cuando se descubre un error grave, unos pocos sectores (o incluso sólo un sector) se reemplazan y el resto no se reemplazan. Esto puede realizarse rápidamente para que los sectores se reemplacen dentro del tiempo límite para la operación de lectura.

[0153] La Figura 12 muestra un ejemplo donde los sectores desde el X hasta el X+11 se almacenan en un bloque original 221. Los sectores desde el X hasta el X+11 se leen a partir del bloque original 221 como parte de una operación de lectura en respuesta a la orden de lectura host. Cuando los sectores desde el X hasta el X+11 se leen, se verifican utilizando el ECC y se descubre que los sectores X+1, X+2 y X+9 tienen errores, y que el X+2 y X+9 tienen errores graves. Se lleva a cabo la corrección de datos para estos sectores utilizando ECC. Las versiones corregidas de los sectores X+1, X+2 y X+9 se indican como (X+1)', (X+2)' y (X+9)' respectivamente. Las versiones corregidas (X+1)', (X+2)' y (X+9)' se envían al host 223 junto con los sectores X, X+3 to X+8, X+10 y X+11. Ya que X+1 tiene errores que no son graves, la versión corregida (X+1)' no necesita almacenarse. Sin embargo, ambos X+2 y X+9 contienen errores graves y copias de las versiones corregidas (X+2)' y (X+9)' almacenadas en el bloque de corrección dedicado 225. El bloque de corrección dedicado 225 también contiene sectores de los datos corregidos que corresponden a sectores almacenados en otros bloques de la matriz de memoria no volátil. Por lo tanto, en este punto, (X+2)' y (X+9)' pueden considerarse reemplazar X+2 y X+9.

[0154] Una operación de lectura llevada a cabo posterior al almacenamiento de (X+2)' y (X+9)' que indica las direcciones lógicas de los sectores X+2 y X+9 devolverán (X+2)' y (X+9)' en lugar de X+2 y X+9 al host. El controlador de memoria puede realizar esto de cualquier manera apropiada. Por ejemplo, el controlador de memoria puede mantener un índice de sectores que se almacenan en el bloque de corrección 225 y verificar la lista cuando se recibe una orden de lectura. Puede mantenerse una copia del índice en el propio bloque de corrección 225, o en otro lugar en la memoria no volátil para conservar la lista si se pierde la energía. Leer únicamente la versión corregida puede ser más rápido que llevar a cabo la corrección ECC de la versión errónea cuando se recibe una orden de lectura. Por lo tanto, el rendimiento de lectura puede mejorarse para las lecturas posteriores. Además, este método tiene la ventaja de generar una copia corregida de un sector cuando los errores en el sector se vuelven graves pero antes de que los errores se vuelvan demasiado graves para permitir su corrección. A veces, los sectores que desarrollan errores continúan deteriorándose, así que corregirlos y reemplazarlos a tiempo puede evitar el desarrollo de errores incorregibles.

[0155] La Figura 13 muestra una operación de consolidación que puede realizarse posterior a la lectura y una operación de reemplazo parcial de la Figura 12. En la operación de consolidación, los sectores corregidos (X+2)' y (X+9)' se copian desde el bloque de corrección 225 en un bloque consolidado 227 y los sectores sin corregir X, X+1, X+3 hasta el X+8, X+10 y X+11 también se copian en el bloque consolidado 227. Por lo tanto, el bloque consolidado 227 contiene todos los sectores que estaban en el bloque original 221 con versiones corregidas de los sectores que tenían errores graves en el bloque original. El sector X+1 se muestra teniendo todavía errores corregibles en el bloque consolidado 227, pese a que en algunos ejemplos, el sector X+1 también se corrige antes del almacenamiento en el bloque consolidado 227. La consolidación generalmente se realiza como una operación secundaria cuando los recursos están disponibles. Una vez que se escribe el bloque consolidado 227, el bloque original 221 puede marcarse como obsoleto y puede borrarse. Los sectores (X+2)' y (X+9)' en el bloque de corrección 225 ya no son necesarios porque han sido copiados en el bloque consolidado 227. Pueden considerarse obsoletos en este punto. El bloque de corrección también puede eliminarse para liberar el espacio si el bloque de corrección contiene sectores no válidos.

[0156] La Figura 14 muestra una operación de compactación que puede llevarse a cabo en el bloque de corrección 225 cuando el bloque de corrección contiene un gran número de sectores obsoletos. El bloque de corrección 225 contiene sectores (X+2)' y (X+9)' que han sido copiados en el bloque consolidado 227. Por lo tanto, los sectores (X+2)' y (X+9)' en el bloque de corrección 225 se consideran obsoletos. Los sectores obsoletos adicionales pueden ser obsoletos como resultado de ser copiados a otros bloques consolidados. Una operación de compactación copia los sectores válidos (no obsoletos) del bloque de corrección 225 al bloque de corrección compactado 229 sin copiar los sectores obsoletos. Esto se lleva a cabo dentro de un límite temporal, en este caso el límite temporal para una operación de lectura.

[0157] La compactación puede realizarse para prevenir que el bloque de corrección se llene demasiado como para almacenar más sectores. Típicamente, esto se determina comparando el número de sectores en el bloque de corrección con un número máximo. Cuando se alcanza este número máximo, se lleva a cabo una operación de compactación. El umbral suele alcanzarse cuando el bloque está lleno o casi lleno. Los sectores válidos A y B están

en el bloque de corrección junto con sectores obsoletos que incluyen los sectores obsoletos (X+2)' y (X+9)'. Posteriormente, se reciben los sectores Y y Z. El almacenamiento de los sectores Y y Z provoca que se alcance el máximo necesario para realizar una operación de compactación. Los sectores válidos A, B, Y y Z del bloque de corrección 225 se copian en el bloque de corrección compacto 229 mientras que los sectores obsoletos no se copian. Después de copiar los sectores A, B, Y y Z en el bloque de corrección compacto 229, el bloque de corrección 225 puede considerarse obsoleto y borrarse. La operación de compactación libera espacio 231 disponible para almacenar sectores adicionales en el bloque de corrección compactado 229. Únicamente los sectores validos se copian para que el tiempo de copiado sea menor al tiempo requerido para copiar un bloque entero y puede realizarse dentro de un tiempo límite.

[0158] A medida que el número de sectores válidos en un bloque de corrección se hace mayor, el tiempo que se necesita para copiar el bloque de corrección también aumenta hasta que en algún momento, el tiempo necesario para copiar el bloque de corrección puede exceder un límite temporal. Es deseable mantener el número de sectores válidos en el bloque de corrección por debajo de este nivel para que los sectores válidos del bloque de corrección siempre puedan copiarse dentro de un límite temporal (por ejemplo, durante una operación de compactación). Por lo tanto, si el número de sectores válidos alcanza el máximo, un nuevo bloque de corrección puede abrirse pese a que todavía haya espacio en el presente bloque de corrección. El umbral se determina por el número de sectores que pueden copiarse dentro del límite temporal. En un ejemplo, el umbral es un cuarto de un bloque (Es decir un 25% de los sectores en un bloque son válidos). De otra manera, una operación de consolidación puede llevarse a cabo para copiar sectores válidos en los bloques consolidados. Esto vuelve obsoletos los sectores en el bloque de corrección y reduce el número de sectores válidos a copiar. Por lo tanto, el número de sectores válidos puede mantenerse por debajo de un umbral para que todos los sectores válidos de un bloque de corrección puedan copiarse dentro de un límite temporal. Este límite es típicamente el límite temporal para una operación de lectura. Otros límites temporales también pueden cumplirse de esta manera.

[0159] El controlador de memoria mantiene seguimiento de múltiples versiones diferentes de un sector que están presentes en la matriz de memoria en el mismo momento y determina qué versión es válida en todo momento. Por ejemplo, la Figura 15 muestra tres versiones diferentes de un sector X+2, (X+2)' y (X+2)", todos con la misma dirección lógica, almacenada en diferentes bloques de la matriz de memoria en el mismo momento. En los ejemplos arriba descritos, un sector en el bloque de corrección 225 reemplazaba el sector correspondiente en el bloque original 221. Sin embargo, además del bloque original 221 y el bloque de corrección 225, la Figura 15 muestra un bloque de actualización desordenada 233. Los bloques de actualización y en particular los bloques de actualización desordenada se describen en la solicitud de patente estadounidense número 10/750,155, titulada "Non-volatile memory and method with block management system". La Figura 15 muestra un bloque de actualización desordenada que contiene un sector (X+2)" que tiene la misma dirección lógica que el sector X+2 y (X+2)'. En general, los sectores en un bloque actualizado reemplazan a los sectores en un bloque original. En la situación mostrada en la Figura 15, donde existen tanto un bloque de corrección como un bloque de actualización que contienen cada uno un sector con la misma dirección lógica, surge un problema particular. Si sólo una de estas versiones estuviese presente, esa versión reemplazaría la versión en el bloque original. Sin embargo, con dos versiones debe determinarse cuál debería utilizarse. Por ejemplo, si el host envía una petición de lectura identificando la dirección lógica correspondiente a (X+2)' y (X+2)", sólo una de estas debería enviarse al host. Aquí, la determinación se basa en el momento en que se escribieron el sector actualizado (X+2)" y el sector corregido (X+2)'. Por ejemplo, si el sector corregido (X+2)' se escribió antes de que se escribiera (X+2)", entonces el sector (X+2)" reemplaza al sector (X+2)' porque representa una actualización de los datos en el sector mediante el host. De manera alternativa, si el sector corregido (X+2)' se escribió después del sector actualizado (X+2)" entonces el sector corregido (X+2)' debe basarse en el sector actualizado (X+2)". En otras palabras, (X+2)' no derivó de X+2 sino de (X+2)" si se escribió después de que (X+2)" se escribiera. Esto es porque la corrección ECC se aplica a la copia más actual del sector disponible en el momento en que se recibe la orden de lectura. Ya que es necesario determinar cuál fue escrito antes si (X+2)' o (X+2)", el controlador de memoria graba el orden de escritura. El controlador de memoria puede marcar la fecha en los sectores a medida que se escriben (por ejemplo, poniendo los datos temporales en el encabezamiento). De manera alternativa, el controlador de la memoria puede mantener un índice de sectores indicando cuándo se escribieron. Cualquier otro método puede utilizarse para seguir el orden en el que los sectores se escriben en la matriz de memoria no volátil.

