(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2017-0012233
(43) 공개일자 2017년02월02일

(51) 국제특허분류(Int. Cl.)
G06F 12/02 (2006.01) *G06F 12/0811* (2016.01)
G06F 12/0817 (2016.01) *G06F 12/0831* (2016.01)
G06F 12/1081 (2016.01)
(52) CPC특허분류
G06F 12/023 (2013.01)
G06F 12/0811 (2013.01)
(21) 출원번호 10-2016-7032011
(22) 출원일자(국제) 2015년05월21일
심사청구일자 없음
(85) 번역문제출일자 2016년11월16일
(86) 국제출원번호 PCT/US2015/031913
(87) 국제공개번호 WO 2015/179606
국제공개일자 2015년11월26일
(30) 우선권주장
62/001,545 2014년05월21일 미국(US)
(뒷면에 계속)

(71) 출원인
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
헤데스, 마테우스, 코르넬리스, 안토니오스, 아드리아누스
미국 92121 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
바이드하야나탄, 나타라잔
미국 92121 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
베릴리, 콜린, 비튼
미국 92121 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인 남앤드남

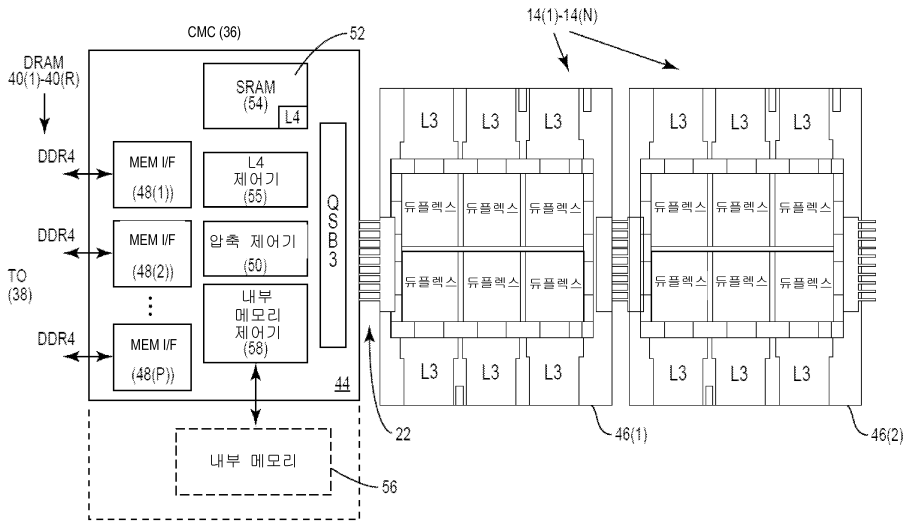
전체 청구항 수 : 총 24 항

(54) 발명의 명칭 **중앙 처리 장치(CPU)-기반 시스템의 압축된 메모리 제어기(CMC)들을 이용한 메모리 대역폭 압축의 제공**

(57) 요 약

중앙 처리 장치(CPU)-기반 시스템의 CMC(compressed memory controller)들을 이용하여 메모리 대역폭 압축을 제공하는 것이 개시된다. 이와 관련하여, 일부 양상들에서, CMC는 시스템 메모리의 물리적 어드레스에 대한 메모리 판독 요청을 수신하고, 물리적 어드레스의 에러 정정 코드(ECC) 비트들 및/또는 마스터 디렉토리로부터 물리적 어드레스에 대한 압축 표시자(CI)를 판독하도록 구성된다. CI에 기초하여, CMC는 메모리 판독 요청에 대해 판독될 메모리 블록들의 수를 결정하고 결정된 수의 메모리 블록들을 판독한다. 일부 양상들에서, CMC는 시스템 메모리의 물리적 어드레스에 대한 메모리 기록 요청을 수신하고 기록 데이터의 압축 패턴에 기초하여 기록 데이터에 대한 CI를 생성하도록 구성된다. CMC는 생성된 CI로 물리적 어드레스의 ECC 비트들 및/또는 마스터 디렉토리를 업데이트한다.

