(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0021186
(43) 공개일자 2022년02월22일

(51) 국제특허분류(Int. Cl.)
G06F 21/78 (2013.01) *G06F 11/08* (2006.01)
G06F 21/60 (2013.01) *G06F 21/72* (2013.01)
(52) CPC특허분류
G06F 21/78 (2013.01)
G06F 11/08 (2013.01)
(21) 출원번호 10-2020-0101705
(22) 출원일자 2020년08월13일
심사청구일자 없음

(71) 출원인
에스케이하이닉스 주식회사
경기도 이천시 부발읍 경충대로 2091
(72) 발명자
이종용
경기도 이천시 부발읍 경충대로 2091
(74) 대리인
신성특허법인(유한)

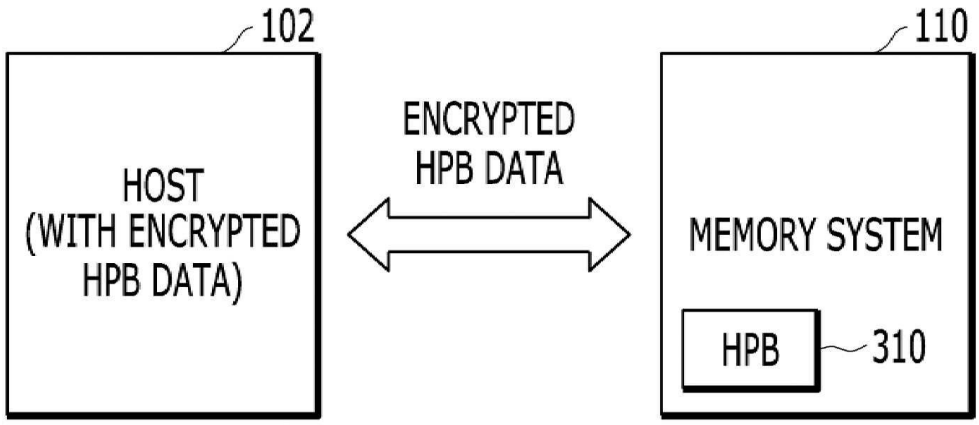
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **데이터 처리 시스템 내 데이터를 공유하는 장치 및 방법**

(57) 요 약

본 기술은 비휘발성 메모리 장치에 저장된 데이터와 관련한 맵 정보에 대해 순환 중복 검사를 수행하고, 맵 정보에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 순환 중복 검사가 수행된 맵 정보와 암호화 데이터에 대해 논리 연산을 통해 암호화된 전송 데이터를 생성하여 호스트에 전송하는 컨트롤러를 제공한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 21/60 (2013.01)
G06F 21/72 (2013.01)

명 세 서

청구범위

청구항 1

비휘발성 메모리 장치에 저장된 데이터와 관련한 맵 정보에 대해 순환 중복 검사를 수행하고, 상기 맵 정보에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 순환 중복 검사가 수행된 상기 맵 정보와 상기 암호화 데이터에 대해 논리 연산을 통해 암호화된 전송 데이터를 생성하여 호스트에 전송하는 컨트롤러.

청구항 2

제1항에 있어서,

상기 맵 정보는 상기 비휘발성 메모리 장치 내 상기 데이터가 저장된 위치를 가리키는 물리 주소를 포함하는,

컨트롤러.

청구항 3

제2항에 있어서,

상기 전송 데이터는 상기 순환 중복 검사의 결과, 상기 논리 주소 및 상기 물리 주소를 포함하고, 상기 결과, 상기 논리 주소를 제외한 상기 물리 주소에 대해 상기 논리 연산을 수행하여 부분 암호화를 수행하는,

컨트롤러.

청구항 4

제1항에 있어서,

상기 논리 주소를 랜덤화한 후, 상기 논리 연산에 대응하는 마스킹 동작을 통해 제1 암호화 데이터를 추출하고, 상기 마스킹 동작에서 제외된 제2 암호화 데이터와 상기 제1 암호화 데이터에 논리 연산을 수행하여 상기 암호화 데이터를 생성하는,

컨트롤러.

청구항 5

제1항에 있어서,

상기 전송 데이터는 상기 맵 정보에 대한 시간 정보를 포함하고, 상기 맵 정보와 상기 시간 정보는 상기 암호화 데이터와의 상기 논리 연산을 통해 암호화되는,

컨트롤러.

청구항 6

제1항에 있어서,

상기 호스트로부터 읽기 요청, 상기 논리 주소 및 상기 암호화된 전송 데이터를 수신하면, 상기 암호화된 전송 데이터를 디코딩하는,

컨트롤러.

청구항 7

제6항에 있어서,

상기 디코딩을 통해, 상기 암호화된 전송 데이터가 정상적으로 수신되었는 가를 판단하고, 상기 암호화된 전송 데이터와 상기 논리 주소의 연관되는 지를 결정하는,

컨트롤러.

청구항 8

호스트로부터 암호화된 전송 데이터를 수신하고, 상기 암호화된 전송 데이터에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 암호화된 전송 데이터와 상기 암호화 데이터에 대해 논리 연산을 수행하여 맵 정보를 얻은 후, 상기 맵 정보에 대해 순환 중복 검사를 수행하여 상기 맵 정보의 유효성을 판단하는,

컨트롤러.

청구항 9

제8항에 있어서,

상기 맵 정보는 비휘발성 메모리 장치 내 저장된 데이터의 위치를 가리키는 물리 주소를 포함하고, 상기 물리 주소를 이용하여 상기 데이터를 읽어 상기 호스트에 출력하는,

컨트롤러.

청구항 10

제9항에 있어서,

상기 전송 데이터는 상기 순환 중복 검사의 결과, 상기 논리 주소 및 상기 물리 주소를 포함하고, 상기 결과, 상기 논리 주소를 제외한 상기 물리 주소에 대해 상기 논리 연산을 통해 부분 암호화된,

컨트롤러.

청구항 11

제8항에 있어서,

상기 논리 주소를 랜덤화한 후, 상기 논리 연산에 대응하는 마스킹 동작을 통해 제1 암호화 데이터를 추출하고, 상기 마스킹 동작에서 제외된 제2 암호화 데이터와 상기 제1 암호화 데이터에 논리 연산을 수행하여 상기 암호화 데이터를 생성하는,

컨트롤러.

청구항 12

제8항에 있어서,

상기 전송 데이터는 상기 맵 정보에 대한 시간 정보를 포함하고, 상기 맵 정보와 상기 시간 정보는 상기 암호화 데이터와의 상기 논리 연산을 통해 암호화된,

컨트롤러.

청구항 13

제8항에 있어서,

상기 순환 중복 검사를 통해, 상기 전송 데이터가 정상적으로 수신되었는 가를 결정하고, 상기 전송 데이터와 상기 논리 주소의 연관되는 지를 결정하는,

컨트롤러.

청구항 14

제8항에 있어서,

비휘발성 메모리 장치에 저장된 데이터와 관련한 맵 정보를 포함하는 상기 암호화된 전송 데이터를 상기 호스트로 전송하는,

컨트롤러.

청구항 15

데이터를 저장할 수 있는 비휘발성 메모리 장치; 및

상기 비휘발성 메모리 장치에 저장된 데이터와 관련한 물리 주소를 포함하는 맵 정보를 암호화하여, 호스트와 암호화된 전송 데이터를 송수신하는 컨트롤러를 포함하고,

상기 컨트롤러는 순환 중복 검사를 통해 상기 물리 주소가 상기 데이터에 대응하는 논리 주소와의 연관성을 판단하는,

메모리 시스템.

청구항 16

제15항에 있어서,

상기 컨트롤러는

상기 맵 정보에 대해 순환 중복 검사를 수행하고, 상기 맵 정보에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 순환 중복 검사가 수행된 상기 맵 정보와 상기 암호화 데이터에 대해 논리 연산을 통해 암호화된 전송 데이터를 생성하는 인코더; 및

상기 암호화된 전송 데이터에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 암호화된 전송 데이터와 상기 암호화 데이터에 대해 논리 연산을 수행하여 맵 정보를 얻은 후, 상기 맵 정보에 대해 상기 순환 중복 검사를 수행하는 디코더

를 포함하는, 메모리 시스템.

청구항 17

제16항에 있어서,

상기 인코더 및 상기 디코더는

상기 논리 주소를 랜덤화한 후, 상기 논리 연산에 대응하는 마스킹 동작을 통해 제1 암호화 데이터를 추출하고,

상기 마스킹 동작에서 제외된 제2 암호화 데이터와 상기 제1 암호화 데이터에 논리 연산을 수행하여 상기 암호화 데이터를 생성하는,

메모리 시스템.

청구항 18

제16항에 있어서,

상기 전송 데이터는 상기 맵 정보에 대한 시간 정보를 포함하고, 상기 맵 정보와 상기 시간 정보는 상기 암호화 데이터와의 상기 논리 연산을 통해 암호화되는,

메모리 시스템.

청구항 19

제15항에 있어서,

상기 전송 데이터는 상기 순환 중복 검사의 결과, 상기 논리 주소 및 상기 물리 주소를 포함하고, 상기 결과, 상기 논리 주소를 제외한 상기 물리 주소에 대해 상기 논리 연산을 수행하여 부분 암호화가 수행된,

를 포함하는, 메모리 시스템.

청구항 20

제15항에 있어서,

상기 컨트롤러는

상기 호스트로부터 읽기 요청, 상기 논리 주소 및 상기 암호화된 전송 데이터를 수신하면, 상기 암호화된 전송 데이터로부터 얻어진 물리 주소를 사용하여 상기 읽기 요청에 대응하는 동작을 수행하는,

메모리 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명의 실시예들은 메모리 시스템을 포함하는 데이터 처리 시스템에 관한 것으로, 보다 구체적으로는 메모리 시스템에서 생성한 데이터를 공유하는 장치 및 방법에 관한 것이다.

배 경 기 술

[0003] 시스템 반도체 장치는 데이터 연산, 제어 등의 정보를 처리하는 역할을 수행하고, 메모리 반도체 장치는 데이터를 저장하는 역할을 수행한다. 메모리 반도체 장치는 데이터를 임시 저장하기 위해 사용되는 휘발성(volatile) 메모리 장치와 데이터를 영구 저장하기 위해 사용되는 비휘발성(non-volatile) 메모리 장치를 포함할 수 있다.

[0004] 자기 디스크와 기계적인 구동장치(예, mechanical arm)을 포함하는 하드 디스크와 비교하면, 비휘발성 메모리 장치는 반도체 공정 기술의 발달로 작은 면적에 많은 데이터를 저장할 수 있을 뿐만 아니라 기계적인 구동장치를 사용할 필요가 없어 데이터를 액세스하는 속도가 빠르고 전력 소모가 적을 수 있다. 이러한 장점을 갖는 비휘발성 메모리 장치를 포함하는 메모리 시스템의 예로서, USB(Universal Serial Bus) 메모리 장치, 다양한 인터페이스를 갖는 메모리 카드, 솔리드 스테이트 드라이브(SSD: Solid State Drive) 등이 있다.

발명의 내용

해결하려는 과제

[0006] 본 발명의 일 실시예는 메모리 시스템의 복잡도 및 성능 저하를 피하고, 메모리 장치의 사용 효율을 개선하여, 메모리 장치에 저장되는 데이터를 안전하게 보호하고 신속하게 처리할 수 있는 메모리 시스템, 데이터 처리 시스템, 혹은 그것의 동작 방법을 제공할 수 있다.

[0007] 또한, 본 발명의 일 실시예는 데이터 처리 시스템 내 메모리 시스템이 호스트 혹은 컴퓨팅 장치에 맵 정보를 전송하고, 호스트 혹은 컴퓨팅 장치가 맵 정보를 포함하는 명령어를 메모리 시스템에 전송하도록 함으로써, 메모리 시스템의 동작 성능을 향상시킬 수 있는 방법 및 장치를 제공할 수 있다.

[0008] 또한, 본 발명의 일 실시예는 메모리 시스템이 호스트에 전송하는 맵 정보를 암호화하여 데이터 처리 시스템의 보안을 강화할 수 있는 방법과 장치를 제공할 수 있다.

[0009] 또한, 본 발명의 일 실시예에서는 메모리 시스템이 호스트에 맵 정보가 포함된 데이터를 암호화하여 전송한 후, 호스트로부터 암호화된 데이터를 수신하면 암호화된 데이터를 디코딩하는 과정에서 맵 정보의 오류를 확인할 수 있는 방법과 장치를 제공할 수 있다.

[0010] 또한, 본 발명의 일 실시예에서는 메모리 시스템이 호스트와 암호화된 데이터를 주고받는 과정에서 발생하는 오버헤드를 줄여, 메모리 시스템의 입출력 성능을 개선하고 데이터 처리 시스템의 성능을 향상시킬 수 있는 방법과 장치를 제공할 수 있다.

[0011] 본 발명에서 이루고자 하는 기술적 과제들은 이상에서 언급한 기술적 과제들로 제한되지 않으며, 언급하지 않은 또 다른 기술적 과제들은 아래의 기재로부터 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0013] 본 발명의 실시 예들은 메모리 시스템, 메모리 시스템에 포함되는 컨트롤러 혹은 메모리 시스템을 포함하는 데이터 처리 장치를 제공할 수 있다.

[0014] 본 발명의 실시 예에 따른 컨트롤러는 비휘발성 메모리 장치에 저장된 데이터와 관련한 맵 정보에 대해 순환 중복 검사를 수행하고, 상기 맵 정보에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 순환 중복 검사가 수행된 상기 맵 정보와 상기 암호화 데이터에 대해 논리 연산을 통해 암호화된 전송 데이터를 생성하여 호스트에 전송할 수 있다.

[0015] 또한, 상기 맵 정보는 상기 비휘발성 메모리 장치 내 상기 데이터가 저장된 위치를 가리키는 물리 주소를 포함할 수 있다.

[0016] 또한, 상기 전송 데이터는 상기 순환 중복 검사의 결과, 상기 논리 주소 및 상기 물리 주소를 포함하고, 상기 결과, 상기 논리 주소를 제외한 상기 물리 주소에 대해 상기 논리 연산을 수행하여 부분 암호화를 수행할 수 있다.

[0017] 또한, 컨트롤러는 상기 논리 주소를 랜덤화한 후, 상기 논리 연산에 대응하는 마스킹 동작을 통해 제1 암호화 데이터를 추출하고, 상기 마스킹 동작에서 제외된 제2 암호화 데이터와 상기 제1 암호화 데이터에 논리 연산을 수행하여 상기 암호화 데이터를 생성할 수 있다.

[0018] 또한, 상기 전송 데이터는 상기 맵 정보에 대한 시간 정보를 포함하고, 상기 맵 정보와 상기 시간 정보는 상기 암호화 데이터와의 상기 논리 연산을 통해 암호화될 수 있다.

[0019] 또한, 상기 호스트로부터 읽기 요청, 상기 논리 주소 및 상기 암호화된 전송 데이터를 수신하면, 상기 암호화된 전송 데이터를 디코딩할 수 있다.

[0020] 또한, 컨트롤러는, 상기 디코딩을 통해, 상기 암호화된 전송 데이터가 정상적으로 수신되었는 가를 판단하고, 상기 암호화된 전송 데이터와 상기 논리 주소의 연관되는 지를 결정할 수 있다.

[0021] 본 발명의 다른 실시예에 따른 컨트롤러는 호스트로부터 암호화된 전송 데이터를 수신하고, 상기 암호화된 전송

데이터에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 암호화된 전송 데이터와 상기 암호화 데이터에 대해 논리 연산을 수행하여 맵 정보를 얻은 후, 상기 맵 정보에 대해 순환 중복 검사를 수행하여 상기 맵 정보의 유효성을 판단할 수 있다.

[0022] 또한, 상기 맵 정보는 비휘발성 메모리 장치 내 저장된 데이터의 위치를 가리키는 물리 주소를 포함하고, 컨트롤러는 상기 물리 주소를 이용하여 상기 데이터를 읽어 상기 호스트에 출력할 수 있다.

[0023] 또한, 상기 전송 데이터는 상기 순환 중복 검사의 결과, 상기 논리 주소 및 상기 물리 주소를 포함하고, 상기 결과, 상기 논리 주소를 제외한 상기 물리 주소에 대해 상기 논리 연산을 통해 부분 암호화될 수 있다.

[0024] 또한, 컨트롤러는 상기 논리 주소를 랜덤화한 후, 상기 논리 연산에 대응하는 마스킹 동작을 통해 제1 암호화 데이터를 추출하고, 상기 마스킹 동작에서 제외된 제2 암호화 데이터와 상기 제1 암호화 데이터에 논리 연산을 수행하여 상기 암호화 데이터를 생성할 수 있다.

[0025] 또한, 상기 전송 데이터는 상기 맵 정보에 대한 시간 정보를 포함하고, 상기 맵 정보와 상기 시간 정보는 상기 암호화 데이터와의 상기 논리 연산을 통해 암호화될 수 있다.

[0026] 또한, 컨트롤러는, 상기 순환 중복 검사를 통해, 상기 전송 데이터가 정상적으로 수신되었는 가를 결정하고, 상기 전송 데이터와 상기 논리 주소의 연관되는 지를 결정할 수 있다.

[0027] 또한, 컨트롤러는 비휘발성 메모리 장치에 저장된 데이터와 관련한 맵 정보를 포함하는 상기 암호화된 전송 데이터를 상기 호스트로 전송할 수 있다.

[0028] 본 발명의 다른 실시예에 따른 메모리 시스템은 데이터를 저장할 수 있는 비휘발성 메모리 장치; 및 상기 비휘발성 메모리 장치에 저장된 데이터와 관련한 물리 주소를 포함하는 맵 정보를 암호화하여, 호스트와 암호화된 전송 데이터를 송수신하는 컨트롤러를 포함하고, 상기 컨트롤러는 순환 중복 검사를 통해 상기 물리 주소가 상기 데이터에 대응하는 논리 주소와의 연관성을 판단할 수 있다.

[0029] 또한, 상기 컨트롤러는 상기 맵 정보에 대해 순환 중복 검사를 수행하고, 상기 맵 정보에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 순환 중복 검사가 수행된 상기 맵 정보와 상기 암호화 데이터에 대해 논리 연산을 통해 암호화된 전송 데이터를 생성하는 인코더; 및 상기 암호화된 전송 데이터에 대응하는 논리 주소에 기반하여 암호화 데이터를 생성한 후, 상기 암호화된 전송 데이터와 상기 암호화 데이터에 대해 논리 연산을 수행하여 맵 정보를 얻은 후, 상기 맵 정보에 대해 상기 순환 중복 검사를 수행하는 디코더를 포함할 수 있다.

[0030] 또한, 상기 인코더 및 상기 디코더는 상기 논리 주소를 랜덤화한 후, 상기 논리 연산에 대응하는 마스킹 동작을 통해 제1 암호화 데이터를 추출하고, 상기 마스킹 동작에서 제외된 제2 암호화 데이터와 상기 제1 암호화 데이터에 논리 연산을 수행하여 상기 암호화 데이터를 생성할 수 있다.

[0031] 또한, 상기 전송 데이터는 상기 맵 정보에 대한 시간 정보를 포함하고, 상기 맵 정보와 상기 시간 정보는 상기 암호화 데이터와의 상기 논리 연산을 통해 암호화될 수 있다.

[0032] 또한, 상기 전송 데이터는 상기 순환 중복 검사의 결과, 상기 논리 주소 및 상기 물리 주소를 포함하고, 상기 결과, 상기 논리 주소를 제외한 상기 물리 주소에 대해 상기 논리 연산을 수행하여 부분 암호화가 수행될 수 있다.

