(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년01월08일
(11) 등록번호 10-1348255
(24) 등록일자 2013년12월30일

(51) 국제특허분류(Int. Cl.)
G06F 12/00 (2006.01) *G06F 12/02* (2006.01)
(21) 출원번호 10-2008-7011330
(22) 출원일자(국제) 2006년09월28일
심사청구일자 2011년09월23일
(85) 번역문제출일자 2008년05월13일
(65) 공개번호 10-2008-0074124
(43) 공개일자 2008년08월12일
(86) 국제출원번호 PCT/US2006/038008
(87) 국제공개번호 WO 2007/047062
국제공개일자 2007년04월26일
(30) 우선권주장
11/250,299 2005년10월13일 미국(US)
11/250,794 2005년10월13일 미국(US)
(56) 선행기술조사문헌
MEMORY MANAGEMENT INTERNATIONAL WORKSHOP "Memory Management in Flash-Memory Disks with Data Compression"
US06145069 A
전체 청구항 수 : 총 29 항

(73) 특허권자
샌디스크 테크놀로지스, 인코포레이티드
미국 텍사스 75024 플라노 노스 달라스 파크웨이 6900 투 리가시 타운 센터
(72) 발명자
신클레어, 알란, 더블유.
영국, 매디스톤 에프케이2 0비유, 브로드헤드 캔디, 더 코테지스
(74) 대리인
박경재

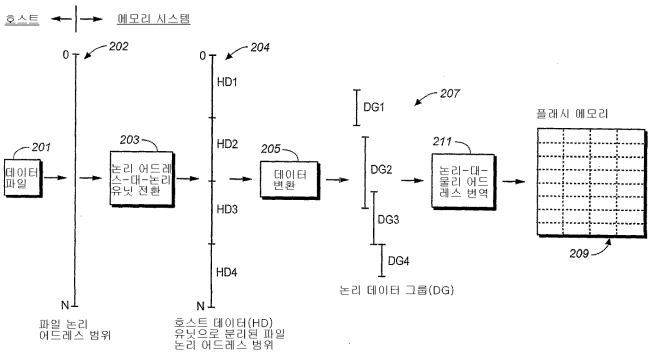
심사관 : 이명진

(54) 발명의 명칭 **고정된 크기의 저장 블록을 가진 메모리 시스템에서데이터의 변환된 유닛의 저장**

(57) 요 약

여러 인코딩, 압축, 암호화 또는 다른 데이터 변환 알고리즘에 기인하는 저장된 데이터의 양의 변화는 변환된 데이터의 별도의 유닛을 개별적으로 식별하며, 플래시 메모리와 같은 메모리 시스템의 저장 블록 내에서 물리적인 연속물에 이러한 유닛을 저장함으로써 처리된다. 저장된 데이터는 메모리 시스템 내의 프로세서 상에서 작동하는 애플리케이션 또는 메모리 시스템의 외부의 호스트 시스템으로부터 나올 수 있다.

대 표 도



특허청구의 범위

청구항 1

비휘발성 메모리 시스템에 있어서,

데이터 저장 소자의 개별 블록으로 분리된 플래시 메모리로서, 상기 개별 블록은 데이터를 기록하기 전에 한번에 물리적으로 삭제 가능한 최소 수의 저장 소자를 포함하고, 상기 각각의 개별 블록은 주어진 양의 데이터까지 저장하는, 상기 플래시 메모리와,

펌웨어에 의해 작동되는 마이크로프로세서를 포함하고 상기 플래시 메모리와 연결된 제어기로서, 상기 제어기는 상기 플래시 메모리에 의해 수신된 데이터를 동일한 크기의 개별 데이터 유닛으로 분할하도록 작동하고, 상기 데이터 유닛은 연속 논리 어드레스 범위 내에 인접한 논리 어드레스를 가지며, 상기 제어기는, 상기 유닛의 적어도 일부에서 데이터의 양을 변화시켜 상기 적어도 일부 유닛에 해당하는 데이터 그룹이 상기 적어도 일부 유닛의 것과 다른 데이터의 양을 포함하는 방식으로 상기 개별 데이터 유닛을 해당 데이터 그룹으로 변환하고, 저장 소자의 블록에 상기 데이터 그룹을 저장하도록 작동하고, 상기 각각의 유닛은 상기 주어진 양의 데이터 이하의 데이터 양을 포함하며, 상기 제어기는, 그룹의 논리 경계와 일치하는 물리 경계를 갖는 데이터 그룹의 물리 어드레스에 의해 저장 위치를 식별하고, 상기 그룹의 일부는 저장 소자의 개별 블록 내에 완전히 저장되고, 상기 그룹의 다른 일부는 그 경계가 저장 소자의 블록 내에 있는 저장 소자의 두 개의 서로 다른 블록에 독립된 서브 그룹(sub-group)으로 저장되며, 상기 그룹과 서브 그룹은 저장 소자의 개별 블록 내에 연속적으로 저장되고, 상기 서브 그룹은 상기 주어진 양 미만의 데이터 양을 함유하며, 상기 제어기는 물리 어드레스를 포함하는 데이터의 상기 개별 그룹과 서브 그룹에 대한 엔트리(entry)를 갖는 중앙 테이블(central table)을 또한 유지하는, 상기 제어기를

포함하는, 비휘발성 메모리 시스템.

청구항 2

제 1항에 있어서, 상기 제어기는, 데이터 그룹의 물리 어드레스와 분리되고 다른 서브 그룹의 물리 어드레스를 구비한 중앙 테이블을 유지하도록 추가 작동하는, 비휘발성 메모리 시스템.

청구항 3

제 1항에 있어서, 상기 제어기는, 상기 데이터를 인코딩, 압축, 또는 암호화하는 것 중 적어도 하나에 의해 상기 수신된 데이터를 변환하도록 추가 작동하는, 비휘발성 메모리 시스템.

청구항 4

제 3항에 있어서, 상기 제어기는, 변환 전에 존재하는 데이터의 양을 감소시키는 방식으로 상기 수신된 데이터를 변환하도록 추가 작동하는, 비휘발성 메모리 시스템.

청구항 5

제 3항에 있어서, 상기 제어기는, 변환 전에 존재하는 데이터의 양을 증가시키는 방식으로 상기 수신된 데이터를 변환하도록 추가 작동하는, 비휘발성 메모리 시스템.

청구항 6

제 1항에 있어서, 상기 제어기는, 동일한 양의 데이터를 포함하는 논리 어드레스의 개별 유닛 내에 수신된 데이터를 변환하도록 추가 작동하는, 비휘발성 메모리 시스템.

청구항 7

제 1항에 있어서, 변환된 제어기에 의해 수신된 데이터는 상기 메모리 시스템의 외부에서 생성되는, 비휘발성 메모리 시스템.

청구항 8

제 1항에 있어서, 변환된 제어기에 의해 수신된 데이터는 상기 제어기에 의해 실행되는 애플리케이션으로 생성되는, 비휘발성 메모리 시스템.

청구항 9

제 1항에 있어서, 상기 저장 소자는 전기 전도성 플로팅 게이트를 포함하는, 비휘발성 메모리 시스템.

청구항 10

제 1항에 있어서, 상기 제어기는, 상기 수신된 데이터의 변환의 적어도 일부를 수행하기 위한 전용 회로를 더 포함하는, 비휘발성 메모리 시스템.

청구항 11

제 1항에 있어서, 상기 제어기는, 상기 저장 소자 중 개별 저장 소자에서 1 비트 이상의 데이터를 저장하도록 추가 작동하는, 비휘발성 메모리 시스템.

청구항 12

제 1항에 있어서, 상기 제어기는, 상기 메모리 시스템에 의해 수신된 데이터의 논리 어드레스 범위를 복수의 연속 논리 어드레스 개별 유닛으로 분할하도록 추가 작동하고, 상기 개별 유닛은 각각 상기 주어진 양의 데이터 이하의 데이터 양을 저장할 수 있는, 비휘발성 메모리 시스템.

청구항 13

제 1항에 있어서, 상기 제어기는 수신된 데이터를 사기 개별 유닛으로 분할하고, 상기 개별 유닛은 크기가 동일하며, 상기 각각의 개별 유닛은 저장 소자의 개별 블록의 저장 용량 이하의 데이터 양을 각각 포함하고, 상기 각각의 개별 유닛 내의 데이터는 논리 어드레스 범위 내에 인접한 논리 어드레스를 갖는, 비휘발성 메모리 시스템.

청구항 14

제 1항에 있어서, 저장 소자의 상기 블록 중 하나는 제 1 데이터 그룹을 저장하고, 이러한 하나의 블록은 상기 제 1 데이터 그룹 외에 데이터에 이용 가능한 저장 용량을 갖고, 상기 제어기는 이러한 블록에서 상기 제 1 데이터 그룹과 다른 제 2 데이터 그룹의 서브 그룹을 저장하는, 비휘발성 메모리 시스템.

청구항 15

논리 어드레스 범위 내에 논리 어드레스를 갖는 메모리 시스템에 의해 수신된 데이터를 저장하기 위해 재프로그래밍 가능한 비휘발성 메모리 시스템을 작동시키는 방법으로서,

상기 메모리 시스템은 셀의 개별 작동 블록으로 구성된 메모리 셀을 포함하고, 개별 블록은 데이터를 기록하기 전에 한번에 물리적으로 삭제 가능한 최소 수의 메모리 셀을 포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템을 작동시키는 방법에 있어서,

상기 메모리 시스템에 의해 수신된 데이터를, 개별 메모리 셀 블록의 저장 용량 이하의 데이터 양을 개별 포함하는 동일 크기의 데이터 유닛으로 분할하는 단계로서, 개별 유닛 내의 데이터는 논리 어드레스 범위 내에 인접한 논리 어드레스를 갖는, 상기 분할 단계와,

상기 수신된 데이터를 인코딩, 압축, 또는 암호화하는 것 중 적어도 하나에 의해 수신 데이터의 개별 유닛을, 변환 이전 데이터의 해당 유닛에 존재하는 것과 다른 양의 데이터를 개별 포함하는 해당 데이터 그룹으로 변환시키는 단계와,

데이터 그룹의 최소 경계 및 메모리 셀 블록의 물리 경계와 일치하는 경계를 갖는 하나 이상의 메모리 셀 블록 내의 인접한 위치에 데이터 그룹을 기록하는 단계로서, 두 개의 메모리 셀 블록 각각에 위치한 서브 그룹 중 하나를 구비한 두 개의 메모리 셀 블록을 지나는 개별 그룹을 두 개의 서브 그룹으로 나누는 단계를 포함하는, 상기 기록 단계와,

데이터의 개별 그룹과 서브 그룹의 위치의 중앙 테이블을 상기 메모리 셀 블록 내에 유지하는 단계로서, 상기 메모리 셀 블록의 경계 내에 있는 데이터의 개별 그룹과 서브 그룹에 대해 한정된 경계를 유지하는 단계를 포함

하는, 상기 유지 단계를

포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 16

제 15항에 있어서, 상기 메모리에 변환되어 기록된 이전 수신 데이터 중 적어도 하나의 개별 유닛 내의 데이터를 갱신하는 데이터를 수신하는 것에 응답하여, 상기 메모리 시스템 내에서,

상기 적어도 하나의 변환된 개별 데이터 유닛을 형성하는 변환 데이터의 하나 이상의 그룹을 판독하는 단계와,

상기 판독된 데이터를 디코딩, 압축풀기, 또는 암호해독하는 것 중 적어도 하나에 의해 변환 데이터의 판독 그룹을 역변환시켜, 상기 적어도 하나의 개별 데이터 유닛을 재구성하는 단계와,

이후 상기 수신된 갱신 데이터를 갖는 상기 재구성된 적어도 하나의 개별 유닛을 수정하는 단계와,

이후 상기 수정된 재구성된 데이터를 인코딩, 압축, 또는 암호화하는 것 중 적어도 하나에 의해 수정된 적어도 하나의 데이터 유닛을 해당 데이터 그룹으로 변환하는 단계와,

상기 데이터 그룹을 하나 이상의 메모리 셀 블록 내의 인접한 위치에 재기록하는 단계로서, 두 개의 메모리 셀 블록 각각에 위치한 서브 그룹 중 하나를 구비한 두 개의 메모리 셀 블록을 지나는 개별 그룹을 두 개의 서브 그룹으로 나누는 단계를 포함하는, 상기 재기록 단계와,

상기 개별 재기록 그룹과 서브 그룹의 위치의 중앙 테이블을 상기 메모리 셀 블록 내에서 개정하는 단계로서, 상기 메모리 블록의 물리 경계 내에 있는 데이터의 개별 그룹과 서브 그룹에 대해 한정된 경계를 유지하는 단계를 포함하는, 상기 개정 단계를

더 포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 17

제 15항에 있어서, 상기 데이터를 변환하는 단계는, 상기 변환 데이터 유닛의 적어도 일부가 변환 전에 존재하는 것보다 더 적은 양의 데이터를 개별 포함하는 유닛을 포함하도록 하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 18

제 15항에 있어서, 상기 데이터를 변환하는 단계는, 상기 변환 데이터 유닛의 적어도 일부가 변환 전에 존재하는 것보다 더 많은 양의 데이터를 개별 포함하는 유닛을 포함하도록 하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 19

제 15항에 있어서, 상기 변환 데이터의 한정 그룹은 서로 다른 셀 블록에 존재하는 데이터 그룹의 서브 그룹을 함께 연결하는 단계를 포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 20

제 15항에 있어서,

적어도 하나의 추가 메모리 셀 블록에서 변환되지 않고 메모리 시스템에 의해 수신된 추가 데이터 유닛을 저장하는 단계와,

상기 메모리 셀 블록 내에서 추가 데이터 그룹에 의해 상기 추가 데이터 유닛의 저장 위치를 식별하는 단계를

더 포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 21

제 15항에 있어서, 상기 방법은, 데이터가 수신되는 메모리 커넥터에 의해 호스트 장치에 분리 가능하게 연결될 수 있는 메모리 시스템 모듈에서 실행되는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 22

제 15항에 있어서, 상기 메모리 시스템에 의해 수신된 데이터는 상기 메모리 시스템 내에서 프로세서에 의해 실행된 애플리케이션 프로그램으로부터 생성되는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 23

제 15항에 있어서, 상기 논리 어드레스 범위는 상기 메모리 시스템의 논리 어드레스 범위인, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 24

제 15항에 있어서, 상기 논리 어드레스 범위는 저장되는 데이터의 파일 오브젝트(file object)의 논리 어드레스 범위인, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 25

논리 어드레스 범위 내에 논리 어드레스를 갖는 메모리 시스템에 의해 수신된 데이터를 압축하고 상기 압축 데이터를 저장하기 위해 재프로그래밍 가능한 비휘발성 메모리 시스템을 작동시키는 방법으로서,

상기 메모리 시스템은 셀의 개별 작동 블록으로 구성된 메모리 셀을 포함하고, 개별 블록은 데이터를 기록하기 전에 한번에 물리적으로 삭제 가능한 최소 수의 메모리 셀을 포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템을 작동시키는 방법에 있어서,

상기 메모리 시스템에 의해 수신된 데이터를, 개별 메모리 셀 블록의 저장 용량 이하의 데이터 양을 개별 포함하는 동일 크기의 데이터 유닛으로 분할하는 단계로서, 개별 그룹 내의 데이터는 논리 어드레스 범위 내에 인접한 논리 어드레스를 갖는, 상기 분할 단계와,

수신 데이터의 개별 유닛을 압축하여, 압축 이전 데이터의 해당 유닛보다 개별적으로 적은 데이터의 해당 그룹을 생성하는 단계와,

데이터 그룹의 최소 경계 및 메모리 셀 블록의 물리 경계와 일치하는 경계를 갖는 하나 이상의 메모리 셀 블록 내의 인접한 위치에 데이터 그룹을 기록하는 단계로서, 두 개의 메모리 셀 블록 각각에 위치한 서브 그룹 중 하나를 구비한 두 개의 메모리 셀 블록을 지나는 개별 그룹을 두 개의 서브 그룹으로 나누는 단계를 포함하는, 상기 기록 단계와,

데이터의 개별 그룹과 서브 그룹의 위치의 중앙 테이블을 상기 메모리 셀 블록 내에 유지하는 단계로서, 상기 메모리 셀 블록의 경계 내에 있는 데이터의 개별 그룹과 서브 그룹에 대해 한정된 경계를 유지하는 단계를 포함하는, 상기 유지 단계를

포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 26

제 25항에 있어서, 상기 논리 어드레스 범위는 상기 메모리 시스템의 논리 어드레스 범위인, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 27

제 25항에 있어서, 상기 논리 어드레스 범위는 저장되는 데이터의 파일 오브젝트의 논리 어드레스 범위인, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 28

제 25항에 있어서, 상기 방법이 실행되는 메모리 시스템은, 데이터가 수신되는 메모리 커넥터에 의해 호스트 장치에 분리 가능하게 연결될 수 있는 모듈 내에 있는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 29

제 25항에 있어서, 상기 개별 셀의 적어도 일부에 1 비트 이상의 상기 수신 데이터를 저장하는 방식으로 상기

메모리 셀을 작동시키는 단계를 더 포함하는, 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동 방법.

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

명 세 서

기 술 분 야

[0001] 본원에 설명된 기술들은 "Direct Data File Applicatoins"로써 이후에 집합적으로 참조된, 2005년 2월 16일 출원된 미국 특허 명세서 일련번호 제 11/060,249호, 제 11/060,174호 및 제 11/060,248호, 및 2005년 8월 3일자로 출원된 가출원 일련번호 제 60/705,388호에서 더욱 완전히 설명된 플래시 메모리 시스템에서 구현될 수 있다.

배 경 기 술

[0002] 이 출원은 반도체 플래시 메모리와 같은 재프로그래밍 가능한 비휘발성 메모리 시스템의 작동에 관한 것이고, 특히, 시간에 걸쳐 변하는 여러 크기를 갖는 데이터 파일의 논리 유닛의 처음 저장 및 연속적인 갱신에 관한 것

이다. 모든 특허, 특허 출원, 아티클 및 다른 출간물, 문서 및 본원에 관련된 것이 모든 목적을 위해 전체로서 본원에 참조의 방법으로 통합되었다.

[0003] 상업적인 플래시 메모리 시스템의 초기 세대에서, 직사각형 어레이의 메모리 셀들은 표준 디스크 드라이브 섹터의 데이터량, 즉, 512 바이트가 각각 저장되는 다수의 셀 그룹으로 분할되었다. 또한 16 바이트와 같은 부가적인 양의 데이터는 통상적으로 오류 정정 코드(ECC) 및 가능하다면 데이터가 저장되는 메모리 셀 그룹 및/또는 사용자 데이터에 관한 다른 오버헤드 데이터를 저장하기 위해 각각의 그룹에 포함된다. 이러한 각 그룹의 메모리 셀은 함께 삭제 가능한 최소의 수의 메모리 셀이다. 즉, 삭제 유닛(erase unit)은 포함되는 임의의 오버헤드 데이터 및 하나의 데이터 섹터를 저장하는 다수의 메모리 셀인 것이 효율적이다. 이러한 유형의 메모리 시스템의 예는 미국 특허 제 5,602,987호 및 제 6,426,893호에서 설명된다. 플래시 메모리의 특징은 메모리 셀이 데이터와 함께 재프로그래밍되기 전에 삭제될 필요가 있는 것이다.