대 표 도 - 도3



(52) CPC특허분류
G06F 12/0817 (2013.01)
G06F 12/0833 (2013.01)
G06F 12/1081 (2013.01)
G06F 2206/1004 (2013.01)
G06F 2212/2532 (2013.01)
G06F 2212/401 (2013.01)
G06F 2212/452 (2013.01)
G06F 2212/621 (2013.01)
G06F 2212/622 (2013.01)

(30) 우선권주장
62/092,326 2014년12월16일 미국(US)
62/092,409 2014년12월16일 미국(US)
14/716,001 2015년05월19일 미국(US)
14/717,552 2015년05월20일 미국(US)

명 세 서

청구범위

청구항 1

압축된 메모리 제어기로서,

시스템 버스를 통해 시스템 메모리에 액세스하도록 구성된 메모리 인터페이스; 및

상기 시스템 메모리의 마스터 디렉토리의 이전에 판독된 마스터 디렉토리 메모리 블록을 저장하도록 각각 구성되는 복수의 CI(compression indicator) 캐시 엔트리들을 포함하는 CI 캐시를 포함하고,

상기 압축된 메모리 제어기는,

상기 시스템 메모리의 메모리 라인 내의 액세스될 메모리 블록의 물리적 어드레스를 포함하는 메모리 판독 요청을 수신하도록;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하는지를 결정하도록;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하지 않는다는 결정에 응답하여:

상기 마스터 디렉토리로부터 상기 물리적 어드레스에 대응하는 CI를 포함하는 마스터 디렉토리 메모리 블록을 판독하도록;

상기 CI 캐시의 CI 캐시 엔트리에 상기 마스터 디렉토리 메모리 블록을 기록하도록; 그리고

상기 CI 캐시의 CI 캐시 엔트리로부터 상기 CI를 판독하도록;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응한다는 결정에 응답하여, 상기 CI 캐시의 CI 캐시 엔트리로부터 상기 CI를 판독하도록;

상기 CI에 기초하여, 상기 메모리 판독 요청에 대해 판독할, 상기 시스템 메모리의 메모리 라인 내의 메모리 블록들의 수를 결정하도록; 그리고

상기 물리적 어드레스에서 시작하는 시스템 메모리의 메모리 라인 내의 결정된 수의 메모리 블록들을 판독하도록 구성되는,

압축된 메모리 제어기.

청구항 2

제 1 항에 있어서,

상기 CI가 상기 마스터 디렉토리로부터 판독되어야 한다고 확률론적으로(probabilistically) 결정하는 것에 응답하여, 상기 마스터 디렉토리 메모리 블록을 판독하도록 구성되고,

상기 압축된 메모리 제어기는, 상기 CI가 상기 마스터 디렉토리로부터 판독되지 않아야 한다고 확률론적으로 결정하는 것에 응답하여, 상기 메모리 블록과 연관된 하나 또는 그 초과의 ECC(error correcting code) 비트들로부터 상기 CI를 판독하도록 추가로 구성되는,

압축된 메모리 제어기.

청구항 3

제 1 항에 있어서,

상기 CI 캐시의 CI 캐시 엔트리에 상기 마스터 디렉토리 메모리 블록을 기록하기 이전에,

상기 CI 캐시의 현재 CI 캐시 엔트리가 퇴거(evict)되어야 하는지를 결정하도록; 그리고

현재 CI 캐시 엔트리가 퇴거되어야 한다는 결정에 응답하여,

상기 현재 CI 캐시 엔트리가 수정되었는지를 결정하도록; 그리고

상기 현재 CI 캐시 엔트리가 수정되었다는 결정에 응답하여, 상기 현재 CI 캐시 엔트리를 상기 마스터 디렉토리에 기록하도록, 추가로 구성되는,

압축된 메모리 제어기.