[0033] 또한, 상기 컨트롤러는 상기 호스트로부터 읽기 요청, 상기 논리 주소 및 상기 암호화된 전송 데이터를 수신하면, 상기 암호화된 전송 데이터로부터 얻어진 물리 주소를 사용하여 상기 읽기 요청에 대응하는 동작을 수행할 수 있다.

[0034] 상기 본 발명의 양태들은 본 발명의 바람직한 실시예들 중 일부에 불과하며, 본원 발명의 기술적 특징들이 반영된 다양한 실시예들이 당해 기술분야의 통상적인 지식을 가진 자에 의해 이하 상술할 본 발명의 상세한 설명을 기반으로 도출되고 이해될 수 있다.

발명의 효과

[0036] 본 발명에 따른 장치에 대한 효과에 대해 설명하면 다음과 같다.

[0037] 본 발명의 일 실시 예에 따른 메모리 시스템 혹은 데이터 처리 시스템은, 데이터 처리 과정에서 보안을 강화할 수 있다.

[0038] 또한, 본 발명의 일 실시 예에 따른 메모리 시스템은 데이터 처리 과정에서 데이터의 에러를 용이하게 확인하도록 하여 오버헤드(overheads)를 줄이고, 메모리 시스템의 동작 성능 혹은 입출력 성능을 개선할 수 있다.

[0039] 본 발명에서 얻을 수 있는 효과는 이상에서 언급한 효과들로 제한되지 않으며 언급하지 않은 또 다른 효과들은 아래의 기재로부터 본 발명이 속하는 분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0041] 도 1은 본 발명의 일 실시예에 따른 데이터 처리 시스템을 설명한다.

도 2는 본 발명의 일 실시예에 따른 데이터 처리 시스템에서 호스트와 메모리 시스템의 구성을 설명한다.

도 3은 본 발명의 다른 실시예에 따른 데이터 처리 시스템을 설명한다.

도 4는 본 발명의 다른 실시예에 따른 메모리 시스템을 설명한다.

도 5는 본 발명의 일 실시예에 따른 데이터 처리 시스템에서 호스트와 메모리 시스템의 읽기 동작을 설명한다.

도 6은 본 발명의 일 실시예에 따른 호스트와 메모리 시스템의 제1 동작을 설명한다.

도 7은 본 발명의 일 실시예에 따른 호스트와 메모리 시스템의 제2 동작을 설명한다.

도 8은 본 발명의 실시 예에 따라 컨트롤러와 호스트가 송수신하는 데이터를 설명한다.

도 9는 본 발명의 일 실시예에 따른 메모리 시스템에서 데이터 인코딩 동작을 설명한다.

도 10은 본 발명의 일 실시예에 따른 메모리 시스템에서 데이터 디코딩 동작을 설명한다.

도 11은 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 인코딩 동작을 설명한다.

도 12는 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 인코딩 방법을 설명한다.

도 13은 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 디코딩 동작을 설명한다.

도 14는 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 디코딩 방법을 설명한다.

발명을 실시하기 위한 구체적인 내용

[0042] 이하, 본 발명에 따른 바람직한 실시 예를 첨부한 도면을 참조하여 상세히 설명한다. 하기의 설명에서는 본 발명에 따른 동작을 이해하는데 필요한 부분만이 설명되며 그 이외 부분의 설명은 본 발명의 요지를 흩뜨리지 않도록 생략될 것이라는 것을 유의하여야 한다.

[0043] 이하, 도면들을 참조하여 본 발명의 실시 예들에 대해서 보다 구체적으로 설명하기로 한다.

[0044] 도 1은 본 발명의 일 실시예에 따른 데이터 처리 시스템을 설명한다.

[0045] 도 1을 참조하면, 데이터 처리 시스템은 메모리 시스템(110) 및 호스트(102)를 포함할 수 있다.

[0046] 실시예에 따라, 호스트(102)는 사용자의 요구에 대응하는 동작을 수행하는 장치이고, 메모리 시스템(110)은 호스트(102)의 동작 중 발생하는 정보를 일시적 혹은 영구적으로 저장하기 위해 사용될 수 있다. 메모리 시스템(110)과 호스트(102)는 데이터, 신호, 명령, 요청 등을 기 설정된 방식(예, 프로토콜)으로 송수신할 수 있도록 연결될 수 있다. 메모리 시스템(110)과 호스트(102)에 대해서는 도 2 내지 도 4를 참조하여 구체적으로 설명한다.

[0047] 호스트(102)와 메모리 시스템(110)은 서로 다른 주소 체계(address scheme)을 가질 수 있다. 이로 인하여, 메모리 시스템(110) 내 비휘발성 메모리셀을 포함하는 저장 공간에 호스트(102)가 요구한 데이터를 저장하기 위해서, 메모리 시스템(110)은 호스트(102)가 사용하는 파일 시스템과 비휘발성 메모리셀을 포함하는 저장 공간을 연결시키는 주소 변환(Address translation)을 수행할 수 있다. 예를 들면, 호스트(102)가 사용하는 파일 시스템에 따른 데이터의 주소를 논리 주소 혹은 논리 블록 주소라고 부를 수 있고, 메모리 시스템(110) 내 비휘발

성 메모리셀을 포함하는 저장 공간에서 데이터의 위치는 물리 주소 혹은 물리 블록 주소라고 부를 수 있다. 호스트(102)가 읽기 요청과 함께 논리 주소를 메모리 시스템(110)에 전달하는 경우, 메모리 시스템(110)은 논리 주소에 대응하는 물리 주소를 탐색한 후 탐색된 물리 주소에 저장된 데이터를 호스트(102)에 출력할 수 있다. 이러한 과정 중 메모리 시스템(110)이 호스트(102)가 전달한 논리 주소에 대응하는 물리 주소를 탐색하는 과정에서 주소 변환(Address translation)이 수행될 수 있다.

[0048] 호스트(102)에서 전달된 요청에 대응하여 메모리 시스템(110)은 데이터 입출력 동작을 수행할 수 있다. 예를 들어, 메모리 시스템(110)이 외부 장치에서 전달된 읽기 요청에 대응하여 리드 동작을 수행하면 복수의 비휘발성 메모리 셀에 저장된 데이터를 호스트(102)로 전달할 수 있다. 리드 동작을 위해, 메모리 시스템(110)은 맵핑 정보를 바탕으로 호스트(102)에서 전달된 논리 주소를 주소 변환한 후, 물리 주소에 대응하는 위치에 저장된 데이터를 액세스할 수 있다. 또한, 메모리 시스템(110)은 호스트(102)에서 전달된 쓰기 요청과 함께 전달된 데이터를 복수의 비휘발성 메모리 셀에 저장할 수 있다. 메모리 시스템(110)은 데이터를 복수의 비휘발성 메모리 셀에 저장한 후, 데이터에 대응하는 논리 주소와 물리 주소를 연관시키는 맵핑 정보를 생성하거나 갱신할 수 있다.

[0049] 메모리 시스템(110)은 데이터 입출력 성능을 향상시키기 위해 호스트 성능 부스터(Host Performance Booster(HPB), 310)를 포함할 수 있다. 호스트 성능 부스터(HPB, 310)는 호스트(102)에서 사용되는 논리 주소(logical address)와 메모리 시스템(110)에서 사용되는 물리 주소(physical address)를 연관시킨 맵핑 정보를 호스트(102)에 전송할 수 있다. 호스트(102)가 맵핑 정보를 저장하고 있다면, 호스트(102)가 읽기 요청을 메모리 시스템(110)에 전송하는 경우 호스트(102)가 맵핑 정보를 참조하여 메모리 시스템(110)에 읽기 요청과 함께 메모리 시스템(110)에서 사용되는 물리 주소를 전달할 수 있다. 메모리 시스템(110)이 읽기 요청과 함께 전달된 물리 주소를 사용하고 메모리 시스템(110)이 주소 변환을 수행하지 않는다면, 메모리 시스템(110)은 호스트(102)의 읽기 요청에 대응하는 데이터를 더 빨리 출력할 수 있다. 맵핑 정보는 메모리 시스템(110)이 독립적으로 생성하고 내부 동작을 통해 사용되는 것이었으나, 호스트(102)와 공유하는 경우 메모리 시스템(110)의 데이터 입출력 성능의 향상을 가져올 수 있다.

[0050] 메모리 시스템(110)의 내부에서 사용되는 맵핑 정보가 외부 장치인 호스트(102)와 공유되면, 정보의 공유에 따라 보안에 취약해지거나 호스트(102)의 동작에 의해 공유되는 정보의 왜곡, 변형이 발생할 가능성이 있다. 또한, 데이터 암호화는 호스트(102)가 맵핑 정보를 사용하는 것을 방지할 수도 있다. 따라서, 호스트 성능 부스터(HPB)는 호스트(102)에 전송하는 맵핑 정보를 암호화하여 호스트(102)에 전송할 수 있다.

[0051] 한편, 메모리 시스템(110)이 암호화된 전송 데이터(Encrypted HPB data)를 호스트(102)에 전송하면, 호스트(102)가 암호화된 전송 데이터를 사용할 수 있어야 한다. 예를 들어 맵핑 정보는 논리 주소 및 물리 주소를 포함할 수 있다. 맵핑 정보의 논리 주소 및 물리 주소를 모두 암호화하면, 호스트(102)는 암호화된 맵핑 정보를 디코딩(즉, 암호화를 해제)하여 논리 주소 및 물리 주소를 인식할 수 있어야 한다. 호스트(102)와 메모리 시스템(110)이 암호화된 맵핑 정보를 송수신하고 암호화된 맵핑 정보를 디코딩하는 경우, 호스트(102)는 맵핑 정보를 인코딩(암호화) 및 디코딩(암호화 해제)할 필요가 있다. 하지만, 호스트(102)가 자신이 사용할 필요가 없는 맵핑 정보, 보다 구체적으로는 물리 주소를 사용하기 위해 인코딩 및 디코딩을 수행하는 것은 메모리 시스템(110)과의 데이터 통신에서 오버헤드(overheads)를 증가시킬 수 있다.

[0052] 본 발명의 일 실시예에서는 메모리 시스템(110)이 맵핑 정보의 전체가 아닌 일부만을 암호화함으로써, 호스트(102)가 암호화된 전송 데이터를 인코딩 및 디코딩하는 동작을 수행할 필요가 없도록 할 수 있다. 예를 들어, 메모리 시스템(110)이 호스트(102)에 맵핑 정보를 전송하는 경우, 맵핑 정보의 논리 주소를 암호화하지 않고, 맵핑 정보의 물리 주소를 암호화할 수 있다. 즉, 메모리 시스템(110)은 맵핑 정보를 부분적으로 암호화하여 암호화된 전송 데이터를 호스트(102)에 전송할 수 있다. 호스트(102)로부터 암호화된 전송 데이터를 수신하면, 메모리 시스템(110)은 암호화된 전송 데이터를 디코딩할 수 있다. 이 경우, 호스트(102)는 맵핑 정보를 인코딩/디코딩하기 위한 모듈, 회로 등을 포함할 필요가 없고, 메모리 시스템(110)이 맵핑 정보의 인코딩/디코딩을 위한 모듈을 포함하여 호스트(102)와의 데이터 통신의 보안을 강화하고, 호스트(102)에 의한 정보의 왜곡, 변형을 감지할 수 있다.

[0053] 논리 주소가 암호화되지 않았기 때문에, 호스트(102)는 메모리 시스템(110)에서 전송된 암호화된 전송 데이터가 특정 논리 주소에 연관되어 있음을 인지할 수 있다. 따라서, 메모리 시스템(110)로부터 암호화된 전송 데이터를 수신하더라도, 호스트(102)는 해당 암호화된 전송 데이터가 특정 논리 주소에 연관되어 있어 호스트(102)의 저장 공간에 해당 정보를 저장할 수 있다. 호스트(102)가 특정 논리 주소에 연관된 암호화된 전송 데이터를 저장하고 있는 경우, 호스트(102)는 해당 논리 주소와 함께 암호화된 전송 데이터를 메모리 시스템(110)에 전달할

수 있다.

[0054] 전술한 바와 같이, 맵핑 정보의 전체가 아닌 부분적으로 암호화하여, 메모리 시스템(110)의 데이터 입출력 성능을 향상시키기 위해 호스트(102)가 논리 주소에 대응하는 암호화된 전송 데이터를 메모리 시스템(110)에 전송할 수 있도록 하면서도, 호스트(102)에 의해 암호화된 전송 데이터가 사용되는 것을 방지하고, 메모리 시스템(110)은 호스트(102)에 의한 왜곡, 변형의 발생을 인지할 수 있다.

[0055] 도 2는 본 발명의 일 실시예에 따른 데이터 처리 시스템에서 호스트(102)와 메모리 시스템(110)의 구성을 설명한다.

[0056] 도 2를 참조하면, 호스트(102)는 프로세서(104), 메모리(106) 및 호스트 컨트롤러 인터페이스(108)를 포함할 수 있다. 메모리 시스템(110)은 컨트롤러(130) 및 메모리 장치(150)를 포함할 수 있다. 컨트롤러(130)는 호스트 인터페이스(132), 플래시 변환 계층(FTL, 240), 메모리 인터페이스(142) 및 메모리(144)를 포함할 수 있다. 실시예에 따라, 도 1에서 설명한 호스트 성능 부스터(HPB, 310)는 호스트 인터페이스(132) 혹은 플래시 변환 계층(FTL, 240)에 포함될 수 있다.

[0057] 도 2에서 설명하는 컨트롤러(130) 및 메모리 장치(150)는 도 3 내지 도 4에서 설명하는 컨트롤러(130) 및 메모리 장치(150)와 유사할 수 있다. 도 2에서 설명하는 컨트롤러(130) 및 메모리 장치(150)와 도 3 내지 도 4에서 후술하는 컨트롤러(130) 및 메모리 장치(150)에서 기술적으로 구분될 수 있는 내용을 중심으로 설명한다.

[0058] 호스트(102)는 호스트(102)와 연동하는 메모리 시스템(110)에 비하여 고성능의 프로세서(104) 및 대용량의 메모리(106)를 포함할 수 있다. 호스트(102) 내 프로세서(104) 및 메모리(106)는 메모리 시스템(110)과 달리 공간적 제약이 적고, 필요에 따라 프로세서(104) 및 메모리(106)의 하드웨어적인 업그레이드(upgrade)가 가능한 장점이 있다. 따라서, 메모리 시스템(110)이 동작 효율성을 높이기 위해, 호스트(102)가 가지는 자원(resource)을 활용할 수 있다.

[0059] 메모리 시스템(110)이 저장할 수 있는 데이터의 양이 증가하면서, 메모리 시스템(110)에 저장되는 데이터에 대응하는 메타 데이터의 양도 증가한다. 메모리 시스템(110) 내 컨트롤러(130)가 메타 데이터를 로딩(loading)할 수 있는 메모리(144)의 공간은 제한적이므로, 메타 데이터의 양이 증가는 컨트롤러(130)의 동작에 부담을 준다. 예를 들어, 컨트롤러(130)가 메타 데이터를 위해 할당할 수 있는 메모리(144) 내 공간의 제약으로 인해, 메타 데이터의 전부가 아닌 일부를 로딩(loading)할 수 있다. 만약 호스트(102)가 액세스하고자 하는 위치가 일부 로딩된 메타 데이터에 포함되지 않은 경우, 컨트롤러(130)는 로딩(loading)한 메타 데이터의 일부가 갱신되었다면 메모리 장치(150)에 다시 저장해야 하고, 호스트(102)가 액세스하고자 하는 위치에 대응하는 메타 데이터를 메모리 장치(150)로부터 읽어야 한다. 이러한 동작들은 컨트롤러(130)가 호스트(102)가 요구하는 읽기 혹은 쓰기 동작을 수행하기 위해 필요적으로 수행될 수 있으며, 메모리 시스템(110)의 동작 성능을 저하시킬 수 있다.

[0060] 실시예에 따라, 컨트롤러(130)가 사용할 수 있는 메모리(144)에 비하여, 호스트(102)가 포함하는 메모리(106)의 저장 공간은 수십배에서 수천배 클 수 있다. 따라서, 메모리 시스템(110)은 컨트롤러(130)가 사용하는 메타 데이터(166)를 호스트(102) 내 메모리(106)에 전달하여, 호스트(102) 내 메모리(106)가 메모리 시스템(110)이 수행하는 주소변환과정을 위한 캐시(cashe) 메모리로 사용되도록 할 수 있다. 여기서, 메타 데이터(166)는 도 1에서 설명한 암호화된 전송 데이터(Encrypted HPB data)를 포함할 수 있다. 이 경우, 호스트(102)는 메모리 시스템(110)에 명령과 함께 논리 주소를 전달하지 않고, 메모리(106)에 저장된 메타 데이터(166)를 바탕으로 논리 주소에 대응하는 암호화된 전송 데이터(Encrypted HPB data)를 명령과 함께 메모리 시스템(110)에 전달할 수 있다. 메모리 시스템(110)은 논리 주소를 물리 주소로 변환하는 과정을 생략할 수 있고, 전달되는 암호화된 전송 데이터(Encrypted HPB data)에 포함된 물리 주소를 바탕으로 메모리 장치(150)에 액세스할 수 있다. 이 경우, 전술했던 컨트롤러(130)가 메모리(144)를 사용하면서 발생하는 동작 부담을 해소할 수 있어, 메모리 시스템(110)의 동작 효율성이 매우 높아질 수 있다.

[0061] 한편, 메모리 시스템(110)이 메타 데이터(166)를 호스트(102)에 전송하더라도, 메모리 시스템(110)이 메타 데이터(166)에 기준이 되는 정보의 관리(즉, 메타 데이터의 갱신, 삭제, 생성 등)를 수행할 수 있다. 메모리 시스템(110) 내 컨트롤러(130)는 메모리 장치(150)의 동작 상태에 따라 가비지 컬렉션, 웨어 레벨링 등의 백그라운 동작을 수행할 수 있고, 호스트(102)에서 전달된 데이터를 메모리 장치(150) 내 저장하는 물리적 위치(물리 주소)를 결정할 수 있기 때문에, 메모리 장치(150) 내 데이터의 물리적인 주소는 변경될 수 있다. 따라서, 메타 데이터(166)의 기준이 되는 정보(source)의 관리는 메모리 시스템(110)이 맡을 수 있다.

[0062] 즉, 메모리 시스템(110)은 이 메타 데이터(166)를 관리하는 과정에서, 호스트(102)에 전달한 메타 데이터(166)

를 수정, 갱신할 필요가 있다고 판단되면, 메모리 시스템(110)은 호스트(102)에 메타 데이터(166)의 갱신을 요청할 수 있다. 호스트(102)는 메모리 시스템(110)의 요청에 대응하여, 메모리(106) 내 저장된 메타 데이터(166)를 갱신할 수 있다. 이를 통해, 호스트(102) 내 메모리(106)에 저장된 메타 데이터(166)가 최근 상태를 유지할 수 있으며, 호스트 컨트롤러 인터페이스(108)가 메모리(106)에 저장된 메타 데이터(166)를 사용하여 메모리 시스템(110)에 전달할 주소값을 변환하더라도 동작에 문제가 발생하지 않을 수 있다.

[0063] 한편, 메모리(106)에 저장되는 메타 데이터(166)는 논리 주소(logical address)에 대응하는 암호화된 전송 데이터(Encrypted HPB data)를 포함할 수 있다. 호스트(102)는 메모리 시스템(110)에 전달하는 목적 혹은 동작이 아닌 다른 목적 혹은 다른 동작을 위해 암호화된 전송 데이터(Encrypted HPB data)를 사용하지 않을 수 있다.