[0004] 플래시 메모리 시스템은 가장 공통적으로 개인용 컴퓨터, 카메라 등과 같은 다양한 호스트에 제거 가능하게 연결된 메모리 카드 또는 플래시 드라이브의 형태로 제공되지만, 이러한 호스트 시스템 내에 임베드될 수도 있다. 메모리에 데이터를 기록할 때, 통상적으로 호스트는 고유 논리 어드레스들을 메모리 시스템의 연속된 가상 어드레스 공간 내의 섹터들, 클러스터들, 혹은 다른 데이터의 유닛들에 할당한다. 디스크 운영 체제(DOS)처럼, 호스트는 메모리 시스템의 논리 어드레스 공간 내 어드레스들에 데이터를 기록하고, 이들 어드레스들로부터 데이터를 판독한다. 메모리 시스템 내의 제어기는 호스트로부터 수신된 논리 어드레스를 메모리 어레이 내의 물리 어드레스로 번역하는데, 여기서 데이터가 실제로 저장되며, 이어서 이들 어드레스 번역들을 주시한다. 메모리 시스템의 데이터 저장 용량은 적어도 메모리 시스템에 대해 규정된 전체 논리 어드레스 공간에 걸쳐 어드레싱이 가능한 데이터량만큼 크다.

[0005] 플래시 메모리 시스템들의 더 최근의 세대에서, 삭제 유닛의 크기는 복수 섹터들의 데이터를 저장하기 위해 충분한 메모리 셀의 블록으로 증가되었다. 메모리 시스템이 연결되는 호스트 시스템들이 섹터와 같은 작은 최소 유닛들로 데이터를 프로그래밍하고 판독할 수 있을지라도, 다수의 섹터들이 플래시 메모리의 단일 삭제 유닛에 저장된다. 호스트가 논리 섹터들의 데이터를 갱신하거나 교체할 때 블록 내의 데이터의 일부 섹터가 무용화될 수 있다는 것이 일반적이다. 전체 블록은 블록 내에 저장된 임의의 데이터가 겹쳐서 기록될 수 있기 전에 삭제되어야만 하기 때문에, 새롭거나 갱신된 데이터는 전형적으로 삭제된 다른 블록에 저장되고, 데이터를 위한 남은 용량을 갖는다. 이러한 프로세스는 원 블록(original block)을 메모리 내의 유용한 공간을 취하는 무용 데이터를 갖게 한다. 그러나 이 블록은 그 안에 임의의 유효한 데이터가 남아 있다면 삭제될 수 없다.

[0006] 그러므로, 메모리의 저장 용량을 더 양호하게 사용하기 위해서, 유효 부분 블록 분량들의 데이터를 삭제된 블록에 복제함으로써 이들 데이터를 통합 또는 수집하고 이에 따라 이들 데이터가 복제된 블록(들)이 삭제될 수 있고, 이들 전체 저장 용량이 재사용되는 것이 일반적이다. 블록 내 데이터 섹터들의 논리 어드레스들의 순서로 이들 데이터 섹터를 그룹화하는 것이 데이터를 판독하며 판독된 데이터를 호스트에 전달하는 속도를 증가시키기 때문에, 이를 행하기 위해서 데이터를 복제하는 것이 또한 바람직하다. 만약 이러한 데이터 복제가 너무 빈번하게 발생한다면, 메모리 시스템의 작동 성능이 저하될 수 있다. 이는 특히 메모리 시스템의 작동에 영향을 미치는데, 여기서 메모리의 저장 용량은 전형적인 경우에 시스템의 논리 어드레스 공간을 통해 호스트에 의해 어드레싱 가능한 데이터의 양보다 조금 많다. 이러한 경우, 데이터 통합 또는 수집은 호스트 프로그래밍 명령이 실행될 수 있기 전에 요구되어 질 수 있다. 그러면 프로그래밍 시간이 증가된다.

[0007] 블록들의 크기는 소정의 반도체 에어리어에 저장될 수 있는 데이터의 비트들의 수를 증가시키기 위해서 메모리 시스템들의 연속적인 세대들에서 증가하고 있다. 256 이상의 데이터 섹터를 저장하는 블록이 일반적이 되고 있다. 또한, 2, 4 혹은 그 이상의 블록들의 서로 다른 어레이들 혹은 서브-어레이들은 데이터 프로그래밍 및 판독에서 병렬 계산(parallelism)의 정도를 증가시키기 위해서 종종 메타 블록들에 함께 논리적으로 링크된다. 이러한 대용량 작동 유닛과 함께, 메모리 시스템을 효율적으로 작동하는데 있어서의 난제들이 발생한다. 저장된 데이터의 변화되고 규칙적이지 않은 크기의 그룹화의 결과를 가져오는 데이터의 압축, 암호화 또는 다른 변환은 메모리 시스템의 성능의 적절한 레벨을 유지하는 것을 더 힘들게 한다.

발명의 상세한 설명

[0008] 메모리 시스템에 저장된 데이터가 어드레스의 별개의 유닛, 바람직하게는 동일한 양의 데이터를 갖는 유닛으로 분리되고, 유닛 내의 데이터가 메모리의 블록 또는 메타 블록에 저장되는 데이터의 변환된 유닛으로, 바람직하게는 개별적으로, 변환된다. 데이터 변환은 예를 들어, 데이터를 인코딩하고, 압축하거나 암호화하는 메모리 시스템에 의해 발생할 수 있다. 변환된 데이터 유닛은 항상 변환 전보다 작거나 큰, 다른 크기를 가지며, 여러 변

환된 데이터 유닛의 크기는 전형적으로 동일하지 않다. 저장된 변환된 데이터 유닛은 변환된 데이터 유닛의 경계 및 변환된 데이터 유닛에 의해 스트래들(straddle)될 수 있는 블록 또는 메타 블록의 임의의 물리 경계와 일치하는 어드레스 경계를 갖는 데이터 그룹에 의해 메모리 내에서 식별된다. 물리 블록 경계에 의해 두 개 이상의 데이터 그룹으로 분리되는 변환된 데이터 유닛에 대해서, 이러한 데이터 그룹이 함께 링크될 수 있다. 가변 크기의 데이터 그룹은 사용 가능한 공간에서 연이어 계속 물리적으로 저장되는 것이 바람직하다. 메모리 시스템 내에서 임의의 부가적인 프로세싱 동안, 변환된 데이터 유닛의 경계와 일치하는 데이터 그룹 경계는 별도로 식별되는 것이 바람직하다.

[0009] 이러한 기술은 각각 자신의 어드레스 범위를 갖는 개별적인 파일 오브젝트의 형태로 데이터와 함께 구현될 수 있거나, 메모리 시스템의 공통 논리 어드레스 공간을 공유하는 다수의 파일 오브젝트와 함께 구현될 수 있다. 둘 중 하나의 경우에, 데이터의 논리 어드레스 범위는 별개의 데이터 유닛으로 분리된다. 변환 후, 변환된 데이터의 하나 이상의 고유하게 식별된 데이터 그룹들이 물리 저장 위치들로 맵핑된다. 실시예에서, 하나의 변환된 데이터 유닛의 데이터 그룹(들)은 다른 변환된 데이터 유닛의 데이터 그룹(들)과 임의의 방법으로 통합되거나 결합되지 않고, 이로써 별도로 식별된 개별적인 유닛의 데이터를 유지시킨다. 이는 판독될 때, 유닛-바이-유닛 기반으로 저장된 그룹의 데이터를 재변환시키도록 한다. 메모리 시스템에 의해 변환되고 저장되는 데이터는 호스트 외부의 메모리 시스템 또는 메모리 시스템 내의 프로세서에 의해 실행되는 애플리케이션으로부터 비롯될 수 있다.

[0010] 본 발명의 다른 양상, 이점, 특징 및 세부 사항은 첨부된 도면에 관한 설명에 따른 그의 예시적인 예들의 설명에 포함된다.

실 시 예

[0032] 현재의 플래시 메모리 시스템 및 호스트 장치와의 통상적인 작동이 도1 내지 도8에 관하여 설명된다. 이러한 시스템에서, 본 발명의 여러 양상이 구현될 수 있다. 도1의 호스트 시스템(1)은 데이터를 플래시 메모리(2)에 저장하고 이로부터 데이터를 검색한다. 플래시 메모리는 호스트 내에 임베드될 수 있을지라도, 기계적이며 전기적인 커넥터의 메이팅 부품(mating parts; 3,4)을 통해 호스트에 제거 가능하게 연결된 더 대중적인 형태의 카드인 메모리(2)가 도시된다. 현재, 예를 들어, 컴팩트 플래시(CF), 멀티미디어 카드(MMC), 보안 디지털(SD), 미니 SD, 메모리 스틱, 스마트미디어 및 트랜스플래시 카드와 같이 상업적으로 사용 가능한 여러 다른 플래시 메모리 카드가 있다. 각각의 이러한 카드가 그의 표준화된 사양에 따라 고유한 기계적이고/이거나 전기적인 인터페이스를 가질지라도, 각각에 포함된 플래시 메모리는 매우 유사하다. 이러한 카드들은 본 출원의 양수인, SanDisk Corporation으로부터 모두 입수 가능하다. SanDisk는 또한 그의 Cruzer trademark하에서 일련의 플래시 드라이브를 제공하는데, 이는 호스트의 USB(Universal Serial Bus) 소켓에 플러그함으로써 호스트와 연결하는 USB 플러그를 갖는 소형 패키지들의 휴대 메모리 시스템들이다. 각각의 이러한 메모리 카드들 및 플래시 드라이브들은 그들 내의 플래시 메모리의 호스트 및 제어 작동과 인터페이스하는 제어기를 포함한다.

[0033] 이러한 메모리 카드들 및 플래시 드라이브들을 사용하는 호스트 시스템은 많고 다양하다. 이들은 개인용 컴퓨터(PC), 랩톱 및 다른 휴대 가능한 컴퓨터, 셀룰러 전화, 개인 휴대용 정보 단말기(PDA), 디지털 스틸 카메라, 디지털 무비 카메라 및 휴대용 오디오 플레이어를 포함한다. 호스트는 전형적으로 하나 이상의 유형의 메모리 카드 또는 플래시 드라이브에 대한 내장형 소켓을 포함하지만, 일부는 메모리 카드가 플러그된 아답터를 필요로 한다.

[0034] 도1의 호스트 시스템(1)은 회로 소자 및 소프트웨어의 결합으로 만들어진, 메모리(2)가 연결되는 한, 두 개의 주요 부품을 갖는 것으로 보여질 수 있다. 이들은 메모리(2)와 인터페이스하는 애플리케이션 부분(5) 및 드라이브 부분(6)이다. 개인용 컴퓨터에서, 예컨대, 애플리케이션 부분(5)은 워드 프로세싱, 그래픽스, 제어 또는 다른 대중적인 애플리케이션 소프트웨어를 실행하는 프로세서를 포함할 수 있다. 카메라, 셀룰러 전화 또는 주로 단일 세트의 기능 수행에 전용된 다른 호스트 시스템에서, 애플리케이션 부분(5)은 카메라가 사진을 찍어서 저장하도록 작동시키고, 셀룰러 전화가 전화를 걸고 받도록 작동시키는 등의 소프트웨어를 포함한다.

[0035] 도1의 메모리 시스템(2)은 플래시 메모리(7), 및 데이터를 주고받기 위해 카드가 접속되는 호스트와 인터페이스하고 메모리(7)을 제어하는 회로들(8)을 포함한다. 통상적으로 제어기(8)는 데이터 프로그래밍 및 판독시 호스트(1)에 의해서 사용되는 데이터의 논리 어드레스들과 메모리(7)의 물리 어드레스들간을 변환한다.

[0036] 도2를 참조하여, 도1의 비휘발성 메모리(2)로서 사용될 수 있는 전형적인 플래시 메모리 시스템의 회로 소자가 설명된다. 시스템 제어기는 시스템 버스(13)를 통해 하나 이상의 집적 회로 메모리 칩과 병렬로 연결된 단일 집

적 회로 칩(11) 상에서 보통 구현되는데, 이러한 단일 메모리 칩(15)은 도2에 도시된다. 도시된 개별적인 버스(13)는 데이터를 전달하기 위한 별도 세트의 컨덕터(17), 메모리 어드레스에 대한 세트(19) 및 제어 및 상태 신호에 대한 세트(21)를 포함한다. 대안적으로, 단일 세트의 컨덕터는 이들의 세 개의 기능 사이에서 시간 공유(time shared)될 수 있다. 게다가, 2004년 8월 9일 출원된, 발명의 명칭이 "Ring Bus Structure and It's Use in Flash Memory Systems"인 미국 특허 출원 일련 번호 제 10/915,039호에서 설명된 링 버스와 같은 다른 구성의 시스템 버스가 사용될 수 있다.

[0037] 전형적인 제어기 칩(11)은 인터페이스 회로(25)를 통해 시스템 버스(13)와 인터페이스 하는 자신의 내부 버스(23)를 갖는다. 일반적으로 버스에 연결된 일차적인 기능은 (마이크로프로세서 또는 마이크로컨트롤러와 같은) 프로세서(27), 시스템을 초기화("부트")하는 코드를 포함하는 읽기 전용 메모리(ROM)(29), 및 버스(23)에 또한 연결될 수 있는 호스트 및 메모리 사이의 제어기를 통과하는 데이터에 대한 오류 정정 코드(ECC)를 계산하고 확인하는 호스트 회로(33) 및 메모리 사이에서 이동되는 데이터를 일차적으로 버퍼하는데 사용되는 랜덤 액세스 메모리(RAM)(31)이다. 제어기를 통과한 데이터를 인코딩하고 디코딩하도록 전용된 회로(34)가 또한 포함될 수 있다. 이러한 인코딩은 압축 및 보안 암호화를 포함하지만, 대부분 임의의 유형의 데이터 변환이 이러한 방법으로 수행될 수 있다. 전용 회로(33,34)는 사용될 때 펌웨어 제어 하에서 프로세서(27)에 의해 다른 방법으로 실행될 수 있는 특정한 알고리즘을 실행한다. 제어기 버스(23)는 회로(35)를 통해 호스트 시스템과 인터페이스하는데, 이는 메모리 카드 내에 포함된 도2의 시스템의 경우에, 커넥터(4) 부분인 카드의 외부 컨택트(37)를 통해 행해진다. 클록(39)은 제어기(11)의 다른 구성 요소들 각각에 의해 연결되어 사용된다.

[0038] 메모리 칩(15) 뿐만 아니라 시스템 버스(13)에 연결된 임의의 다른 것이 다수의 서브-어레이들 또는 플레인(plane)들에 구성된 메모리 셀의 어레이를 전형적으로 포함하는데, 두 개의 이러한 플레인(41,43)이 간략성을 위해 도시되지만, 네 개 또는 여덟 개의 이러한 플레인과 같은 부가적인 플레인이 대신 사용될 수 있다. 대안적으로, 칩(15)의 메모리 셀 어레이는 플레인으로 분리될 수 없다. 그러나 분리될 때, 각각의 플레인은 서로 독립적으로 작동할 수 있는 자신의 열 제어 회로(column control circuit)(45,47)를 갖는다. 회로(45,47)는 시스템 버스(13)의 어드레스 부분(19)으로부터 그들의 개별적인 메모리 셀 어레이의 어드레스를 수신하며, 특정한 하나 이상의 개별적인 비트 라인(49,51)을 어드레싱하도록 이들을 디코딩한다. 워드 라인(53)은 어드레스 버스(19) 상에서 수신된 어드레스에 응답하여 행 제어 회로(row control circuit)(55)를 통해 어드레싱된다. 소스 전압 제어 회로(57,59)는 또한 p-웰 전압 제어 회로(p-well voltage control circuit)(61,63)와 같이 개별적인 플레인과 연결된다. 메모리 칩(15)이 단일 어레이의 메모리 셀을 갖는다면, 그리고 두 개 이상의 이러한 칩이 시스템에 존재한다면, 각각의 칩의 어레이는 상술된 다수의 플레인 칩 내에서 플레인 또는 서브-어레이와 유사하게 작동될 수 있다.

[0039] 데이터는 시스템 버스(13)의 데이터 부분(17)과 연결된 개별적인 데이터 입력/출력 회로(65,67)을 통해 플레인(41,43)의 안팎으로 이동된다. 회로(65,67)는 메모리 셀 내로 데이터를 프로그래밍하고, 개별적인 열 제어 회로(45,47)를 통해 플레인에 연결된 라인(69,71)을 통해, 그들의 개별적인 플레인의 메모리 셀로부터 데이터를 판독한다.

[0040] 제어기(11)가 데이터를 프로그래밍하고, 데이터를 판독하며, 삭제하며 여러 하우스키핑 문제에 참여하도록 메모리 칩의 작동을 제어할지라도, 각각의 메모리 칩은 또한 이러한 기능을 수행하기 위해서 제어기(11)로부터 명령들을 실행하는 어떤 제어 회로 소자를 포함한다. 인터페이스 회로(73)는 시스템 버스(13)의 제어 및 상태 부분(21)에 연결된다. 제어기로부터의 명령은 이들 명령을 실행하기 위해서 다른 회로의 특정한 제어를 제공하는 상태 머신(75)에 제공된다. 제어 라인(77-81)은 도2에 도시된 바와 같이 이들 다른 회로와 상태 머신(75)을 연결시킨다. 상태 머신(75)으로부터의 상태 정보는 버스 부분(21)을 통해 제어기(11)로 전송을 위해 인터페이스(73)에 라인(83)을 통해 전달된다.

[0041] NOR와 같은 다른 아키텍처가 대신 사용될 수 있을지라도, 메모리 셀 어레이(41,43)의 NAND 아키텍처는 현재 바람직하다. NAND 플래시 메모리 및 메모리 시스템의 일부로써의 이들의 작동의 예가 미국 특허 제 5,570,315호, 5,774,397호, 6,046,935호, 6,373,746호, 6,456,528호, 6,522,580호, 6,771,536호 및 6,781,877호, 및 미국 특허 출원 공보 제 2003/0147278호에서 참조될 수 있다.

[0042] 예시적인 NAND 어레이는 도3의 회로 다이어그램에 의해 설명되는데, 이는 도2의 메모리 시스템의 메모리 셀 어레이(41)의 일부이다. 많은 수의 글로벌 비트 라인이 제공되는데, 단지 네 개의 이러한 라인(91-94)이 설명의 간략성을 위해 도2에 도시된다. 다수의 직렬 연결된 메모리 셀 스트링(string)(97-104)이 이러한 비트 라인들 중 하나 및 기준 포텐셜 사이에 연결된다. 대표적인 것으로서 메모리 셀 스트링(99)을 사용하면, 다수의 전하

저장 메모리 셀(107-110)이 스트링의 어느 한쪽의 단부에서 선택 트랜지스터(111,112)와 직렬로 연결된다. 선택 트랜지스터의 스트링이 전도성이 있을 때, 스트링은 그의 비트 라인 및 기준 포텐셜 사이에 연결된다. 스트링 내의 하나의 메모리 셀은 동시에 프로그래밍되거나 판독된다.

[0043] 도3의 워드 라인(115-118)은 메모리 셀의 다수의 스트링 각각에서 하나의 메모리 셀의 전하 저장 소자에 걸쳐 개별적으로 확장하며, 게이트(119,220)는 스트링의 각각의 단부에서 선택 트랜지스터의 상태를 제어한다. 공통 워드 및 제어 게이트 라인(115-120)을 공유하는 메모리 셀 스트링은 함께 삭제되는 메모리 셀의 블록(123)을 형성하도록 만들어진다. 셀의 이러한 블록은 한번에 물리적으로 삭제 가능한 최소 수의 셀을 포함한다. 워드 라인(115-118) 중 하나를 따라, 메모리 셀의 하나의 행이 한번에 프로그래밍된다. 전형적으로, NAND 어레이의 행은 접지 또는 다른 공통 포텐셜에 연결된 스트링의 단부에 가장 가까운 워드 라인(118)을 따라 행으로 시작하는 경우에, 규정된 순서로 프로그래밍된다. 워드 라인(117)을 따라 메모리 셀의 행이 블록(123) 도처에서 다음에 프로그래밍 등이 된다. 워드 라인(115)을 따라 행이 마지막에 프로그래밍된다.