[0160] Mientras que los ejemplos anteriores se refieren a sectores como la unidad de datos que se corrige dentro de un bloque, también pueden utilizarse otras unidades. Por ejemplo, una página es la unidad de datos de escritura en una matriz de memoria. Típicamente, una página contiene múltiples sectores de datos y un bloque contiene múltiples páginas. Una vez que la página se escribe no puede volver a escribirse generalmente sin una operación de borrado. Por lo tanto, si un único sector se escribe en una página, el resto de la página no puede usarse. Una página puede ser una unidad conveniente de corrección para algunos ejemplos de la presente invención, para que cuando un sector tenga un error grave, ese sector se escriba en un bloque de corrección como parte de una escritura de página que incluye otros sectores que pueden no tener errores graves o no tener ningún error. La figura 16 muestra un bloque original 251 con seis páginas (páginas de la 0-5) con cuatro sectores por página. El sector S2 en la página 1 de un bloque original tiene un error grave. Cuando se detecta utilizando el ECC, el sector se corrige utilizando el ECC

y la versión corregida S2' se envía al host 233. Una copia de S2' también se almacena en el bloque de corrección 253. Sin embargo, en este ejemplo, los otros sectores (S1, S3 y S4) de la página 1 del bloque original 251 también se escriben en el bloque de corrección 253 aunque los sectores S1, S3 y S4 no tengan errores graves que requieran una corrección y copia inmediatas. Por lo tanto, la página 3 del bloque de corrección 253 está completamente ocupada con datos, incluyendo tanto datos corregidos como sin corregir. Todos los sectores en la página 1 del bloque original 251 pueden considerarse obsoletos en este punto. Por lo tanto, la parte de datos que se corrige y almacena en el bloque de corrección no tiene que tener un tamaño particular para sacar beneficio de los aspectos de la presente invención. Si cualquier parte de los datos que sea menor que un bloque entero se copia de esta manera, se ahorra tiempo.

[0161] Como alternativa para conservar los sectores de reemplazo corregidos en un bloque de corrección dedicado, dichos sectores pueden almacenarse en un bloque que también se utiliza para almacenar otros sectores para otros fines. Se dan ejemplos de almacenamiento de diversos sectores con direcciones lógicas no secuenciales en un bloque y manteniendo el seguimiento de los sectores almacenados en la solicitud de patente estadounidense número 11/016.285, titulada "Scratch Pad Block". Los métodos variados utilizados para gestionar datos en bloques de memoria provisional también pueden utilizarse para bloques de corrección.

[0162] Mientras que ciertos ejemplos proporcionados hace referencia a límites temporales durante la ejecución de órdenes de lectura, otros límites temporales pueden aplicarse en otros momentos haciendo una corrección rápida de los datos y almacenando los datos corregidos deseables. En tales situaciones, las partes de datos que contienen errores pueden corregirse y almacenarse sin almacenar otras partes que no contienen errores. Incluso donde no se aplican límites temporales reales, pueden emplearse dichas técnicas para mejorar la velocidad en ciertos momentos.

Conclusión

[0163] Pese a que varios aspectos de la presente invención se han descrito aquí con respecto a los modos de realización ejemplares de la misma, se entenderá que la presente invención tiene derecho a su protección dentro del alcance de las reivindicaciones adjuntas.

Reivindicaciones

1. Un método para gestionar datos en un sistema de memoria con una matriz de memoria no volátil que incluye múltiples bloques, un bloque representando la unidad de borrado mínima, comprendiendo:
recibir una orden de lectura a partir de un host, identificar una pluralidad de sectores en un primer bloque para leer dentro de un límite temporal predeterminado menor al tiempo requerido para copiar un bloque de la matriz de memoria;
leer la pluralidad de sectores a partir de un primer bloque y determinar si los sectores de la pluralidad deben corregirse y reemplazarse; si un sector individual debe ser corregido y reemplazado, corrigiendo después el sector individual y escribiendo el sector corregido en un segundo bloque dentro del límite temporal predeterminado;
mantener los sectores que no deben corregirse ni reemplazarse en el primer bloque en la forma sin corregir sin escribirlos en el segundo bloque;
enviar el sector individual y los sectores que no deben ser corregidos ni reemplazados al host; y
copiar consecuentemente el sector individual y los sectores que no deben ser corregidos ni reemplazados a un tercer bloque.

2. El método de la Reivindicación 1, comprendiendo también corregir otros sectores desde el primer bloque y escribir los otros sectores corregidos en el segundo bloque con el sector individual dentro del límite temporal.

3. El método de la Reivindicación 1 comprendiendo también la escritura adicional de sectores en el segundo bloque, los sectores adicionales reemplazando los sectores de datos en uno o más bloques de la matriz de memoria distintos al primer bloque; y
compactar consecuentemente los datos en el segundo bloque copiando sectores del segundo bloque que no están obsoletos en un cuarto bloque**,** sin copiar en el cuarto bloque los sectores del segundo bloque que no están obsoletos.

4. El método de la Reivindicación 3, en el que la compactación ocurre en respuesta al umbral de sectores almacenados en el segundo bloque.

5. El método de la Reivindicación 4, en el que el umbral de sectores se determina mediante la cantidad máxima de datos que pueden copiarse dentro de un límite temporal para una orden de lectura.

6. El método de la Reivindicación 1, comprendiendo consecuentemente la lectura únicamente de las copias de sectores corregidos que tienen tanto una copia corregida como una copia no corregida en la matriz de memoria no volátil.

7. El método de la Reivindicación 6, en el que la copia corregida de un sector se identifica a partir de una lista de sectores corregidos y reemplazados.

8. Un método para gestionar datos en un sistema de memoria con una matriz de memoria no volátil que incluye múltiples bloques, un bloque representando la unidad de borrado mínima, comprendiendo:

identificar sectores de datos a corregir y reemplazar, y escribir sectores de reemplazo corregidos únicamente para los sectores identificados en un bloque dedicado, mientras los sectores sin corregir permanecen en los bloques originales con los sectores identificados;
recibir sectores actualizados desde un host, con los sectores actualizados reemplazando sectores almacenados en la matriz de memoria no volátil;
almacenar los sectores actualizados en la memoria no volátil;
recibir consecuentemente una orden de lectura desde un host, identificando un sector con una dirección lógica para la que existe un primer sector corregido de reemplazo en el bloque dedicado y para la que un primer sector actualizado existe en la memoria no volátil; y
enviar al host el sector más recientemente escrito del primer sector corregido de reemplazo y el primer sector actualizado.

9. El método de la Reivindicación 8, en el que los sectores actualizados se almacenan en bloques actualizados y el primer sector actualizado se almacena en un primer bloque actualizado, un bloque actualizado almacenando datos actualizados dentro de un rango de dirección lógica que corresponde a uno más bloques originales.

10. El método de la Reivindicación 9 en el que el primer bloque de actualización almacena sectores en un orden lógico no secuencial.