청구항 4

제 1 항에 있어서,

상기 CI가 상기 마스터 디렉토리로부터 판독되어야 한다고 확률론적으로 결정하는 것에 추가로 응답하여, 상기 마스터 디렉토리 메모리 블록을 판독하도록 구성되고,

상기 압축된 메모리 제어기는 추가로, 상기 CI가 상기 마스터 디렉토리로부터 판독되지 않아야 한다고 확률론적으로 결정하는 것에 응답하여, 상기 메모리 블록과 연관된 하나 또는 그 초과의 ECC(error correcting code) 비트들로부터 상기 CI를 판독하도록 구성되는,

압축된 메모리 제어기.

청구항 5

제 1 항에 있어서,

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하는지를 결정하는 것과 병렬로, 상기 시스템 메모리에 얼리(early) 메모리 판독 요청을 전송하도록 추가로 구성되고,

상기 압축된 메모리 제어기는, 상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응한다는 결정에 응답하여, 상기 CI에 기초하여 상기 얼리 메모리 판독 요청을 수정하도록 구성됨으로써 상기 결정된 수의 메모리 블록들을 판독하도록 구성되는,

압축된 메모리 제어기.

청구항 6

제 1 항에 있어서,

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하는지를 결정하는 것과 병렬로, 레벨 4(L4) 캐시 상에서 캐시 판독 동작을 수행하도록 추가로 구성되는,

압축된 메모리 제어기.

청구항 7

제 1 항에 있어서,

상기 CI에 의해 표시된 압축 패턴을 결정하도록 구성됨으로써, 상기 CI에 기초하여, 상기 메모리 판독 요청에 대해 판독할, 상기 시스템 메모리의 메모리 라인 내의 메모리 블록들의 수를 결정하도록 구성되는,

압축된 메모리 제어기.

청구항 8

제 7 항에 있어서,

제로(zero) 메모리 블록들이 판독되어야 한다고 표시하는 제로-라인 표시자를 상기 CI가 포함한다고 결정하도록 구성됨으로써 상기 CI에 의해 표시된 압축 패턴을 결정하도록 구성되는,

압축된 메모리 제어기.

청구항 9

압축된 메모리 제어기로서,

시스템 버스를 통해 시스템 메모리에 액세스하도록 구성된 메모리 인터페이스; 및

상기 시스템 메모리의 마스터 디렉토리의 이전에 판독된 마스터 디렉토리 메모리 블록을 저장하도록 각각 구성되는 복수의 CI 캐시(compression indicator) 엔트리들을 포함하는 CI 캐시를 포함하고,

상기 압축된 메모리 제어기는,

기록 데이터 및 상기 시스템 메모리의 메모리 라인 내에 기록될 메모리 블록의 물리적 어드레스를 포함하는 메모리 기록 요청을 수신하도록;

상기 기록 데이터에 대한 압축 패턴을 결정하도록;

상기 압축 패턴에 기초하여 상기 기록 데이터에 대한 CI를 생성하도록;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하는지를 결정하도록;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하지 않는다는 결정에 응답하여:

생성된 CI로 상기 마스터 디렉토리의 물리적 어드레스에 대응하는 저장된 CI를 업데이트하도록; 그리고

상기 CI 캐시의 CI 캐시 엔트리에 상기 저장된 CI를 기록하도록;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응한다는 결정에 응답하여, 상기 생성된 CI로 상기 CI 캐시의 CI 캐시 엔트리를 업데이트하도록;

상기 생성된 CI에 기초하여 상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들에 상기 기록 데이터를 기록하도록; 그리고

상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들 각각의 하나 또는 그 초과의 ECC(error correcting code) 비트들에 상기 생성된 CI를 기록하도록 구성되는,

압축된 메모리 제어기.

청구항 10

제 9 항에 있어서,

상기 저장된 CI가 상기 마스터 디렉토리에서 업데이트되어야 한다고 확률론적으로 결정하는 것에 응답하여 상기 저장된 CI를 업데이트하도록 구성되는,

압축된 메모리 제어기.

청구항 11

제 9 항에 있어서,

상기 마스터 디렉토리가 업데이트되어야 한다고 확률론적으로 결정하는 것에 추가로 응답하여, 상기 저장된 CI를 업데이트하고 상기 CI 캐시의 CI 캐시 엔트리에 상기 저장된 CI를 기록하도록 구성되는,

압축된 메모리 제어기.