[0044] 제2 블록(125)은 제1 블록(123)의 스트링처럼 동일한 글로벌 비트 라인에 연결되지만, 상이한 세트의 워드 및 제어 게이트 라인을 갖는 메모리 셀의 유사한 스트링이다. 워드 및 제어 게이트 라인은 행 제어 회로(55)에 의해 그들의 적합한 작동 전압으로 구동된다. 도2의 플레인(1,2)과 같이, 시스템에 하나 이상의 플레인 또는 서브-어레이가 더 존재한다면, 하나의 메모리 아키텍처가 그들 사이에서 확장하는 공통 워드라인을 사용한다. 대안적으로 공통 워드 라인을 공유하는 두 개 이상의 플레인 또는 서브-어레이가 존재할 수 있다. 다른 메모리 아키텍처에서, 개별적인 플레인 또는 서브-어레이의 워드 라인이 별도로 구동된다.

[0045] 상기 참조된 여러 NAND 특허 및 출원 공보에서 설명된 바와 같이, 메모리 시스템은 각각의 전하 저장 소자 또는 영역에서 두 개 이상의 검출 가능한 레벨의 전하를 저장하도록 작동될 수 있어서, 각각에 1비트 이상의 데이터를 저장할 수 있다. 메모리 셀의 전하 저장 소자는 가장 공통적으로 전도성 있는 플로팅 게이트이지만, 대안적으로 미국 특허 출원 공보 제 2003/0109093호에서 설명되는 바와 같이 비-전도성 유전체 전하 트랩핑 물질일 수 있다.

[0046] 도4는 아래에서 더 설명되는 예와 같이 사용되는 플래시 메모리 셀 어레이(7)(도1)의 구성을 개념적으로 도시한다. 메모리 셀의 네 개의 플레인 또는 서브-어레이(131-134)는 단일 집적 메모리 셀 칩, 두 개의 칩(각각의 칩 상의 두 개의 플레인) 또는 네 개의 별도의 칩 상에 있을 수 있다. 특정한 배열은 후술될 논의에 중요하지 않다. 물론, 1,2,8,16 또는 그 이상의 수의 플레인이 시스템에 존재할 수 있다. 플레인은 개별적인 플레인(131-134)에 위치된 블록(137,138,139,140)과 같은 직사각형으로 도 4에 도시된 메모리 셀의 블록으로 개별적으로 분리된다. 각각의 플레인에는 수십 또는 수백 개의 블록이 존재할 수 있다. 상술된 바와 같이, 메모리 셀의 블록은 삭제 유닛, 함께 물리적으로 삭제될 수 있는 가장 작은 수의 메모리 셀이다. 그러나 증가된 병렬 계산(parallelism)에 대해서, 블록은 더 큰 메타 블록 유닛에서 작동된다. 각각의 플레인으로부터 하나의 블록은 메타 블록을 형성하기 위해서 함께 논리적으로 링크된다. 네 개의 블록(137-140)은 하나의 메타 블록(141)을 형성하도록 도시된다. 메타 블록 내의 모든 셀은 전형적으로 함께 삭제된다. 메타 블록을 형성하는데 사용되는 블록은 블록(145-148)으로 만들어진 제2 메타 블록(143)에 도시된 바와 같이, 그들의 개별적인 플레인 내에서 동일한 관련 위치에 제한될 필요가 없다. 모든 플레인에 걸쳐 메타 블록을 확장시키는 것이 보통 바람직할지라도, 높은 시스템 성능을 위해서, 메모리 시스템은 여러 플레인에서, 임의의 또는 모든 한 개, 두 개 또는 세 개의 블록의 메타 블록을 다이나믹하게 형성하기 위한 능력으로 작동될 수 있다. 이는 메타 블록의 크기가 하나의 프로그래밍 작동에서 저장을 위해 사용 가능한 데이터의 양에 매우 부합하도록 한다.

[0047] 그 후에 개별적인 블록들은 도5에 도시된 바와 같이, 메모리 셀의 페이지로 작동 목적으로 분리된다. 각각의 블록(131-134)의 메모리 셀은 예를 들어, 8 개의 페이지(P0-P7)로 각각 분리된다. 대안적으로, 각각의 블록 내에 메모리 셀의 16,32 또는 그 이상의 페이지가 존재할 수 있다. 페이지는 한번에 프로그래밍되는 최소량의 데이터를 포함하는, 블록 내의 데이터 프로그래밍 및 판독의 유닛이다. 도3의 NAND 아키텍처에서, 페이지는 블록 내의 워드 라인을 따라 메모리 셀로 형성된다. 그러나, 메모리 시스템 작동 병렬 계산을 증가시키기 위해서, 두 개 이상의 블록 내의 이러한 페이지가 메타 페이지에 논리적으로 링크될 수 있다. 네 개의 블록(131-134) 각각으로부터 하나의 물리 페이지로 형성되는 메타 페이지(151)는 도5에 도시된다. 메타 페이지(151)는 예를 들어, 각각의 네 개의 블록에 페이지(P2)를 포함하지만, 메타 페이지의 페이지는 각각의 블록 내의 동일한 관련된 위치를 필수적으로 가질 필요가 없다. 모든 네 개의 플레인에 걸쳐 동시에 최대량의 데이터를 프로그래밍하여 판독하는 것이 바람직할지라도, 높은 시스템 성능을 위해서, 메모리 시스템은 또한 여러 플레인의 별도의 블록에서 임의의 또한 모든 한 개, 두 개 또는 세 개의 페이지의 메타 페이지를 형성하도록 작동될 수 있다. 이는 작동을 프로그래밍하여 판독하는 것이 동시에 편리하게 처리될 수 있는 데이터의 양에 적응할 수 있게 일치하도록 하고,

메타 페이지의 일부가 데이터와 함께 프로그래밍되지 않는 경우를 감소시킨다.

[0048] 도5에 도시된 바와 같이, 다수의 플레인의 물리 페이지로 형성된 메타 페이지는 이런 다수의 플레인의 워드 라인 행을 따라 메모리 셀을 포함한다. 동시에 하나의 워드 라인 행에서 모든 셀을 프로그래밍하는 것보다, 이들은 또한 두 개 이상의 인터리빙된 그룹에서 보다 일반적으로 교대로 프로그래밍되는데, 각각의 그룹은 (단일 블록에서) 데이터의 페이지 또는 (다수의 블록에 걸쳐) 데이터의 메타 페이지를 저장한다. 동시에 교체용 메모리 셀(alternate memory cell)을 프로그래밍함으로써, 데이터 레지스터 및 감지 증폭기를 포함하는 주변 회로 유닛은 각각의 비트 라인에 대해 제공될 필요가 없다기보다, 근접 비트 라인들 사이에서 시간 공유된다. 이는 주변 회로에 대해 필요로 되는 기판 공간의 양을 절약하며, 메모리 셀이 행을 따라 증가된 밀도로 패킹되도록 한다. 반면, 소정의 메모리 시스템으로부터 사용 가능한 병렬 계산을 최대화하기 위해서 행을 따라 모든 셀을 동시에 프로그래밍하는 것이 바람직하다.

[0049] 도3을 참조하면, 행을 따라 모든 다른 메모리 셀로 데이터를 동시에 프로그래밍하는 것은 도시된 단일 행 대신 NAND 스트링의 적어도 한 단부를 따라 두 개의 행 선택 트랜지스터(도시되지 않음)를 제공함으로써 가장 편리하게 성취된다. 그 후에 선택 트랜지스터의 한 행은 하나의 제어 신호에 응답하여 블록 내의 모든 다른 스트링을 그들의 개별적인 비트 라인에 연결시키며, 선택 트랜지스터의 다른 행은 다른 제어 신호에 응답하여 모든 다른 스트링을 개재하는 것을 그들의 개별적인 비트 라인에 연결시킨다. 그러므로 두 개의 페이지의 데이터는 메모리 셀의 각각의 행에 기록된다.

[0050] 각각의 논리 페이지의 데이터 양은 종래대로, 각각의 섹터가 512 바이트의 데이터를 포함하는, 전형적으로 정수의 하나 이상의 데이터의 섹터이다. 도6은 페이지 또는 메타 페이지의 데이터의 두 개의 섹터(153,155)의 논리 데이터 페이지를 도시한다. 각각의 섹터는 저장된 사용자 또는 시스템 데이터의 512 바이트의 부분(157) 및 상기 부분(157)의 데이터 또는 데이터가 저장되는 물리 페이지 또는 저장 블록 중 어느 하나에 관련된 오버헤드 데이터에 대한 다른 수의 바이트의 부분(159)을 통상적으로 포함한다. 오버헤드 데이터의 바이트 수는 전형적으로 16바이트이며, 각각의 섹터(153,155)에 대해 총 528 바이트이다. 오버헤드 부분(159)은 프로그래밍 동안 데이터 부분(157)으로부터 계산된 ECC, 그의 논리 어드레스, 블록이 몇번 삭제되고 재프로그래밍되는지에 대한 경험 카운트, 하나 이상의 제어 플래그, 작동 전압 레벨 및/또는 이와 같은 것을 포함할 수 있고, 이러한 오버헤드 데이터(159)로부터 계산된 ECC를 더할 수 있다. 대안적으로, 오버헤드 데이터(159) 또는 그의 일부가 다른 블록의 다른 페이지에 저장도리 수 있다.

[0051] 메모리의 병렬 계산이 증가함에 따라, 메타 블록의 데이터 저장 능력이 증가하며, 데이터 페이지 및 페타 페이지의 크기가 결과적으로 증가한다. 그 후에 데이터 페이지는 데이터의 두 개 이상의 섹터를 포함할 수 있다. 메타 페이지당 두 개의 데이터 페이지 및 하나의 데이터 페이지의 두 개의 섹터와 함께, 메타 페이지에 네 개의 섹터가 존재한다. 그러므로 각각의 메타 페이지는 2048 바이트의 데이터를 저장한다. 이는 높은 정도의 병렬 계산이며, 행에서 메모리 셀의 수가 증가함에 따라 더 증가될 수 있다. 이러한 이유로, 플래시 메모리의 폭은 페이지 및 메타 페이지의 데이터 양을 증가시키기 위해서 확장되고 있다.

[0052] 상기 식별된 물리적으로 작은 재프로그래밍 가능한 비휘발성 메모리 카드 및 플래시 드라이브는 512 메가바이트(MB), 1 기가바이트(GB), 2GB 및 4GB의 데이터 저장 능력이 상업적으로 가능하며, 더 높아질 수 있다. 도7은 호스트 및 이러한 대용량 메모리 시스템 사이의 가장 공통적인 인터페이스를 도시한다. 호스트는 호스트에 의해 실행되는 펌웨어 프로그램 또는 애플리케이션 소프트웨어에 의해 발생되거나 사용되는 데이터 파일을 처리한다. PC, 랩탑 컴퓨터 등과 같은 범용 컴퓨터 호스트에서 주로 발견되는 워드 프로세싱 데이터 파일이 한 예이고, CAD(computer aided design) 소프트웨어의 도면 파일이 또 다른 예이다. pdf 포맷의 문서가 또한 이러한 파일이다. 스틸 디지털 비디오 카메라는 메모리 카드에 저장되는 각각의 픽처에 대한 데이터 파일을 발생시킨다. 셀룰러 전화기는 전화 디렉토리와 같은 내부 메모리 카드 상의 파일로부터 데이터를 사용한다. PDA는 어드레스 파일, 달력 파일 등과 같은 여러 다른 파일을 저장하고 사용한다. 임의의 이러한 애플리케이션에서, 메모리 카드는 또한 호스트를 작동시키는 소프트웨어를 포함할 수 있다.

[0053] 호스트 및 메모리 시스템 사이의 공통 논리 인터페이스는 도7에 도시된다. 연속적인 논리 어드레스 공간(161)은 메모리시스템에 저장될 수 있는 모든 데이터에 대한 어드레스를 제공하기에 충분히 크다. 호스트 어드레스 공간은 데이터 클러스터의 인크리먼트로 전형적으로 분리된다. 각각의 클러스터는 전형적인 4 섹터 및 64 섹터 사이의 어딘가에 다수의 데이터 섹터를 포함하도록 소정의 호스트 시스템에 디자인될 수 있다. 표준 섹터는 512 바이트의 데이터를 포함한다.

[0054] 세 개의 파일(1,2,3)은 생성되는 것으로 도7의 예에서 도시된다. 호스트 시스템 상에서 작동하는 애플리케이션

프로그램은 순서화된 세트의 데이터와 같은 각각의 파일을 생성하며 이들을 고유 이름 또는 다른 기준에 따라 식별한다. 다른 파일에 이미 할당되지 않은 사용 가능한 충분한 논리 어드레스 공간이 파일(1)에 호스트에 의해 할당된다. 파일(1)은 연속적인 범위의 사용 가능한 논리 어드레스가 할당되는 것으로 도시된다. 어드레스 범위는 또한 호스트 작동 소프트웨어에 대한 특정한 범위와 같은 특별한 목적으로 일반적으로 할당되는데, 이는 그 후에 호스트가 논리 어드레스를 데이터에 할당하는 때에 이러한 어드레스가 사용되지 않을지라도 데이터를 저장하는 것을 피한다.

[0055] 도2가 호스트에 의해 나중에 생성될 때, 호스트는 도7에 도시된 바와 같이, 논리 어드레스 공간(161) 내에서 두 개의 상이한 범위의 연속적인 어드레스를 유사하게 할당한다. 파일이 연속적인 논리 어드레스를 할당받을 필요가 없지만, 다른 파일에 이미 할당된 어드레스 범위들 사이에서 어드레스 조각일 수 있다. 그 후에 이러한 예는 호스트에 의해 생성된 또 다른 파일(3)이 파일(1,2) 및 다른 데이터에 이미 할당되지 않은 호스트 어드레스 공간의 다른 일부를 할당받았다는 것을 나타낸다.

[0056] 호스트는 파일 할당 테이블(FAT)을 유지함으로써 메모리 논리 어드레스 공간을 주시하는데, 여기서 호스트가 여러 호스트 파일에 할당하는 논리 어드레스가 유지된다. FAT 테이블은 전형적으로 비휘발성 메모리뿐만 아니라 호스트 메모리에 저장되고, 새로운 파일이 저장되며, 다른 파일이 삭제되고, 파일이 수정되는 등에 따라 호스트에 의해 주기적으로 갱신된다. 호스트 파일이 삭제될 때, 예컨대, 호스트는 논리 어드레스가 이제 다른 데이터 파일과 함께 사용가능하다는 것을 나타내기 위해서 FAT 테이블을 갱신함으로써 삭제된 파일에 이미 할당된 논리 어드레스를 할당 해제한다.

[0057] 호스트는 메모리 시스템 제어기가 파일을 저장하는 것을 선택하는 곳인 물리 위치에 대해 고려되지 않는다. 전형적인 호스트만이 그의 논리 어드레스 공간 및 그의 여러 파일에 할당된 논리 어드레스를 인식한다. 반면, 전형적인 호스트/카드 인터페이스를 통해 메모리 시스템은 단지 데이터가 기록되는 논리 어드레스 공간의 일부를 인식하지만, 특정한 호스트 파일에 할당된 논리 어드레스를 인식하지 않거나 호스트 파일의 수조차 인식하지 않는다. 메모리 시스템 제어기는 데이터의 저장 또는 검색을 위해 호스트에 의해 제공되는 논리 어드레스를 호스트 데이터가 저장되는 플래시 메모리 셀 어레이 내의 고유한 물리 어드레스로 변환시킨다. 블록(163)은 이러한 논리-대-물리 어드레스 변환의 워킹 테이블(working table)을 나타내는데, 이는 메모리 시스템 제어기에 의해 유지된다.

[0058] 메모리 시스템 제어기는 높은 레벨로 시스템의 성능을 유지하는 방법으로 메모리 어레이(165)의 블록 및 메타 블록 내에 데이터 파일을 저장하도록 프로그래밍된다. 네 개의 플레인 또는 서브-어레이들은 이러한 설명에서 사용된다. 데이터는 시스템이 각각의 플레인으로부터 블록으로 형성된 전체 메타 블록에 걸쳐 허용하는 최대 병렬 계산 정도와 함께 프로그래밍되고 판독되는 것이 바람직하다. 적어도 하나의 메타 블록(167)은 통상적으로 메모리 제어기에 의해 사용하는 데이터 및 작동 펌웨어를 저장하는 리저브드 블록(reserved block)으로써 할당된다. 다른 메타블록(169) 또는 다수의 메타블록들은 호스트 작동 소프트웨어, 호스트 FAT 테이블 등의 저장을 위해 할당될 수 있다. 대부분의 물리 저장 공간은 데이터 파일의 저장을 위해 남아 있다. 그러나 메모리 제어기는 수신된 데이터가 그의 여러 파일 오브젝트들 사이에서 호스트에 의해 할당받는 방법을 모른다. 모든 메모리 제어기는 전형적으로 특정한 논리 어드레스에 호스트에 의해 기록된 데이터가 제어기의 논리-대-물리 어드레스 테이블(163)에 의해 유지됨에 따라 상응하는 물리 어드레스에 저장되는 것을 호스트와의 상호작용으로부터 인식한다.

[0059] 전형적인 메모리 시스템에서, 몇몇 여분의 저장 용량의 블록이 어드레스 공간(161) 내에 데이터의 양을 저장하는데 필요 이상으로 제공된다. 하나 이상의 이러한 여분의 블록들은 메모리의 수명 동안 결함이 있을 수 있는 다른 블록에 대한 대용으로 리던던트 블록(redundant block)으로서 제공될 수 있다. 개별적인 메타 블록 내에 포함된 논리 그룹의 블록들은 통상적으로 메타 블록에 원래 할당된 결함 있는 블록에 대한 리던던트 블록의 대용을 포함한 여러 이유로 변경된다. 메타 블록(171)과 같은 하나 이상의 부가적인 블록은 삭제된 블록 풀(erased block pool)에서 전형적으로 유지된다. 호스트가 메모리 시스템에 데이터를 기록할 때, 제어기는 호스트에 의해 할당된 논리 어드레스를 삭제된 블록 풀의 메타 블록 내 물리 어드레스로 전환한다. 그 후에 논리 어드레스 공간(161) 내에 데이터를 저장하는데 사용되지 않는 다른 메타 블록은 후속하는 데이터 기록 작동 동안 사용을 위해 삭제된 풀 블록으로써 삭제되고 지정된다.

[0060] 특정한 호스트 논리 어드레스에 저장된 데이터는 원래 저장된 데이터가 무용화됨으로써 새로운 데이터에 의해 주기적으로 겹쳐서 기록된다. 응답으로 메모리 시스템 제어기는 새로운 데이터를 삭제된 블록에 기록하고, 그 후에 논리 어드레스에 대한 논리-대-물리 어드레스 테이블을 변화시켜 논리 어드레스에서 데이터가 저장되는 새

로운 물리 블록을 식별하도록 한다. 논리 어드레스에 원래 데이터를 포함하는 블록이 삭제되어 새로운 데이터의 저장을 위해 사용 가능하게 만들어진다. 이러한 삭제는 기록 시작시 삭제 블록 풀로부터 선-삭제된 블록에 충분하지 않은 저장 용량이 존재한다면, 현재 데이터 기록 작동이 완료될 수 있기 전에 종종 행해져야만 한다. 이는 시스템 데이터 프로그래밍 속도에 불리한 영향을 미칠 수 있다. 메모리 제어기는 호스트가 그들의 동일한 논리 어드레스에 새로운 데이터를 기록할 때만 소정의 논리 어드레스에서 데이터가 호스트에 의해 무용화된다는 것을 전형적으로 인식한다. 그러므로 메모리의 많은 블록은 임시로 이러한 유효하지 않은 데이터를 저장할 수 있다.