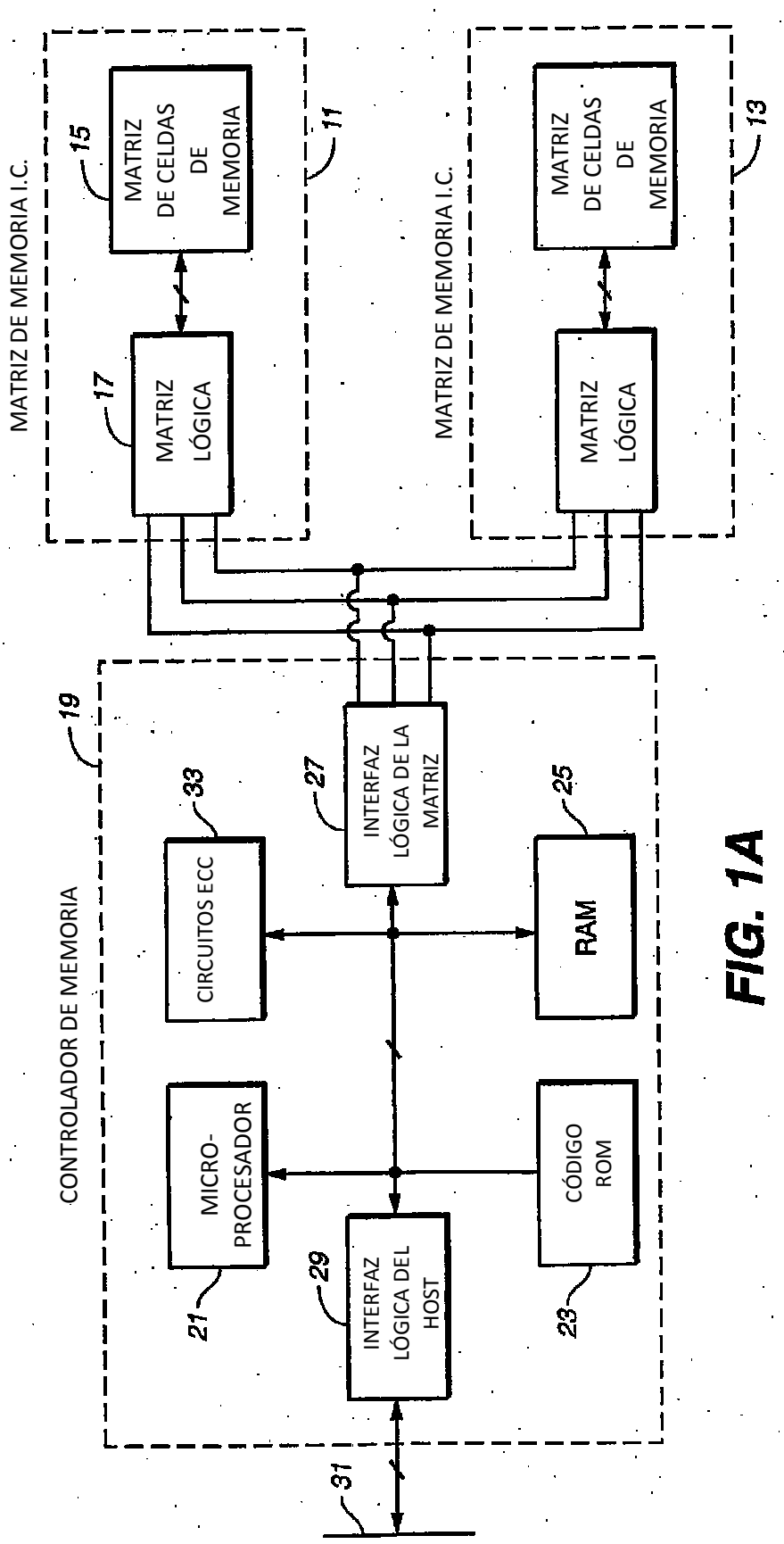

MATRIZ DE MEMORIA I.C.
15
MATRIZ DE CELDAS DE MEMORIA
11
17
MATRIZ LÓGICA
MATRIZ DE MEMORIA I.C.
MATRIZ DE CELDAS DE MEMORIA
13
MATRIZ LÓGICA
19
CONTROLADOR DE MEMORIA
33
CIRCUITOS ECC
27
INTERFAZ LÓGICA DE LA MATRIZ
25
RAM
MICRO-PROCESADOR
21
29
INTERFAZ LÓGICA DEL HOST
CÓDIGO ROM
23
31