청구항 12

제 9 항에 있어서,

상기 CI 캐시의 CI 캐시 엔트리에 상기 저장된 CI를 기록하기 이전에,

상기 CI 캐시의 현재 CI 캐시 엔트리가 퇴거되어야 하는지를 결정하도록; 그리고

현재 CI 캐시 엔트리가 퇴거되어야 한다는 결정에 응답하여,

상기 현재 CI 캐시 엔트리가 수정되었는지를 결정하도록; 그리고

상기 현재 CI 캐시 엔트리가 수정되었다는 결정에 응답하여, 상기 CI 캐시 엔트리를 상기 마스터 디렉토리에 기록하도록, 추가로 구성되는,

압축된 메모리 제어기.

청구항 13

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법으로서,

시스템 버스를 통해 압축된 메모리 제어기에 의해, 시스템 메모리의 메모리 라인 내의 액세스될 메모리 블록의 물리적 어드레스를 포함하는 메모리 판독 요청을 수신하는 단계;

상기 물리적 어드레스가 CI(compression indicator) 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하는지를 결정하는 단계;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하지 않는다는 결정에 응답하여:

상기 시스템 메모리의 마스터 디렉토리로부터 상기 물리적 어드레스에 대응하는 CI를 포함하는 마스터 디렉토리 메모리 블록을 판독하는 단계;

상기 CI 캐시의 CI 캐시 엔트리에 상기 마스터 디렉토리 메모리 블록을 기록하는 단계; 및

상기 CI 캐시의 CI 캐시 엔트리로부터 상기 CI를 판독하는 단계;

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응한다는 결정에 응답하여, 상기 CI 캐시의 CI 캐시 엔트리로부터 상기 CI를 판독하는 단계;

상기 CI에 기초하여, 상기 메모리 판독 요청에 대해 판독할, 상기 시스템 메모리의 메모리 라인 내의 메모리 블록들의 수를 결정하는 단계; 및

상기 물리적 어드레스에서 시작하는 시스템 메모리의 메모리 라인 내의 결정된 수의 메모리 블록들을 판독하는 단계를 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 14

제 13 항에 있어서,

상기 마스터 디렉토리 메모리 블록을 판독하는 단계는 상기 CI가 상기 마스터 디렉토리로부터 판독되어야 한다고 확률론적으로 결정하는 것에 응답하며;

상기 방법은, 상기 CI가 상기 마스터 디렉토리로부터 판독되지 않아야 한다고 확률론적으로 결정하는 것에 응답하여, 상기 메모리 블록과 연관된 하나 또는 그 초과의 ECC(error correcting code) 비트들로부터 상기 CI를 판독하는 단계를 더 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 15

제 13 항에 있어서,

상기 CI 캐시의 CI 캐시 엔트리에 상기 마스터 디렉토리 메모리 블록을 기록하기 이전에,

상기 CI 캐시의 현재 CI 캐시 엔트리가 퇴거되어야 하는지를 결정하는 단계; 및

현재 CI 캐시 엔트리가 퇴거되어야 한다는 결정에 응답하여,

상기 현재 CI 캐시 엔트리가 수정되었는지를 결정하는 단계; 및

상기 현재 CI 캐시 엔트리가 수정되었다는 결정에 응답하여, 상기 현재 CI 캐시 엔트리를 상기 마스터 디렉토리에 기록하는 단계를 더 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 16

제 13 항에 있어서,

상기 마스터 디렉토리 메모리 블록을 판독하는 단계는 상기 CI가 상기 마스터 디렉토리로부터 판독되어야 한다고 확률론적으로 결정하는 것에 추가로 응답하며,

상기 방법은, 상기 CI가 상기 마스터 디렉토리로부터 판독되지 않아야 한다고 확률론적으로 결정하는 것에 응답하여, 상기 메모리 블록과 연관된 하나 또는 그 초과의 ECC(error correcting code) 비트들로부터 상기 CI를 판독하는 단계를 더 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 17