[0061] 블록 및 메타 블록의 크기는 집적 회로 메모리 칩의 에어리어를 효율적으로 사용하기 위해 증가한다. 이는 메타 블록의 저장 용량보다 작으며, 많은 경우에는 심지어 한개 블록의 저장 용량보다 작은 데이터의 양을 저장하는 큰 비율의 개별적인 데이터 기록의 결과를 가져온다. 메모리 시스템 제어기가 일반적으로 새로운 데이터를 삭제된 풀 메타 블록으로 향하게 하기 때문에, 이는 메타 블록의 일부가 채워지지 않는 결과를 가져올 수 있다. 새로운 데이터가 다른 메타블록에 저장된 일부 데이터로 갱신된다면, 새로운 데이터 메타페이지의 것과 연속적인 논리 어드레스를 갖는 다른 메타 블록으로부터 데이터의 나머지 유효한 메타 페이지가 또한 새로운 메타블록으로 논리 어드레스의 순서로 복제되는 것이 바람직하다. 낡은 메타 블록은 다른 유효 데이터 메타페이지를 계속 유지할 수 있다. 이는 시간에 걸쳐, 개별적인 메타 블록의 임의의 메타페이지의 데이터가 여러 메타 블록에 기록되는 동일한 논리 어드레스로 새로운 데이터에 의해 교체되며, 무용화되며 유효하지 않게 되는 결과를 가져온다.

[0062] 전체 논리 어드레스 공간(161)에 걸쳐 데이터를 저장하도록 충분한 물리 메모리 공간을 유지하기 위해서, 이러한 데이터는 주기적으로 콤팩트화되거나 통합된다(가비지 콜렉션). 이는 더 효율적으로 연속적인 논리 어드레스에서 데이터를 판독하게 하기 때문에, 실질적인 만큼 그들의 논리 어드레스와 동일한 순서로 메타블록 내에 데이터 섹터를 유지하는 것이 또한 바람직하다. 그래서 데이터 콤팩션 및 가비지 콜렉션은 전형적으로 이러한 부가적인 목적으로 수행된다. 부분적인 블록 데이터 갱신을 수신할 때 메모리를 관리하는 일부 양상 및 메타 블록의 사용이 미국 특허 제 6,763,424호에서 설명된다.

[0063] 데이터 콤팩션은 전형적으로 메타 블록으로부터 모든 유효한 데이터 메타 페이지를 판독하는 단계, 이들을 새로운 블록에 기록하는 단계, 프로세스에서 유효하지 않은 데이터를 갖는 메타페이지를 무시하는 단계를 포함한다. 유효한 데이터를 갖는 메타 페이지는 거기에 저장된 데이터의 논리 어드레스 순서에 부합하는 물리 어드레스 순서대로 배열되는 것이 또한 바람직하다. 유효하지 않은 데이터를 포함하는 메타페이지가 새로운 메타 블록에 복제되지 않기 때문에 새로운 메타 블록에 점유된 메타 페이지의 수는 낡은 메타 블록에 점유된 것보다 적을 것이다. 그래서 낡은 블록은 삭제되고 새로운 데이터를 저장할 수 있게 만들어진다. 그 후에 통합에 의해 얻어진 부가적인 메타 페이지의 용량은 다른 데이터를 저장하는데 사용될 수 있다.

[0064] 가비지 콜렉션 동안, 연속적이거나 거의 연속적인 논리 어드레스를 갖는 유효 데이터의 메타 페이지가 두 개 이상의 메타 블록으로부터 모아지고, 다른 메타 블록에 재기록되는데, 통상적으로 하나는 삭제된 블록 풀에 있다. 모든 유효한 데이터 메타 페이지가 원래의 두 개 이상의 메타블록으로부터 복제될 때, 이들은 미래에 사용하기 위해 삭제될 수 있다.

[0065] 데이터 통합 및 가비지 콜렉션은 시간이 걸리고, 특히 데이터 통합 또는 가비지 콜렉션이 호스트로부터의 명령이 실행될 수 있기 전에 행해질 필요가 있다면, 메모리 시스템의 수행에 영향을 미칠 수 있다. 이런 작동은 메모리 시스템 제어기에 의해 일반적으로 스케줄링되어, 가능한 많이 백그라운드에서 행해지지만, 이러한 작동을 수행하기 위한 필요는 제어기가 이러한 작동이 완료될 때까지 바쁜 상태 신호를 호스트에게 제공해야만 하도록 할 수 있다. 호스트 명령의 실행이 지연될 수 있는 곳의 예는 호스트가 메모리에 기록하기를 원하는 모든 데이터를 저장하기 위해 삭제된 블록 풀에서 충분히 사전 삭제된 메타블록이 없으며, 이 후에 삭제될 수 있는 유효 데이터의 하나 이상의 메타블록을 클리어하기 위해 데이터 통합 또는 가비지 콜렉션은 우선 필요로 되는 곳이다. 그러므로 이러한 왜곡을 최소화하기 위해서 메모리의 제어를 관리하는 것에 주의를 기울인다. 이러한 많은 기술이 다음의 미국 특허 출원에서 설명된다: 2003년 12월 30일자로 출원된 발명의 명칭이 "Management of Non-Volatile Memory Systems Having Large Erase Blocks"인 일련 번호 제 10/749,831호; 2003년 12월 30일자로 출원된 발명의 명칭이 "Non-Volatile Memory and Method with Block Management System"인 일련 번호 제 10/750,155호; 2004년 8월 13일자로 출원된 발명의 명칭이 "Non-Volatile Memory and Method with Memory Planes Alignment"인 일련 번호 제 10/917,888호; 2004년 8월 13일자로 출원된 일련 번호 제 10/917,867호; 2004년 8월 13일자로 출원된 발명의 명칭이 "Non-Volatile Memory and Method with Phased Program Failure Handling"인 일련 번호 제 10/917,889호; 및 2004년 8월 13일자로 출원된 발명의 명칭이 "Non-Volatile Memory

and Method with Control Data Management"인 일련 번호 제 10/917,725호.

[0066] 매우 큰 삭제 블록을 갖는 메모리 어레이의 작동을 효율적으로 제어하기 위한 하나의 챌린지(challenge)는 메모리 블록의 경계 및 용량을 갖는 소정의 기록 작동동안 저장된 데이터 섹터의 수에 부합하게 하고 정렬시키는 것이다. 하나의 접근법은 전체 메타 블록을 채우는 양보다 적은 데이터 양을 저장하는 것을 필요로 함에 따라, 최대 수의 블록보다 적은 블록을 갖는 호스트로부터 새로운 데이터를 저장하는데 사용되는 메타 블록을 구성하는 것이다. 적응형 메타 블록의 사용은 2003년 12월 30일자로 출원된 발명의 명칭이 "Adaptive Metablocks"인 미국 특허 출원 일련 번호 제 10/749,189호에서 설명된다. 데이터의 블록들 사이의 경계 및 메타 블록들 사이의 물리 경계의 일치(fitting)는 2004년 5월 7일자로 출원된 특허 출원 일련 번호 제 10/841,118호 및 2004년 12월 16일에 출원된 발명의 명칭이 "Data Run Programming"인 일련 번호 제 11/016,271호에서 설명된다.

[0067] 메모리 제어기는 또한 비휘발성 메모리에서 호스트에 의해 저장되는, FAT 테이블로부터 데이터를 사용할 수 있어서 메모리 시스템을 더 효율적으로 작동시킨다. 이러한 하나의 사용은 데이터가 그들의 논리 어드레스를 할당해제함으로써 무용화되었다는 것이 호스트에 의해 식별될 때를 인식하는 것이다. 이러한 것을 아는 것은 논리 어드레스에 새로운 데이터를 기록하는 호스트에 의해 이것을 일반적으로 인식하기 전에 메모리 제어기가 이러한 유효하지 않은 데이터를 포함하는 블록의 삭제를 스케줄링하도록 한다.이는 2004년 7월 21일자로 출원된, 발명의 명칭이 "Method and Apparatus for Maintaining Data on Non-Volatile Memory System"인 미국 특허 출원 일련 번호 제 10/897,049호에서 설명된다. 다른 기술은 소정의 기록 작동이 파일 사이의 경계가 놓인 단일 파일인지 또는 다수의 파일인지 여부를 추론하기 위해서 메모리에 새로운 데이터를 기록하는 호스트 패턴을 모니터링하는 것을 포함한다. 2004년 12월 23일자로 출원된 발명의 명칭이 "FAT Analysis for Optimized Sequential Cluster Management"인 미국 특허 출원 일련 번호 제 11/022,369호는 이러한 유형의 기술의 사용을 설명한다.

[0068] 메모리 시스템을 효율적으로 작동시키기 위해서, 얼마나 많은 논리 어드레스가 그의 개별적인 파일의 데이터로 호스트에 의해 할당되는지 제어기가 아는 것이 바람직하다. 그 후에 데이터 파일은 파일 경계가 공지되지 않을 때, 더 많은 수의 메타 블록들 사이에서 스캐터링(scatter)되는 것보다, 단일 메타 블록 또는 메타 블록의 그룹 내에서 제어기에 의해 저장될 수 있다. 이는 가비지 콜렉션 작동 및 데이터 통합의 수 및 복잡성이 감소되는 결과를 가져온다. 메모리 시스템의 성능은 결과적으로 개선된다. 그러나, 상술된 바와 같이, 호스트/메모리 인터페이스가 논리 어드레스 공간(161)을 포함할 때(도7) 호스트 데이터 파일 구조에 대한 많은 것을 메모리 제어기가 아는 것이 어렵다.

[0069] 도8을 참조하면, 전형적인 논리 어드레스 호스트/메모리 인터페이스가 도7에 이미 도시된 것과는 다르게 도시된다. 호스트 발생 데이터 파일(host generated data files)은 호스트에 의해 논리 어드레스를 할당받는다. 그래서 메모리 시스템은 이들 논리 어드레스를 인지하며 이들을 데이터가 실제로 저장되는 메모리 셀 블록의 물리 어드레스로 맵핑한다.

[0070] 직접적인 데이터 파일 저장

[0071] 많은 양의 데이터의 저장을 위해 호스트 및 메모리 시스템 사이의 개선된 인터페이스가 논리 어드레스 공간의 사용을 제거한다. 이는 상술된 직접적인 데이터 파일 애플리케이션에 관한 것이다. 호스트는 대신 파일 내의 (바이트와 같은) 데이터 유닛의 오프셋 어드레스 및 고유한 파일ID(또는 다른 고유한 기준)에 의해 각각의 파일을 논리적으로 어드레싱한다. 이러한 파일 어드레스는 메모리 시스템 제어기에 직접 제공되는데, 이는 그 후에 각각의 호스트 파일의 데이터가 물리적으로 저장되는 자신의 테이블을 유지한다. 이러한 새로운 인터페이스는 도2 내지 도6을 참조하여 상술된 바와 동일한 메모리 시스템과 함께 구현될 수 있다. 상술된 것과 함께 일차적인 차이점은 메모리 시스템이 호스트 시스템과 통신하고 파일 데이터를 저장하는 방법이다.

[0072] 도7의 논리 어드레스 인터페이스와 비교되어야 하는 파일 기반 인터페이스는 도9에 도시된다. 도9의 파일 내에서 데이터 오프셋 및 각각의 파일(1,2,3)의 식별이 메모리 제어기에 직접 통과된다. 그래서 이러한 논리 어드레스 정보는 메모리 제어기 기능(173)에 의해 메모리(165)의 메타 페이지 및 메타 블록의 물리 어드레스로 번역된다.

[0073] 도8의 논리 어드레스 인터페이스와 비교되어야 하는 파일 기반 인터페이스가 도10에 또한 도시된다. 도8의 호스트 유지 FAT 테이블 및 논리 어드레스 공간이 도10에서 보여지지 않는다. 오히려, 호스트에 의해 발생된 데이터 파일이 파일 내의 데이터의 오프셋 및 파일 넘버에 의해 메모리 시스템에 식별된다. 그 후에 메모리 시스템은 메모리 셀 어레이의 물리 블록으로 파일을 직접 맵핑한다.

[0074] 새로운 데이터 파일이 직접적인 데이터 파일 저장 기술로 메모리 내로 프로그래밍될 때, 데이터는 블록의 제1

물리 위치에서 시작하며 순차적인 순서대로 블록의 위치를 통해 계속되는 메모리 셀의 삭제된 블록 내로 기록된다. 데이터는 파일 내의 데이터의 오프셋의 순서에 관계없이, 호스트로부터 수신된 순서로 프로그래밍된다. 프로그래밍은 파일의 모든 데이터가 메모리 내로 기록될 때까지 계속된다. 파일의 데이터 양이 단일 메모리 블록의 용량을 초과한다면, 그리고 제1 블록이 채워질 때, 프로그래밍은 제2 삭제된 블록에서 계속된다. 제2 메모리 블록은 파일의 모든 데이터가 저장되거나 제2 블록이 채워질 때까지 제1 위치로부터의 순서로 첫 번째와 동일한 방법으로 프로그래밍된다. 제3 또는 부가적인 블록이 파일의 임의의 나머지 데이터와 함께 프로그래밍될 수 있다. 단일 파일의 데이터를 저장하는 다수의 블록 또는 메타 블록은 물리적으로 또는 논리적으로 계속될 필요가 없다. 설명의 용이성을 위해서, 다르게 규정되지 않는 한, 본원에서 사용되는 "블록"이라는 용어는 메타 블록들이 특정한 시스템에서 사용되고 있는지 여부에 따라 삭제하는 블록 유닛(block unit of erase) 또는 다수의 블록 "메타 블록"이라 칭해지는 것으로 의도된다.

[0075] 도11 내지 도13은 메모리 시스템 내에서 행해지는 인코딩, 압축, 암호화 또는 다른 데이터 변환 없이 플래시 메모리의 직접적인 데이터 파일 작동의 예를 제공한다. 도11A를 참조하면, 데이터 파일을 메모리 시스템에 기록하는 것이 도시된다. 이러한 예에서, 데이터 파일(181)은 메모리 시스템의 하나의 블록 또는 메타 블록(183)의 저장 용량보다 크며, 이는 실선 수직 라인 사이에서 확장하는 것으로 도시된다. 그러므로 데이터 파일(181)의 일부(184)는 또한 제2 블록(185)에 기록된다. 이러한 메모리 셀 블록은 물리적으로 연속적인 것으로 도시되지만, 꼭 그럴 필요는 없다. 파일(181)로부터의 데이터는 파일의 모든 데이터가 메모리에 기록될 때까지 호스트로부터 스트림이 수신됨에 따라 기록된다. 도11A의 예에서, 데이터(181)는 기록 명령이 호스트에 의해 발행된 후에 호스트로부터 수신된 파일에 대한 초기 데이터이다.

[0076] 메모리 시스템이 저장된 데이터를 관리하고 주시하는 바람직한 방법은 가변 크기의 데이터 그룹의 사용이다. 즉, 파일의 데이터는 완성 파일을 형성하기 위한 규정된 순서로 함께 체인화될 수 있는 다수의 데이터 그룹으로서 저장된다. 그러나, 바람직하게는 파일 내의 데이터 그룹의 순서가 파일 인덱스 테이블(FIT)의 사용을 통해 메모리 시스템 제어기에 의해 유지된다. 호스트로부터의 데이터 스트림이 기록됨에 따라, 새로운 데이터 그룹은 파일 데이터의 논리 오프셋 어드레스에서 또는 데이터가 저장되는 물리 공간 중 어느 하나에서 불연속적일 때마다 시작된다. 이러한 물리적인 비연속성의 예는 파일의 데이터가 하나의 블록을 채우고, 다른 블록에 기록되기 시작할 때이다. 이는 도11A에 도시되는데, 여기서 제1 데이터 그룹은 제1 블록(183)을 채우며, 파일의 나머지 부분(184)은 제2 데이터 그룹처럼 제2 블록(185)에 저장된다. 제1 데이터 그룹은 (F0,D0)으로 나타내질 수 있는데, F0은 데이터 파일의 시작의 논리 오프셋이고, D0은 파일이 시작하는 메모리 내의 물리 위치이다. 제2 데이터 그룹은 (F1,D1)으로 나타내지는데, F1은 제2 블록(185)의 시작에서 저장되는 논리 파일 오프셋의 데이터이고, D1은 데이터가 저장되는 물리 위치이다.

[0077] 호스트 메모리 인터페이스를 통해 전달되는 데이터의 양은 다수의 데이터 바이트, 다수의 데이터 섹터 또는 일부 다른 그래뉼래러티(granularity)로 표현될 수 있다. 호스트는 주로 바이트 그래뉼래러티로 그의 파일 데이터를 규정하지만, 현재 논리 어드레스 인터페이스를 통해 대용량 메모리 시스템과 통신할 때, 512 바이트의 섹터로 바이트를 각각 그룹화하거나 다수의 섹터의 클러스터로 각각 그룹화한다. 이는 메모리 시스템의 작동을 간략화시키기 위하여 통상적으로 행해진다. 본원에서 설명되는 파일 기반 호스트 메모리 인터페이스가 데이터의 일부 다른 유닛을 사용할 수 있을지라도, 원래의 호스트 파일 바이트 그래뉼래러티가 일반적으로 바람직하다. 즉, 데이터 오프셋, 길이 등은 섹터(들), 클러스터(들) 등으로 나타내지는 것보다, 바이트(들), 데이터의 가장 작은 합리적인 유닛의 형태로 나타내지는 것이 바람직하다. 이는 본원에서 설명되는 기술과 함께 플래시 메모리 저장의 용량의 좀 더 효율적인 사용을 허용한다.

[0078] 공통적인 기존 논리 어드레스 인터페이스에서, 호스트는 또한 기록된 데이터의 길이를 규정한다. 이는 또한 본원에서 설명되는 파일 기반 인터페이스와 함께 행해질 수 있지만, 기록 명령의 수행을 위해 필수적이지 않기 때문에, 호스트가 기록된 데이터의 길이를 제공하지 않는 것이 바람직하다.

[0079] 그래서 도11A에서 도시된 방법으로 메모리에 기록된 새로운 파일은 상기 순서로, 데이터 그룹에 대한 인덱스 엔트리인 (F0,D0), (F1,D1)의 시퀀스로써 FIT에 나타내진다. 즉, 호스트 시스템이 특정한 파일에 액세스하기를 원할 때마다, 호스트는 그의 파일ID 또는 다른 아이덴티피케이션(identification)을 메모리 시스템으로 송신하고, 이는 그 파일을 만드는 데이터 그룹을 식별하기 위해서 그의 FIT에 액세스한다. 개별적인 데이터 그룹의 길이는 또한 메모리 시스템의 작동을 편리하게 하기 위해서 그들의 개별적인 엔트리에 포함될 수 있다. 사용될 때, 메모리 제어기는 데이터 그룹의 길이를 계산하고 저장한다.

[0080] 호스트가 열린 상태에서 도11A의 파일을 유지하는 한, 물리 기록 포인터(P)는 또한 그 파일에 대해 호스트로부

터 수신된 임의의 부가적인 데이터를 기록하기 위한 위치를 한정하기 위해 유지되는 것이 바람직하다. 파일에 대한 임의의 새로운 데이터는 파일 내의 새로운 데이터의 논리 위치에 관계없이 물리 메모리의 파일의 단부에 기록된다. 메모리 시스템은 4 또는 5개의 파일과 같은 다수의 파일이 동시에 열려 있게 하며, 그들 각각에 대한 기록 포인터(P)를 유지한다. 상이한 파일에 대한 기록 포인터는 상이한 메모리 블록의 위치를 가리킨다. 다수의 열린 파일에 대한 메모리 시스템의 제한이 이미 존재할 때 호스트 시스템이 새로운 파일을 열기를 원한다면, 열린 파일들 중 하나는 먼저 닫히고, 새로운 파일이 그 후 열린다. 파일이 닫힌 후에, 그 파일에 대한 기록 포인터(P)를 유지할 필요가 더 이상 없다.