[0064] 한편, 실시예에 따라, 논리 주소(logical address)와 물리 주소(physical address)를 대응시키는 메타 데이터에는 논리 주소에 대응하는 물리 주소를 확인하기 위한 제1 맵핑 정보와 물리 주소에 대응하는 논리 주소를 확인하기 위한 제2 맵핑 정보가 포함될 수 있다. 이 중, 메모리(106)에 저장되는 메타 데이터(166)는 제1 맵핑 정보를 포함할 수 있다. 제2 맵핑 정보는 주로 메모리 시스템(110)의 내부 동작을 위해 사용되며, 호스트(102)가 데이터를 메모리 시스템(110)에 저장하거나 특정 논리 주소에 대응하는 데이터를 메모리 시스템(110)으로부터 읽기 위한 동작에는 사용되지 않을 수 있다. 실시예에 따라, 제2 맵핑 정보는 메모리 시스템(110)이 호스트(102)에 전송하지 않을 수 있다.

[0065] 한편, 메모리 시스템(110) 내 컨트롤러(130)는 제1 맵핑 정보 혹은 제2 맵핑 정보를 관리(생성, 삭제, 갱신 등)하면서, 제1 맵핑 정보 혹은 제2 맵핑 정보를 메모리 장치(150)에 저장할 수 있다. 호스트(102) 내 메모리(106)는 휘발성 메모리 장치이므로, 호스트(102) 및 메모리 시스템(110)에 전원 공급이 중단되는 등의 이벤트가 발생하는 경우에 호스트(102) 내 메모리(106)에 저장된 메타 데이터(166)는 사라질 수 있다. 따라서, 메모리 시스템(110) 내 컨트롤러(130)는 호스트(102) 내 메모리(106)에 저장된 메타 데이터(166)를 최근 상태로 유지시킬 뿐만 아니라 최근 상태의 제1 맵핑 정보 혹은 제2 맵핑 정보를 메모리 장치(150)에 저장할 수 있다.

[0066] 도 3는 본 발명의 다른 실시예에 따른 데이터 처리 시스템을 설명한다.

[0067] 도 3를 참조하면, 데이터 처리 시스템(100)은 호스트(102) 및 메모리 시스템(110)을 포함한다. 예를 들면, 호스트(102)와 메모리 시스템(110)은 데이터 버스(data bus), 호스트 케이블(host cable) 등과 같은 데이터 전달 수단을 통해 연결되어, 데이터를 송수신할 수 있다.

[0068] 호스트(102)는 전자 장치, 예컨대 휴대폰, MP3 플레이어, 랩탑 컴퓨터 등과 같은 휴대용 전자 장치들, 또는 데스크탑 컴퓨터, 게임기, TV, 프로젝터 등과 같은 비휴대용 전자 장치들을 포함할 수 있다. 예를 들어, 호스트(102)는 컴퓨팅 장치 혹은 유무선 전자 장치들을 포함할 수 있다.

[0069] 또한, 호스트(102)는, 적어도 하나의 운영 시스템(OS: operating system)를 포함하며, 운영 시스템은, 호스트(102)의 기능 및 동작을 전반적으로 관리 및 제어하고, 데이터 처리 시스템(100) 또는 메모리 시스템(110)을 사용하는 사용자와 호스트(102) 간에 상호 동작을 제공한다. 여기서, 운영 시스템은, 사용자의 사용 목적 및 용도에 상응한 기능 및 동작을 지원하며, 예컨대, 호스트(102)의 이동성(mobility)에 따라 일반 운영 시스템과 모바일 운용 시스템으로 구분할 수 있다. 또한, 운영 시스템에서의 일반 운영 시스템 시스템은, 사용자의 사용 환경에 따라 개인용 운영 시스템과 기업용 운영 시스템으로 구분할 수 있으며, 일 예로, 개인용 운영 시스템은 일반 사용자를 위한 서비스 제공 기능을 지원하도록 특성화된 시스템을 포함할 수 있고, 기업용 운영 시스템은 고성능을 확보 및 지원하도록 특성화된 시스템을 포함할 수 있다. 한편, 호스트(102)는 복수의 운영 시스템들을 포함할 수 있으며, 또한 사용자 요청(user request)에 상응한 메모리 시스템(110)과의 동작 수행을 위해 운영 시스템을 실행한다. 호스트(102)는 사용자 요청에 해당하는 복수의 커맨드들을 메모리 시스템(110)으로 전송하며, 메모리 시스템(110)에서는 복수의 커맨드들에 해당하는 동작들(즉, 사용자 요청에 상응하는 동작들)을 수행한다.

[0070] 메모리 시스템(110) 내 컨트롤러(130)는 호스트(102)로부터의 요청에 응답하여 메모리 장치(150)를 제어할 수 있다. 예를 들면, 컨트롤러(130)는 리드 동작을 수행하여 메모리 장치(150)로부터 리드된 데이터를 호스트(102)로 제공할 수 있고, 쓰기 동작(프로그램 동작)을 수행하여 호스트(102)로부터 제공된 데이터를 메모리 장치(150)에 저장할 수 있다. 이러한 데이터 입출력 동작을 수행하기 위해, 컨트롤러(130)는 리드, 프로그램(program), 이레이즈(erase) 등의 동작을 제어할 수 있다.

[0071] 실시예에 따라, 컨트롤러(130)는 호스트 인터페이스(132), 프로세서(134), 에러 정정부(138), 파워 관리 유닛(Power Management Unit, PMU)(140), 메모리 인터페이스(142), 및 메모리(144)를 포함할 수 있다. 도 3에서 설

명한 컨트롤러(130)에 포함된 구성 요소들은 메모리 시스템(110)의 구현 형태, 동작 성능 등에 따라 달라질 수 있다. 예를 들면, 메모리 시스템(110)은 솔리드 스테이트 드라이브(SSD: Solid State Drive), MMC, eMMC(embedded MMC), RS-MMC(Reduced Size MMC), micro-MMC 형태의 멀티 미디어 카드(MMC: Multi Media Card), SD, mini-SD, micro-SD 형태의 시큐어 디지털(SD: Secure Digital) 카드, USB(Universal Storage Bus) 저장 장치, UFS(Universal Flash Storage) 장치, CF(Compact Flash) 카드, 스마트 미디어(Smart Media) 카드, 메모리 스틱(Memory Stick) 등과 같은 다양한 종류의 저장 장치들 중 어느 하나로 구현될 수 있다. 컨트롤러(130)의 내부에 포함되는 구성 요소들은 메모리 시스템(110)의 구현 형태에 따라 추가되거나 제거될 수 있다.

[0072] 호스트(102)와 메모리 시스템(110)은 약속된 규격에 대응하여 신호, 데이터 등을 송수신하기 위한 컨트롤러 혹은 인터페이스를 포함할 수 있다. 예를 들면, 메모리 시스템(110) 내 호스트 인터페이스(132)는 호스트(102)에 신호, 데이터 등을 송신하거나 호스트(102)로부터 전달되는 신호, 데이터 등을 수신할 수 있는 장치를 포함할 수 있다.

[0073] 컨트롤러(130)에 포함된 호스트 인터페이스(132)는 호스트(102)로부터 전달되는 신호, 커맨드(command) 또는 데이터를 수신할 수 있다. 즉, 호스트(102)와 메모리 시스템(110)은 서로 약속된 규격을 통해 데이터를 송수신할 수 있다. 데이터를 송수신하기 위한 약속된 규격의 예로서 USB(Universal Serial Bus), MMC(Multi-Media Card), PATA(Parallel Advanced Technology Attachment), SCSI(Small Computer System Interface), ESDI(Enhanced Small Disk Interface), IDE(Integrated Drive Electronics), PCIE(Peripheral Component Interconnect Express), SAS(Serial-attached SCSI), SATA(Serial Advanced Technology Attachment), MIPI(Mobile Industry Processor Interface) 등과 같은 다양한 인터페이스 프로토콜이 있다. 실시예에 따라, 호스트 인터페이스(132)는 호스트(102)와 데이터를 주고받는 영역으로 호스트 인터페이스 계층(HIL: Host Interface Layer, 이하 'HIL'이라 칭하기로 함)이라 불리는 펌웨어(firmware)를 통해 구현되거나 구동될 수 있다.

[0074] 데이터를 송수신하기 위한 규격 중 하나인 IDE(Integrated Drive Electronics) 혹은 ATA(Advanced Technology Attachment)는 40개의 선이 병렬로 연결된 케이블을 사용하여 호스트(102)와 메모리 시스템(110) 간의 데이터의 송수신을 지원할 수 있다. 하나의 호스트(102)에 복수의 메모리 시스템(110)이 연결되는 경우, 복수의 메모리 시스템(110)이 연결되는 위치 혹은 딥스위치를 이용하여 복수의 메모리 시스템(110)을 마스터 혹은 슬레이브로 구분할 수 있다. 마스터로 설정된 메모리 시스템(110)이 주된 메모리 장치로 사용될 수 있다. IDE(ATA)는 Fast-ATA, ATAPI, EIDE(Enhanced IDE) 방식 등으로 발전해왔다.

[0075] SATA(Seral Advanced Technology Attachment, S-ATA)는 IDE(Integrated Drive Electronics) 장치의 접속 규격인 병렬 데이터 송수신 방식의 각종 ATA 규격과 호환성을 갖는 직렬 데이터 송수신 방식으로서, 연결선은 병렬 신호 40개에서 직렬 신호 6개로 줄일 수 있다. SATA는 IDE보다 데이터 송수신 속도가 빠르고, 데이터 송수신에 사용되는 호스트(102) 내 자원을 소모가 적은 이유로 널리 사용되어 왔다. SATA는 호스트(102)에 포함된 하나의 송수신 장치에 최대 30개의 외부 장치를 연결할 수 있다. 또한, SATA는 데이터 통신이 실행 중에도 외부 장치를 탈착할 수 있는 핫 플러깅을 지원하기 때문에, 호스트(102)에 전원이 공급된 상태에서도 유니버설 시리얼 버스(USB)처럼 메모리 시스템(110)을 추가 장치로서 연결하거나 분리할 수 있다. 예를 들어, eSATA 포트가 있는 장치의 경우, 호스트(102)에 메모리 시스템(110)을 외장 하드처럼 자유롭게 탈착할 수 있다.

[0076] SCSI(Small Computer System Interface)는 컴퓨터, 서버 등과 주변 장치를 연결하는 데 사용하는 직렬 연결 방식으로서, IDE 및 SATA와 같은 인터페이스에 비하여 전송 속도가 빠른 장점이 있다. SCSI에서는 호스트(102)와 복수의 주변 장치(예, 메모리 시스템(110)이 직렬로 연결되지만, 호스트(102)와 각 주변 장치 간 데이터 송수신은 병렬 데이터 송수신 방식으로 구현될 수 있다. SCSI에서는 호스트(102)에 메모리 시스템(110)과 같은 장치의 연결과 분리가 쉽다. SCSI는 호스트(102)에 포함된 하나의 송수신 장치에 15개의 외부 장치가 연결되는 것을 지원할 수 있다.

[0077] SAS(Serial Attached SCSI)는 SCSI의 직렬 데이터 송수신 버전으로 이해할 수 있다. SAS는 호스트(102)와 복수의 주변 장치가 직렬로 연결될 뿐만 아니라, 호스트(102)와 각 주변 장치간 데이터 송수신도 직렬 데이터 송수신 방식으로 수행될 수 있다. SAS는 많은 연결선을 포함하는 넓은 병렬 케이블 대신 시리얼 케이블로 연결하여 장비 관리가 쉽고 신뢰성과 성능이 개선될 수 있다. SAS는 호스트(102)에 포함된 하나의 송수신 장치에 최대 8개의 외부 장치를 연결할 수 있다.

[0078] NVMe(Non-volatile memory express)는 비휘발성 메모리 시스템(110)을 탑재한 서버, 컴퓨팅 장치 등의 호스트(102)의 성능 향상과 설계 유연성을 높일 수 있도록 만든 PCIe(Peripheral Component Interconnect Express,

PCI Express) 인터페이스 기반의 프로토콜을 가리킬 수 있다. 여기서, PCIe는 컴퓨팅 장치와 같은 호스트(102)와 컴퓨팅 장치와 연결되는 주변 장치와 같은 메모리 시스템(110)을 연결하기 위한 슬롯(slot) 혹은 특정 케이블을 이용하여, 복수의 핀(예, 18개, 32개, 49개, 82개 등)과 적어도 하나의 배선(예, x1, x4, x8, x16 등)을 통해 배선 당 초당 수백 MB이상(예, 250 MB/s, 500 MB/s, 984.6250 MB/s, 1389 MB/s 등)의 대역폭을 가질 수 있다. 이를 통해, PCIe는 초당 수십~수백 Gbit의 대역폭을 구현할 수 있다. NVMe는 하드 디스크보다 더 빠른 속도로 동작하는 SSD와 같은 비휘발성 메모리 시스템(110)의 속도를 지원할 수 있다.

[0079] 실시예에 따라, 호스트(102)와 메모리 시스템(110)은 범용 직렬 버스(Universal Serial Bus, USB)를 통해 연결될 수 있다. 범용 직렬 버스(USB)는 키보드, 마우스, 조이스틱, 프린터, 스캐너, 저장 장치, 모뎀, 화상 회의 카메라 등과 같은 주변 장치에 대한 경제적인 표준 연결을 보장하는 확장성이 뛰어난 핫 플러그형 플러그 앤 플레이 직렬 인터페이스를 포함할 수 있다. 호스트(102)에 포함된 하나의 송수신 장치에 메모리 시스템(110)과 같은 복수의 주변 장치를 연결할 수 있다.

[0080] 도 3를 참조하면, 컨트롤러(130) 내 에러 정정부(error correction circuitry, 138)는 메모리 장치(150)에서 처리되는 데이터의 에러 비트를 정정할 수 있다. 실시예에 따라, 에러 정정부(138)는 ECC 인코더와 ECC 디코더를 포함할 수 있다. 여기서, ECC 인코더(ECC encoder)는 메모리 장치(150)에 프로그램될 데이터를 에러 정정 인코딩(error correction encoding)하여, 패리티(parity) 비트가 부가된 데이터를 생성할 수 있다. 패리티 비트가 부가된 데이터는 메모리 장치(150)에 저장될 수 있다. ECC 디코더(ECC decoder)는, 메모리 장치(150)에 저장된 데이터를 리드할 경우, 메모리 장치(150)로부터 리드된 데이터에 포함되는 에러를 검출 및 정정한다. ECC 유닛(138)은 메모리 장치(150)로부터 리드한 데이터를 에러 정정 디코딩(error correction decoding)한 후, 에러 정정 디코딩의 성공 여부를 판단하고, 판단 결과에 따라 지시 신호, 예컨대 에러 정정 성공(success)/실패(fail) 신호를 출력하며, ECC 인코딩 과정에서 생성된 패리티(parity) 비트를 사용하여 리드된 데이터의 에러 비트를 정정할 수 있다. ECC 유닛(138)은 에러 비트 개수가 정정 가능한 에러 비트 한계치 이상 발생하면, 에러 비트를 정정할 수 없으며, 에러 비트를 정정하지 못함에 상응하는 에러 정정 실패 신호를 출력할 수 있다.

[0081] 실시예에 따라, 에러 정정부(138)는 LDPC(low density parity check) 코드(code), BCH(Bose, Chaudhri, Hocquenghem) 코드, 터보 코드(turbo code), 리드-솔로몬 코드(Reed-Solomon code), 컨벌루션 코드(convolution code), RSC(recursive systematic code), TCM(trellis-coded modulation), BCM(Block coded modulation) 등의 코디드 모듈레이션(coded modulation)을 사용하여 에러 정정을 수행할 수 있으며, 이에 한정되는 것은 아니다. 또한, 에러 정정부(138)는 데이터에 포함된 오류를 정정하기 위한 프로그램, 회로, 모듈, 시스템, 또는 장치를 포함할 수 있다.

[0082] 예를 들어, ECC 디코더(ECC decoder)는 메모리 장치(150)에서 전달된 데이터에 대해 경판정 복호(hard decision decoding) 혹은 연판정 복호(soft decision decoding)를 수행할 수 있다. 여기서, 경판정 복호(hard decision decoding)는 에러 정정을 크게 구분한 두 가지 방법 중 하나로 이해할 수 있다. 경판정 복호(hard decision decoding)는 '0' 또는 '1'의 디지털 데이터를 메모리 장치(150) 내 비휘발성 메모리 셀에서 읽어서 에러를 정정하는 동작을 포함할 수 있다. 경판정 복호(hard decision decoding)는 2진의 논리 신호를 다루기 때문에, 회로 또는 알고리즘의 설계가 간단할 수 있고, 처리 속도가 빠를 수 있다.

[0083] 한편, 경판정 복호(hard decision decoding)와 구별되는 연판정 복호(soft decision decoding)는 메모리 장치(150) 내 비휘발성 메모리 셀의 문턱 전압을 2 이상의 양자화된 값(예, 여러 비트 데이터, 근사값, 또는 아날로그값 등)에 근거해서 에러를 정정하는 동작을 포함할 수 있다. 컨트롤러(130)는 메모리 장치(150) 내 복수의 비휘발성 메모리 셀로부터 2 이상의 알파벳 또는 양자화된 값을 수신한 후, 양자화된 값들을 조건확률 또는 우도 등 정보의 조합으로 특징지어 생성된 정보들을 토대로 복호(decoding)를 수행할 수 있다.

[0084] 실시예에 따라, ECC 디코더(ECC decoder)는 연판정 복호(soft decision decoding)를 위한 방법 중 LDPC-GM(low-density parity-check and generator matrix) 코드를 사용할 수 있다. 여기서, LDPC(low-density parity-check) 코드는 메모리 장치(150)에서 데이터의 값을 단순히 1 또는 0이 아니라(경판정 복호가 아니라) 신뢰도에 따라 여러 비트로 읽고, 이를 메시지 교환 방식을 통해서 반복적으로 신뢰도 정보를 향상시켜서 1 또는 0의 최종값을 결정할 수 있는 알고리즘을 사용한다. 예를 들어, LDPC 코드를 이용한 복호 알고리즘은 확률적 복호법(probabilistic decoding)으로 이해할 수 있으며, 메모리 장치(150)에서 일어날 수 있는 에러인 비트 반전(Bit-flipping)에 대해 비휘발성 메모리 셀에서 출력되는 값을 0 또는 1로 부호화한 경판정 복호(hard-decision decoding)에 비하여, 비휘발성 메모리 셀에 저장된 값을 확률적 정보를 기초로 판단할 수 있기 때문에, 복구 가능성을 높일 수 있고 정정되는 정보의 신뢰성과 안정성을 높일 수 있다. LDPC-GM 코드는 내부

LDGM 코드들이 고속의 LDPC 코드들에 직렬로 연쇄(concatenated)될 수 있는 구조(scheme)을 가질 수 있다.

[0085] 실시예에 따라, ECC 디코더(ECC decoder)는 연판정 복호(soft decision decoding)를 위한 방법 중 LDPC-CCs(low-density parity-check conventional convolutional codes) 코드를 사용할 수 있다. 여기서, LDPC-CCs 코드는 가변 블록 길이, 시프트 레지스터를 기반으로 하는 선형 시간 인코딩 및 파이프 라인 디코딩을 이용하는 구조를 가질 수 있다.