제 13 항에 있어서,

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하는지를 결정하는 것과 병렬로, 상기 시스템 메모리에 얼리 메모리 판독 요청을 전송하는 단계를 더 포함하고,

상기 결정된 수의 메모리 블록들을 판독하는 단계는, 상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응한다는 결정에 응답하여, 상기 CI에 기초하여 상기 얼리 메모리 판독 요청을 수정하는 단계를 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 18

제 13 항에 있어서,

상기 물리적 어드레스가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리에 대응하는지를 결정하는 것과 병렬로, 레벨 4(L4) 캐시 상에서 캐시 판독 동작을 수행하는 단계를 더 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 19

제 13 항에 있어서,

상기 CI에 기초하여, 상기 메모리 판독 요청에 대해 판독할, 상기 시스템 메모리의 메모리 라인 내의 메모리 블록들의 수를 결정하는 단계는, 상기 CI에 의해 표시된 압축 패턴을 결정하는 단계를 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 20

제 19 항에 있어서,

상기 CI에 의해 표시된 압축 패턴을 결정하는 단계는, 제로 메모리 블록들이 판독되어야 한다고 표시하는 제로-라인 표시자를 상기 CI가 포함한다고 결정하는 단계를 포함하는,

메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 21

메모리 기록 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법으로서,

시스템 버스를 통해 압축된 메모리 제어기에 의해, 기록 데이터 및 시스템 메모리의 메모리 라인 내에 기록될 메모리 블록의 물리적 어드레스를 포함하는 메모리 기록 요청을 수신하는 단계;

상기 기록 데이터에 대한 압축 패턴을 결정하는 단계;

상기 압축 패턴에 기초하여 상기 기록 데이터에 대한 CI(compression indicator)를 생성하는 단계;

상기 물리적 어드레스에 대응하는 CI가 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리 내에 존재하는지를 결정하는 단계;

상기 물리적 어드레스에 대응하는 CI가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리 내에 존재하지 않는다는 결정에 응답하여:

생성된 CI로 마스터 디렉토리의 물리적 어드레스에 대응하는 저장된 CI를 업데이트하는 단계; 및

상기 CI 캐시의 CI 캐시 엔트리에 상기 저장된 CI를 기록하는 단계;

상기 물리적 어드레스에 대응하는 CI가 상기 CI 캐시의 복수의 CI 캐시 엔트리들의 CI 캐시 엔트리 내에 존재한다는 결정에 응답하여, 상기 생성된 CI로 상기 CI 캐시의 CI 캐시 엔트리를 업데이트하는 단계;

상기 생성된 CI에 기초하여 상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들에 상기 기록 데이터를 기록하는 단계; 및

상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들 각각의 하나 또는 그 초과의 ECC(error correcting code) 비트들에 상기 생성된 CI를 기록하는 단계를 포함하는,

메모리 기록 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 22

제 21 항에 있어서,

상기 저장된 CI를 업데이트하는 단계는 상기 CI가 상기 마스터 디렉토리에서 업데이트되어야 한다고 확률론적으로 결정하는 것에 응답하는,

메모리 기록 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 23

제 21 항에 있어서,

상기 저장된 CI를 업데이트하는 단계 및 상기 저장된 CI를 상기 CI 캐시의 CI 캐시 엔트리에 기록하는 단계는 상기 마스터 디렉토리가 업데이되어야 한다고 확률론적으로 결정하는 것에 추가로 응답하는,

메모리 기록 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

청구항 24

제 21 항에 있어서,

상기 CI 캐시의 CI 캐시 엔트리에 상기 저장된 CI를 기록하기 이전에,

상기 CI 캐시의 현재 CI 캐시 엔트리가 퇴거되어야 하는지를 결정하는 단계; 및

현재 CI 캐시 엔트리가 퇴거되어야 한다는 결정에 응답하여,

상기 현재 CI 캐시 엔트리가 수정되었는지를 결정하는 단계; 및

상기 현재 CI 캐시 엔트리가 수정되었다는 결정에 응답하여, 상기 CI 캐시 엔트리를 상기 마스터 디렉토리에 기록하는 단계를 포함하는,

메모리 기록 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법.