[0081] 도11B는 또한 기록 명령의 사용에 의해, 이전에 기록되었지만 도11A의 여전히 오픈인 파일의 단부에 호스트에 의해 데이터를 부가하는 것을 도시한다. 데이터(187)가 파일의 단부에 호스트 시스템에 의해 부가되는 것을 도시하는데, 이는 그 파일에 대한 데이터 단부에서 제2 블록(185)에 또한 기록된다. 첨부된 데이터는 데이터 그룹의 일부(F1,D1)가 되므로, 기존 데이터 그룹(184) 및 첨부된 데이터(189) 사이에 논리 어드레스 불연속성도, 물리 어드레스 불연속성도 존재하지 않기 때문에, 이제 부가적인 데이터를 포함한다. 그러므로 완전한 파일은 FIT에서 인덱스 엔트리인 (F0,D0),(F1,D1)의 시퀀스로써 여전히 나타내진다. 포인터(P)의 어드레스는 또한 저장된 첨부 데이터 단부의 어드레스로 바뀐다.

[0082] 데이터(191)의 블록을 도11A에서 이전에 기록된 파일에 삽입하는 예는 도11C에 도시된다. 호스트가 데이터(191)를 파일에 삽입할지라도, 메모리 시스템은 이전에 기록된 파일 데이터의 단부의 위치(193)에 삽입된 데이터를 첨부한다. 이는 호스트가 파일을 닫은 후에 백그라운드에서 나중에 행해질 수 있을지라도, 데이터가 열린 파일에 삽입될 때 그들의 논리 순서로 파일의 데이터를 재기록하는 것이 필수적이지 않다. 삽입된 데이터가 제2 메모리 블록(185) 내에 전체적으로 저장되기 때문에, 단일의 새로운 그룹(F1,D3)을 형성한다. 그러나, 이러한 삽입을 행하는 것은 도11A의 이전 데이터 그룹(F0,D0)이 삽입 전 하나(F0,D0) 및 삽입 후 하나(F2,D1)의 두 개의 그룹으로 분리되는 결과를 가져온다. 이는 삽입하는 시작 부(F1)에서 그리고 삽입하는 단부(F2)에서 발생하는 바와 같이 데이터의 논리 불연속성이 존재할 때마다 새로운 데이터 그룹이 형성될 필요가 있기 때문이다. 그룹(F3,D2)은 제2 블록(185)의 시작인 물리 어드레스(D2)의 결과이다. 그룹(F1,D3) 및 (F3,D2)는 동일한 메모리 블록에 저장될지라도, 별도로 유지되는데, 이는 동일한 메모리 블록에 저장된 데이터의 오프셋에 불연속성이 있기 때문이다. 삽입이 있는 원래 파일은 데이터 그룹 인덱스 엔트리인 (F0,D0), (F1,D3), (F2,D1), (F3,D2)에 의해 순서대로 메모리 시스템(FIT)에 나타내진다. 새롭거나 기존 파일에 대한 새로운 데이터가 메모리의 임의의 데이터를 무용화하지 않고 기록될 수 있다는 것이 도11A, 도11B 및 도11C의 예로부터 주목되어야 한다. 즉, 기록 및 삽입 명령의 실행은 임의의 다른 데이터가 무효화되거나 무용화되도록 유발하지 않는다.

[0083] 도11D는 갱신 명령을 사용하여, 도11A에 도시된 방법으로 원래 기록된 데이터의 임의의 부분이 갱신되는 다른 예를 도시한다. 데이터 파일의 일부(195)는 갱신되는 것으로 도시된다. 메모리 시스템 내의 전체 파일이 갱신되어 재기록되는 것보다, 파일의 갱신된 일부(197)가 이전에 기록된 데이터에 부가된다. 이전에 기록된 데이터의 일부(199)는 이제 무용화된다. 무용 데이터에 의해 자리잡힌 공간을 비우기 위하여 갱신된 파일을 통합하는 것이 통상적으로 바람직할지라도, 이는 통상적으로 호스트가 열린 파일을 유지하는 동안 행해지는 것이 아니라, 오히려 파일이 닫힌 후에 백그라운드에서 행해질 수 있다. 갱신 이후에, 파일은 순서대로, 데이터 그룹 인덱스 엔트리인 (F0,D0), (F1,D3), (F2,D1), (F3,D2)에 의해 메모리 시스템(FIT)에 나타내진다. 도11A의 단일 데이터 그룹(F0,D0)은 다시 도11D에서 갱신된 일부 전에 하나, 갱신된 일부, 갱신된 일부 이후에 하나의 조각들로 분리된다.

[0084] 변화 가능한 길이의 데이터 그룹의 사용을 더 설명하기 위해서, 동일한 파일을 포함하는 여러 기록 작동의 시퀀스가 순서대로 도12A 내지 도12E에서 도시된다. 원래 파일 데이터(W1)는 기록 명령을 사용하여 도12A에 도시된 바와 같이 메모리 시스템의 두 개의 블록에 우선 기록된다. 그 후에, 파일은 두 개의 데이터 그룹으로 규정되는데, 제1 그룹은 물리 메모리 블록의 시작부에서 시작하고, 제2 그룹은 물리 메모리 블록 경계 이후에 필요로 된다. 그 후에, 도12A의 파일은 데이터 그룹에 대한 인덱스 엔트리의 다음 시퀀스: (F0,D0),(F1,D1)에 의해 식별된다.

[0085] 도12B에서, 도12A에 기록된 파일 데이터가 갱신 명령을 사용하여 갱신된다. 갱신된 파일 데이터(U1)는 이전 그룹(F1,D1)의 바로 다음에 기록되며, 갱신된 데이터의 이전 버전은 무용화된다. 도12A의 이전 그룹(F0,D0)은 도12B의 개정된 그룹(F0,D0)으로 짧아지며, 이전 그룹(F1,D1)은 그룹(F4,D2)으로 짧아진다. 갱신된 데이터는 이들이 메모리 블록의 경계를 오버랩(overlap)하기 때문에 두 개의 그룹인 (F2,D3), (F3,D4)로 기록된다. 일부 데이터는 제3 메모리 블록에 저장된다. 파일은 이제 데이터 그룹에 대한 인덱스 엔트리의 다음 시퀀스: (F0,D0),

(F2,D3), (F3,D4) 및 (F4,D2)에 의해 이제 설명된다.

[0086] 도12B의 파일은 삽입 명령을 사용하여 새로운 파일 데이터(I1)의 삽입에 의해 도12C에서 더 수정된다. 새로운 데이터(I1)는 삽입된 데이터가 메모리 블록의 경계를 오버랩하기 때문에 도12C의 새로운 그룹인 (F5,D6) 및 (F6,D7)로서, 도12B의 이전 그룹(F4,D2)의 바로 다음에 메모리에 기록된다. 제4 메모리 블록이 사용된다. 도12B의 이전 그룹(F0,D0)은 새로운 데이터(I1)의 삽입으로 인해, 도12C에서 짧아진 그룹인 (F0,D0) 및 (F7,D5)으로 분리된다. 파일은 이제 데이터 그룹에 대한 인덱스 엔트리의 다음 시퀀스: (F0,D0), (F5,D6), (F6,D7), (F7,D5), (F8,D3), (F9,D4), (F10,D2)에 의해 설명된다.

[0087] 도12D는 기록 명령을 사용하여, 파일의 단부에 새로운 데이터(W2)를 부가하는 도12C의 데이터 파일의 부가적인 수정을 도시한다. 새로운 데이터(W2)는 도12D의 새로운 그룹(F11,D8)으로서, 도12C의 이전 그룹(F10,D2)의 바로 다음에 기록된다. 파일은 이제 데이터 그룹에 대한 인덱스 엔트리의 다음 시퀀스:(F0,D0),(F5,D6),(F6,D7),(F7,D5),(F8,D3),(F9,D4),(F10,D2),(F11,D8)에 의해 설명된다.

[0088] 열린 파일로의 제2 갱신은 도12E에 도시되는데, 여기서 갱신된 파일 데이터(U2)는 갱신 명령을 발행하는 호스트에 의해 도12D의 파일에 기록된다. 갱신된 데이터(U2)는 도12D의 이전 그룹(F11,D8)의 바로 다음에 도12E에서 기록되는데, 이 데이터의 이전 버전은 무용화된다. 도12D의 이전 그룹(F9,D4)은 도12E에서 개정된 그룹(F9,D4)로 짧아지며, 이전 그룹(F10,D2)은 완전히 무용화되며, 이전 그룹(F11,D8)은 새로운 그룹(F14,D9)을 형성하도록 짧아진다. 갱신된 데이터는 도12E의 새로운 그룹인 (F12,D10) 및 (F13,D11)에 기록되어 블록 경계를 오버랩한다. 이제 제5 블록이 파일을 저장하기 위해 필요해진다. 파일은 이제 데이터 그룹에 대한 인덱스 엔트리의 다음 시퀀스: (F0,D0), (F5,D6), (F6,D7), (F7,D5), (F8,D3), (F9,D4), (F12,D10), (F13,D11), (F14,D9)에 의해 설명된다.

[0089] 각각의 파일의 데이터의 오프셋은 이전 설명에 따라 파일의 생성 또는 수정 이후에 정확한 논리 순서로 연속적으로 유지되는 것이 바람직하다. 그러므로 삽입 명령을 수행하는 일부로서(도12C), 예를 들어, 호스트에 의해 제공되는 삽입되는 데이터의 오프셋은 삽입에 바로 선행하는 오프셋으로부터 연속적이며, 삽입 이후에 파일에서 이미 있는 데이터는 삽입된 데이터의 양만큼 증가된다. 갱신 명령은 기존 파일에서 소정의 어드레스 범위 내에 있는 데이터가 갱신된 데이터의 양만큼 교체되게 하는 결과를 가져오므로, 파일의 다른 데이터의 오프셋이 통상적으로 교체될 필요가 없다.

[0090] 도11 및 도12에 의해 설명되고, 상기 설명된 모든 데이터 할당 및 인덱싱 기능이 메모리 시스템의 제어기에 의해 수행된다. 기록, 삽입 또는 갱신 명령들 중 하나에 따라, 호스트는 메모리 시스템으로 송신되는 파일 내의 데이터의 오프셋 및 파일ID를 단지 전달한다. 메모리 시스템은 휴식을 취한다.

[0091] 단지 상술된 방법으로 호스트로부터 플래시 메모리로 파일 데이터를 직접 기록하는 이점은 그렇게 저장된 데이터의 그래뉼래러티 또는 분해능(resolution)이 호스트의 것과 동일하게 유지될 수 있다는 것이다. 호스트 애플리케이션이 1 바이트의 그래뉼래러티를 갖는 파일 데이터를 기록한다면, 예컨대, 데이터는 또한 1 바이트 그래뉼래러티의 플래시 메모리에 기록될 수 있다. 개별적인 데이터 그룹 내의 데이터의 양 및 위치는 다수의 바이트에서 측정된다. 즉, 호스트 애플리케이션 파일 내에 별도로 어드레싱 가능한 데이터의 동일한 오프셋 유닛은 또한 플래시 메모리에 저장될 때 그 파일 내에 별도로 어드레싱 가능하다. 블록 내의 동일한 파일의 데이터 그룹들 사이의 임의의 경계는 가장 근접한 바이트 또는 다른 호스트 오프셋 유닛으로 인덱스 테이블에서 규정된다. 유사하게는, 블록 내의 상이한 파일의 데이터 그룹 사이의 경계가 호스트 오프셋의 유닛에서 규정된다.

[0092] 도12B 및 도12E로부터 갱신 명령의 실행이 파일을 저장하는데 필요한 물리적 공간이 파일에 있는 데이터량보다 더 커지도록 하는 결과를 가져온다는 것이 주의될 것이다. 이는 갱신에 의해 교체된 데이터가 메모리에 저장되어 있기 때문이다. 그러므로 무용하고, 유효하지 않은 데이터를 제거함으로써 적은 물리 저장 공간에 파일의 데이터를 통합(가비지 콜렉트)하는 것이 매우 바람직하다. 그러므로 더 많은 저장 공간이 다른 데이터에 대해 사용될 수 있다.

[0093] 도12B 및 도12E의 파일 데이터 갱신 외에도, 도12C의 데이터 삽입은 파일 데이터가 순서를 벗어나 저장되게 하는 결과를 가져온다는 것이 또한 주의될 수 있다. 즉, 갱신 및 삽입은 그들이 행해지는 동시에 메모리에 저장된 파일의 단부에 부가되는 반면, 이들은 파일 내의 어디든 거의 항상 논리적으로 위치된다. 이는 도12B, 도12C 및 도12E의 예의 경우이다. 그러므로, 파일 내의 오프셋의 순서에 부합하도록 메모리에 저장된 파일의 데이터를 재정리하는 것이 바람직할 수 있다. 이는 그 후에 순서대로 페이지 및 블록을 판독하는 것이 그들의 오프셋 순서로 파일 데이터를 제공할 것이기 때문에, 저장된 데이터를 판독하는 속도가 개선된다. 이는 파일의 최대 가능한

조각화를 제공한다. 그러나 더 효율적인 판독을 행하기 위해서 파일 데이터를 재정리하는 것은 메모리 시스템의 성능에 파일 데이터 통합만큼 중요하지 않은데, 이는 다른 데이터를 저장하는데 사용하기 위해 하나 이상의 메모리 블록을 잠재적으로 비워둔다. 그러므로 파일에서 데이터를 재정리하는 것은 통상적으로 자신에 의해 행해지지 않을 것이고, 여기서 이득은 부가된 작동 오버헤드의 가치가 아니라, 약간의 부가된 작동 오버헤드와 함께, 또는 어떠한 부가된 작동 오버헤드 없이 많은 가비지 콜렉션 작동의 일부로서 행해질 수 있다는 것이다.

[0094] 도12E의 파일은 행해진 두 개의 데이터 갱신(U1,U2)으로 인해 메모리에 저장된 무용 데이터 그룹(회색 부분)을 포함한다. 파일을 저장하는데 사용되는 메모리 용량의 양은 결과적으로 도12E를 통해 명백한 바와 같이, 파일의 크기보다 실질적으로 더 크다. 그러므로 가비지 콜렉션은 적합하다. 도13은 도12E의 데이터 파일을 가비지 콜렉팅하는 결과의 도면을 제공한다. 그 파일은 가비지 콜렉션 전에, 거의 다섯 개의 블록의 저장 용량을 취하는 반면(도12E), 가비지 콜렉션 이후에 동일한 파일이 세 개보다 조금 많은 메모리 셀 블록에 부합된다(도13). 가비지 콜렉션 작동의 일부로서, 데이터는 블록으로부터 복제되며, 여기서 이들은 다른 삭제된 블록 내에 처음에 기록되므로, 원 블록은 삭제된다. 전체 파일이 데이터 콜렉트된다면, 이들의 데이터는 파일 내의 데이터 논리 오프셋 순서와 동일한 물리 순서로 새로운 블록에 복제될 수 있다. 갱신(U1,U2) 및 삽입(I1)은 예컨대, 호스트 파일에서 나타나는 바와 동일한 순서로 가비지 콜렉션(도13) 이후에 저장된다.

[0095] 가비지 콜렉션은 또한 통합되는 파일 내에 새롭고 상이한 데이터 그룹의 생성의 결과를 일반적으로 가져온다. 도13의 경우에, 파일은 새로운 데이터 그룹에 대한 인덱스 엔트리의 다음의 새로운 시퀀스: (F0,D0),(F1,D1),(F2,D2),(F3,D3)에 의해 설명된다. 이는 도12E에 도시된 파일의 상태에 존재하는 것보다 매우 적은 수의 데이터 그룹이다. 이제 파일의 데이터가 복제되는 메모리 셀 블록의 각각에 대해 하나의 데이터 그룹이 존재한다. 가비지 콜렉션 작동의 일부로서, 파일 인덱스 테이블(FIT)은 파일을 형성하는 새로운 데이터 그룹을 반영하기 위해서 갱신된다.

[0096] 변화 가능한 크기로 정해진 데이터 유닛의 저장

[0097] 플래시 메모리의 현재 작동에서, 논리 어드레스 공간의 소정의 데이터 블록을 데이터 블록의 데이터 양과 동일한 데이터 저장 용량을 갖는 메모리의 물리 블록 또는 메타 블록으로 맵핑하는 것이 전형적이다. 이는 메모리 시스템의 작동을 더욱 효율적이게 한다. 그러나, 저장 전에 논리 데이터 블록의 크기를 변화시키는 물리 메모리 및 논리 어드레스 공간 사이의 데이터 변환이 있을 때, 논리 데이터 블록 및 물리 데이터 블록 사이에 이러한 요망되는 크기의 동일함이 더 이상 존재하지 않는다. 그래서 종래 맵핑 방식은 사용될 수 없다.

[0098] 이러한 문제점을 극복하기 위한 하나의 기술은 변환된 데이터의 불규칙한 크기의 그룹으로 중간 논리 어드레스를 할당하여, 중간 어드레스 블록을 동일한 크기를 갖는 플래시 메모리 블록에 맵핑하는 것이다. 그러나, 후속하는 갱신에 의한 데이터의 이러한 중간 크기로 정해진 그룹의 크기의 변화는 중간 어드레스의 물리 메모리 블록으로의 할당에서 중요한 조각화를 생성한다. 이는 더 빈번한 데이터 통합 및 가비지 컬렉션을 필요하게 하는 결과를 가져오는데 이는 메모리 시스템의 성능을 매우 감소시킨다.

[0099] 메모리 시스템의 양호한 성능을 여전히 유지하는 한편, 이러한 변환되거나 불규칙적한 크기의 데이터의 유닛을 수용하기 위하여 발견된 것은 상기 섹션에서 설명된 직접적인 데이터 파일 기술의 사용이다. 가변 크기의 데이터 그룹의 사용은 특히 데이터 유닛의 양을 다르게 하고 변경하는 것을 처리하도록 적응된다.

[0100] 도14는 상술된 직접적인 데이터 파일 기술을 사용하여 데이터 파일을 기록하는 것의 예를 도시하지만, 여기서 데이터는 호스트에 의해 제공되는 것과 다른 양의 데이터를 저장하는 방식으로 메모리 시스템 내에서 변환된다. 데이터 변환은 메모리에 기록되기 전에 수신된 데이터의 압축, 암호화 또는 다른 인코딩에 기인할 수 있다. 압축 알고리즘은 호스트로부터 소정량의 데이터를 저장하기 위해 필요한 물리 메모리 양을 감소시키기 위해서 데이터의 양을 상당히 감소시키는 것이 주요 목적이다. 반면, 데이터 암호화 프로세스는 저장되는 데이터의 양을 종종 증가시킨다. 본원에서 설명되는 기술은 어느 경우에나 동일하게 적용되며, 또한 저장된 데이터의 일부분이 증가된 크기이고 다른 부분의 크기가 감소되는 경우에도 적용된다.

[0101] 도14를 참조하면, 파일(201) 내의 데이터는 0으로부터 N으로 신장하는 논리 오프셋 어드레스(202)를 갖는다. 파일 오프셋 어드레스는 데이터를 따라 호스트에 의해 제공된다. 이러한 어드레스 공간은 데이터(HD1,HD2,HD3,HD4)의 연속적인 유닛(203)으로 전환(203)에 의해 메모리 시스템 내에서 분리된다. 이러한 호스트 데이터 유닛은 그 후에 207로 나타내지는 바와 같이 개별적인 논리 데이터 그룹(DG1,DG2,DG3,DG4)으로 블록(205)에 의해 나타내지는 바와 같이 변환된다. 이를 설명하기 위해서, 변환된 데이터 그룹(DG2)은 대응하는 호스트 데이터 유닛(HD2)보다 크고, 변환된 데이터 그룹(DG3)은 호스트 유닛(HD3)과 크기가 같으며, 다른 두 개의

데이터 그룹은 그들의 변환되지 않은 호스트 데이터 사본보다 적은 데이터를 포함한다. 그러나 특히, 변환된 데이터 그룹은 수행되는 변환의 유형에 따라 그들의 대응하는 호스트 데이터 유닛보다 모두 더 크거나 모두 더 작을 것이다. 데이터 변환 이후에, 논리 데이터 그룹(207)은 상술된 직접적인 데이터 파일 기술과 유사한, 블록(211)에 의해 나타내지는 바와 같은, 어레이 내의 물리 어드레스에 그들을 할당함으로써 플래시 메모리 셀 어레이(209)에 저장된다.