[0086] 실시예에 따라, ECC 디코더(ECC decoder)는 연판정 복호(soft decision decoding)를 위한 방법 중 LLR-TC(Log Likelihood Ratio Turbo Code)를 사용할 수 있다. 여기서, LLR(Log Likelihood Ratio)은 샘플링된 값(sampled value)과 이상적인 값(ideal value) 사이의 거리(distance)에 대한 비선형 함수(non-linear function)로 계산될 수 있다. 또한, TC(Turbo Code)는 간단한 부호(예를 들면 Hamming code 등)를 이차원 또는 삼차원으로 구성하고 횡 방향 (row direction)과 열 방향 (column direction)의 디코딩을 반복해서 역시 신뢰도를 개선하는 구조를 가질 수 있다.

[0087] 한편, 도시되지 않았지만, 컨트롤러(130)는 데이터를 메모리 장치(150)에 쓰거나, 메모리 장치(150)에 저장된 데이터를 읽는 과정에서 이레이저 코딩(erasure coding, EC)을 통해 데이터를 인코딩(encoding) 혹은 디코딩(decoding)할 수 있다. 여기서, 이레이저 코딩(erasure coding, EC)은 이레이저 코드(Erasure Code)를 이용하여 데이터를 인코딩하고, 데이터 손실시 디코딩 과정을 거쳐 원본 데이터를 복구하는 데이터 복구 기법으로 이해할 수 있다. 소거 코드(Erasure Codes)로 생성된 패리티가 데이터 복제본 생성보다 적은 저장공간을 차지하므로, 이레이저 코딩(EC)은 메모리 시스템(110)의 신뢰성을 제공하면서 저장공간 효율성을 높일 수 있다. 사용되는 이레이저 코드(Erasure Code)는 다양할 수 있다. 예를 들어, 이레이저 코드로는 리드 솔로몬 부호(Reed-Solomon Code), 타호-LAFS(Tahoe-LAFS, Tahoe Least-Authority File System), 에베노드 코드(EVENODD code), 위버 코드(Weaver code), 엑스 코드(X-code) 등이 있다. 소거 코드(Erasure Codes) 별로 다른 알고리즘이 사용될 수 있으며, 컨트롤러(130)는 연산 복잡도를 줄이면서 복구 성능을 높이기 위한 이레이저 코드를 사용할 수 있다.

[0088] PMU(140)는 메모리 시스템(110)에 인가되는 전원(예, 컨트롤러(130)에 공급되는 전압)을 감시하고, 컨트롤러(130)에 포함된 구성 요소들에 파워를 제공할 수 있다. PMU(140)는 전원의 온(On) 혹은 오프(Off)를 감지할 뿐만 아니라, 공급되는 전압 레벨이 불안정한 경우, 메모리 시스템(110)이 긴급하게 현재 상태를 백업할 수 있도록 트리거 신호를 생성할 수 있다. 실시예에 따라, PMU(140)는 긴급 상황에서 사용될 수 있는 전력을 축적할 수 있는 장치를 포함할 수 있다.

[0089] 메모리 인터페이스(142)는, 컨트롤러(130)가 호스트(102)로부터의 요청에 응답하여 메모리 장치(150)를 제어하기 위해, 컨트롤러(130)와 메모리 장치(150) 간의 신호, 데이터를 송수신할 수 있다. 메모리 장치(150)가 플래시 메모리(예, NAND 플래시 메모리)일 경우, 메모리 인터페이스(142)는 NAND 플래시 컨트롤러(NAND Flash Controller, NFC)를 포함할 수 있다. 프로세서(134)의 제어에 따라, 메모리 인터페이스(142)는 메모리 장치(150)의 동작을 제어하기 위한 신호를 생성할 수 있고, 메모리 장치(150)에서 출력된 데이터를 수신하거나, 메모리 장치(150)에 저장될 데이터를 송신할 수 있다. 실시예에 따라, 메모리 인터페이스(142)는 메모리 장치(150) 간 데이터 입출력을 지원하며, 메모리 장치(150)와 데이터를 주고받는 영역으로 플래시 인터페이스 계층(FIL: Flash Interface Layer, 이하 'FIL'이라 칭하기로 함)이라 불리는 펌웨어(firmware)를 통해 구현되거나 구동될 수 있다.

[0090] 실시예에 따라, 메모리 인터페이스(142)는 메모리 장치(150) 간 데이터 입출력을 위해 Open NAND Flash Interface(ONFi), 토글(toggle) 모드 등을 지원할 수 있다. 예를 들면, ONFi는 8-비트 혹은 16-비트의 단위 데이터에 대한 양방향(bidirectional) 송수신을 지원할 수 있는 신호선을 포함하는 데이터 경로(예, 채널, 웨이 등)를 사용할 수 있다. 컨트롤러(130)와 메모리 장치(150) 사이의 데이터 통신은 비동기식 SDR(Asynchronous Single Data Rate), 동기식 DDR(Synchronous Double Data Rate) 및 토글 DDR(Toggle Double Data Rate) 중 적어도 하나에 대한 인터페이스(interface)를 지원하는 장치를 통해 수행될 수 있다.

[0091] 메모리(144)는 메모리 시스템(110) 및 컨트롤러(130)의 동작 메모리(working memory)로서, 메모리 시스템(110) 및 컨트롤러(130)의 구동을 위해 필요한 데이터 혹은 구동 중 발생한 데이터를 저장할 수 있다. 예를 들어, 메모리(144)는 컨트롤러(130)가 호스트(102)로부터의 요청에 응답하여 메모리 장치(150)로부터 제공된 리드 데이터를 호스트(102)로 제공하기 전 임시 저장할 수 있다. 또한, 컨트롤러(130)는 호스트(102)로부터 제공된 쓰기 데이터를 메모리 장치(150)에 저장하기 전, 메모리(144)에 임시 저정할 수 있다. 메모리 장치(150)의 리드, 라이트, 프로그램, 이레이즈(erase) 등의 동작을 제어할 경우, 메모리 시스템(110) 내 컨트롤러(130)와 메모리 장

치(150) 사이에 전달되거나 발생하는 데이터는 메모리(144)에 저장될 수 있다. 리드 데이터 또는 쓰기 데이터뿐만 아니라, 메모리(144)는 호스트(102)와 메모리 장치(150) 간 데이터 라이트 및 리드 등의 동작을 수행하기 위해 필요한 정보(예, 맵 데이터, 리드 명령, 프로그램 명령 등)를 저장할 수 있다. 메모리(144)는 명령큐(command queue), 프로그램 메모리, 데이터 메모리, 라이트 버퍼(buffer)/캐시(cache), 리드 버퍼/캐시, 데이터 버퍼/캐시, 맵(map) 버퍼/캐시 등을 포함할 수 있다. 여기서, 맵 버퍼/캐시는 도 1에서 설명한 맵 정보를 저장하기 위한 장치 혹은 영역일 수 있다.

[0092] 실시예에 따라, 메모리(144)는 휘발성 메모리로 구현될 수 있으며, 예컨대 정적 랜덤 액세스 메모리(SRAM: Static Random Access Memory), 또는 동적 랜덤 액세스 메모리(DRAM: Dynamic Random Access Memory) 등으로 구현될 수 있다. 아울러, 메모리(144)는, 도 3에서 도시한 바와 같이, 컨트롤러(130)의 내부에 존재하거나, 또는 컨트롤러(130)의 외부에 존재할 수 있으며, 이때 메모리 인터페이스를 통해 컨트롤러(130)로부터 데이터가 입출력되는 외부 휘발성 메모리로 구현될 수도 있다.

[0093] 프로세서(134)는 컨트롤러(130)의 동작을 제어할 수 있다. 호스트(102)로부터의 라이트 요청 또는 리드 요청에 응답하여, 프로세서(134)는 메모리 장치(150)에 대한 프로그램 동작 또는 리드 동작을 수행할 수 있다. 프로세서(134)는, 컨트롤러(130)의 데이터 입출력 동작을 제어하기 위해 플래시 변환 계층(FTL: Flash Translation Layer, 이하 'FTL'이라 칭하기로 함)이라 불리는 펌웨어(firmware)를 구동할 수 있다. 플래시 변환 계층(FTL)은 도 3에서 보다 구체적으로 설명한다. 실시예에 따라, 프로세서(134)는 마이크로프로세서 또는 중앙 처리 장치(CPU) 등으로 구현될 수 있다.

[0094] 또한, 실시예에 따라, 프로세서(134)는 서로 구별되는 연산 처리 영역인 코어(core)가 두 개 이상이 집적된 회로인 멀티 코어(multi-core) 프로세서로 구현될 수도 있다. 예를 들어, 멀티 코어 프로세서 내 복수의 코어는 복수의 플래시 변환 계층(FTL)을 각각 구동하면, 메모리 시스템(110)의 데이터 입출력 속도를 향상시킬 수 있다.

[0095] 컨트롤러(130) 내 프로세서(134)는 호스트(102)로부터 입력된 커맨드에 대응하는 동작을 수행할 수도 있고, 호스트(102)와 같은 외부 장치에서 입력되는 커맨드와 무관하게 메모리 시스템(110)이 독립적으로 동작을 수행할 수도 있다. 통상적으로 호스트(102)로부터 전달된 커맨드에 대응하여 컨트롤러(130)가 수행하는 동작이 포그라운드(foreground) 동작으로 이해될 수 있고, 호스트(102)로부터 전달된 커맨드와 무관하게 컨트롤러(130)가 독립적으로 수행하는 동작이 백그라운드(background) 동작으로 이해될 수 있다. 포그라운드(foreground) 동작 또는 백그라운드(background) 동작으로, 컨트롤러(130)는 메모리 장치(150)에 저장된 데이터에 대한 읽기(read), 쓰기(write) 혹은 프로그램(program), 삭제(erase) 등을 위한 동작을 수행할 수도 있다. 또한, 호스트(102)로부터 전달된 셋 커맨드(set command)로 셋 파라미터 커맨드(set parameter command) 또는 셋 픽쳐 커맨드(set feature command)에 해당하는 파라미터 셋 동작 등도 포그라운드 동작으로 이해될 수 있다. 한편, 호스트(102)에서 전달되는 명령없이 백그라운드 동작으로, 메모리 장치(150)에 포함된 복수의 메모리 블록들(152, 154, 156)과 관련하여, 메모리 시스템(110)은 가비지 컬렉션(Garbage Collection, GC), 웨어 레벨링(Wear Leveling, WL), 배드 블록을 확인하여 처리하는 배드 블록 관리(bad block management) 등을 위한 동작들을 수행할 수도 있다.

[0096] 한편, 포그라운드(foreground) 동작 또는 백그라운드(background) 동작으로 실질적으로 유사한 동작이 수행될 수도 있다. 예를 들어, 메모리 시스템(110)이 호스트(102)의 명령에 대응하여 수동 가비지 컬렉션(Manual GC)을 수행하면 포그라운드 동작으로 이해될 수 있고, 메모리 시스템(110)이 독립적으로 자동 가비지 컬렉션(Auto GC)을 수행하면 백그라운드 동작으로 이해될 수 있다.

[0097] 메모리 장치(150)가 비휘발성 메모리 셀을 포함하는 복수의 다이(dies) 혹은 복수의 칩(chips)으로 구성된 경우, 컨트롤러(130)는 메모리 시스템(110)의 성능 향상을 위해 호스트(102)에서 전달된 요청 혹은 명령들을 메모리 장치(150) 내 복수의 다이(dies) 혹은 복수의 칩(chips)에 나누어 동시에 처리할 수 있다. 컨트롤러(130) 내 메모리 인터페이스(142)은 메모리 장치(150) 내 복수의 다이(dies) 혹은 복수의 칩(chips)과 적어도 하나의 채널(channel)과 적어도 하나의 웨이(way)를 통해 연결될 수 있다. 컨트롤러(130)가 비휘발성 메모리 셀로 구성되는 복수의 페이지에 대응하는 요청 혹은 명령을 처리하기 위해 데이터를 각 채널 혹은 각 웨이를 통해 분산하여 저장할 경우, 해당 요청 혹은 명령에 대한 동작이 동시에 혹은 병렬로 수행될 수 있다. 이러한 처리 방식 혹은 방법을 인터리빙(interleaving) 방식으로 이해할 수 있다. 메모리 장치(150) 내 각 다이(die) 혹은 각 칩(chip)의 데이터 입출력 속도보다 인터리빙 방식으로 동작할 수 있는 메모리 시스템(110)의 데이터 입출력 속도는 빠를 수 있으므로, 메모리 시스템(110)의 데이터 입출력 성능을 향상시킬 수 있다.

[0098] 컨트롤러(130)는 메모리 장치(150)에 포함된 복수의 메모리 다이들과 연결된 복수의 채널들 또는 웨이들의 상태를 확인할 수 있다. 예컨대, 채널들 또는 웨이들의 상태는 비지(busy) 상태, 레디(ready) 상태, 액티브(active) 상태, 아이들(idle) 상태, 정상(normal) 상태, 비정상(abnormal) 상태 등으로 구분할 수 있다. 컨트롤러(130)가 명령, 요청 및/또는 데이터가 전달되는 채널 또는 웨이에 대응하여, 저장되는 데이터의 물리 주소가 결정될 수 있다. 한편, 컨트롤러(130)는 메모리 디바이스 (150)로부터 전달된 디스크립터(descriptor)를 참조할 수 있다. 디스크립터는 미리 결정된 포맷 또는 구조를 갖는 데이터로서, 메모리 장치(150)에 관한 무언가를 기술하는 파라미터의 블록 또는 페이지를 포함할 수 있다. 예를 들어, 디스크립터는 장치 디스크립터, 구성 디스크립터, 유닛 디스크립터 등을 포함할 수 있다. 컨트롤러(130)는 명령 또는 데이터가 어떤 채널(들) 또는 방법(들)을 통해 교환되는지를 결정하기 위해 디스크립터를 참조하거나 사용한다.

[0099] 메모리 시스템(110) 내 메모리 장치(150)는 복수의 메모리 블록(152, 154, 156)을 포함할 수 있다. 복수의 메모리 블록(152, 154, 156) 각각은 복수의 비휘발성 메모리 셀을 포함한다. 도시되지 않았지만, 실시예에 따라, 복수의 메모리 블록(152, 154, 156) 각각은 3차원(dimension) 입체 스택(stack) 구조를 가질 수 있다.

[0100] 메모리 장치(150)에 포함된 복수의 메모리 블록들(152, 154, 156)은, 하나의 메모리 셀에 저장 또는 표현할 수 있는 비트의 수에 따라, 단일 레벨 셀(Single Level Cell, SLC) 메모리 블록 및 멀티 레벨 셀(Multi Level Cell, MLC) 메모리 블록 등으로 구분될 수 있다. SLC 메모리 블록은 하나의 메모리 셀에 1 비트 데이터를 저장하는 비휘발성 메모리 셀들로 구현된 복수의 페이지들을 포함할 수 있다. MLC 메모리 블록에 비하여, SLC 메모리 블록은 데이터 연산 성능이 빠르며 내구성이 높을 수 있다. MLC 메모리 블록은 하나의 메모리 셀에 멀티 비트 데이터(예를 들면, 2 비트 또는 그 이상의 비트)를 저장하는 메모리 셀들로 구현된 복수의 페이지들을 포함할 수 있다 SLC 메모리 블록에 비하여, MLC 메모리 블록은 동일한 면적, 공간에 더 많은 데이터를 저장할 수 있다. 메모리 장치(150)에 포함된 MLC 메모리 블록은 하나의 메모리 셀에 2 비트 데이터를 저장할 수 있는 메모리 셀들에 의해 구현된 복수의 페이지들을 포함하는 더블 레벨 셀(Double Level Cell, DLC), 하나의 메모리 셀에 3 비트 데이터를 저장할 수 있는 메모리 셀들에 의해 구현된 복수의 페이지들을 포함하는 트리플 레벨 셀(Triple Level Cell, TLC), 하나의 메모리 셀에 4 비트 데이터를 저장할 수 있는 메모리 셀들에 의해 구현된 복수의 페이지들을 포함하는 쿼드러플 레벨 셀(Quadruple Level Cell, QLC), 또는 하나의 메모리 셀에 5 비트 또는 그 이상의 비트 데이터를 저장할 수 있는 메모리 셀들에 의해 구현된 복수의 페이지들을 포함하는 다중 레벨 셀(multiple level cell) 등을 포함할 수 있다.

[0101] 실시예에 따라, 컨트롤러(130)는 메모리 시스템(150)에 포함된 멀티 레벨 셀(MLC) 메모리 블록을 하나의 메모리 셀에 1 비트 데이터를 저장하는 SLC 메모리 블록과 같이 운용할 수 있다. 예를 들어, 멀티 레벨 셀(MLC) 메모리 블록의 일부에서 다른 블록에 비하여 더 빠를 수 있는 데이터 입출력 속도를 활용하여, 컨트롤러(130)는 멀티 레벨 셀(MLC) 메모리 블록의 일부를 SLC 메모리 블록으로 운용함으로써 데이터를 임시로 저장하기 위한 버퍼(buffer)로 사용할 수도 있다.

[0102] 또한, 실시예에 따라, 컨트롤러(130)는 메모리 시스템(150)에 포함된 멀티 레벨 셀(MLC) 메모리 블록에 삭제 동작 없이 복수 번 데이터를 프로그램할 수 있다. 일반적으로, 비휘발성 메모리 셀은 덮어 쓰기(overwrite)를 지원하지 않는 특징을 가지고 있다. 하지만, 멀티 레벨 셀(MLC) 메모리 블록이 멀티 비트 데이터를 저장할 수 있는 특징을 이용하여, 컨트롤러(130)는 비휘발성 메모리 셀에 1비트 데이터를 복수 번 프로그램할 수도 있다. 이를 위해, 컨트롤러(130)는 비휘발성 메모리 셀에 데이터를 프로그램한 횟수를 별도의 동작 정보로 저장할 수 있고, 동일한 비휘발성 메모리 셀에 다시 프로그램하기 전 비휘발성 메모리 셀의 문턱 전압의 레벨을 균일하게 하기 위한 균일화(uniformity) 동작을 수행할 수도 있다.

[0103] 실시예에 따라, 메모리 장치(150)는 ROM(Read Only Memory), MROM(Mask ROM), PROM(Programmable ROM), EPROM(Erasable ROM), EEPROM(Electrically Erasable ROM), FRAM(Ferromagnetic ROM), PRAM(Phase change RAM), MRAM(Magnetic RAM), RRAM(Resistive RAM), NAND 혹은 NOR 플래시 메모리(flash memory), 상변환 메모리(PCRAM: Phase Change Random Access Memory), 저항 메모리(RRAM(ReRAM): Resistive Random Access Memory), 강유전체 메모리(FRAM: Ferroelectrics Random Access Memory), 또는 스핀 주입 자기 메모리(STT-RAM(STT-MRAM): Spin Transfer Torque Magnetic Random Access Memory) 등과 같은 메모리 장치로 구현될 수 있다.

[0104] 도 4는 본 발명의 다른 실시예에 따른 메모리 시스템을 설명한다.

[0105] 도 4를 참조하면, 호스트(102) 및 메모리 장치(150)와 연동하는 컨트롤러(130)는 호스트 인터페이스(132), 플래시 변환 계층(FTL, 240), 메모리 인터페이스(142) 및 메모리(144)를 포함할 수 있다. 도 4에서 설명하는 플래시 변환 계층(Flash Translation Layer (FTL), 240)의 하나의 실시예로서, 플래시 변환 계층(FTL, 240)은 메모리

시스템(110)의 동작 성능에 따라 다양한 형태로 구현될 수 있다.