FIG. 1A

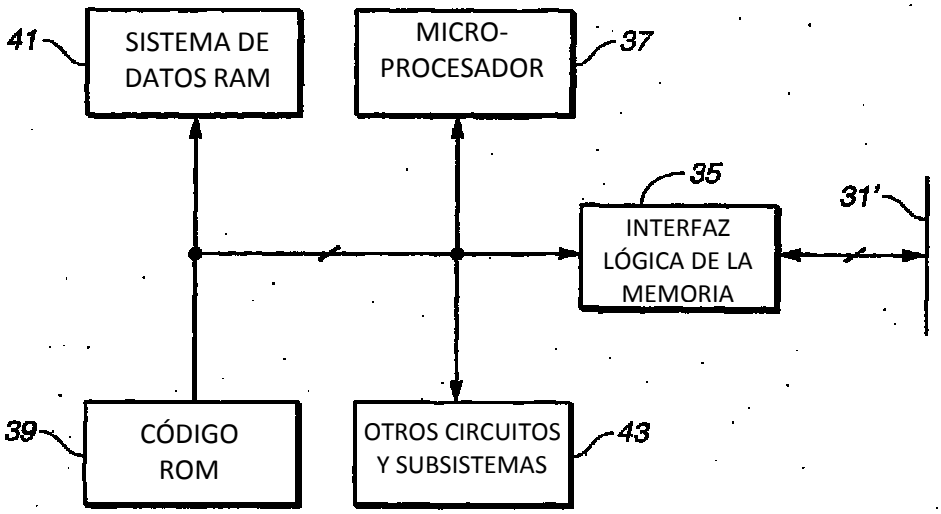


FIG. 1B

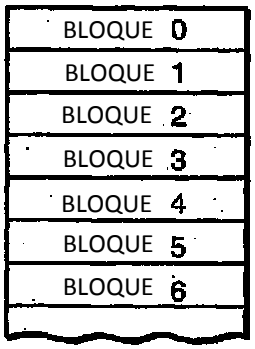


FIG. 2

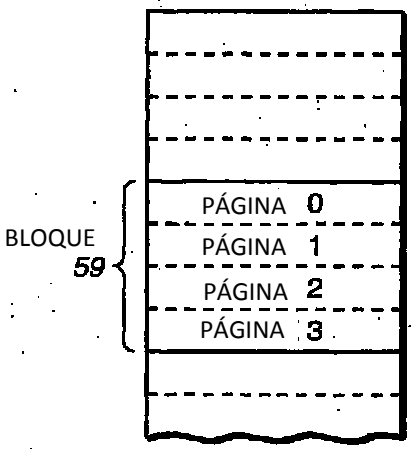


FIG. 4

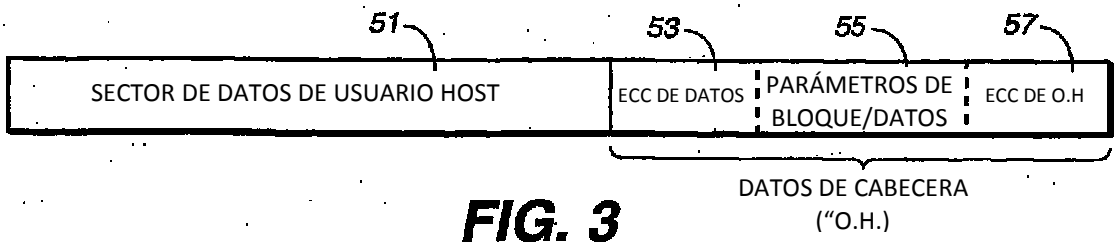


FIG. 3

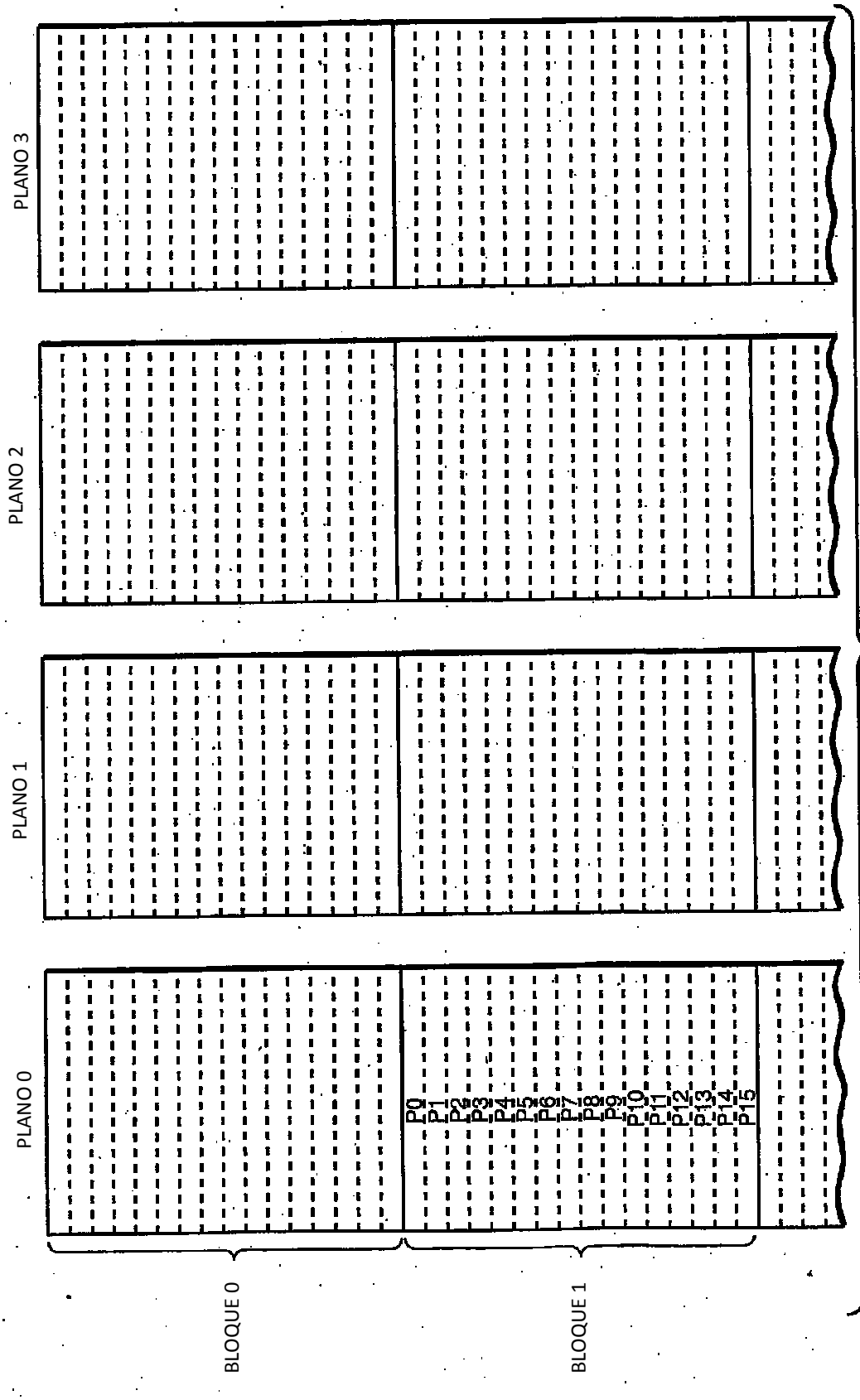

PLANO 3
PLANO 2
PLANO 1
PLANO 0
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
BLOQUE 0
BLOQUE 1