[0102] 파일 논리 어드레스 범위(203)의 개별적인 호스트 데이터 유닛(HD1) 등은 데이터의 동일하거나 실질적으로 동일한 양을 포함하도록 선택되는 것이 바람직하다. 이러한 양의 데이터는 또한 데이터 변환(205)에 의해 사용되는 것과 동일하게 설정되는 것이 바람직하다. 즉, 적어도 많은 압축 및 암호화 알고리즘은 동시에 파일의 설정된 양의 데이터만을 처리하며, 호스트 데이터 유닛(HD1) 등의 크기는 그 양과 동일하게 만들어질 수 있다. 이러한 양은 또한 변환 알고리즘의 파라미터를 설정함으로써 종종 선택될 수 있고, 그러므로 호스트 데이터 유닛(HD1) 등의 이러한 양 및 크기는 특정한 시스템의 작동을 최적화하기 위해 함께 선택될 수 있다.

[0103] 도15에 도시된 바와 같이, 이들 기술은 상술된 호스트 데이터 파일 인터페이스에 국한되는 것이 아니라, 단일 논리 어드레스 공간이 호스트 및 메모리 시스템 사이의 다수의 파일을 인터페이스하는데 또한 사용될 수 있다. 이는 종래 디스크 저장 시스템 인터페이스이다. 데이터 파일(213)은 어드레스 전환 기능(214)에 의해 논리 어드레스 공간(215) 내에서 호스트에 의해 고유한 어드레스를 할당받는다. 공지된 바와 같이, 호스트는 새로운 파일의 논리 어드레스를 특정한 파일에 대해 연속적이지 않을 수 있거나 종종 연속적이지 않은 어드레스 공간(215) 내의 점유되지 않은 어드레스에 할당한다. 그러므로 메모리 시스템은 임의의 특정한 파일에 할당된 논리 어드레스에 대한 직접적인 정보를 갖지 않는다.

[0104] 그러나 도15의 단일 논리 어드레스 인터페이스(215)가 데이터의 연속적인 세트(217)로 216으로 나타내지는 바와 같이 분리될 수 있는데, 각각의 세트는 동일한 양의 데이터를 포함하는 것이 바람직하다. 그래서 각각의 세트는 219에 도시된 바와 같이, 논리 호스트 데이터 유닛(HD1,HD2,HD3)으로 218에서 분리된 그의 논리 어드레스 범위를 갖는, 데이터 파일 오브젝트와 같이 처리된다. 이는 2005년 8월 3일자로 출원된 Sergey A. Gorobets에 의한 미국 특허 출원 일련 번호 제 11/196,869호에서 설명되는 기술의 구현이다. 각각의 데이터 세트(217) 내의 어드레스는 논리 어드레스 전환(218)에 의해 호스트 데이터 유닛(219)으로 맵핑된다. 호스트 데이터(HD) 유닛의 크기는 바람직하게는 동일하거나, 실질적으로 동일하고, 변환 기능(223)에 의해 한번에 변환된 데이터의 양에 대응하도록 선택되는 것이 바람직하다. 그러므로 이러한 호스트 데이터(HD) 유닛은 도14를 참조하여 상술된 프로세스에서와 동일한 방식으로 225에서 나타내지는 바와 같이, 여러 크기의 논리 데이터 그룹(DG)들 중 대응하는 것으로 전환된다. 그 후에 논리 데이터 그룹은 어드레스 번역(227)에 의해 결정된 물리 어드레스에서 메모리(209)에 저장된다.

[0105] 데이터 변환(205(도14) 또는 223(도15))의 수행에서, 논리 세트(204 또는 219)의 호스트 데이터(HD) 유닛의 완전한 하나는 전형적으로 RAM(31)에서와 같이 한번에 메모리 제어기에 일시적으로 저장된다(도2). 그 후에 저장된 HD 유닛은 인코딩되거나, 그렇지 않으면 변환된다. 유사하게, 변환된 방식으로 저장된 호스트 데이터 유닛을 수정할 때, 데이터가 판독되고, 재변환되며, 그 후에 재변환된 데이터 유닛의 일부 또는 전부가 수정되는 동안 일시적으로 RAM에 저장된다. 그러므로 이는 현재 플래시 메모리 시스템이 프로그래밍 및 판독 기능을 수행할 때 데이터의 단지 하나의 페이지 또는 메타 페이지에 대한 저장 용량을 전형적으로 필요로 하기 때문에 더 큰 용량 RAM을 필요로 할 수 있다.

[0106] 도14 및 도15의 메모리 시스템이 외부 호스트로 그리고 외부 호스트로부터 데이터를 전달하는 것으로 도시될지라도, "호스트"는 대안적으로 메모리 시스템의 내부에서 실행되는 소프트웨어 애플리케이션 프로그램일 수 있다. 플래시 메모리 어레이 내에 저장된 애플리케이션 소프트웨어는 RAM(31)에 로딩되며, 메모리의 이러한 프로세서 제어 작동 외에도, 제어기 프로세서(27)에 의해 실행된다(도2). 또는, 필요하다면, 부가적인 마이크로프로세서가 이러한 목적을 위해 메모리 시스템에 부가될 수 있다.

[0107] 도16을 처음으로 참조하면, 도14 또는 도15 중 어느 하나의 예의 변환된 논리 데이터 그룹(DG)을 번역하고 저장하는 기술이 설명될 것이다. 도16에서 데이터의 세트(231) 또는 예시적인 파일이 호스트 데이터 파일 인터페이스의 하나의 호스트 파일(201)(도14) 또는 논리적인 어드레스 공간(215)의 일부 또는 전체(도15) 중 어느 하나의 것을 나타낸다. 호스트 데이터(231)의 논리 어드레스 범위는 동일한 크기의 호스트 데이터 유닛(HD1,HD2,HD3 등)으로 분리되는데, 이들 각각은 A1,A2,A3등의 개별적인 시작논리 어드레스를 갖는다. 호스트 데이터 유닛(HD1,HD2, HD3등)은 압축, 암호화 또는 다른 유형의 인코딩에 의해서와 같이, 메모리 시스템 내에서, 변하는 길이(L1,L2,L3등)를 각각 갖는 대응하는 데이터 그룹(DG1,DG2,DG3등)으로 개별적으로 변환된다.

[0108] 그 후에 변환된 데이터 그룹(DG1,DG2,DG3등)은 도11 내지 도13을 참조하여 상술된 직접적인 데이터 파일 작동의 데이터 그룹으로서 처리된다. 데이터 그룹은 물리 메모리(233)에 저장되는데, 두 개의 블록 또는 메타 블록(X,Y)가 도시된다. 데이터 그룹은 서로 인접한 순서로 저장되는데, 데이터 그룹(DG1)은 메모리 블록(X)의 어드레스(0)에서 시작하고, 데이터 그룹(DG2)은 DG1의 단부 등의 바로 다음에 블록(X)의 어드레스(1)에서 시작한다. 데이터 그룹(DG5)은 블록(X,Y)들 사이에서 분리되는데, 이러한 예에서, 두 개의 서브 그룹(DG5a, DG5b)으로 분리되는데, 이들은 개별적인 길이(L5a,L5b)를 가지며, 블록(X)의 메모리 어드레스(4) 및 블록(Y)의 메모리 어드레스(0)에서 시작한다. 블록(X) 내에 적합한 그룹(DG5)의 시작으로부터 많은 데이터가 그룹(DG5)에 저장됨에 따라, 그룹(DG5)의 나머지 양은 블록(Y)에 저장된다.

[0109] 도16의 예에서 호스트는 메모리 시스템에 저장된 데이터에 대한 그의 논리 어드레스 공간이 호스트 데이터 유닛(HD1,HD2,HD3등)으로 분리됨을 인식하지 않는다. 호스트는 또한 메모리 시스템에서 행해지는 데이터 변환을 인식하지 않는다. 호스트는 메모리 시스템으로부터 데이터를 판독할 때, 도16을 참조하여 설명된 기록 프로세스의 역이 발생한다. 규정된 데이터 파일(도14) 또는 시스템의 논리 어드레스 공간 내에서 규정된 어드레스의 데이터(도15)가 호스트에 의해 요청될 때, 저장된 데이터 그룹(DG1,DG2,DG3등)이 저장 블록(X,Y)으로부터 판독되며, 호스트 데이터(HD1,HD2,HD3등)의 개별적인 유닛으로 개별적으로 역변환되고 그 후 호스트로 송신된 데이터 그룹(231)에 결합된다.

[0110] 호스트 데이터(HD)의 유닛에 대응하는 데이터 그룹(DG)의 물리 위치는 파일 인덱스 테이블(FIT)에 의해 메모리 시스템 오버헤드의 일부로서 유지된다. 이러한 테이블의 일부의 예가 도16에 도시된 데이터 저장에 대해서 도17에서 주어진다. 이러한 테이블은 데이터가 우선 기록될 때 형성되고, 기록된 데이터가 수정됨에 따라 수정되며, 호스트에 의해 요청되는 데이터를 찾기 위해 판독 동안 사용된다. 데이터 유닛(231)의 각각의 저장된 데이터 그룹(DG)에 대한 테이블에서 적어도 하나의 라인이 존재한다. 각각의 데이터 그룹에 대한 정보는, 테이블의 열에서, 데이터 그룹이 대응하는 호스트 데이터 유닛(HD)의 오프셋 어드레스(A)를 제공하는 열(235)을 포함한다. 열(237)은 변환된 데이터 그룹(DG)의 길이(L)를 저장한다. 시작 바이트 위치 또는 데이터 그룹이 시작하는 물리 메모리(233)의 다른 어드레스가 포인터 열(239)에서 제공된다.

[0111] 도11 내지 도13을 참조하여 상술된 것으로부터 직접적인 데이터 파일 기술의 이러한 애플리케이션에 하나의 상당한 차이가 있다. 변환되지 않은 데이터 저장의 경우에 데이터 그룹이 이들의 수를 감소시키기 위해 쉽게 결합되므로, 이들을 주시하는데 있어서의 다른 오버헤드 및 FIT의 크기를 감소시키는데, 이는 변환된 데이터 그룹을 갖는 경우는 아니다. 개별적인 호스트 데이터(HD) 유닛은 개별적으로 변환되기 때문에, 각각 개별적인 대응하는 변환된 데이터 그룹(DG)을 계속 주시하는 것이 바람직하다. 이는 저장 블록 내에 저장된 데이터 그룹 사이의 경계의 기록을 유지하므로, 저장 블록으로부터 판독된 데이터를 개별적인 데이터 그룹으로 분리하도록 한다. 그래서 대응하는 호스트 데이터(HD) 유닛은 FIT의 라인들 중 하나에 의해 식별되는 대응하는 데이터 그룹(DG)으로부터 개별적으로 재변환된다.

[0112] 그러므로, 열(241)은 고정된 상태로 있는지 여부를 각각의 데이터 그룹에 대해 나타내는 단일 비트와 함께 도17의 FIT에 포함된다. "1" 비트는 이것이 그 케이스인 것을 나타낸다. 데이터 그룹을 병합하지 않음으로써, 저장된 데이터 그룹 사이의 경계 식별이 유지된다. "0" 비트는 데이터 그룹이 도11 내지 도13에 관하여 상술된 일반적이 직접적인 데이터 파일 설명 및 직접적인 데이터 파일 애플리케이션의 방식으로 다른 것과 결합될 수 있다는 것을 나타낸다. 이는 메모리 시스템이 변환된 데이터 및 변환되지 않은 데이터 둘 다를 저장할 수 있도록 한다. 변환되는 데이터 그룹에 대해서, 열(241)의 비트는 "1"이고, 변환되지 않은 데이터 그룹에 대해서, 이러한 비트는 "0"이다.

[0113] 변환된 데이터 및 변환되지 않은 데이터 둘 다를 저장하는 시스템에서, 호스트로부터 수신된 기록 명령은 메모리 시스템 내에서 두 개의 특정한 기록 명령들 중 하나로 전환된다. 기록 명령은 수신된 데이터가 직접적인 데이터 파일 저장을 설명하는 상기 섹션의 기술에 따라 병합될 수 있는 자유로운 데이터 그룹으로서 메모리에 기록되도록 한다. 기록_그룹 명령은 수신된 데이터가 이러한 섹션의 설명에 따라 병합될 수 없는 고정된 데이터 그룹으로서 메모리에 기록되도록 한다. 고정된 데이터 그룹은 그룹을 형성하는 스트리밍된 데이터에 이어 기록_그룹 명령과 그 후 데이터 그룹의 단부를 규정하기 위한 닫힘_그룹 명령의 데이터 변환 기능으로부터 전송에 의해 규정된다.

[0114] 도17의 FIT의 다른 열(243)은 각각의 데이터 그룹에 대해서 서로 링크되었는지 여부를 나타낸다. 이는 데이터 그룹이 도16의 예에서, 저장 블록(X,Y)들 사이에서 분리되는 데이터 그룹(DG5)과 같이, 두 개의 메모리 블록으로 분리될 때 발생한다. 두 개의 서브-그룹들이 결과적으로 형성된다. "1"은 각각의 링크된 데이터 서브-그룹에

대한 이러한 열에 저장되고, "0"은 링크되지 않은 그룹에 대한 것이다. 이러한 예에서, 각각의 서브 그룹(244,246)은 데이터 그룹(A5)의 일부만을 보여주기 위해서 이러한 열에서 "1"을 갖는다. 이는 두 개가 전체 데이터 그룹(A5)이 단일 메모리 블록에 저장되는 결과를 가져온다면 임의의 미래의 데이터 통합 또는 가비지 콜렉션 동안 단일 데이터 그룹(A5)에 다시 결합될 수 있는 서브 그룹이라는 것을 말한다. 이는 개별적인 데이터 그룹이 결합되도록 하지 않는 열(241)에서 비트 "1"에 의해 야기되는 제한에 대한 예외이다. 데이터 그룹(A5)의 오프셋 어드레스 및 길이는 단 한번 도17에서, 제1의 두 개의 서브 그룹에 대한 기록(244)의 일부로 포함된다. 대안적으로, 이들 데이터는 기록(244,246) 둘 다에서 반복될 수 있다.

[0115] 최종 열(245)은 호스트 파일의 단부인 데이터 그룹을 마킹하기 위해서 도17의 예시적인 FIT에 제공된다. 도17의 테이블에서 마지막 데이터 그룹은 "1"로 마킹되고, 다른 것은 "0"으로 마킹됨으로써 호스트 파일의 마지막 데이터 그룹과 다르다고 나타내진다. 이는 메모리 시스템이 호스트와의 데이터 파일 오브젝트 인터페이스를 가질 때 사용된다, 도14에 도시된 경우. 그러나, 호스트 인터페이스가 도15의 연속적인 단일 논리 어드레스 공간일 때, 통상적으로 사용되지는 않을 것이다.

[0116] 데이터 그룹(DG3)의 길이(L3)와 같은 데이터 그룹의 길이는 이러한 데이터 그룹의 시작 어드레스(A3) 및 다음 데이터 그룹의 시작 어드레스(A4) 사이의 차이와 동일하지 않다는 것이 인지될 것이다. 이는 물론 호스트 데이터 유닛(HD3)의 변환이 그의 길이를 바꾸었기 때문이다.

[0117] 도18 및 도19는 데이터가 호스트 데이터 유닛(HD1-HD8)에 의해 처음에 나타내지는 데이터 파일에 나중에 추가된다면 발생할 수 있는 개별적인 도16 및 도17에 대한 변화를 도시한다. 이러한 경우에, 호스트 데이터 그룹(HD9)은 파일의 단부에 부가된다. 이러한 호스트 데이터 유닛은 데이터 그룹(DG9)으로 개별적으로 변환된다. 변환된 데이터 그룹(DG9)의 제1 일부는 이러한 예에서, 메모리 저장 블록(Y)의 단부에 남은 공간에 저장되고, 나머지 데이터 그룹(DG9)은 새로운 저장 블록(Z)의 시작부에 저장된다. 대안적으로, 데이터 그룹(DG9)은 다른 파일의 하나 이상의 데이터 그룹을 또한 저장하는데 사용되는 공통 블록에 저장될 수 있다. 공통 블록의 사용은 직접적인 데이터 파일 애플리케이션에서 설명된다.

[0118] 이러한 예에서, 도17의 FIT는 도19에 도시된 바와 같이 그의 단부에서 두 개의 엔트리를 부가함으로써 갱신된다. 하나의 엔트리는 저장 블록(Y)의 서브 그룹에 대한 것이고, 다른 엔트리는 저장 블록(Z)의 서브 그룹에 대한 것이다. "링크된" 비트는 각각의 이들 서브 그룹에 대해서 "1"로 설정되며, "파일의 단부" 비트는 마지막 서브 그룹으로 이동된다.

[0119] 도20 및 도21은 개별적인 도16 및 도17의 저장된 데이터 그룹에 대한 다른 유형의 변화를 도시한다. 이러한 경우에, 호스트 데이터 유닛(HD4)의 일부가 갱신된다. 갱신 프로세스는 우선 메모리 저장 블록(X)으로부터 전체 대응하는 변환된 데이터 그룹(DG4)을 판독하는 것을 포함하며, 원래 호스트 데이터 유닛(HD4)(249)을 재구성하기 위해 도20의 247로 나타내지는 바와 같이 이러한 그룹의 데이터를 역변환시키는 것을 포함한다. 일단 원래 호스트 데이터 유닛이 획득되면, 이는 원래 데이터의 일부를 교체하는 경우에 251에서 나타내지는 바와 같이 호스트에 의해 갱신된다. 그 후에 갱신된 호스트 데이터 유닛은 253에서 나타내지는, 원래 호스트 데이터 유닛(HD4)을 변환시키는데 사용되는 동일한 알고리즘에 의해 변환된다. 이는 갱신된 데이터 그룹(DG4)(255)이 원래 데이터 그룹의 길이(L4)와 전형적으로 다른 길이(L9)를 갖는 결과를 가져온다.

[0120] 갱신된 데이터 그룹은 도20에서 새로운 저장 블록(Z)내에 전체적으로 저장되는 것이 아니라, 도18의 부가된 데이터 그룹(DG9)의 저장에 대한 대안으로서 저장 블록(Y)에 부분적으로 저장되는 것으로 도시된다. 이는 부가된 데이터 그룹이 두 개의 서브 그룹으로 분리하는 것을 피한다. 그 후에 원래 데이터 그룹(DG4)은 저장된 데이터가 통합되거나 가비지 콜렉트된 후에 무용화되고 제거되는 것으로 마킹된다.

[0121] 그 후에 도17의 원래 FIT는 이러한 변화를 반영하도록 갱신된다. 도21에 도시된 바와 같이, 데이터 그룹(DG4)에 대한 라인 상의 정보는 그의 새로운 길이(L9) 및 그의 새로운 시작 메모리 어드레스(Z0)를 나타내도록 변경된다. 파일 비트의 단부는 또한 유닛의 마지막 데이터 그룹이기 때문에, 데이터 그룹(DG8)에 대한 세트로 남는다.