[0106] 호스트 인터페이스(132)은 호스트(102)로부터 전달되는 명령, 데이터 등을 주고받기 위한 것이다. 예를 들어, 호스트 인터페이스 유닛(132)은 호스트(102)로부터 전달되는 명령, 데이터 등을 순차적으로 저장한 뒤, 저장된 순서에 따라 출력할 수 있는 명령큐(56), 명령큐(56)로부터 전달되는 명령, 데이터 등을 분류하거나 처리 순서를 조정할 수 있는 버퍼관리자(52), 및 버퍼관리자(52)로부터 전달된 명령, 데이터 등의 처리를 위한 이벤트를 순차적으로 전달하기 위한 이벤트큐(54)를 포함할 수 있다.

[0107] 호스트(102)로부터 명령, 데이터는 동일한 특성의 복수개가 연속적으로 전달될 수도 있고, 서로 다른 특성의 명령, 데이터가 뒤 섞여 전달될 수도 있다. 예를 들어, 데이터를 읽기 위한 명령어가 복수 개 전달되거나, 읽기 및 프로그램 명령이 교번적으로 전달될 수도 있다. 호스트 인터페이스(132)은 호스트(102)로부터 전달된 명령, 데이터 등을 명령큐(56)에 먼저 순차적으로 저장한다. 이후, 호스트(102)로부터 전달된 명령, 데이터 등의 특성에 따라 컨트롤러(130)가 어떠한 동작을 수행할 지를 예측할 수 있으며, 이를 근거로 명령, 데이터 등의 처리 순서나 우선 순위를 결정할 수도 있다. 또한, 호스트(102)로부터 전달된 명령, 데이터 등의 특성에 따라, 호스트 인터페이스(132) 내 버퍼관리자(52)는 명령, 데이터 등을 메모리(144)에 저장할 지, 플래시 변환 계층(FTL, 240)으로 전달할 지도 결정할 수도 있다. 이벤트큐(54)는 호스트(102)로부터 전달된 명령, 데이터 등에 따라 메모리 시스템 혹은 컨트롤러(130)가 내부적으로 수행, 처리해야 하는 이벤트를 버퍼관리자(52)로부터 수신한 후, 수신된 순서대로 플래시 변환 계층(FTL, 240)에 전달할 수 있다.

[0108] 실시예에 따라, 플래시 변환 계층(FTL, 240)은 이벤트큐(54)로부터 수신된 이벤트를 관리하기 위한 호스트 요구 관리자(Host Request Manager(HRM), 46), 맵 데이터를 관리하는 맵데이터 관리자(Map Manger(MM), 44), 가비지 컬렉션 또는 웨어 레벨링을 수행하기 위한 상태 관리자(42), 메모리 장치 내 블록에 명령을 수행하기 위한 블록 관리자(48)를 포함할 수 있다. 도 4에서 도시되지 않았지만, 실시예에 따라, 도 3에서 설명한 ECC 유닛(138)은 플래시 변환 계층(FTL, 240)에 포함될 수 있다. 실시예에 따라, ECC 유닛(138)은 컨트롤러(130) 내 별도의 모듈, 회로, 또는 펌웨어 등으로 구현될 수도 있다.

[0109] 호스트 요구 관리자(HRM, 46)는 맵데이터 관리자(MM, 44) 및 블록 관리자(48)를 사용하여 호스트 인터페이스(132)으로부터 수신된 읽기 및 프로그램 명령, 이벤트에 따른 요청을 처리할 수 있다. 호스트 요구 관리자(HRM, 46)는 전달된 요청의 논리 주소에 해당하는 물리 주소를 파악하기 위해 맵데이터 관리자(MM, 44)에 조회 요청을 보내고, 맵데이터 관리자(MM, 44)는 주소 변환(address translation)을 수행할 수 있다. 호스트 요구 관리자(HRM, 46)는 물리 주소에 대해 메모리 인터페이스 유닛(142)에 플래시 읽기 요청을 전송하여 읽기 요청을 처리할 수 있다. 한편, 호스트 요구 관리자(HRM, 46)는 먼저 블록 관리자(48)에 프로그램 요청을 전송함으로써 미기록된(데이터가 없는) 메모리 장치의 특정 페이지에 데이터를 프로그램한 다음, 맵데이터 관리자(MM, 44)에 프로그램 요청에 대한 맵 갱신(update) 요청을 전송함으로써 논리-물리 주소의 매핑 정보에 프로그램한 데이터에 대한 내용을 업데이트할 수 있다.

[0110] 여기서, 블록 관리자(48)는 호스트 요구 관리자(HRM, 46), 맵데이터 관리자(MM, 44), 및 상태 관리자(42)가 요청한 프로그램 요청을 메모리 장치(150)를 위한 프로그램 요청으로 변환하여 메모리 장치(150) 내 블록을 관리할 수 있다. 메모리 시스템(110, 도 3 참조)의 프로그램 혹은 쓰기 성능을 극대화하기 위해 블록 관리자(48)는 프로그램 요청을 수집하고 다중 평면 및 원샷 프로그램 작동에 대한 플래시 프로그램 요청을 메모리 인터페이스(142)으로 보낼 수 있다. 또한, 다중 채널 및 다중 방향 플래시 컨트롤러의 병렬 처리(예, 인터리빙 동작)를 최대화하기 위해 여러 가지 뛰어난 플래시 프로그램 요청을 메모리 인터페이스(142)으로 전송할 수도 있다.

[0111] 한편, 블록 관리자(48)는 유효 페이지 수에 따라 플래시 블록을 관리하고 여유 블록이 필요한 경우 유효한 페이지가 없는 블록을 선택 및 지우고, 쓰레기(garbage) 수집이 필요한 경우 가장 적게 유효한 페이지를 포함하고 있는 블록을 선택할 수 있다. 블록 관리자(48)가 충분한 빈 블록을 가질 수 있도록, 상태 관리자(42)는 가비지 수집을 수행하여 유효 데이터를 모아 빈 블록으로 이동시키고, 이동된 유효 데이터를 포함하고 있었던 블록들을 삭제할 수 있다. 블록 관리자(48)가 상태 관리자(42)에 대해 삭제될 블록에 대한 정보를 제공하면, 상태 관리자(42)는 먼저 삭제될 블록의 모든 플래시 페이지를 확인하여 각 페이지가 유효한지 여부를 확인할 수 있다. 예를 들어, 각 페이지의 유효성을 판단하기 위해, 상태 관리자(42)는 각 페이지의 스페어(Out Of Band, OOB) 영역에 기록된 논리 주소를 식별한 뒤, 페이지의 실제 주소와 맵 관리자(44)의 조회 요청에서 얻은 논리 주소에 매핑된 실제 주소를 비교할 수 있다. 상태 관리자(42)는 각 유효한 페이지에 대해 블록 관리자(48)에 프로그램 요청을 전송하고, 프로그램 작업이 완료되면 맵 관리자(44)의 갱신을 통해 매핑 테이블이 업데이트될 수 있다.

[0112] 맵 관리자(44)는 논리-물리 매핑 테이블을 관리하고, 호스트 요구 관리자(HRM, 46) 및 상태 관리자(42)에 의해

생성된 조회, 업데이트 등의 요청을 처리할 수 있다. 맵 관리자(44)는 전체 매핑 테이블을 플래시 메모리에 저장하고, 메몰시 소자(144) 용량에 따라 매핑 항목을 캐시할 수도 있다. 조회 및 업데이트 요청을 처리하는 동안 맵 캐시 미스가 발생하면, 맵 관리자(44)는 메모리 인터페이스(142)에 읽기 요청을 전송하여 메모리 장치(150)에 저장된 매핑 테이블을 로드(load)할 수 있다. 맵 관리자(44)의 더티 캐시 블록 수가 특정 임계 값을 초과하면 블록 관리자(48)에 프로그램 요청을 보내서 깨끗한 캐시 블록을 만들고 더티 맵 테이블이 메모리 장치(150)에 저장될 수 있다.

[0113] 한편, 가비지 컬렉션이 수행되는 경우, 상태 관리자(42)가 유효한 페이지를 복사하는 동안 호스트 요구 관리자(HRM, 46)는 페이지의 동일한 논리 주소에 대한 데이터의 최신 버전을 프로그래밍하고 업데이트 요청을 동시에 발행할 수 있다. 유효한 페이지의 복사가 정상적으로 완료되지 않은 상태에서 상태 관리자(42)가 맵 업데이트를 요청하면 맵 관리자(44)는 매핑 테이블 업데이트를 수행하지 않을 수도 있다. 맵 관리자(44)는 최신 맵 테이블이 여전히 이전 실제 주소를 가리키는 경우에만 맵 업데이트를 수행하여 정확성을 보장할 수 있다.

[0114] 도 5는 본 발명의 일 실시예에 따른 데이터 처리 시스템에서 호스트와 메모리 시스템의 읽기 동작을 설명한다. 도 2 내지 도 5를 참조하여, 호스트(102) 내 메모리(106)에 메타 데이터(166)가 저장된 경우, 호스트(102)가 메모리 시스템(110) 내 데이터를 읽는 동작을 설명한다.

[0115] 호스트(102)와 메모리 시스템(110)에 전원이 공급되고, 호스트(102)와 메모리 시스템(110)이 연동할 수 있다. 호스트(102)와 메모리 시스템(110)이 연동하면, 메모리 장치(150)에 저장된 맵핑 정보(L2P MAP)가 호스트 메모리(106)로 전송될 수 있다.

[0116] 호스트(102) 내 프로세서(104)에 의해 읽기 요청이 발생하면, 읽기 요청은 호스트 컨트롤러 인터페이스(108)에 전달된다. 호스트 컨트롤러 인터페이스(108)는 읽기 요청을 수신한 후, 호스트 메모리(106)에 읽기 요청에 대응하는 논리 주소(Logical Address)를 전달한다. 호스트 메모리(106) 내 저장된 맵핑 정보(L2P MAP)를 바탕으로, 호스트 컨트롤러 인터페이스(108)는 논리 주소(Logical Address)에 대응하는 물리 주소(Physical Address)를 인지할 수 있다.

[0117] 호스트 컨트롤러 인터페이스(108)는 물리 주소(Physical Address)와 함께 읽기 요청(Read CMD)을 메모리 시스템(110) 내 컨트롤러(130)에 전달한다. 컨트롤러(130)는 수신된 읽기 요청과 물리 주소를 바탕으로, 메모리 장치(150)를 액세스할 수 있다. 메모리 장치(150) 내 물리 주소에 대응하는 위치에 저장된 데이터는 호스트 메모리(106)로 전달될 수 있다.

[0118] 비휘발성 메모리 장치를 포함하는 메모리 장치(150)에서 데이터를 읽는 과정은 다른 비휘발성 메모리인 호스트 메모리(106) 등에서 데이터를 읽는 과정에 비해 많은 시간이 소요될 수 있다. 전술한 읽기 과정에는 컨트롤러(130)가 호스트(102)로부터 논리 주소를 수신하여 대응하는 물리 주소를 찾는 과정이 생략될 수 있다. 특히, 컨트롤러(130)가 물리 주소를 찾아내는 과정에서 메모리 장치(150)를 액세스하여 메타 데이터를 읽어내는 동작이 사라질 수 있다. 이를 통해, 호스트(102)가 메모리 시스템(110)에 저장된 데이터를 읽어 내는 과정이 더욱 빨라질 수 있다.

[0119] 도 1 및 도 5를 참조하면, 컨트롤러(130)는 메모리 장치(150)에 저장된 맵핑 정보(L2P MAP)의 일부를 암호화할 수 있다. 컨트롤러(130)는 맵핑 정보(L2P MAP)의 일부를 암호화하여 암호화된 전송 데이터를 호스트 메모리(106)에 저장되도록 할 수 있다. 호스트 메모리(106)는 암호화된 전송 데이터를 저장하고, 호스트 컨트롤러 인터페이스(108)는 논리 주소(Logical Address)와 연관된 암호화된 전송 데이터를 호스트 메모리(106)로부터 확보한 후, 읽기 요청(Read CMD)과 함께 암호화된 전송 데이터를 컨트롤러(130)로 전달할 수 있다. 컨트롤러(130)는 암호화된 전송 데이터를 디코딩하여, 논리 주소에 연관된 물리 주소를 주소 변환 과정없이 인식할 수 있다. 이하에서, 암호화된 전송 데이터를 송수신하는 메모리 시스템(110)과 호스트(102)에 대해 구체적으로 설명한다.

[0120] 도 6은 본 발명의 일 실시예에 따른 호스트와 메모리 시스템의 제1 동작을 설명한다. 구체적으로, 도 6은 호스트와 메모리 시스템이 연동하는 과정 중 메모리 시스템이 호스트에 맵 정보를 전송하는 방법의 일 예를 설명한다.

[0121] 도 6을 참조하면, 메모리 시스템(110, 도 1 내지 4 참조)은 호스트(102, 도 1 내지 4 참조)로부터 전달된 명령에 대응하는 동작이 완료되었는 지를 확인할 수 있다(862). 메모리 시스템은 명령에 대응하는 동작이 완료된 후, 명령에 대응하는 응답(RESPONSE)을 전송하기 전에 호스트에 전송할 맵핑 정보가 있는 지를 확인할 수 있다(864). 만약 호스트에 전송할 맵핑 정보가 없다면(864의 NO), 메모리 시스템은 호스트에서 전달된 명령에 대응하는 동작이 완료되었는 지에 대한 여부(성공 또는 실패)에 대한 정보를 포함하는 응답(RESPONSE)을 전송할 수

있다(866).

[0122] 메모리 시스템이 호스트에 전송할 맵핑 정보가 있는 경우(864의 YES), 메모리 시스템은 맵핑 정보를 전송하기 위한 통지(NOTICE)가 이루어졌는 지를 확인할 수 있다(868). 만약, 메모리 시스템이 맵 정보를 전송하고자 하지만, 메모리 시스템이 호스트에 맵핑 정보를 전송하는 것과 관련한 통지가 사전에 이루어지지 않았다면(868의 NO), 메모리 시스템은 응답(RESPONSE)에 통지(NOTICE)를 추가하여 호스트에 전달할 수 있다(870).

[0123] 만약 맵 정보를 전송하기 위한 통지(NOTICE)가 이미 이루어진 경우(868의 YES), 메모리 시스템은 응답에 맵핑 정보를 인코딩한 후, 추가할 수 있다(872). 이때, 메모리 시스템은 맵핑 정보의 일부를 암호화할 수 있다. 이후, 메모리 시스템은 맵핑 정보를 암호화된 전송 데이터(Encrypted HPB data)를 포함하는 응답을 전송할 수 있다(874). 도 6에서는 메모리 시스템(110)이 호스트(102)의 요청에 대한 동작을 수행한 후, 요청에 대응하는 응답에 암호화된 전송 데이터를 포함시켜, 호스트(102)에 전달한다.

[0124] 호스트는 메모리 시스템으로부터 전송되는 응답(RESPONSE), 통지를 포함하는 응답(RESPONSE WITH NOTICE), 암호화된 전송 데이터(Encrypted HPB data)를 포함하는 응답(RESPOSNE WITH MAP INFO.) 중 적어도 하나를 수신할 수 있다(842).

[0125] 호스트는 수신한 응답에 통지가 포함되어 있는 지를 확인할 수 있다(844). 만약 수신한 응답에 통지가 포함되어 있다면(844의 YES), 호스트는 이후에 전달될 수 있는 맵 정보를 수신하고 저장할 수 있도록 준비할 수 있다(846). 이후, 호스트는 이전 명령에 대응하는 응답을 확인할 수 있다(852). 예를 들어, 호스트는 응답을 확인하여, 이전 명령의 성공 또는 실패 여부를 확인할 수 있다.

[0126] 수신한 응답에 통지가 포함되지 않은 경우(844의 NO), 호스트는 응답에 맵 정보가 포함되어 있는 지를 확인할 수 있다(848). 응답에 맵 정보가 포함되어 있지 않은 경우(848의 NO), 호스트는 이전 명령에 대응하는 응답을 확인할 수 있다(852).

[0127] 수신한 응답에 암호화된 전송 데이터(Encrypted HPB data)가 포함된 경우(848의 YES), 호스트는 응답에 포함된 암호화된 전송 데이터를 논리 주소에 대응시켜 호스트 내부 저장 공간에 저장하거나, 이미 저장된 논리 주소에 대응하는 암호화된 전송 데이터를 갱신할 수 있다(850). 이후, 호스트는 이전 명령에 대응하는 응답을 확인할 수 있다(852).

[0128] 도 7은 본 발명의 일 실시예에 따른 호스트와 메모리 시스템의 제2 동작을 설명한다. 구체적으로, 도 7은 도 6에서 설명한 호스트(102)와 메모리 시스템(110)과 같이 논리 주소(LBA)와 암호화된 전송 데이터(Encrypted HPB data)를 포함하는 명령어를 전달하는 호스트와 수신하는 메모리 시스템의 구체적인 동작의 일 예에 대해서 설명한다.

[0129] 도 7를 참조하면, 호스트는 논리 주소(LBA)를 포함하는 명령(COMMAND)를 생성할 수 있다(812). 이후, 호스트는 논리 주소(LBA)에 대응하는 암호화된 전송 데이터(Encrypted HPB data)가 맵 정보에 있는 지를 확인할 수 있다(814). 암호화된 전송 데이터(Encrypted HPB data)가 없는 경우(814의 NO), 호스트는 논리 주소(LBA)를 포함하는 명령(COMMAND)을 전송할 수 있다(818).

[0130] 반면, 암호화된 전송 데이터(Encrypted HPB data)가 있는 경우(814의 YES), 호스트는 논리 주소(LBA)를 포함하는 명령(COMMAND)에 암호화된 전송 데이터(Encrypted HPB data)를 추가할 수 있다(816). 호스트는 논리 주소(LBA)와 암호화된 전송 데이터(Encrypted HPB data)를 포함하는 명령(COMMAND)을 전송할 수 있다(818).

[0131] 메모리 시스템은 외부에서 전달되는 명령을 수신할 수 있다(820). 메모리 시스템은 수신한 명령에 암호화된 전송 데이터(Encrypted HPB data)가 포함되어 있는 지를 확인할 수 있다(822). 만약 수신한 명령에 암호화된 전송 데이터(Encrypted HPB data)가 없다면(822의 NO), 메모리 시스템은 수신한 명령에 포함된 논리 주소에 대응하는 물리 주소를 탐색할 수 있다(832).

[0132] 만약, 수신한 명령에 암호화된 전송 데이터(Encrypted HPB data)가 포함되어 있다면(822의 YES), 메모리 시스템은 암호화된 전송 데이터(Encrypted HPB data)를 디코딩할 수 있다(824). 메모리 시스템은 암호화된 전송 데이터(Encrypted HPB data)를 디코딩하여 얻어진 물리 주소(PPN)가 유효한 지를 확인할 수 있다(826). 도 6에서 설명한 바와 같이, 메모리 시스템이 암호화된 전송 데이터(Encrypted HPB data)를 호스트에 전달하면, 호스트는 메모리 시스템이 전달한 맵 정보를 바탕으로 맵핑을 수행하여 암호화된 전송 데이터(Encrypted HPB data)를 명령에 포함시켜 전달할 수 있다. 하지만, 메모리 시스템이 호스트에 암호화된 전송 데이터(Encrypted HPB data)를 전달한 후 메모리 시스템이 관리하는 암호화된 전송 데이터(Encrypted HPB data)에 대응하는 맵핑 정보가 변

경, 갱신될 수 있다. 이렇듯 맵핑 정보가 더티(dirty) 상태인 경우 호스트가 전달한 물리 주소(PPN)를 그대로 사용할 수 없으므로, 메모리 시스템은 수신한 명령에 포함된 물리 주소(PPN)가 유효한 지를 판단할 수 있다. 수신한 명령에 포함된 물리 주소(PPN)가 유효한 경우(826의 YES), 메모리 시스템은 물리 주소(PPN)를 사용하여 명령에 대응하는 동작을 수행할 수 있다(830).