FIG. 5

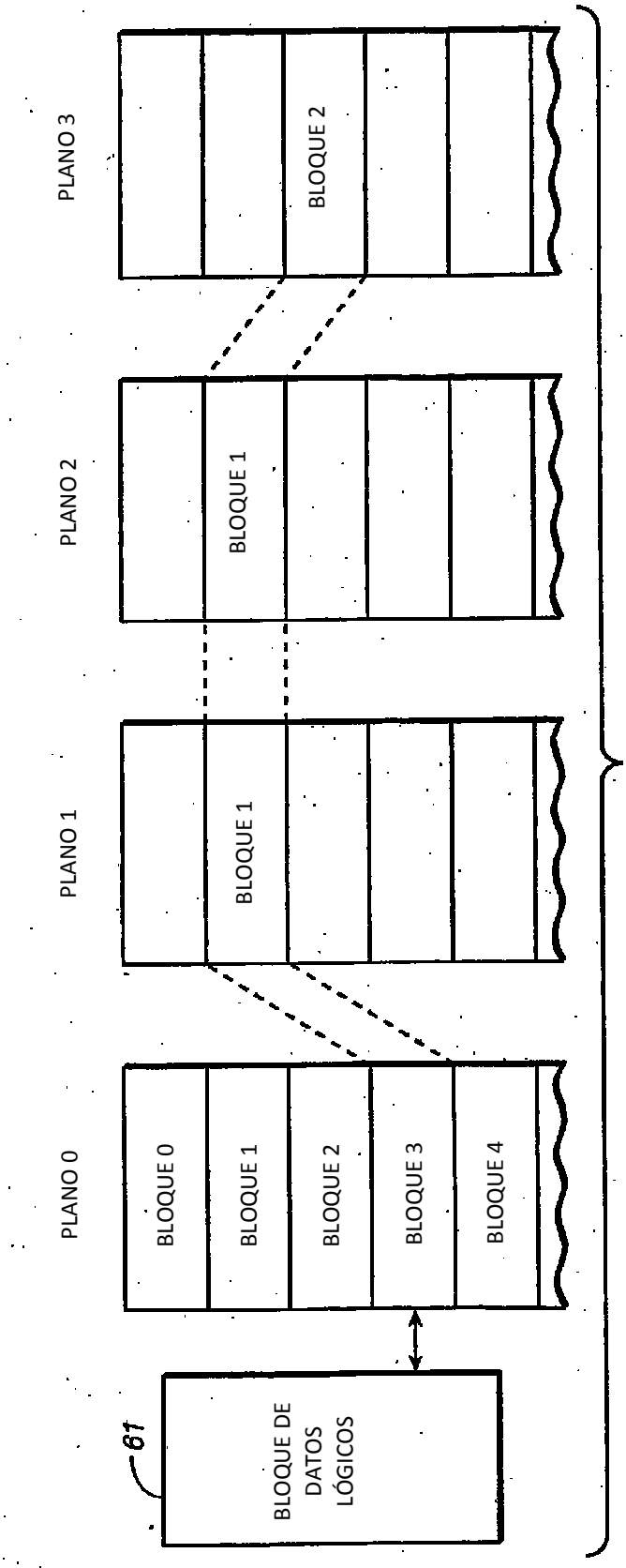

PLANO 3
PLANO 2
PLANO 1
PLANO 0
BLOQUE 0
BLOQUE 1
BLOQUE 2
BLOQUE 3
BLOQUE 4
BLOQUE DE DATOS LÓGICOS
61

FIG. 6

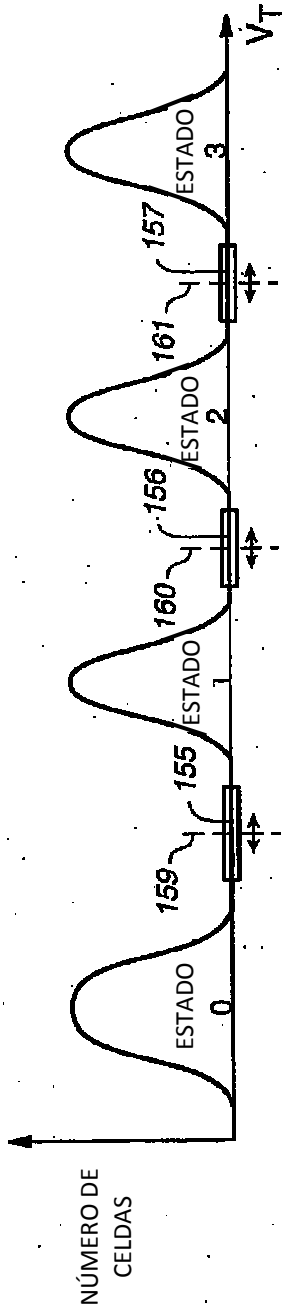

V_T
ESTADO 0
ESTADO 1
ESTADO 2
ESTADO 3
155
156
157
159
160
161
NÚMERO DE CELDAS

FIG. 10

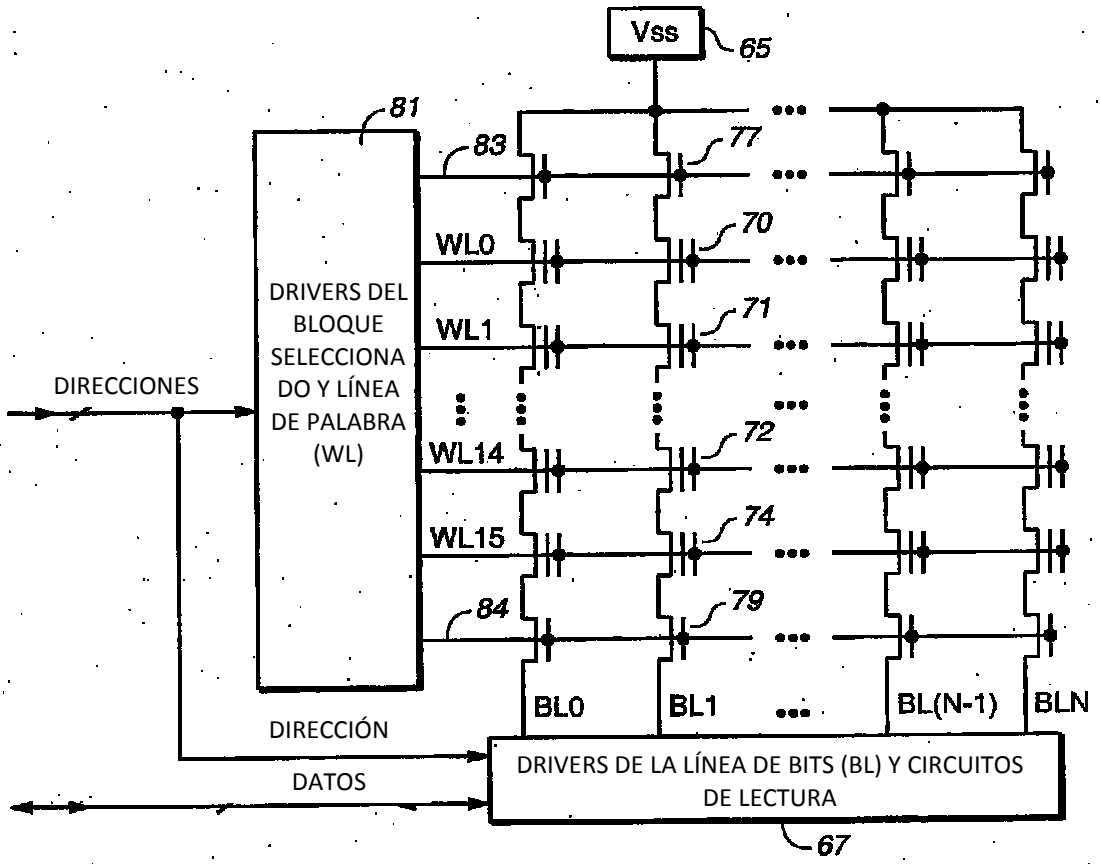

Vss
65
81
83
77
70
71
72
74
79
84
WL0
WL1
WL14
WL15
DRIVERS DEL BLOQUE SELECCIONADO Y LÍNEA DE PALABRA (WL)
DIRECCIONES
DIRECCIÓN
DATOS
BL0
BL1
BL(N-1)
BLN
DRIVERS DE LA LÍNEA DE BITS (BL) Y CIRCUITOS DE LECTURA
67

FIG. 7

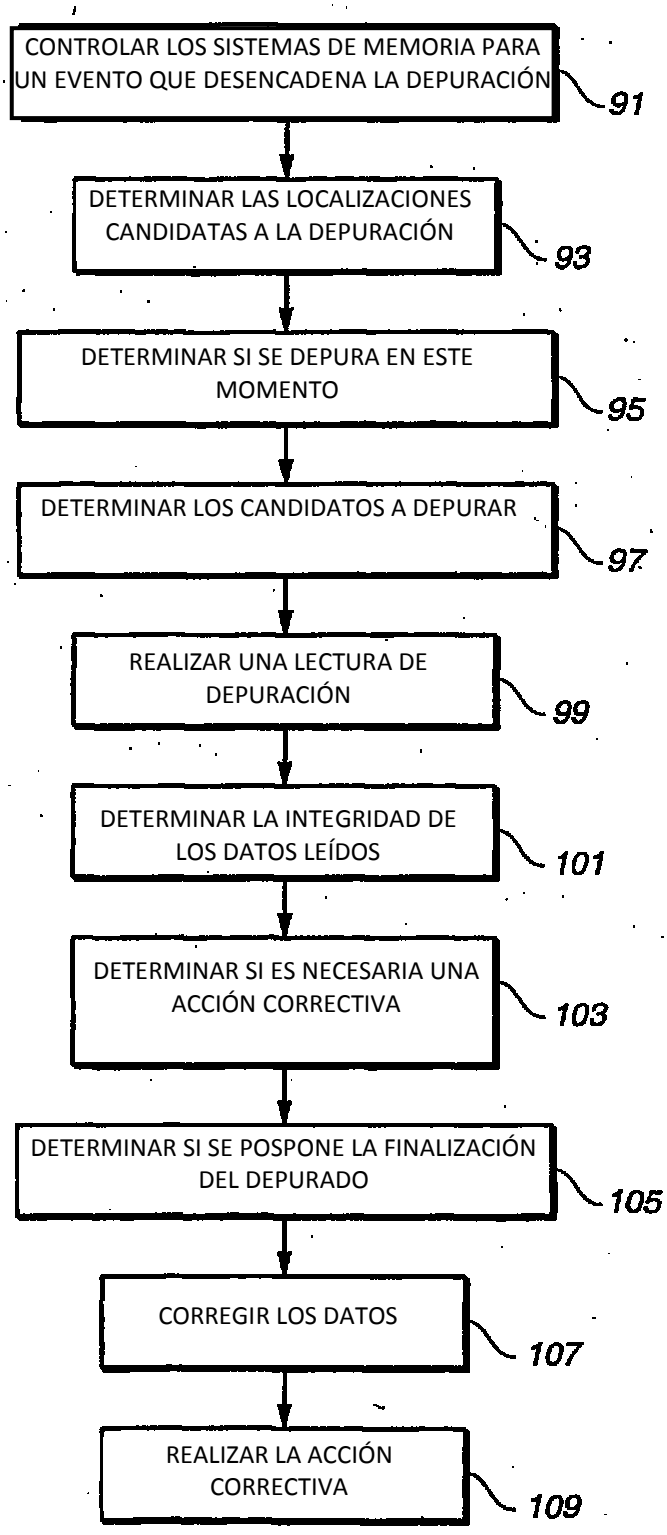

CONTROLAR LOS SISTEMAS DE MEMORIA PARA UN EVENTO QUE DESENCADENA LA DEPURACIÓN
91
DETERMINAR LAS LOCALIZACIONES CANDIDATAS A LA DEPURACIÓN
93
DETERMINAR SI SE DEPURA EN ESTE MOMENTO
95
DETERMINAR LOS CANDIDATOS A DEPURAR
97
REALIZAR UNA LECTURA DE DEPURACIÓN
99
DETERMINAR LA INTEGRIDAD DE LOS DATOS LEÍDOS
101
DETERMINAR SI ES NECESARIA UNA ACCIÓN CORRECTIVA
103
DETERMINAR SI SE POSPONE LA FINALIZACIÓN DEL DEPURADO
105
CORREGIR LOS DATOS
107
REALIZAR LA ACCIÓN CORRECTIVA
109

FIG. 8

FIG. 9

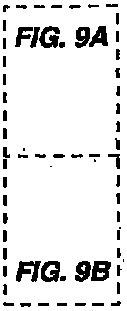
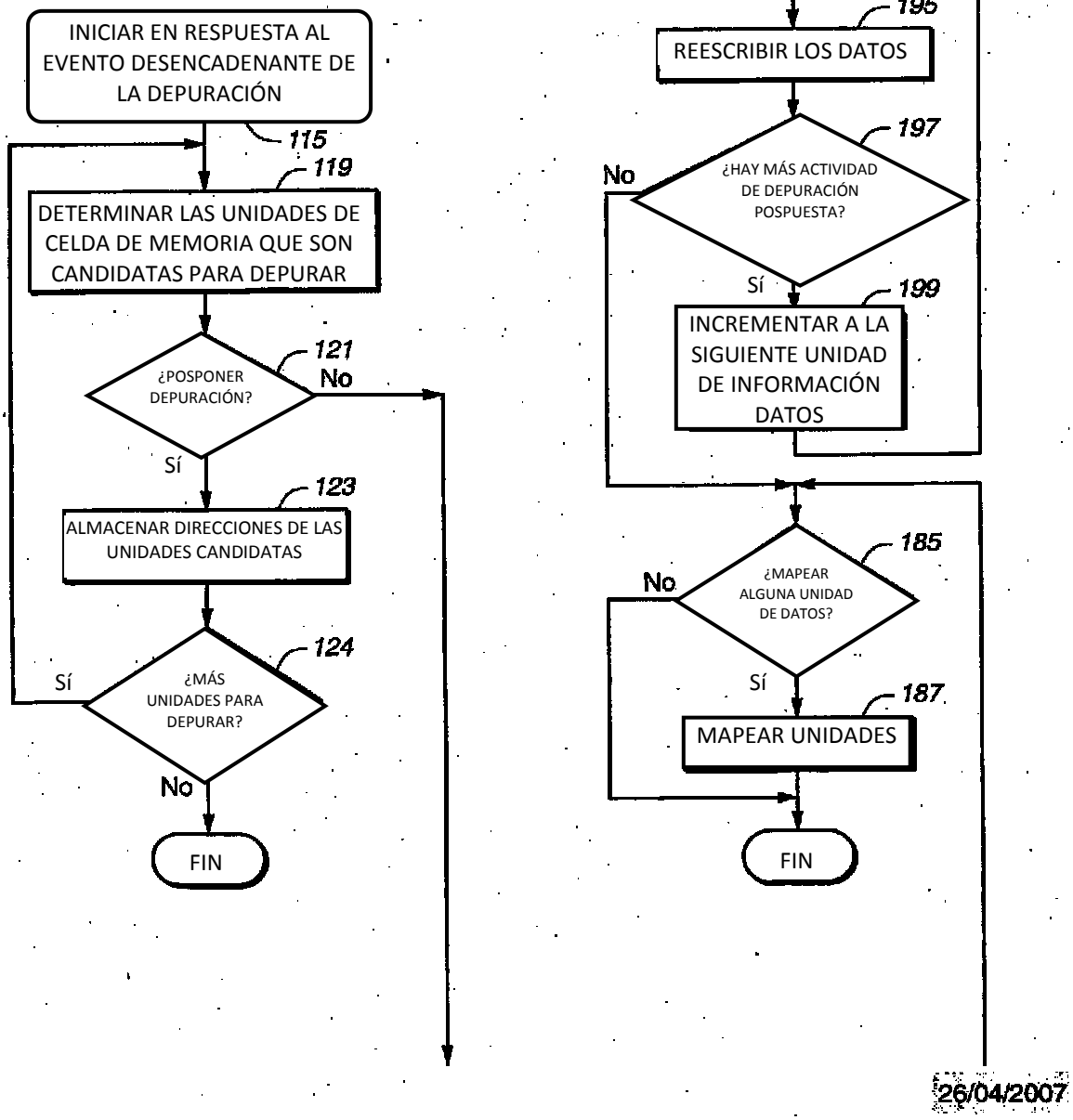