발명의 설명

기 술 분 야

[0001] 우선권 출원들

[0002] [0001] 본 출원은 2014년 5월 21일 출원되고 발명의 명칭이 “MEMORY CONTROLLERS EMPLOYING MEMORY CAPACITY AND/OR BANDWIDTH COMPRESSION, AND RELATED PROCESSOR-BASED SYSTEMS AND METHODS”인 미국 가특허 출원 번호 제62/001,545호를 우선권으로 청구하며, 이는 그 전체가 인용에 의해 본원에 포함된다.

[0003] [0002] 본 출원은 또한, 2014년 12월 16일 출원되고 발명의 명칭이 “MEMORY CONTROLLERS EMPLOYING MEMORY CAPACITY AND/OR BANDWIDTH COMPRESSION, AND RELATED PROCESSOR-BASED SYSTEMS AND METHODS”인 미국 가특허 출원 번호 제62/092,326호를 우선권으로 청구하며, 이는 그 전체가 인용에 의해 본원에 포함된다.

[0004] [0003] 본 출원은 또한, 2014년 12월 16일 출원되고 발명의 명칭이 “MEMORY CONTROLLERS EMPLOYING MEMORY CAPACITY AND/OR BANDWIDTH COMPRESSION WITH NEXT READ ADDRESS PREFETCHING, AND RELATED PROCESSOR-BASED SYSTEMS AND METHODS”인 미국 가특허 출원 번호 제62/092,409호를 우선권으로 청구하며, 이는 그 전체가 인용에 의해 본원에 포함된다.

[0005] [0004] 본 출원은 또한, 2015년 5월 19일 출원되고 발명의 명칭이 “MEMORY CONTROLLERS EMPLOYING MEMORY CAPACITY COMPRESSION, AND RELATED PROCESSOR-BASED SYSTEMS AND METHODS”인 미국 특허 출원 번호 제14/716,001호를 우선권으로 청구하며, 이는 그 전체가 인용에 의해 본원에 포함된다.

[0006] [0005] 본 출원은 또한, 2015년 5월 20일 출원되고 발명의 명칭이 “PROVIDING MEMORY BANDWIDTH COMPRESSION USING COMPRESSED MEMORY CONTROLLERS (CMCs) IN A CENTRAL PROCESSING UNIT (CPU)-BASED SYSTEM”인 미국 특

허 출원 번호 제14/717,552호를 우선권으로 청구하며, 이는 그 전체가 인용에 의해 본원에 포함된다.

[0007] 본 개시의 분야

[0008] [0006] 본 개시의 기술은 일반적으로, 컴퓨터 메모리 시스템들에 관한 것으로서, 특히 메모리에 대한 메모리 액세스 인터페이스를 중앙 처리 장치(CPU)들에 제공하기 위한 컴퓨터 메모리 시스템의 메모리 제어기들에 관한 것이다.

배 경 기 술

[0009] [0007] 마이크로프로세서들은 매우 다양한 애플리케이션들에서 계산 작업들을 수행한다. 통상적인 마이크로프로세서 애플리케이션은 소프트웨어 명령들을 실행하는 하나 또는 그 초과의 중앙 처리 장치들(CPU들)을 포함한다. 소프트웨어 명령들은 메모리의 위치로부터 데이터를 패치(fetch)하고, 패치된 데이터를 이용하여 하나 또는 그 초과의 CPU 동작들을 수행하고, 결과를 생성하도록 CPU에 지시한다. 결과는 그 후 메모리에 저장될 수 있다. 비-제한적인 예들로서, 이 메모리는 CPU에 대해 로컬적인 캐시, CPU 블록의 CPU들 간의 공유 로컬 캐시, 다수의 CPU 블록 간의 공유 캐시, 또는 마이크로프로세서의 메인 메모리일 수 있다.