[0133] 수신한 명령에 포함된 물리 주소(PPN)가 유효하지 않은 경우(826의 NO), 메모리 시스템은 수신한 명령에 포함된 물리 주소(PPN)를 무시할 수 있다(828). 이 경우, 메모리 시스템은 수신한 명령에 포함된 논리 주소(LBA)를 바탕으로 물리 주소(PPN)를 탐색할 수 있다(832).

[0134] 도 8은 본 발명의 실시 예에 따라 컨트롤러와 호스트가 송수신하는 데이터를 설명한다.

[0135] 도 8을 참조하면, 메모리 시스템(110)에 포함된 호스트 성능 부스터(HPB)는 소스 데이터(SD)의 일부를 암호화하여 암호화된 전송 데이터(HPB data)를 생성하고, 암호화된 전송 데이터(HPB data)를 호스트(102)에 출력할 수 있다. 호스트(102)는 메모리 시스템(110)에서 출력된 암호화된 전송 데이터(HPB data)를 호스트 메모리(H_MEM)에 저장할 수 있다.

[0136] 소스 데이터(SD)는 메모리 시스템(110)에서 어드레스들에 대한 맵핑 정보(Mapping Data, MD), 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(physical page number; PPN)를 포함할 수 있다. 맵핑 정보(MD)는 물리 페이지 넘버(PPN)를 토대로 연속된 논리 블록 어드레스(logical block address; LBA)의 개수를 나타낼 수 있다. 맵핑 시간 데이터(TD)는 맵핑 정보(MD)가 생성된 시간을 나타낼 수 있다. 실시예에 따라, 맵핑 시간 데이터(TD)는 도 7에서 물리 주소(PPN)의 유효성을 판단하기 위한 정보로 사용될 수 있다. 맵핑 시간 데이터(TD)는 물리 페이지 넘버(PPN)는 메모리 장치(150) 내 프로그램 동작 시 데이터가 저장된 위치를 가리키는 주소일 수 있다. 예를 들면, 호스트(102)가 메모리 시스템(110)에게 데이터와 함께 프로그램 요청을 전송하면, 메모리 시스템(110)는 메모리 장치(150)에 포함된 메모리 블록들 중에서 하나의 메모리 블록을 선택하고, 선택된 메모리 블록의 복수의 페이지들에 데이터를 프로그램할 수 있다. 따라서 물리 페이지 넘버(PPN)는 데이터가 저장된 메모리 블록 및 페이지의 주소들을 포함할 수 있다. 암호화된 전송 데이터(HPB data)와 함께 전달되는 논리 블록 어드레스(LBA)는 데이터가 저장된 영역에 대한 물리 페이지 넘버(PPN)에 대응되는 논리 어드레스로써, 호스트(102)에서 사용되는 주소일 수 있다.

[0137] 도 8을 참조하면, 메모리 시스템(110)는 호스트(102)가 요청한 프로그램 동작이 완료되면, 데이터가 프로그램된 물리 페이지 넘버(PPN)와, 물리 페이지 넘버(PPN)에 맵핑된 논리 블록 어드레스(LBA)를 호스트(102)에게 전송할 수 있다. 이때, 맵핑된 어드레스 외에도 맵핑에 관련된 다양한 정보들을 호스트(102)에게 전송할 수 있다.

[0138] 본 실시 예에서, 메모리 시스템(110)는 맵핑 정보(MD), 맵핑 시간 데이터(TD), 물리 페이지 넘버(PPN) 및 논리 블록 어드레스(LBA)를 호스트(102)에게 전송하지만, 이 외에도 다양한 데이터들을 호스트(102)에게 전송할 수 있다. 이 중에서, 논리 블록 어드레스(LBA)는 호스트(102)가 사용하는 데이터이므로 메모리 시스템(110)는 논리 블록 어드레스(LBA)를 암호화하지 아니하고 출력한다. 맵핑 정보(MD), 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)는 메모리 시스템(110)에서 사용되는 데이터이므로, 메모리 시스템(110)는 맵핑 정보(MD), 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)를 암호화하여 호스트(102)에게 출력할 수 있다. 하지만, 이 중에서 일부 데이터를 호스트(102)가 사용할 수도 있으므로, 본 실시 예에 따른 메모리 시스템(110)는 맵핑 정보(MD), 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)를 포함하는 소스 데이터(SD)에서 일부 데이터만 암호화하고, 나머지 데이터는 원본 그대로 호스트(102)에게 출력할 수 있다.

[0139] 실시예에 따라, 메모리 시스템(110)은 호스트(102)가 논리 주소(LBA)에 대응하여 참조할 수 있는 정보인 맵핑 정보(MD)는 암호화하지 않고, 호스트(102)가 참조할 필요 없이 저장한 후 읽기 요청에 덧붙여 전송하는 데이터인 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)는 암호화할 수 있다. 예를 들면, 메모리 시스템(110)가 소스 데이터(SD)에서 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PNN)를 선택적으로 암호화하고 맵핑 정보(MD)를 원본 그대로 유지하여 호스트(102)에게 출력하는 경우, 메모리 시스템(110)는 부분 암호화 동작을 수행하여 소스 데이터(SD) 중에서 일부 데이터만 암호화된 암호화된 전송 데이터(HPB data)를 출력할 수 있다.

[0140] 암호화된 전송 데이터(HPB data)는 순환 중복 검사 값(cyclic redundancy check value; CRCV), 맵핑 정보(MD) 및 암호화된 데이터(EN_D)를 포함할 수 있다. 여기서, 암호화된 데이터(EN_D)는 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PNN)가 암호화된 데이터일 수 있다. 메모리 시스템(110)는 소스 데이터(SD)를 부분 암호화하여 암호화된 전송 데이터(HPB data)를 호스트(102)에게 출력할 때, 물리 페이지 넘버(PNN)에 대응되는 논리 블록 어드레스(LBA)도 호스트(102)에게 출력할 수 있다.

[0141] 호스트(102)는 논리 블록 어드레스(LBA)의 리드 동작을 위하여 리드 요청(read request)을 메모리 시스템(110)에게 출력할 때, 논리 블록 어드레스(LBA)에 대응되는 암호화된 전송 데이터(HPB data) 및 논리 블록 어드레스(LBA)를 메모리 시스템(110)에게 출력할 수 있다. 메모리 시스템(110)는 호스트(102)에서 출력된 암호화된 전송 데이터(HPB data)를 디코딩하여 소스 데이터(SD)로 복원할 수 있다.

[0142] 실시예에 따라, 전술한 부분 암호화 동작을 수행할 수 있는 메모리 시스템(110)의 호스트 성능 부스터(HPB)는 인코더(encoding circuitry, 312) 및 디코더(decoding circuitry, 314)를 포함할 수 있다. 인코더(312)는 소스 데이터(SD)를 인코딩하여 암호화된 전송 데이터(HPB data)로 변경할 수 있고, 디코더(314)는 암호화된 전송 데이터(HPB data)를 디코딩하여 소스 데이터(SD)로 복원할 수 있다. 이하에서는 도 9 내지 도 14를 참조하며, 메모리 시스템(110) 내 호스트 성능 부스터(HPB, 310)에 포함된 인코더(312) 및 디코더(314)를 구체적으로 설명한다.

[0143] 도 9는 본 발명의 일 실시예에 따른 메모리 시스템에서 데이터 인코딩 동작을 설명한다. 인코딩 동작(312A)은 도 8에서 설명하는 인코더(312)의 동작에 대한 일 예일 수 있다.

[0144] 도 9를 참조하면, 인코더(312)는 맵핑 정보를 암호화하기 위해 소스 데이터(SD)와 논리 주소(LBA)를 사용할 수 있다. 소스 데이터(SD)는 12비트의 맵핑 정보(MD), 4비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)를 포함할 수 있다. 논리 주소는 32비트의 길이를 가질 수 있다. 도 9에 도시된 복수의 데이터에 대한 비트 정보(길이 정보)는 메모리 시스템(110) 혹은 메모리 장치(150)의 내부 구성에 따라 달라질 수 있다.

[0145] 먼저, 인코딩 동작(312A)은 32비트의 논리 주소 및 시드(seed)에 기반하여 64비트의 난수를 발생시키는 랜덤화(randomize) 과정을 포함한다. 64비트의 난수는 마스킹 과정을 통해 36비트 암호화 데이터(552)로 변경된다. 이때, 64비트의 난수에서 28비트의 일부분은 사용되지 않을 수 있다. 암호 데이터(552)를 36비트로 마스킹 하는 이유는 암호화되는 부분인 4 비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)가 36비트의 길이를 가지기 때문이다.

[0146] 이후, 암호화된 데이터(522)와 소스 데이터(SD)의 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)에 대해 논리 연산을 수행한다. 여기서 논리 연산은 배타적 논리합(XOR) 연산일 수 있다. 배타적 논리합(XOR) 연산을 통해 도 8에서 설명한 암호화된 데이터(EN_D)를 생성할 수 있다.

[0147] 논리 연산 후, 인코딩 동작(312A)은 순환 중복 검사(CRC)를 수행하여 16비트의 순환 중복 검사 결과를 생성하는 과정을 포함한다. 이를 통해, 16비트의 순환 중복 검사 결과, 암호화되지 않은 12비트의 맵핑 정보(MD) 및 36비트의 암호화된 데이터(mixed encryption)을 포함하는 암호화된 전송 데이터(HPB_D)가 생성될 수 있다. 도 9에서 설명한 36비트의 암호화된 데이터(mixed encryption)와 도 8에서 설명한 암호화된 데이터(EN_D)는 실질적으로 동일할 수 있다.

[0148] 도 10은 본 발명의 일 실시예에 따른 메모리 시스템에서 데이터 디코딩 동작을 설명한다. 디코딩 동작(314A)은 도 8에서 설명하는 디코더(314)의 동작에 대한 일 예일 수 있다.

[0149] 도 10을 참조하면, 호스트(102)로부터 전달된 논리 주소(LBA)를 이용하여 암호화된 전송 데이터(HPB data)를 디코딩할 수 있다.

[0150] 먼저, 디코딩 동작(314A)은 호스트(102)로부터 전달된 논리 주소(LBA) 및 시드(seed)에 기반하여 64비트의 난수를 발생시키는 랜덤화(randomize) 과정을 포함한다. 64비트의 난수는 마스킹 과정을 통해 36비트 암호화 데이터(552)로 변경된다. 이때, 64비트의 난수에서 24비트의 일부분은 사용되지 않을 수 있다. 디코딩 동작(314A)에서도 논리 주소에 대응하여 인코딩 동작(312A)에서 생성되었던 암호화 데이터(552)를 동일하게 생성할 수 있다.

[0151] 16비트의 순환 중복 검사 결과, 암호화되지 않은 12비트의 맵핑 정보(MD) 및 36비트의 암호화된 데이터(mixed encryption)을 포함하는 암호화된 전송 데이터(HPB_D)가 수신되면, 디코더(314)는 순환 중복 검사(CRC)를 수행할 수 있다. 순환 중복 검사(CRC)를 통해 암호화된 전송 데이터(HPB_D)이 송수신 과정에서 정상적으로 전달되었는 지를 확인할 수 있다.

[0152] 순환 중복 검사를 통해 암호화된 전송 데이터(HPB_D)에 오류가 없다고 판단되면, 암호화 데이터(552)와 암호화된 전송 데이터(HPB_D)에 포함된 36비트의 암호화된 데이터(mixed encryption)에 논리 연산을 수행할 수 있다. 여기서 논리 연산은 배타적 논리합(XOR) 연산일 수 있다. 배타적 논리합(XOR) 연산을 통해 36비트의 암호화된 데이터(mixed encryption)로부터 4 비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)를 확보할 수 있다.

[0153] 도 9 및 도 10을 참조하면, 논리 주소를 랜덤화하여 각 논리 주소에 대응하는 암호화 데이터(552)를 사용하여 단순한 논리 연산을 통해 맵핑 정보 중 일부를 암호화하는 인코딩과 암호화를 해제하는 디코딩을 수행할 수 있다.

[0154] 도 9을 참조하면, 논리 주소(LBA)와 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)들을 암호화(encryption)하여 호스트(102)로 전송되는 암호화된 전송 데이터(HPB_D)를 생성할 수 있다. 일부 암호화를 통해, 암호화, 비암호화 정보를 선택함으로써, 암호화된 전송 데이터(HPB_D)를 수신하는 호스트(102) 및 메모리 시스템(110)이 암호화되지 않은 정보를 유용하게 활용할 수 있다. 예를 들어, 12비트의 맵핑 정보(MD)를 통해, 미리 물리 주소(PPN)가 연속적임을 알고 요청 혹은 명령(command)에 대응하는 동작을 준비할 수 있다.

[0155] 하지만, 도 9 및 도 10에서 설명한 인코딩 및 디코딩을 수행하는 메모리 시스템(110)에서는 호스트(102)의 내부 동작 중 오류 등으로 인해, 논리 주소에 연관되는 암호화된 전송 데이터(HPB_D)가 잘못될 가능성이 있다. 논리 주소에 대응하여 전송된 암호화된 전송 데이터(HPB_D)가 잘못 연관되어 암호화된 전송 데이터(HPB_D)에 포함되는 경우, 메모리 시스템(110)은 암호화된 전송 데이터(HPB_D)를 디코딩하는 과정을 통해 오류를 발견하기 어려울 수 있다. 예를 들어, 호스트(102)는 논리 주소에 대응하는 12비트의 맵핑 정보(MD) 및 36비트의 암호화된 데이터(mixed encryption)를 바탕으로 순환 중복 검사의 결과를 더하여 암호화된 전송 데이터(HPB_D)르 메모리 시스템(110)에 전송할 수 있다. 순환 중복 검사의 결과를 바탕으로 호스트(102)와 메모리 시스템(110) 사이의 데이터 통신 과정에서 발생한 에러를 확인할 수 있으나, 논리 주소에 대응하는 12비트의 맵핑 정보(MD) 혹은 36비트의 암호화된 데이터(mixed encryption)가 잘못 매칭된 경우에는 순환 중복 검사를 통해서 에러를 확인할 수 없다. 이 경우, 도 7을 참조하여 디코딩을 통해 확보된 물리 주소가 최신 맵핑 정보라고 판단된다면, 메모리 시스템(110)은 논리 주소와 연관 없는 물리 주소를 이용하여 데이터를 읽을 수도 있는 오동작이 일어날 수 있다.

[0156] 또한, 도 9 및 도 10에서 설명한 인코딩 및 디코딩에서 32비트 논리 주소를 랜덤화하여 64비트의 난수를 생성한 후, 36비트에 대응하는 마스킹 동작을 통해 28비트의 난수를 사용하지 않고 있다. 랜덤화 과정을 통해 생성된 난수의 일부를 사용하고 다른 일부를 사용하지 않기 때문에, 랜덤화 과정에서 지원할 수 있는 랜덤 특성을 모두 활용하지 못할 수 있다. 랜덤 특성이 줄어드는 경우 암호화 데이터(552)가 중복된 값을 가질 수 있는 가능성이 높아질 수 있다. 암호화 데이터(552)가 중복된 값을 가질 경우, 암호화가 가져오는 효과를 줄일 수 있다.

[0157] 도 11은 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 인코딩 동작을 설명한다. 인코딩 동작(312B)은 도 8에서 설명하는 인코더(312)의 동작에 대한 다른 예일 수 있다.

[0158] 도 11을 참조하면, 인코더(312)는 맵핑 정보를 암호화하기 위해 소스 데이터(SD)와 논리 주소(LBA)를 사용할 수 있다. 소스 데이터(SD)는 12비트의 맵핑 정보(MD), 4비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)를 포함할 수 있다. 논리 주소는 32비트의 길이를 가질 수 있다. 도 11에 도시된 복수의 데이터에 대한 비트 정보(길이 정보)는 메모리 시스템(110) 혹은 메모리 장치(150)의 내부 구성에 따라 달라질 수 있다.

[0159] 먼저, 인코딩 동작(312B)은 32비트의 논리 주소 및 시드(seed)에 기반하여 64비트의 난수를 발생시키는 랜덤화(randomize) 과정을 포함한다. 64비트의 난수는 마스킹 과정을 통해 36비트의 제1 암호화 데이터(552)와 28비트의 제2 암호화 데이터(554)로 구분할 수 있다. 이후, 36비트의 제1 암호화 데이터(552)와 28비트의 제2 암호화 데이터(554)에 대해 논리 연산을 수행할 수 있다. 도 11을 참조하면, 36비트의 제1 암호화 데이터(552)와 28비트의 제2 암호화 데이터(554)에 대해 배타적 논리합(XOR) 연산을 수행하여, 36비트의 제3 암호화 데이터(556)를 생성할 수 있다. 제3 암호 데이터(556)를 36비트로 생성하는 이유는 암호화되는 부분인 4 비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)가 36비트의 길이를 가지기 때문이다.

[0160] 12비트의 맵핑 정보(MD), 4비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)를 포함하는 소스 데이터(SD)에 대해 순환 중복 검사를 수행하여 순환 중복 검사의 결과를 생성할 수 있다.

[0161] 이후, 제3 암호화된 데이터(526)와 소스 데이터(SD)의 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)에 대해 논리 연산을 수행한다. 여기서 논리 연산은 배타적 논리합(XOR) 연산일 수 있다. 배타적 논리합(XOR) 연산을 통해 도 8에서 설명한 암호화된 데이터(EN_D)를 생성할 수 있다. 이를 통해, 16비트의 순환 중복 검사 결과, 암호화되지 않은 12비트의 맵핑 정보(MD) 및 36비트의 암호화된 데이터(mixed encryption)을 포함하는 암호화된 전송 데이터(HPB_D)가 생성될 수 있다. 도 11에서 설명한 36비트의 암호화된 데이터(mixed encryption)와 도 8에서 설명한 암호화된 데이터(EN_D)는 실질적으로 동일할 수 있다.

[0162] 도 12는 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 인코딩 방법을 설명한다.

[0163] 도 12를 참조하면, 데이터 인코딩 방법은 호스트로 전송할 HPB 데이터를 결정하는 단계(930), 호스트로 전송할 HPB 데이터에 대해 순환 중복 검사를 수행하는 단계(932), 논리 주소를 기반으로 암호화 데이터를 추출하는 단계(934), 및 HPB 데이터와 암호화 데이터를 논리 연산하여 암호화된 HPB 데이터를 생성하는 단계(936)를 포함할 수 있다.

[0164] 여기서, 논리 주소를 기반으로 암호화 데이터를 추출하는 단계(934)는 도 11에서 설명한 것과 같이 32비티의 논리 주소를 랜덤화 과정을 통해 64비트의 난수를 생성한 후, 64비트의 난수를 마스킹 동작을 통해 복수의 암호화 데이터로 분류한 후, 복수의 암호화 데이터에 대해 논리 연산을 수행하는 동작을 포함할 수 있다. 랜덤화 과정을 통해 생성한 난수의 일부를 사용하지 않는 방법(도 9 참조)에 비하여 난수의 전부를 이용하는 방법(도 11 및 도 12 참조)은 랜덤화 과정을 통한 랜덤 특성이 줄어드는 것을 감소시킬 수 있다.

[0165] 또한, HPB 데이터와 암호화 데이터를 논리 연산하는 것(936)은 도 11에서 설명하는 3 암호화된 데이터(526)와 소스 데이터(SD)의 맵핑 시간 데이터(TD) 및 물리 페이지 넘버(PPN)에 대해 배타적 논리합(XOR)을 수행하는 것에 대응할 수 있다.