FIG. 9A
FIG. 9B

FIG. 9A


INICIAR EN RESPUESTA AL EVENTO DESENCADENANTE DE LA DEPURACIÓN
115
DETERMINAR LAS UNIDADES DE CELDA DE MEMORIA QUE SON CANDIDATAS PARA DEPURAR
119
¿POSPONER DEPURACIÓN?
121
No
Sí
ALMACENAR DIRECCIONES DE LAS UNIDADES CANDIDATAS
123
¿MÁS UNIDADES PARA DEPURAR?
124
Sí
No
FIN
INICIAR EN RESPUESTA AL EVENTO DESENCADENANTE DE LA DEPURACIÓN POSPUESTA
117
LEER DATOS Y PARÁMETROS ALMACENADOS PARA SU CORRECCIÓN Y/O REESCRITURA
191
CORREGIR LOS DATOS
193
REESCRIBIR LOS DATOS
195
¿HAY MÁS ACTIVIDAD DE DEPURACIÓN POSPUESTA?
197
No
Sí
INCREMENTAR A LA SIGUIENTE UNIDAD DE INFORMACIÓN DATOS
199
¿MAPEAR ALGUNA UNIDAD DE DATOS?
185
No
Sí
MAPEAR UNIDADES
187
FIN
26/04/2007

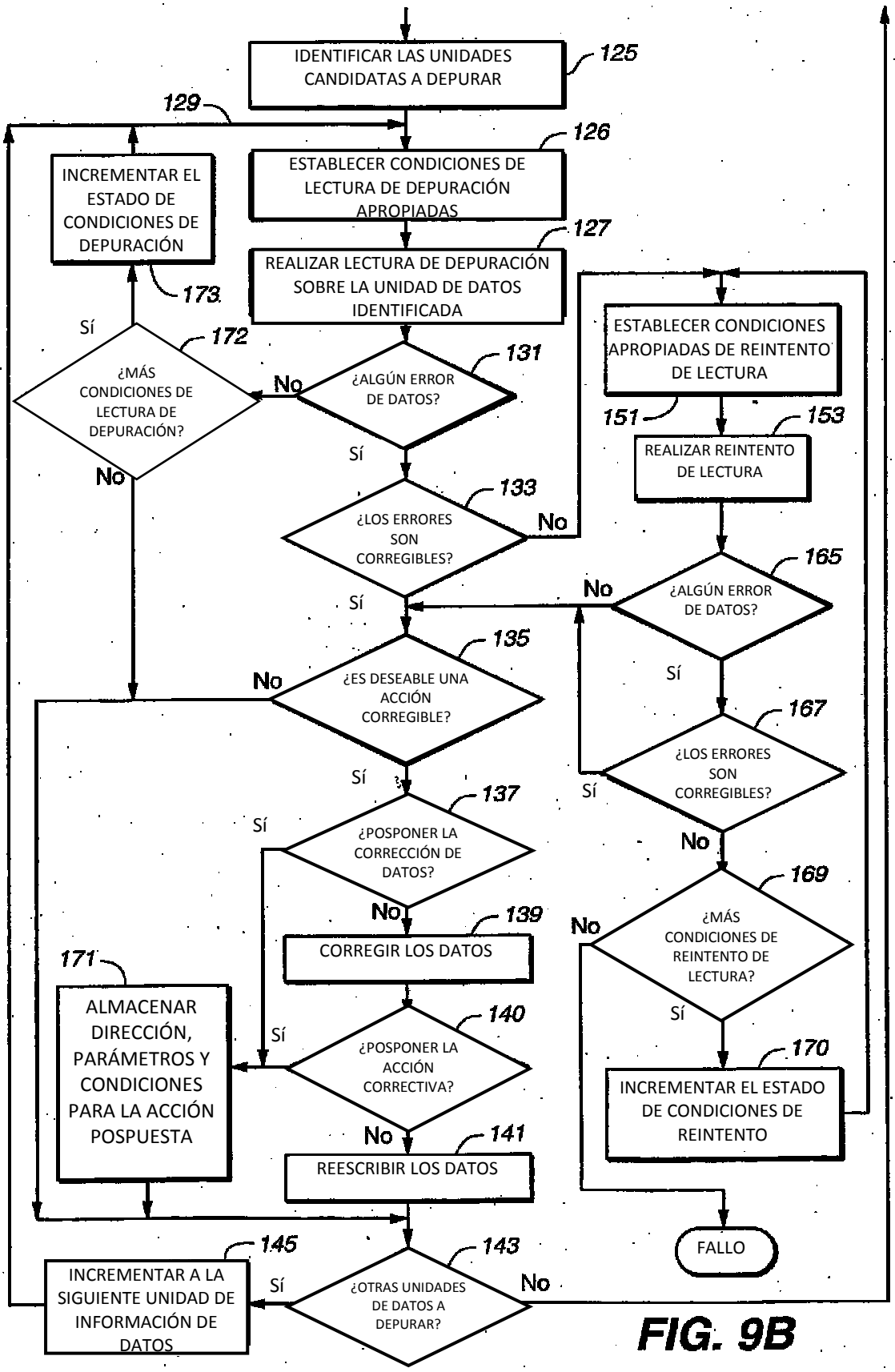

IDENTIFICAR LAS UNIDADES CANDIDATAS A DEPURAR
125
129
126
ESTABLECER CONDICIONES DE LECTURA DE DEPURACIÓN APROPIADAS
INCREMENTAR EL ESTADO DE CONDICIONES DE DEPURACIÓN
127
REALIZAR LECTURA DE DEPURACIÓN SOBRE LA UNIDAD DE DATOS IDENTIFICADA
173
Sí
172
¿MÁS CONDICIONES DE LECTURA DE DEPURACIÓN?
No
131
¿ALGÚN ERROR DE DATOS?
No
Sí
133
¿LOS ERRORES SON CORREGIBLES?
No
Sí
135
¿ES DESEABLE UNA ACCIÓN CORREGIBLE?
No
Sí
137
¿POSPONER LA CORRECCIÓN DE DATOS?
Sí
No
139
CORREGIR LOS DATOS
171
ALMACENAR DIRECCIÓN, PARÁMETROS Y CONDICIONES PARA LA ACCIÓN POSPUESTA
Sí
140
¿POSPONER LA ACCIÓN CORRECTIVA?
No
141
REESCRIBIR LOS DATOS
145
INCREMENTAR A LA SIGUIENTE UNIDAD DE INFORMACIÓN DE DATOS
Sí
143
¿OTRAS UNIDADES DE DATOS A DEPURAR?
No
ESTABLECER CONDICIONES APROPIADAS DE REINTENTO DE LECTURA
151
153
REALIZAR REINTENTO DE LECTURA
165
¿ALGÚN ERROR DE DATOS?
No
Sí
167
¿LOS ERRORES SON CORREGIBLES?
Sí
No
169
¿MÁS CONDICIONES DE REINTENTO DE LECTURA?
No
Sí
170
INCREMENTAR EL ESTADO DE CONDICIONES DE REINTENTO
FALLO