[0122] 도20 및 도21은 판독/수정/기록에 관한 것의 형태를 도시한다. 이는 RAM(31)(도1)과 같이 상이한 양의 버퍼 메모리를 사용하는 방법으로 실행될 수 있다. 하나의 방법은 수정된 전체 데이터 그룹(DG)을 판독하고, 이를 재변환하고, 버퍼 메모리의 전체의 결과적인 호스트 데이터 그룹(DG)을 저장하고, 새로운 도입 데이터와 함께 그의 일부를 겹쳐서 기록함으로써 저장된 데이터를 수정하며, 그 후에 메모리 내에서 재기록하기 위해 저장된 수정된 데이터를 변환시키는 것이다. 적은 버퍼 메모리를 필요로 하는 다른 방법은 수정되지 않은 재변환된 데이터의

일부만을 저장하고, 그 후에 저장된 데이터를 판독하고, 새로운 도입 교체 데이터와 함께 이들을 결합하며, 메모리에 기록하기 위해 수정된 데이터를 변환하는 것이다.

[0123] 결론

[0124] 본 발명의 여러 양상이 그의 예시적인 실시예에 관하여 설명되었을지라도, 본 발명은 첨부된 청구항의 전체 범위 내에서 보호를 위해 권한이 주어진다는 것을 인식할 것이다.

도면의 간단한 설명

[0011] 도1은 현재 구현되는 호스트 및 연결된 비휘발성 메모리 시스템의 개략도;

[0012] 도2는 도1의 비휘발성 메모리와 같이 사용하기 위한 예시적인 플래시 메모리 시스템의 블록도;

[0013] 도3은 도2의 시스템에서 사용될 수 있는 메모리 셀 어레이를 대표하는 회로도;

[0014] 도4는 도2의 시스템의 예시적인 물리 메모리 구성을 도시하는 도면;

[0015] 도5는 도4의 물리 메모리 일부에 대한 확대도;

[0016] 도6은 도4 및 도5의 물리 메모리 일부에 대한 부가적인 확대도;

[0017] 도7은 호스트 및 재프로그래밍 가능한 메모리 시스템 사이의 공통적인 종래 논리 어드레스 인터페이스를 도시하는 도면;

[0018] 도8은 도7과는 다른 방법의 호스트 및 재프로그래밍 가능한 메모리 시스템 사이의 공통적인 종래 논리 어드레스 인터페이스를 도시하는 도면;

[0019] 도9는 직접적인 데이터 파일 애플리케이션에 따른, 호스트 및 재프로그래밍 가능한 메모리 시스템 사이의 직접적인 데이터 파일 저장 인터페이스를 도시하는 도면;

[0020] 도10는 도9와는 다른 방법의 직접적인 데이터 파일 애플리케이션에 따른, 호스트 및 재프로그래밍 가능한 메모리 시스템 사이의 직접적인 데이터 파일 저장 인터페이스를 도시하는 도면;

[0021] 도11A-도11D는 데이터 파일을 메모리에 직접 기록하는 네 가지 다른 예를 도시하는 도면;

[0022] 도12A-도12E는 단일 데이터 파일을 메모리에 직접 기록하는 순서를 도시하는 도면;

[0023] 도13은 도12E에 도시된 데이터 파일을 가비지 콜렉트한 결과를 도시하는 도면;

[0024] 도14는 본 발명에 따른, 데이터 변환과의 직접적인 데이터 파일 메모리 시스템의 작동을 도시하는 도면;

[0025] 도15는 본 발명에 따른, 데이터 변환과의 논리 어드레스 공간 메모리 시스템의 작동을 도시하는 도면;

[0026] 도16은 도14 또는 도15 중 어느 하나에서 데이터 변환의 도시적인 예;

[0027] 도17은 도16의 예에 저장된 데이터의 파일 인덱스 테이블(FIT);

[0028] 도18은 도16의 예에 따라 저장된 데이터에 데이터를 추가하는 것을 도시하는 도면;

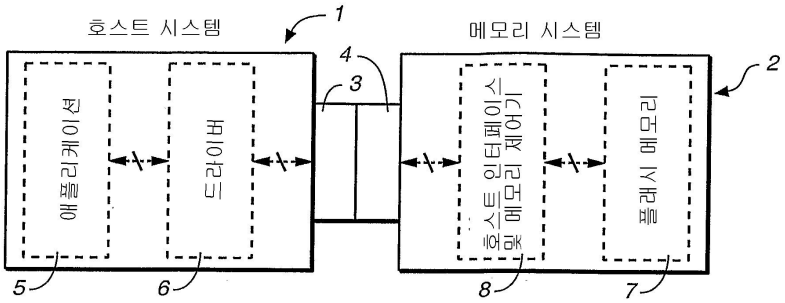
[0029] 도19는 도18에 따라 추가된 일부 데이터 다음에 저장된 데이터의 파일 인덱스 테이블;

[0030] 도20은 도13의 예에 따라 저장된 데이터의 갱신을 도시하는 도면; 및

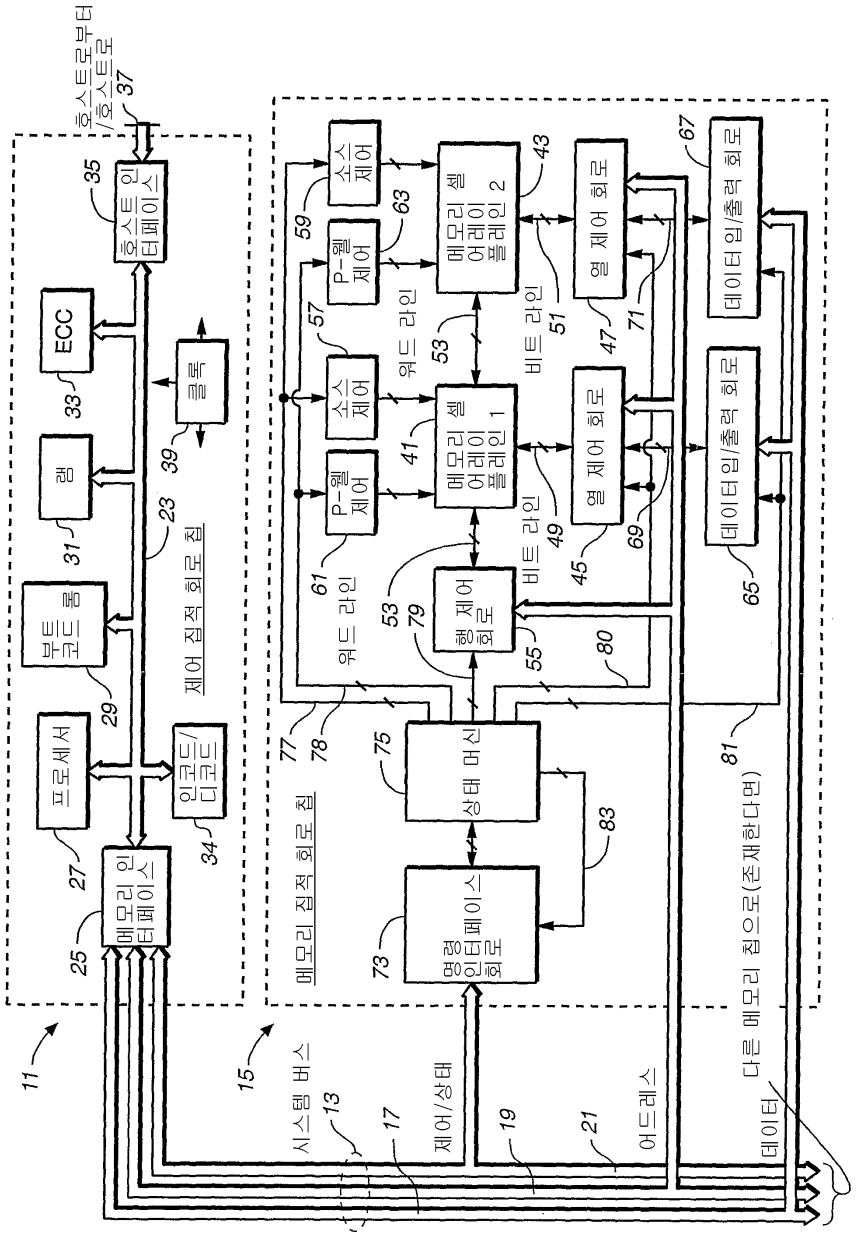
[0031] 도21은 도20에 따라 갱신된 후에 저장된 데이터의 파일 인덱스 테이블.

도면

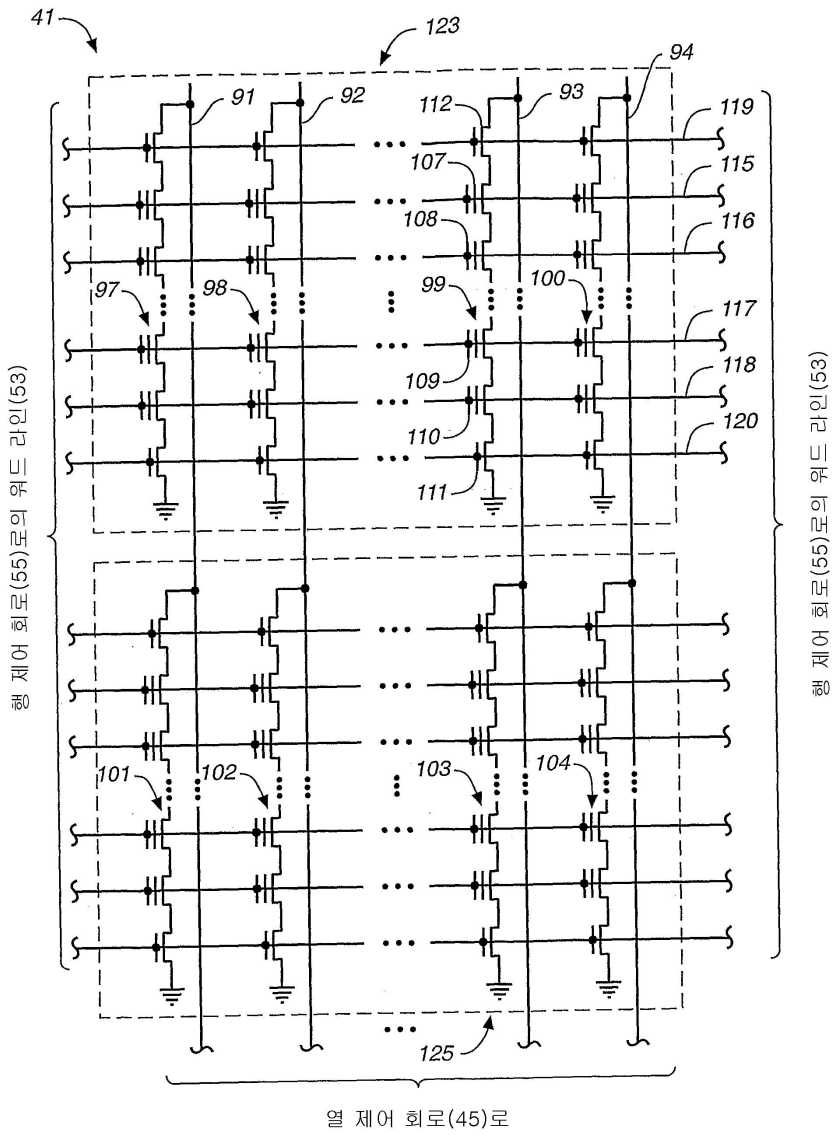
도면1


호스트 시스템
1
메모리 시스템
2
4
3
애플리케이션
드라이버
호스트 인터페이스 및 메모리 제어기
플래시 메모리
5
6
8
7

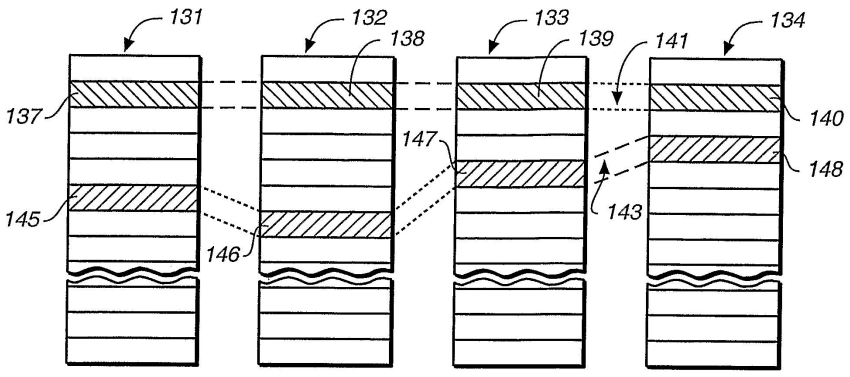
도면2


호스트로부터/호스트로
37
35
호스트 인터페이스
ECC
33
램
31
부트 코드 롬
29
프로세서
27
메모리 인터페이스
25
클록
39
23
제어 집적 회로 칩
입코드/디코드
34
11
15
메모리 집적 회로 칩
소스 제어
59
63
P-웰 제어
57
소스 제어
61
P-웰 제어
워드 라인
53
메모리 셀 어레이 플레인 2
43
메모리 셀 어레이 플레인 1
41
비트 라인
51
49
열 제어 회로
47
열 제어 회로
45
69
71
67
데이터 입/출력 회로
65
제어 행 회로
79
53
55
80
77
78
상태 머신
75
명령 인터페이스 회로
73
83
81
다른 메모리 칩으로(존재한다면)
시스템 버스
13
제어/상태
17
19
어드레스
21
데이터

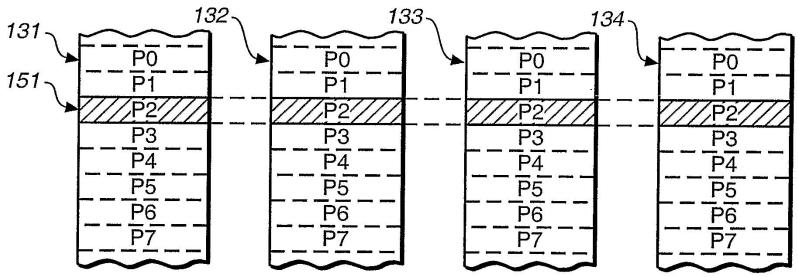
도면3


41
123
94
91
92
112
93
119
107
115
108
116
97
98
99
100
117
109
118
110
120
111
101
102
103
104
125
행 제어 회로(55)로의 워드 라인(53)
행 제어 회로(55)로의 워드 라인(53)
열 제어 회로(45)로

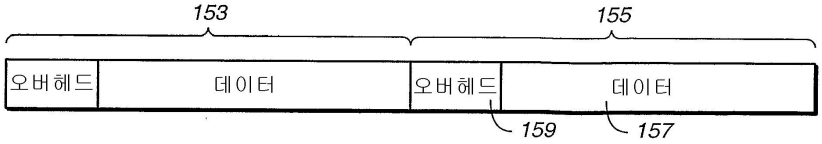
도면4


131
132
138
133
139
141
134
137
140
147
148
145
143
146

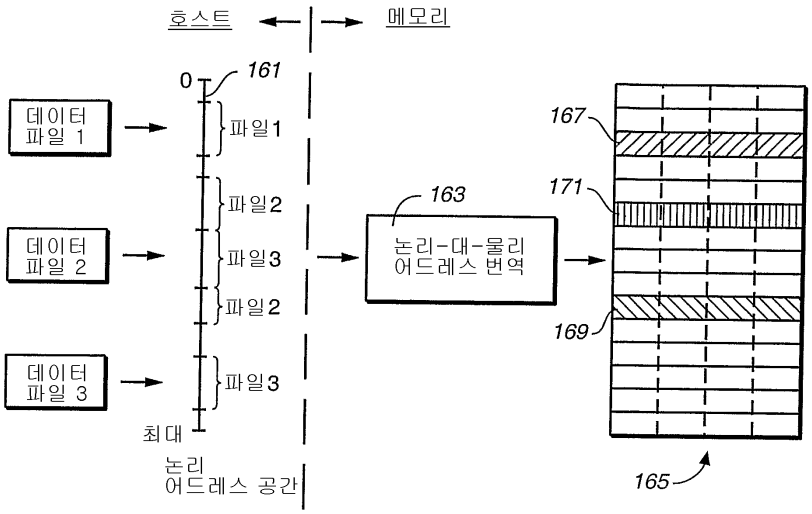
도면5


131
132
133
134
151
P0
P1
P2
P3
P4
P5
P6
P7

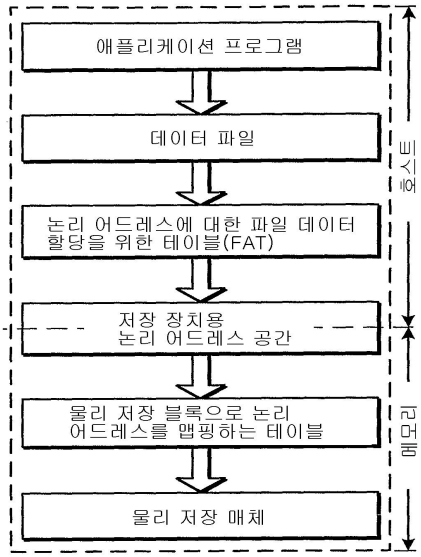
도면6


153
155
오버헤드
데이터
오버헤드
데이터
159
157

도면7

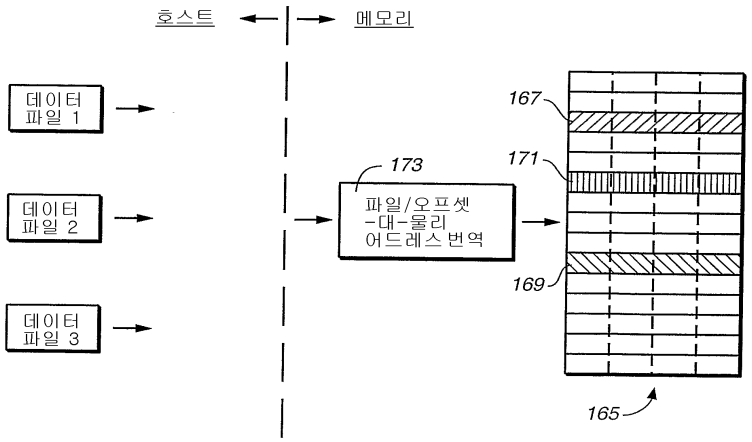

호스트
메모리
0
161
데이터 파일 1
데이터 파일 2
데이터 파일 3
파일1
파일2
파일3
파일2
파일3
최대
논리 어드레스 공간
163
논리-대-물리 어드레스 번역
167
171
169
165