[0166] 도 12에서 설명하는 데이터 인코딩 방법은 메모리 시스템(110)이 호스트(102)에 전송하기 위한 맵핑 정보가 결정되면, 맵핑 정보의 일부를 암호화하기 전에 순환 중복 검사를 먼저 수행하고 순환 중복 검사 이후 일부 데이터에 대해 암호화를 수행할 수 있다. 도 9에서 설명한 인코딩 동작(312A)의 경우 일부 데이터에 대해 암호화를 수행한 후 순환 중복 검사를 수행한다. 이러한 차이로 인하여 암호화된 HPB 데이터를 디코딩하는 과정에서 메모리 시스템(110)이 확인할 수 있는 오류, 에러가 달라질 수 있다.

[0167] 도 13은 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 디코딩 동작을 설명한다.

[0168] 도 13을 참조하면, 호스트(102)로부터 전달된 논리 주소(LBA)를 이용하여 암호화된 전송 데이터(HPB data)를 디코딩할 수 있다.

[0169] 먼저, 디코딩 동작(314B)은 호스트(102)로부터 전달된 논리 주소(LBA) 및 시드(seed)에 기반하여 64비트의 난수를 발생시키는 랜덤화(randomize) 과정을 포함한다. 64비트의 난수는 마스킹 과정을 통해 36비트의 제1 암호화 데이터(552)와 28비트의 제2 암호화 데이터(554)로 구분할 수 있다. 이후, 36비트의 제1 암호화 데이터(552)와 28비트의 제2 암호화 데이터(554)에 대해 논리 연산을 수행할 수 있다. 도 13을 참조하면, 36비트의 제1 암호화 데이터(552)와 28비트의 제2 암호화 데이터(554)에 대해 배타적 논리합(XOR) 연산을 수행하여, 36비트의 제3 암호화 데이터(556)를 생성할 수 있다. 제3 암호 데이터(556)를 36비트로 생성하는 이유는 암호화되는 부분인 4비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)가 36비트의 길이를 가지기 때문이다.

[0170] 도 10에서 설명한 디코딩 동작(314A)과 달리, 도 13에서 설명하는 디코딩 동작(314B)에서는 랜덤화 과정을 통해 생성된 64비트의 난수의 일부를 버리지 않고 논리 연산을 위한 입력으로 사용함으로써, 제3 암호화 데이터(556)의 랜덤 특성이 제1 암호화 데이터(552)의 랜덤 특성보다 개선될 수 있다.

[0171] 디코딩 동작(314B)은 제3 암호화 데이터(556)와 암호화된 전송 데이터(HPB_D) 내 36비트의 암호화된 데이터(mixed encryption)에 대해 논리 연산을 수행할 수 있다. 이때, 논리 연산은 배타적 논리합(XOR)일 수 있다. 논리 연산을 통해 암호화된 데이터(mixed encryption)로부터 4비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)를 얻을 수 있다.

[0172] 이후, 디코더(314)는 순환 중복 검사(CRC)를 수행할 수 있다. 순환 중복 검사(CRC)를 통해 암호화된 전송 데이터(HPB_D)이 송수신 과정에서 정상적으로 전달되었는 지를 확인할 수 있을 뿐만 아니라, 논리 주소와 32비트의 물리 페이지 넘버(PPN)의 연관성을 확인할 수도 있다. 도 10에서 설명한 디코딩 동작(314A)에서 수행하는 순환 중복 검사(CRC)는 호스트(102)와 메모리 시스템(110) 사이에 암호화된 전송 데이터(HPB_D)의 송수신 과정에서 오류가 있는 지를 확인할 수 있으나, 호스트(102)의 내부 동작에 오류 등으로 인하여 논리 주소와 연관되어 메모리 시스템(110)에 전송되는 암호화된 전송 데이터(HPB_D)가 잘못 매칭되는 경우 연관성을 확인하기는 어렵다. 하지만, 도 13에서 설명하는 디코딩 동작(314B)에서는 36비트의 암호화된 데이터(mixed encryption)가 아닌 4비트의 맵핑 시간 데이터(TD) 및 32비트의 물리 페이지 넘버(PPN)에 대해 순환 중복 검사(CRC)를 수행하기 때문에, 32비트의 물리 페이지 넘버(PPN)가 논리 주소와 연관성이 없는 경우 순환 중복 검사(CRC)를 통해 확인될 수 있다.

[0173] 디코더(314)는 순환 중복 검사(CRC)를 통해 32비트의 물리 페이지 넘버(PPN)와 논리 주소의 연관성이 확인되면, 메모리 시스템(110)은 32비트의 물리 페이지 넘버(PPN)에 대응하는 4비트의 맵핑 시간 데이터(TD)를 바탕으로

최신 맵핑 정보인지를 확인하는 과정을 통해 32비트의 물리 페이지 넘버(PPN)의 유효성을 최종적으로 결정할 수 있다.

[0174] 도 14는 본 발명의 다른 실시예에 따른 메모리 시스템에서 데이터 디코딩 방법을 설명한다. 실시예에 따라, 데이터 디코딩 방법은 도 7에서 설명하는 암호화된 전송 데이터(Encrypted HPB data)를 디코딩하는 단계(824)의 구체적인 과정을 설명할 수 있다.

[0175] 도 14를 참조하면, 데이터 디코딩 방법은 호스트로부터 암호화된 HPB 데이터를 수신하는 단계(910), 논리 주소를 기반으로 암호화 데이터를 추출하는 단계(912), 암호화된 HPB 데이터와 암호화 데이터를 논리 연산하는 단계(914), 순환 중복 검사를 수행하는 단계(916), 및 순환 중복 검사에서 에러가 없는 경우, 물리 주소를 결정하는 단계(918)를 포함할 수 있다.

[0176] 데이터 디코딩 방법은 메모리 시스템(110)이 호스트(102)로부터 암호화된 HPB 데이터를 수신하는 것으로 시작할 수 있다(910). 암호화된 HPB 데이터는 논리 주소와 요청 혹은 명령(예, 읽기 요청)과 함께 메모리 시스템(110)에 전달될 수 있다. 이 후, 메모리 시스템(110) 내 디코더(314)는 암호화된 HPB 데이터를 디코딩할 수 있다.

[0177] 먼저, 메모리 시스템(110)은 인코딩 과정과 동일하게 암호화된 HPB 데이터와 연관된 논리 주소를 기반으로 암호화 데이터를 추출할 수 있다(912). 암호화 데이터는 논리 주소를 기초로 생성되며, 실시예에 따라 도 13에서 설명한 바와 같이 랜덤화 과정, 마스킹 과정 및 논리 연산을 통해 생성될 수 있다.

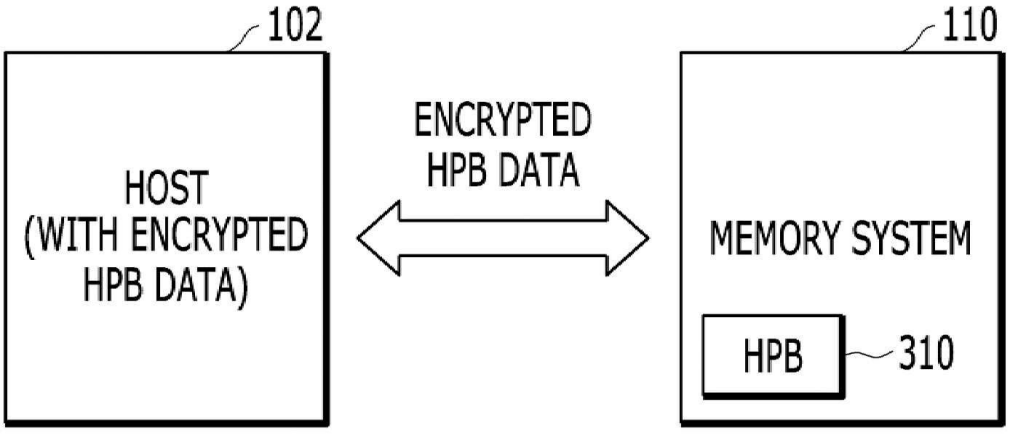
[0178] 데이터 디코딩 방법은 암호화 데이터를 생성한 후, 암호화된 HPB 데이터 내 암호화된 부분에 대해 암호화를 해제할 수 있다. 암호화된 HPB 데이터 내 일부 암호화된 영역에 암호화를 해제하는 것은 암호화 데이터를 이용한 논리 연산을 통해 가능할 수 있다(914).

[0179] 암호화된 HPB 데이터 내 암호화된 데이터를 원래의 데이터로 복원한 후, 디코더(314)는 순환 중복 검사를 수행할 수 있다(916). 암호화를 해제한 후 순환 중복 검사를 수행함으로써, 복원된 데이터와 논리 주소의 연관성 여부(즉, 물리 페이지 넘버(PPN)가 논리 주소와 연관되는 지)를 확인할 수 있다. 순환 중복 검사를 통해 오류가 없는 경우, 디코더(314)는 호스트(102)로부터 전달된 물리 페이지 넘버(PPN)를 결정할 수 있다.

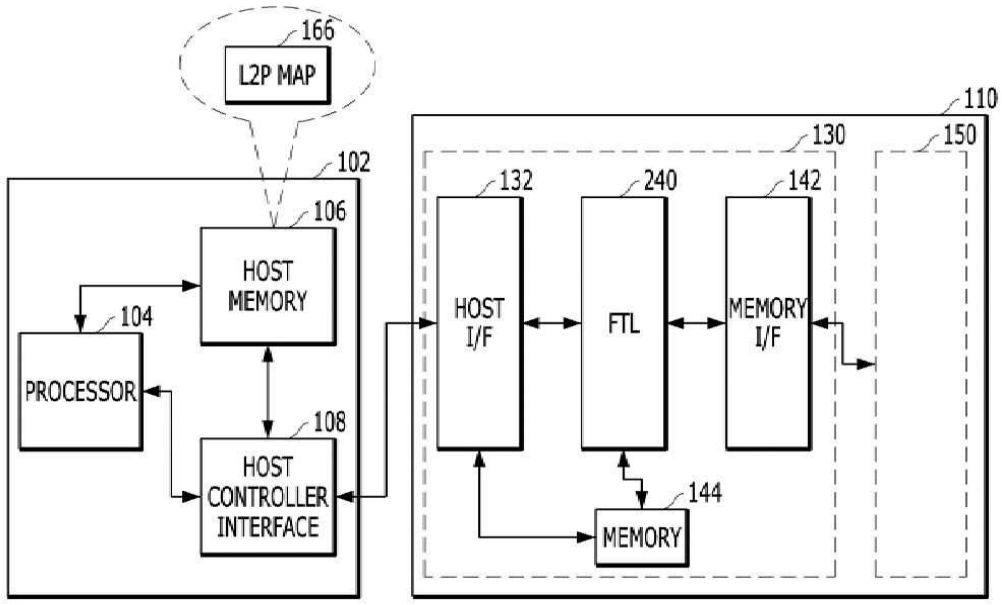
[0180] 한편, 본 발명의 상세한 설명에서는 구체적인 실시 예에 관해 설명하였으나, 본 발명의 범위에서 벗어나지 않는 한도 내에서 여러 가지 변형이 가능함은 물론이다. 그러므로, 본 발명의 범위는 설명된 실시 예에 국한되어 정해져서는 안되며 후술하는 특허청구의 범위뿐만 아니라 이 특허청구의 범위와 균등한 것들에 의해 정해져야 한다.

도면

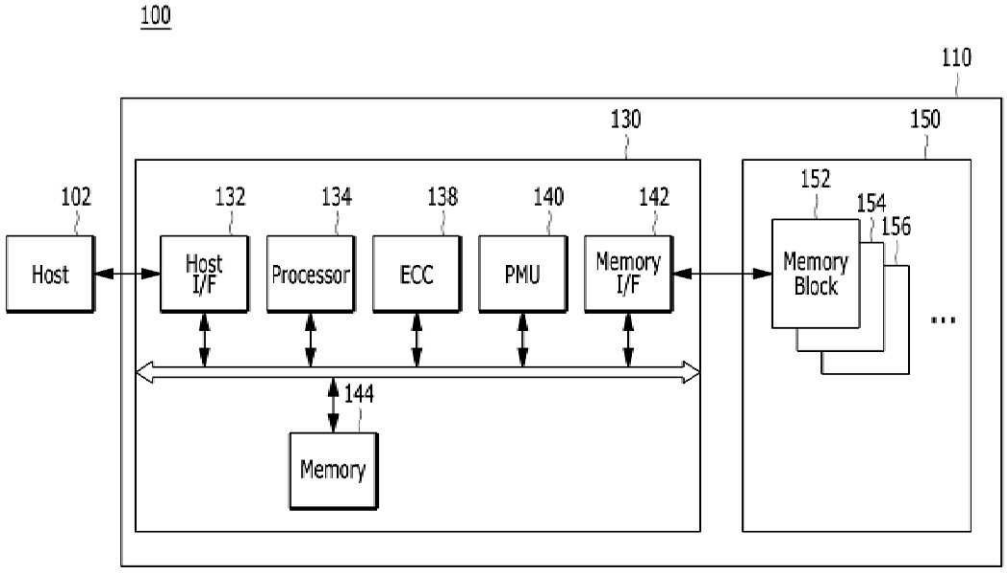
도면1



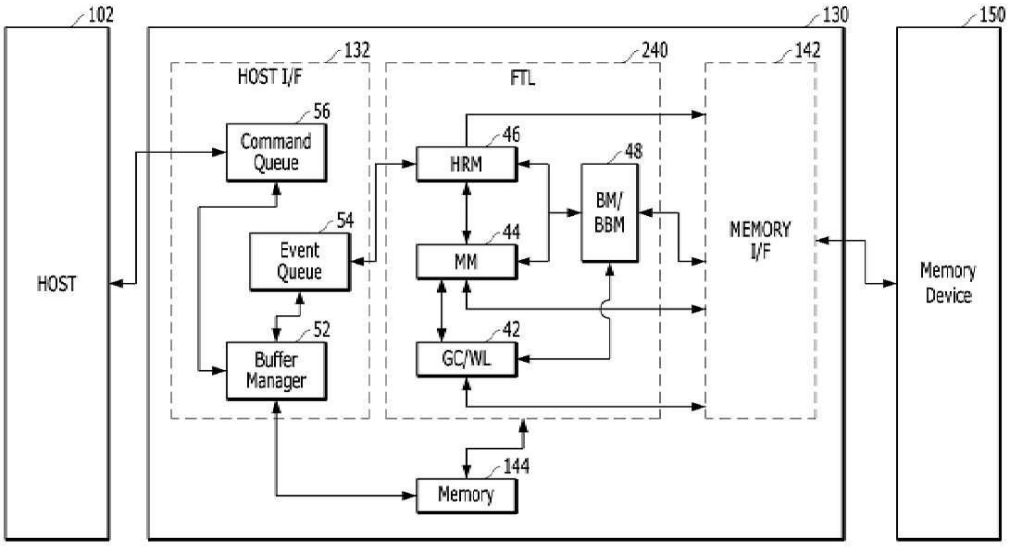
도면2


166
L2P MAP
102
106
HOST
MEMORY
104
PROCESSOR
108
HOST
CONTROLLER
INTERFACE
110
130
150
132
240
142
HOST
I/F
FTL
MEMORY
I/F
144
MEMORY

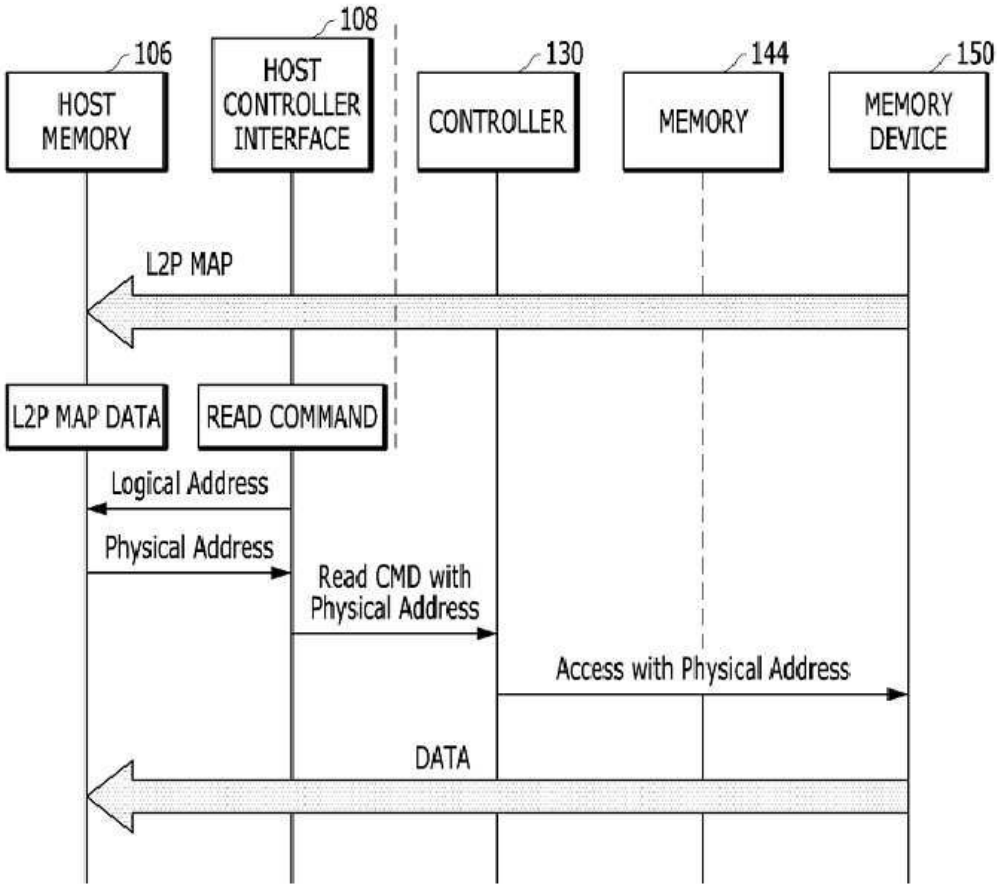
도면3


100
110
130
150
102
132
134
138
140
142
152
154
156
Host
Host
I/F
Processor
ECC
PMU
Memory
I/F
Memory
Block
...
144
Memory