FIG. 9B

LECTURA HOST CON REEMPLAZAMIENTO PARCIAL

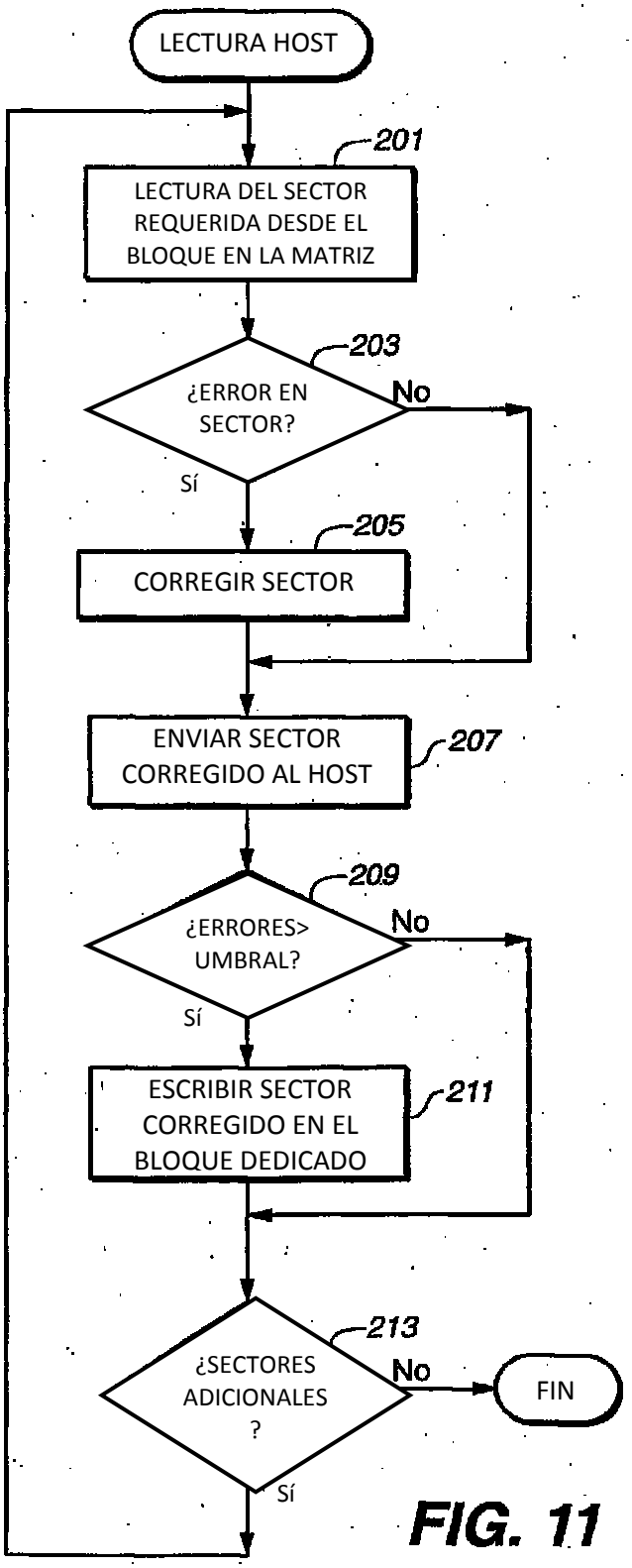


FIG. 11

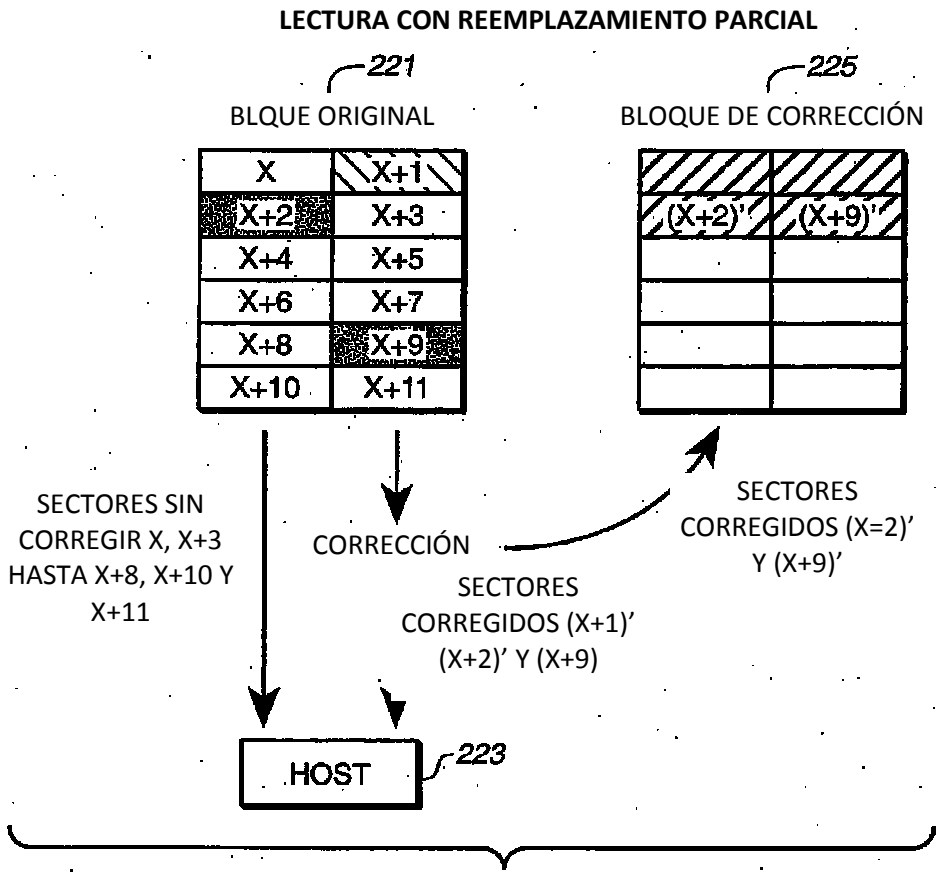

LECTURA CON REEMPLAZAMIENTO PARCIAL
221
BLQUE ORIGINAL
225
BLOQUE DE CORRECCIÓN
X
X+1
X+2
X+3
X+4
X+5
X+6
X+7
X+8
X+9
X+10
X+11
(X+2)'
(X+9)'
SECTORES SIN CORREGIR X, X+3 HASTA X+8, X+10 Y X+11
CORRECCIÓN
SECTORES CORREGIDOS (X+1)' (X+2)' Y (X+9)
SECTORES CORREGIDOS (X=2)' Y (X+9)'
HOST
223

FIG. 12

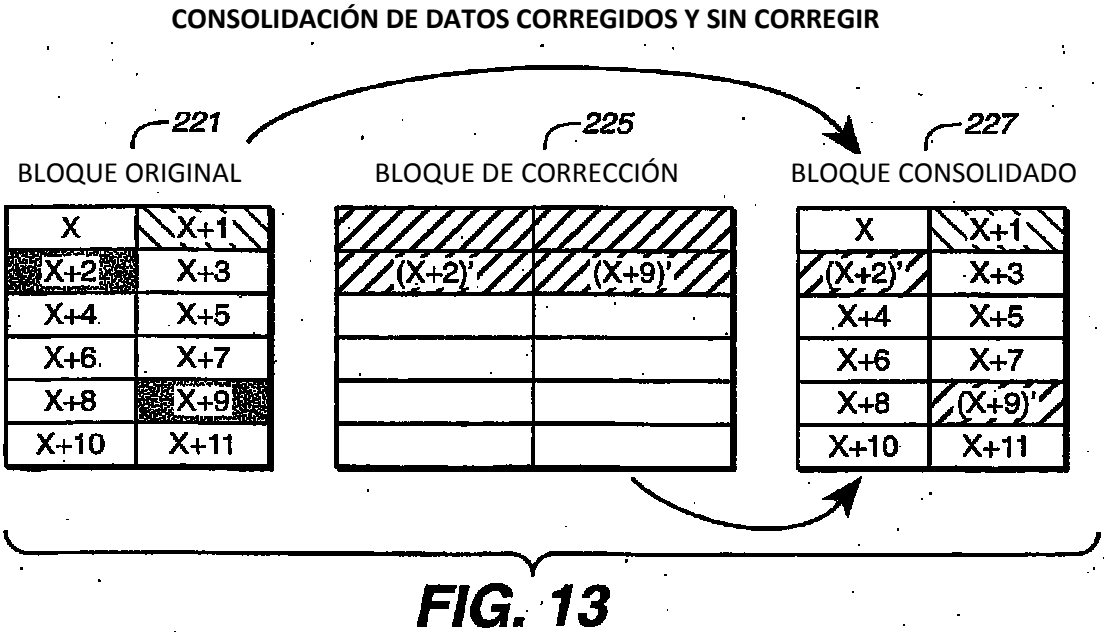

CONSOLIDACIÓN DE DATOS CORREGIDOS Y SIN CORREGIR
221
BLOQUE ORIGINAL
225
BLOQUE DE CORRECCIÓN
227
BLOQUE CONSOLIDADO
X
X+1
X+2
X+3
X+4
X+5
X+6
X+7
X+8
X+9
X+10
X+11
(X+2)'
(X+9)'