도면8

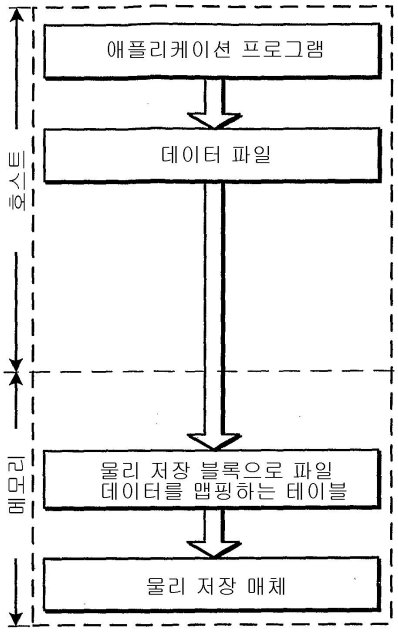


종래 시스템

도면9

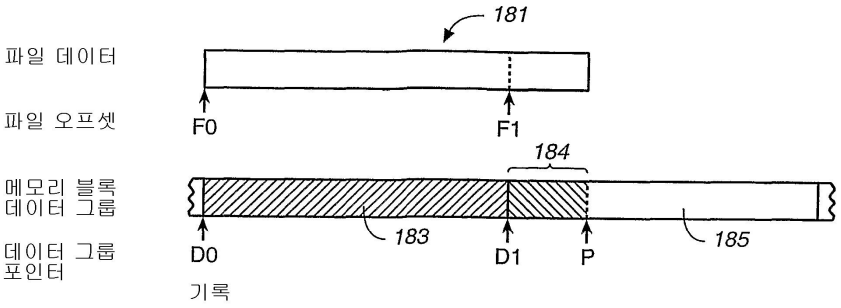


도면10

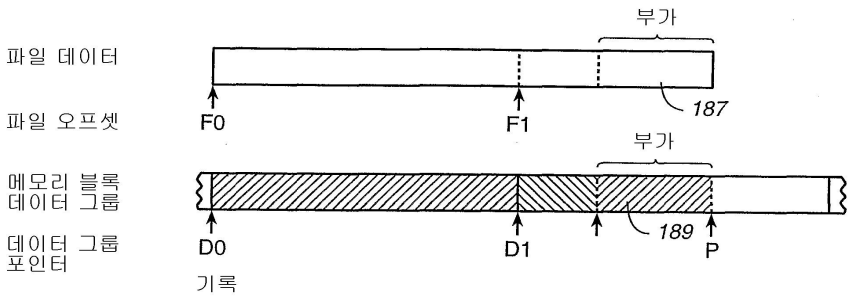


직접적인 데이터 파일 저장 시스템

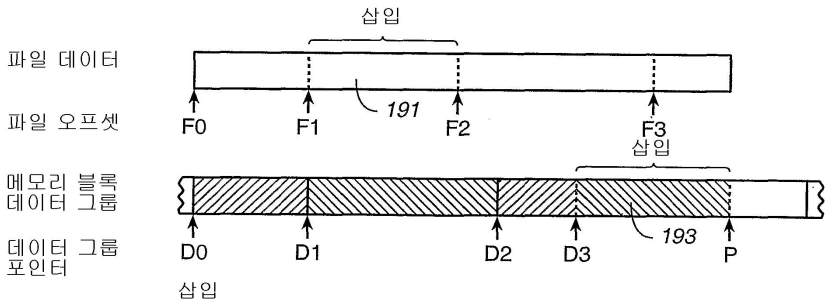
도면11A



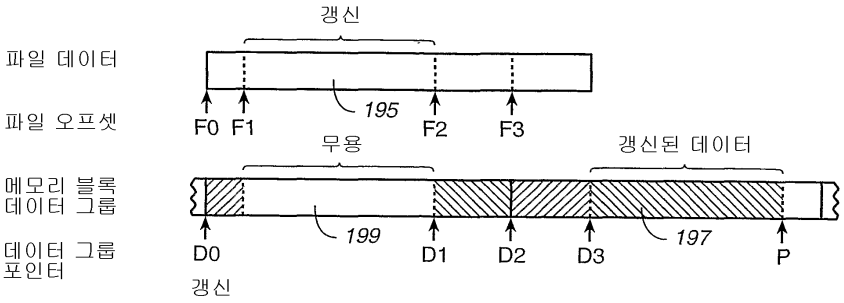
도면11B



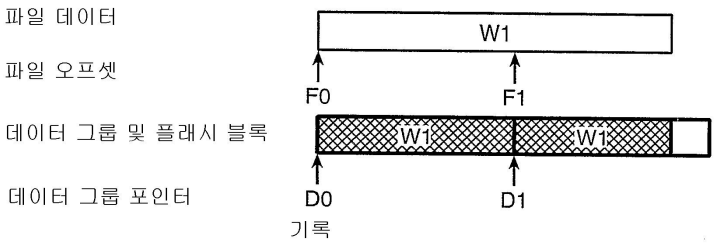
도면11C



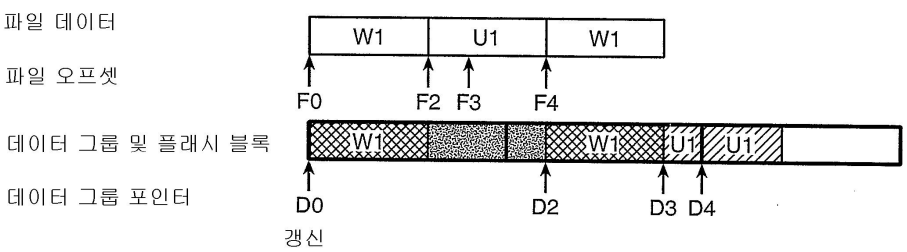
도면11D



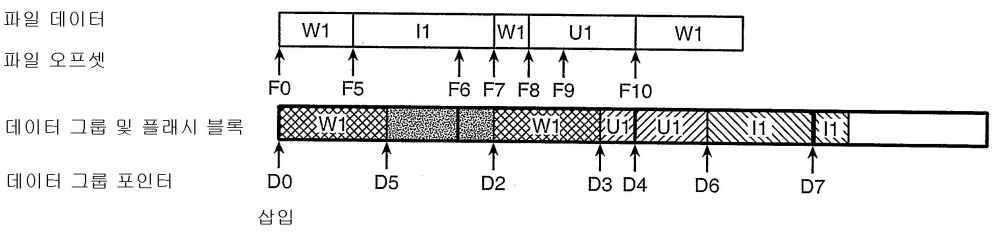
도면12A



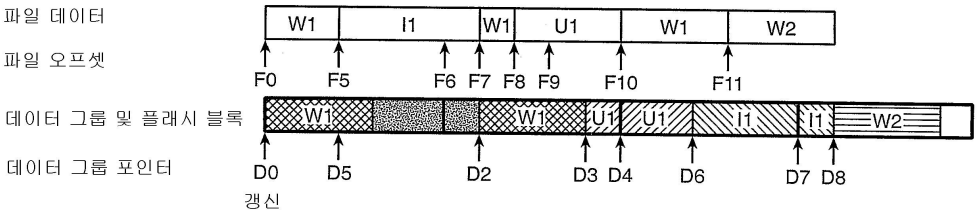
도면12B



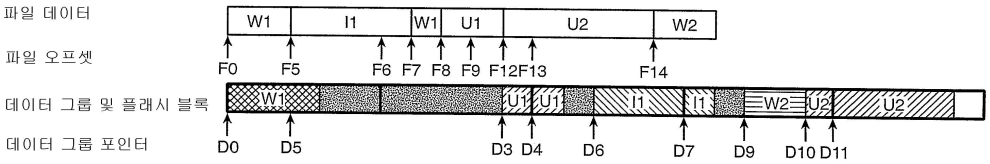
도면12C



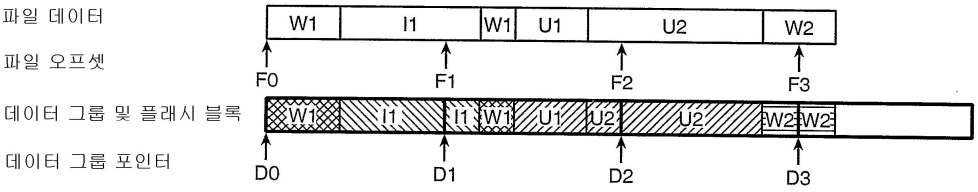
도면12D



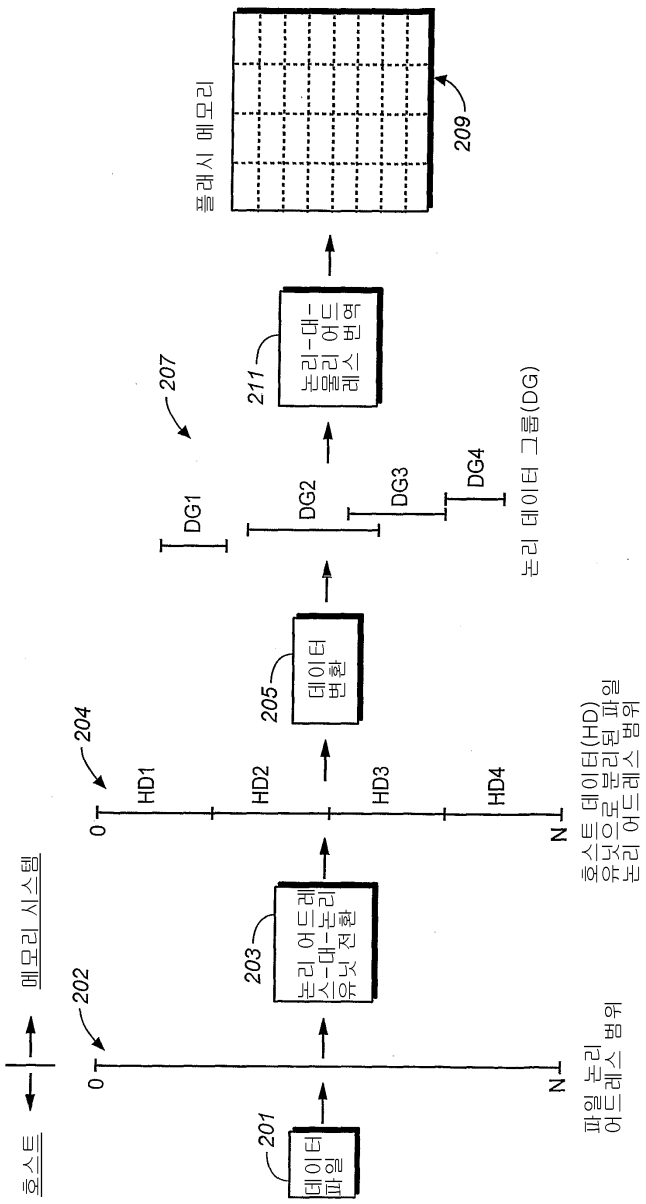
도면12E



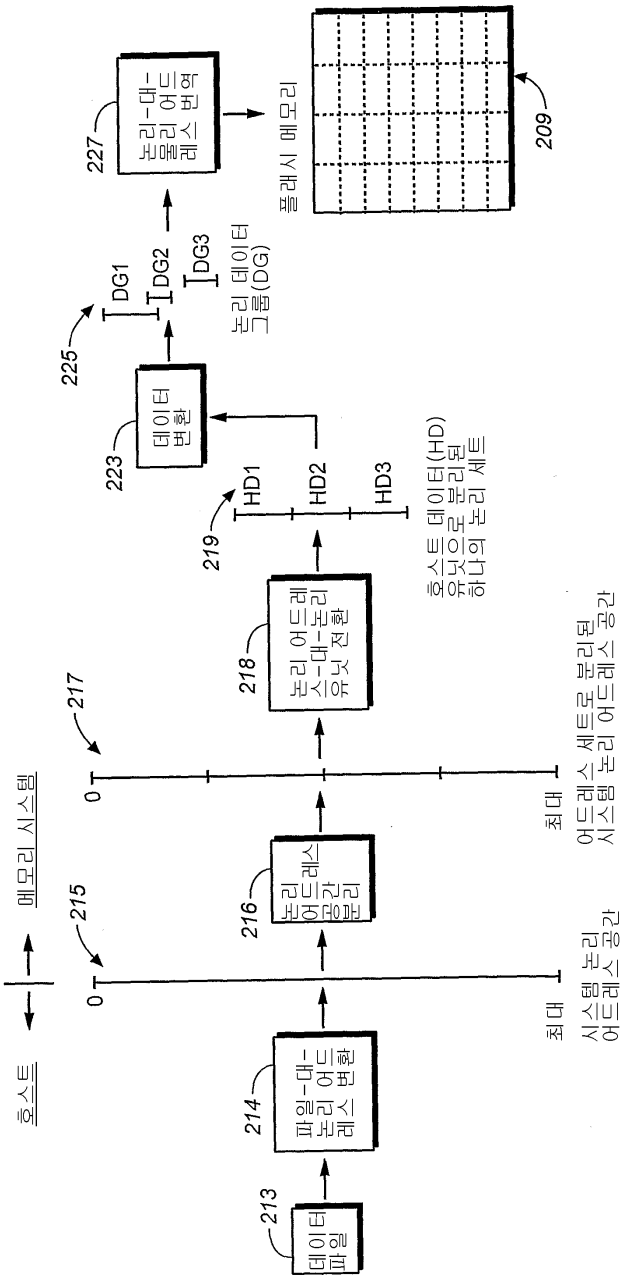
도면13



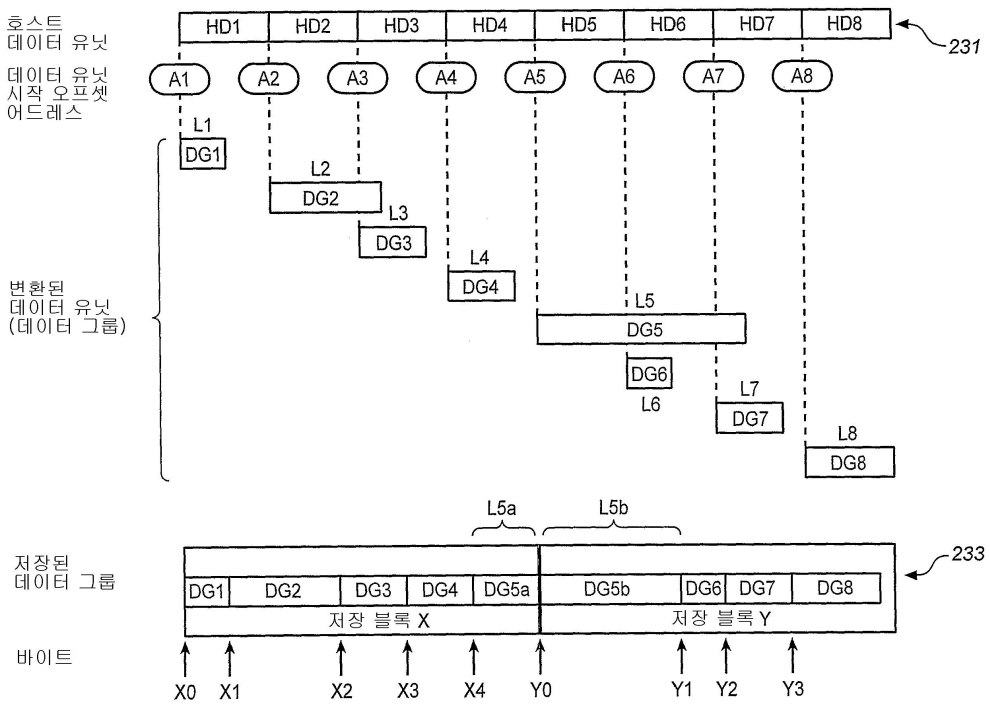
도면14


호스트
메모리 시스템
201
데이터 파일
202
0
N
파일 논리 어드레스 범위
203
논리 어드레스-대-논리 유닛 전환
204
0
N
HD1
HD2
HD3
HD4
호스트 데이터(HD) 유닛으로 분리된 파일 논리 어드레스 범위
205
데이터 변환
207
DG1
DG2
DG3
DG4
논리 데이터 그룹(DG)
211
논리-대-물리 어드레스 변환
209
플래시 메모리

도면15


호스트
메모리 시스템
215
0
최대
시스템 논리
어드레스 공간
214
파일-대-논리 어드레스 변환
213
데이터 파일
216
논리 어드레스 오프셋
217
0
최대
어드레스 세트로 분리된
시스템 논리 어드레스 공간
218
논리 어드레스 승-대-논리 전환
219
HD1
HD2
HD3
호스트 데이터(HD)
유닛으로 분리된
하나의 논리 세트
223
데이터 변환
225
DG1
DG2
DG3
논리 데이터 그룹(DG)
227
논리-대-물리 어드레스 번역
플래시 메모리
209

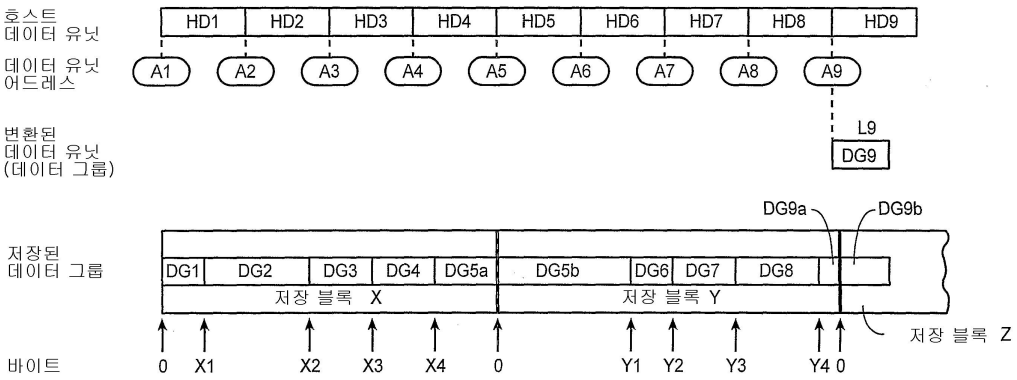
도면16



도면17

오프셋 어드레스 (235)	길이 (237)	포인터 (239)	고정 (241)	링크 (243)	파일의 단부 (245)
A1	L1	X0	1	0	0
A2	L2	X1	1	0	0
A3	L3	X2	1	0	0
A4	L4	X3	1	0	0
A5 (244)	L5	X4	1	1	0
-- (246)	--	Y0	1	1	0
A6	L6	Y1	1	0	0
A7	L7	Y2	1	0	0
A8	L8	Y3	1	0	1

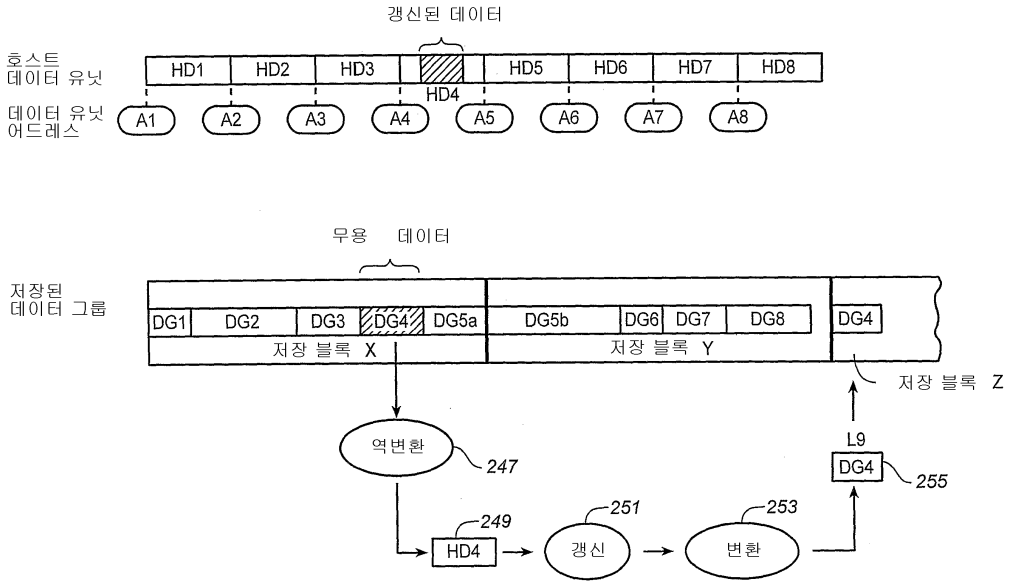
도면18



도면19

오프셋 어드레스	길이	포인터	고정	링크	파일의 단부
A1	L1	X0	1	0	0
A2	L2	X1	1	0	0
A3	L3	X2	1	0	0
A4	L4	X3	1	0	0
A5a	L5a	X4	1	1	0
A5b	L5b	Y0	1	1	0
A6	L6	Y1	1	0	0
A7	L7	Y2	1	0	0
A8	L8	Y3	1	0	0
A9a	L9a	Y4	1	1	0
A9b	L9b	Z0	1	1	1

도면20



도면21

오프셋 어드레스	길이	포인터	고정	링크	파일의 단부
A1	L1	X0	1	0	0
A2	L2	X1	1	0	0
A3	L3	X2	1	0	0
A4	L9	Z0	1	0	0
A5	L5	X4	1	1	0
--	--	Y0	1	1	0
A6	L6	Y1	1	0	0
A7	L7	Y2	1	0	0
A8	L8	Y3	1	0	1