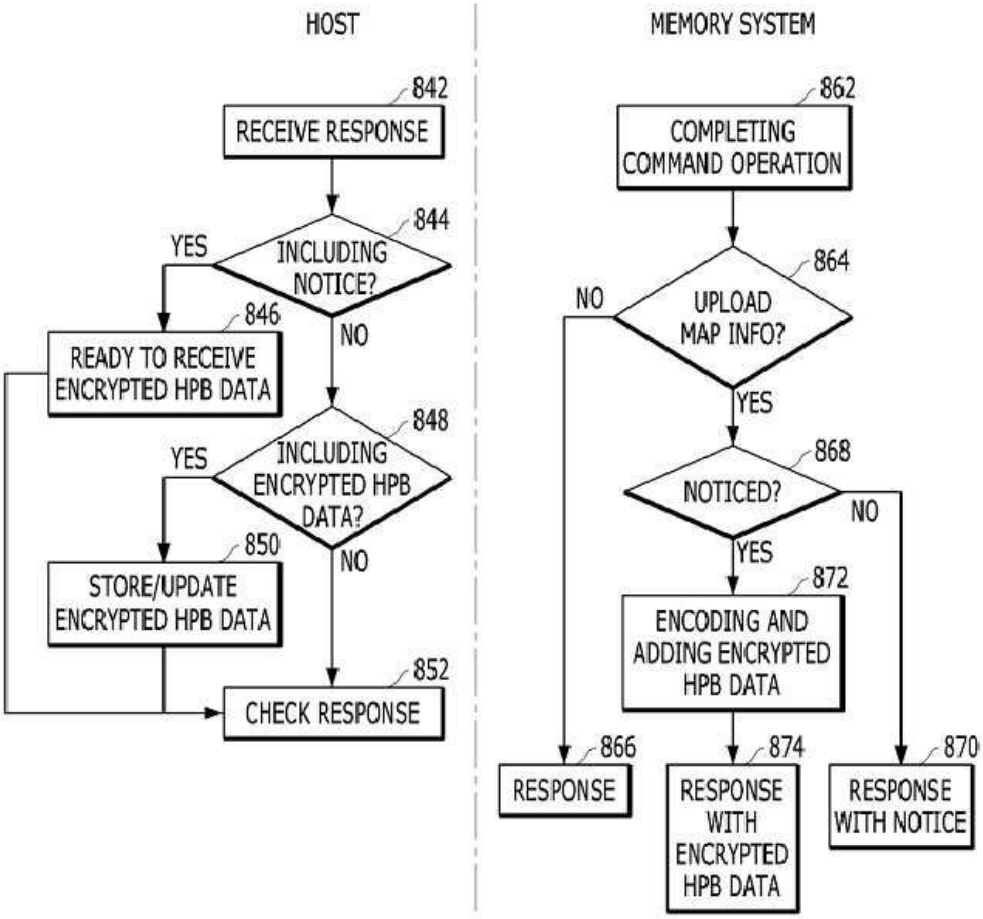
도면4


102
130
150
132
240
142
HOST I/F
FTL
56
Command Queue
46
HRM
48
BM/ BBM
54
Event Queue
44
MM
MEMORY I/F
HOST
Memory Device
52
Buffer Manager
42
GC/WL
144
Memory

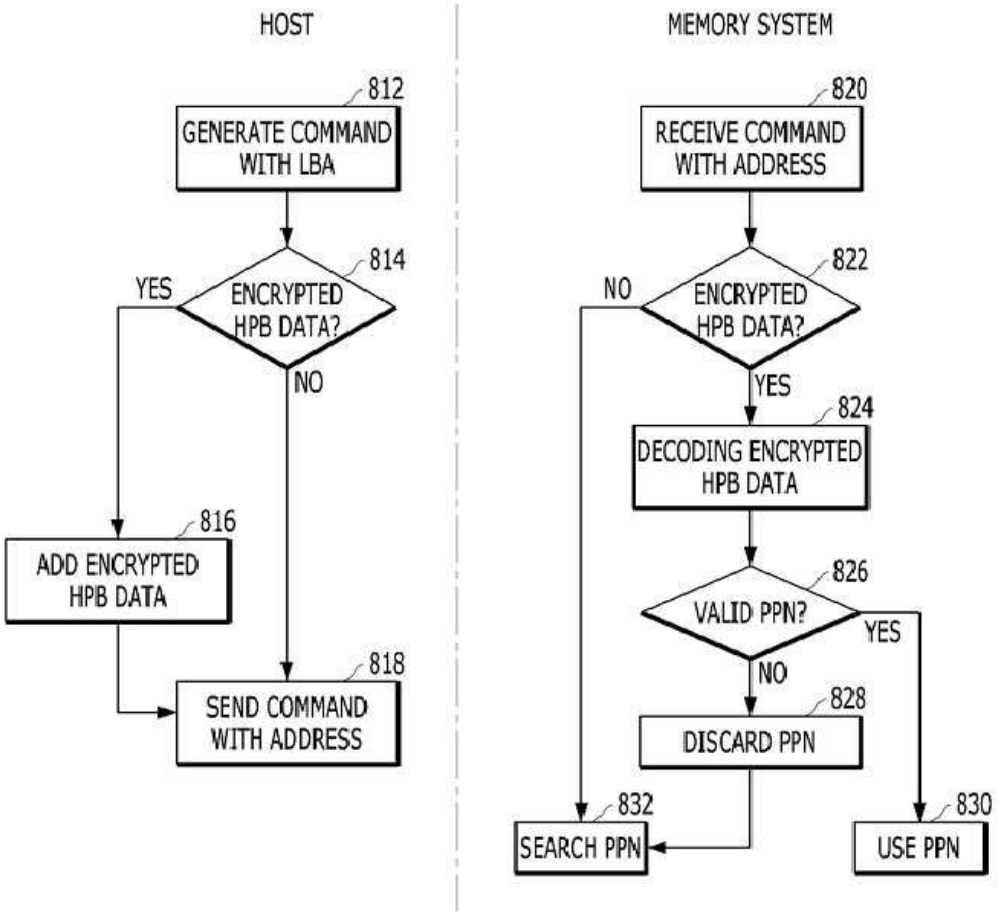
도면5


108
106
HOST MEMORY
HOST CONTROLLER INTERFACE
130
CONTROLLER
144
MEMORY
150
MEMORY DEVICE
L2P MAP
L2P MAP DATA
READ COMMAND
Logical Address
Physical Address
Read CMD with Physical Address
Access with Physical Address
DATA

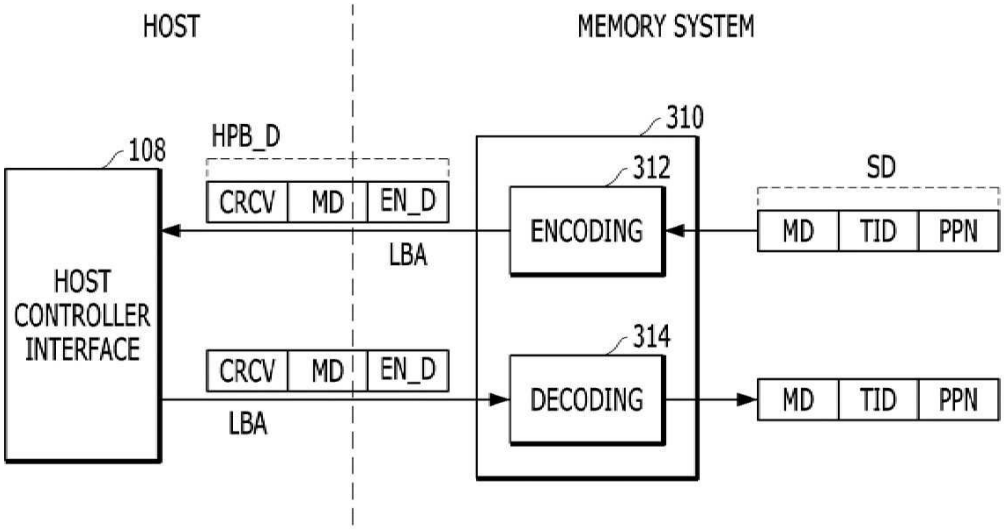
도면6


HOST
MEMORY SYSTEM
842
RECEIVE RESPONSE
844
INCLUDING NOTICE?
YES
NO
846
READY TO RECEIVE ENCRYPTED HPB DATA
848
INCLUDING ENCRYPTED HPB DATA?
YES
NO
850
STORE/UPDATE ENCRYPTED HPB DATA
852
CHECK RESPONSE
862
COMPLETING COMMAND OPERATION
864
UPLOAD MAP INFO?
NO
YES
868
NOTICED?
NO
YES
872
ENCODING AND ADDING ENCRYPTED HPB DATA
866
RESPONSE
874
RESPONSE WITH ENCRYPTED HPB DATA
870
RESPONSE WITH NOTICE

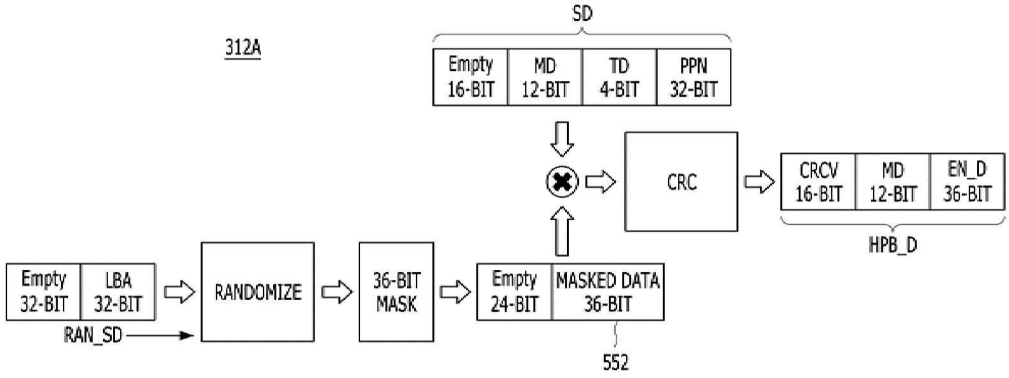
도면7


HOST
MEMORY SYSTEM
812
GENERATE COMMAND WITH LBA
814
ENCRYPTED HPB DATA?
YES
NO
816
ADD ENCRYPTED HPB DATA
818
SEND COMMAND WITH ADDRESS
820
RECEIVE COMMAND WITH ADDRESS
822
ENCRYPTED HPB DATA?
NO
YES
824
DECODING ENCRYPTED HPB DATA
826
VALID PPN?
YES
NO
828
DISCARD PPN
832
SEARCH PPN
830
USE PPN

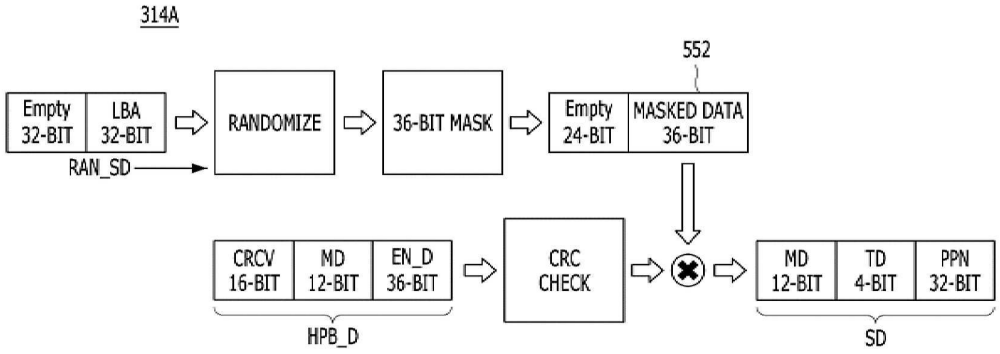
도면8


HOST
MEMORY SYSTEM
108
HOST CONTROLLER INTERFACE
HPB_D
CRCV
MD
EN_D
LBA
310
312
ENCODING
SD
MD
TID
PPN
314
DECODING

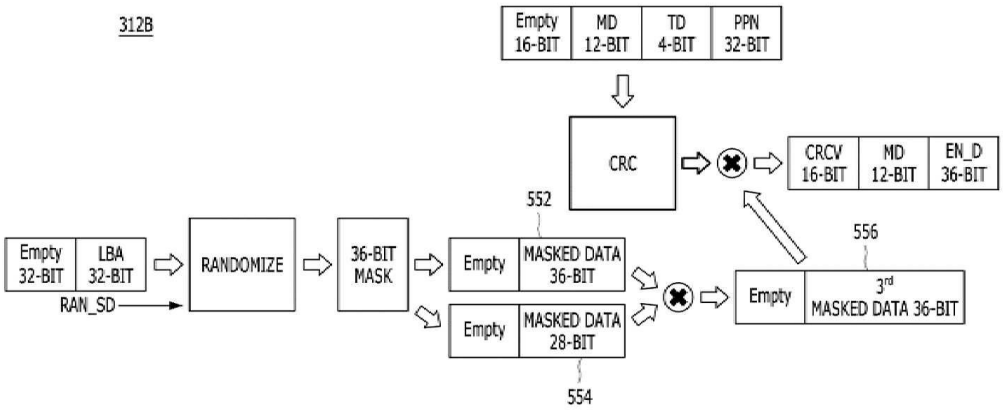
도면9



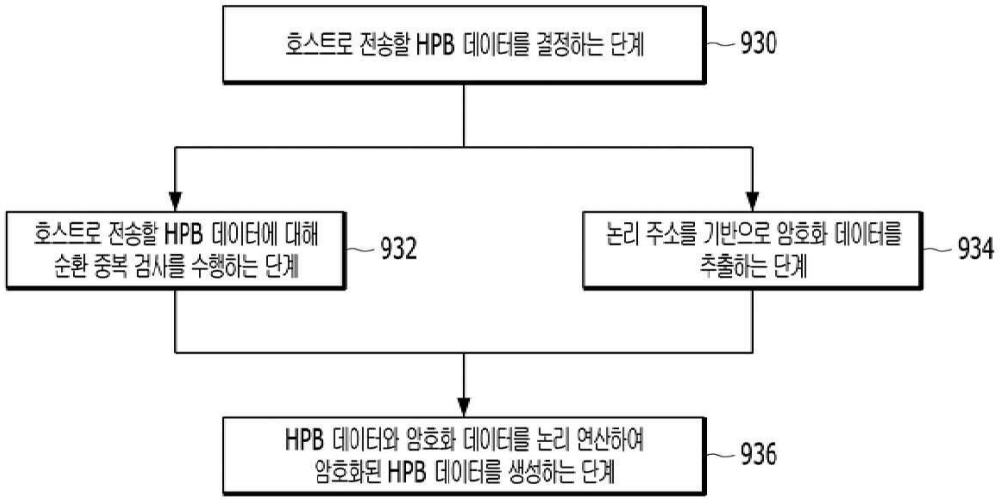
도면10



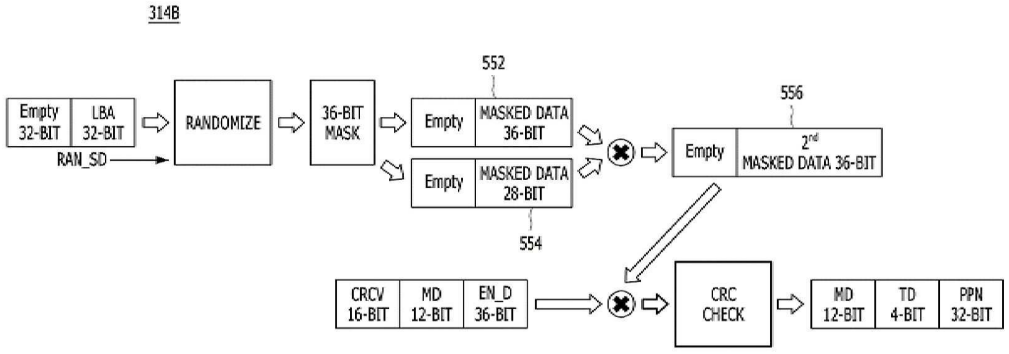
도면11



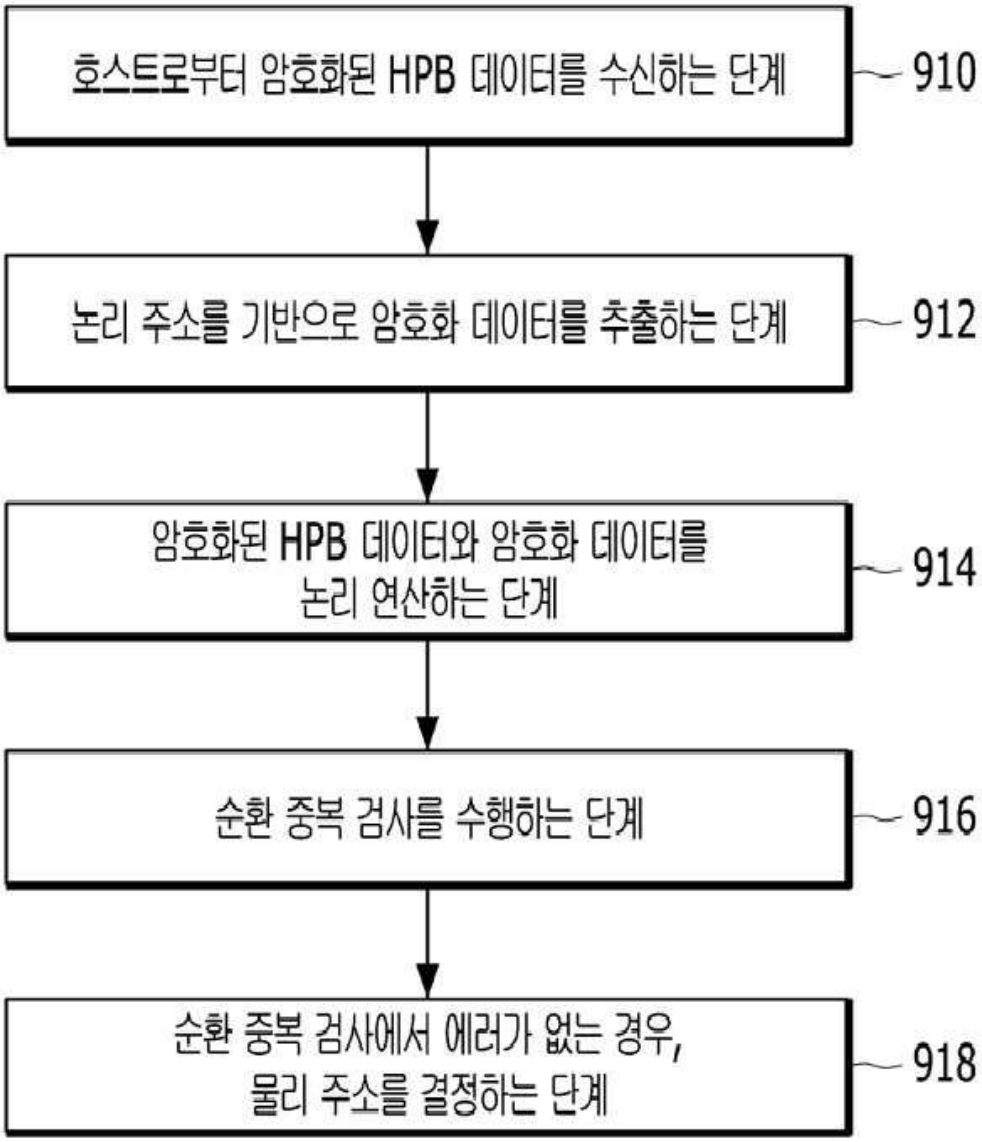
도면12


호스트로 전송할 HPB 데이터를 결정하는 단계
930
호스트로 전송할 HPB 데이터에 대해 순환 중복 검사를 수행하는 단계
932
논리 주소를 기반으로 암호화 데이터를 추출하는 단계
934
HPB 데이터와 암호화 데이터를 논리 연산하여 암호화된 HPB 데이터를 생성하는 단계
936

도면13


314B
Empty 32-BIT
LBA 32-BIT
RAN_SD
RANDOMIZE
36-BIT MASK
552
Empty
MASKED DATA 36-BIT
Empty
MASKED DATA 28-BIT
554
556
Empty
2nd MASKED DATA 36-BIT
CRCV 16-BIT
MD 12-BIT
EN_D 36-BIT
CRC CHECK
MD 12-BIT
TD 4-BIT
PPN 32-BIT

도면14


호스트로부터 암호화된 HPB 데이터를 수신하는 단계
910
논리 주소를 기반으로 암호화 데이터를 추출하는 단계
912
암호화된 HPB 데이터와 암호화 데이터를
논리 연산하는 단계
914
순환 중복 검사를 수행하는 단계
916
순환 중복 검사에서 에러가 없는 경우,
물리 주소를 결정하는 단계
918