FIG. 13

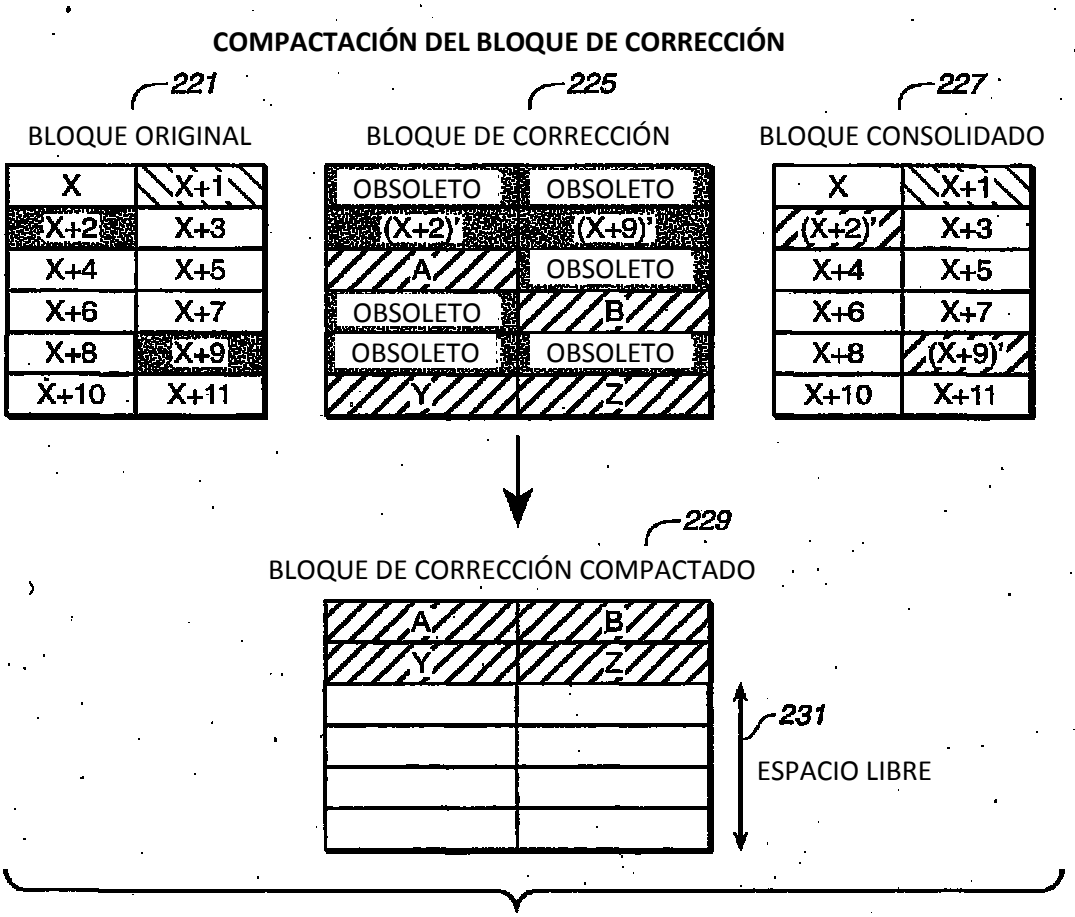

COMPACTACIÓN DEL BLOQUE DE CORRECCIÓN
221
BLOQUE ORIGINAL
X
X+1
X+2
X+3
X+4
X+5
X+6
X+7
X+8
X+9
X+10
X+11
225
BLOQUE DE CORRECCIÓN
OBSOLETO
OBSOLETO
(X+2)'
(X+9)'
A
OBSOLETO
OBSOLETO
B
OBSOLETO
OBSOLETO
Y
Z
227
BLOQUE CONSOLIDADO
X
X+1
(X+2)'
X+3
X+4
X+5
X+6
X+7
X+8
(X+9)'
X+10
X+11
229
BLOQUE DE CORRECCIÓN COMPACTADO
A
B
Y
Z
231
ESPACIO LIBRE

FIG. 14

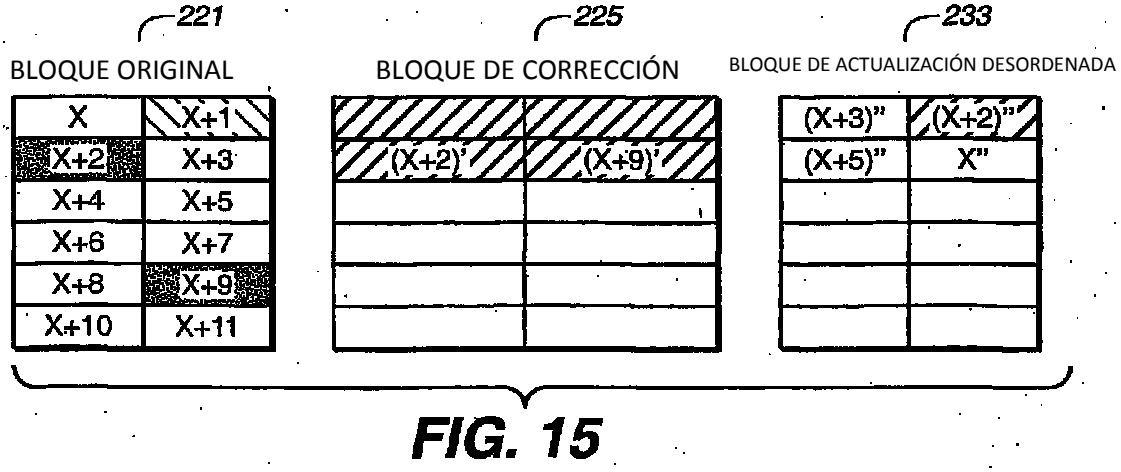

BLOQUES DE CORRECCIÓN Y ACTUALIZACIÓN
221
BLOQUE ORIGINAL
X
X+1
X+2
X+3
X+4
X+5
X+6
X+7
X+8
X+9
X+10
X+11
225
BLOQUE DE CORRECCIÓN
(X+2)'
(X+9)'
233
BLOQUE DE ACTUALIZACIÓN DESORDENADA
(X+3)"
(X+2)"
(X+5)"
X"

FIG. 15

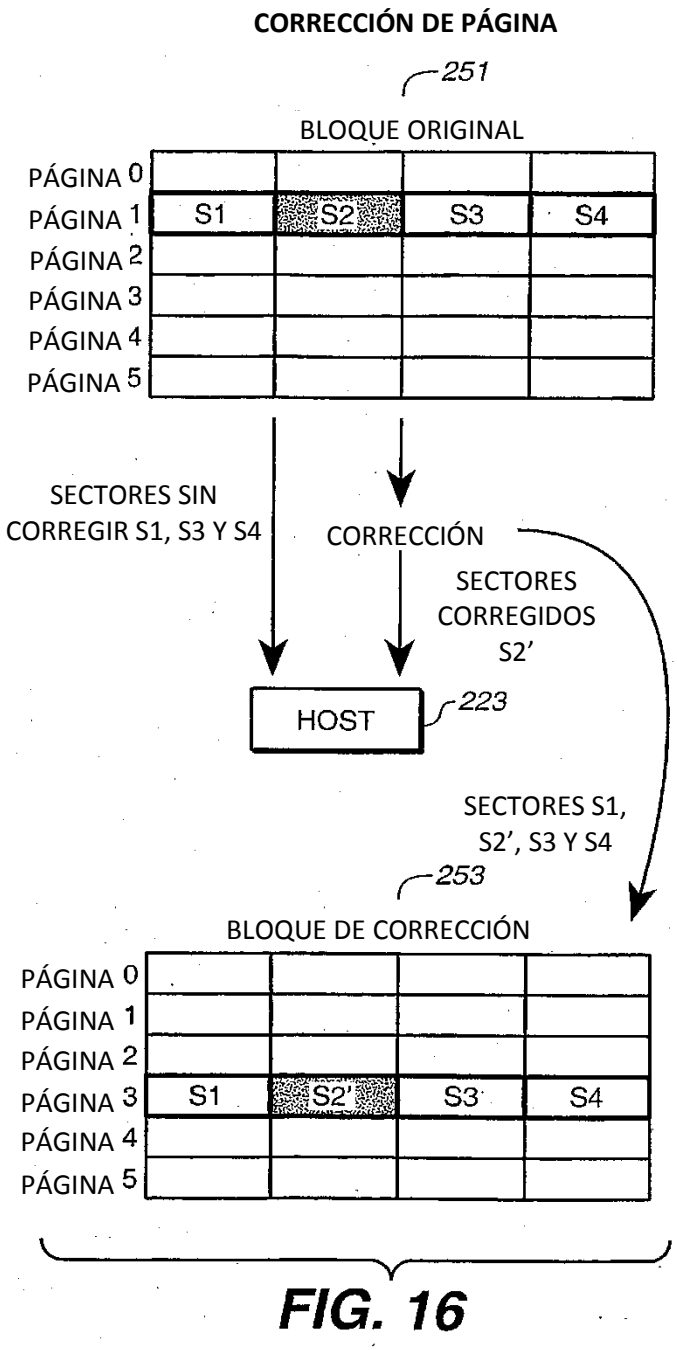

CORRECCIÓN DE PÁGINA
251
BLOQUE ORIGINAL
PÁGINA 0
PÁGINA 1
PÁGINA 2
PÁGINA 3
PÁGINA 4
PÁGINA 5
S1
S2
S3
S4
SECTORES SIN CORREGIR S1, S3 Y S4
CORRECCIÓN
SECTORES CORREGIDOS S2'
HOST
223
SECTORES S1, S2', S3 Y S4
253
BLOQUE DE CORRECCIÓN
PÁGINA 0
PÁGINA 1
PÁGINA 2
PÁGINA 3
PÁGINA 4
PÁGINA 5
S1
S2'
S3
S4

FIG. 16