[0010] [0008] 이와 관련하여, 도 1은 CPU-기반 시스템(12)을 포함하는 예시적인 SOC(system-on-a-chip)(10)의 개략도이다. CPU-기반 시스템(12)은 이 예에서 복수의 CPU 블록들(14(1)-14(N))을 포함하며, 여기서 'N'은 원하는 임의의 수의 CPU 블록들(14(1)-14(N))과 동일하다. 도 1의 이 예에서, CPU 블록들(14(1)-14(N)) 각각은 2개의 CPU들(16(1), 16(2))을 포함한다. CPU 블록들(14(1)-14(N))은 추가로, 공유 레벨 2(L2) 캐시들(18(1)-18(N))을 각각 포함한다. 공유 레벨 3(L3) 캐시(20)는 또한, 각각의 CPU 블록들(14(1)-14(N)) 또는 이들 간에 공유되는 임의의 것에 의해 이용되는 캐시된 데이터를 저장하기 위해 제공된다. 내부 시스템 버스(22)는 CPU 블록들(14(1)-14(N)) 각각이 공유 L3 캐시(20)는 물론 다른 공유 자원들에 액세스하는 것을 인에이블하도록 제공된다. 내부 시스템 버스(22)를 통해 CPU 블록들(14(1)-14(N))에 의해 액세스되는 다른 공유 자원들은 메인 외부 메모리(예를 들어, 비-제한적인 예로서 DDR(double-rate DRAM(dynamic random access memory))에 액세스하기 위해 메모리 제어기(24), 주변기기들(26), 다른 저장소(28), PCI-e(express PCI(peripheral component interconnect)) 인터페이스(30), DMA(direct memory access) 제어기(32), 및/또는 IMC(integrated memory controller)(34)를 포함할 수 있다.

[0011] [0009] 도 1의 CPU-기반 시스템(12)에서 실행중인 CPU-기반 애플리케이션들은 복잡도 및 성능 면에서 증가하고, 공유 L2 캐시(18(1)-18(N)) 및 공유 L3 캐시(20)와 메모리 제어기(24)를 통해 액세스 가능한 외부 메모리의 메모리 용량 요건들을 또한 증가시킬 수 있다. 데이터 압축은 물리적 메모리 용량을 증가시킴 없이 CPU-기반 시스템(12)의 유효 메모리 용량을 증가시키기 위해 이용될 수 있다. 그러나 데이터 압축의 이용은, 데이터가 압축되었는지 또는 압축되지 않았는지에 의존하여, 다수의 메모리 액세스 요청들이 데이터를 리트리브하도록 요구될 때 메모리 액세스 레이턴시를 증가시키고 부가적인 메모리 대역폭을 소비할 수 있다. 이에 따라, 메모리 액세스 레이턴시 및 메모리 대역폭에 관한 영향을 완화하면서 데이터 압축을 이용하여 CPU-기반 시스템(12)의 메모리 용량을 증가시키는 것이 바람직하다.

발명의 내용

[0012] [0010] 본원에서 개시되는 양상들은 중앙 처리 장치(CPU)-기반 시스템의 CMC(compressed memory controller)들을 이용하여 메모리 대역폭 압축을 제공하는 것을 포함한다. 이와 관련하여, 일부 양상들에서, CMC는 메모리 판독 요청들 및/또는 메모리 기록 요청들을 위한 메모리 대역폭 압축을 제공하도록 구성된다. 일부 양상들에 따라, 시스템 메모리의 물리적 어드레스에 대한 메모리 판독 요청의 수신 시에, CMC는 시스템 메모리의 물리적 어드레스와 연관된 에러 정정 코드(ECC) 비트들 및/또는 마스터 디렉토리로부터 물리적 어드레스에 대한 압축 표시자(CI)를 판독할 수 있다. CI는 일부 양상들에서, (예를 들어, 데이터가 압축되었는지 또는 압축되지 않았는지를 표시하고 및/또는 압축된 데이터가 물리적 어드레스에 저장된 메모리 블록들의 수를 표시하는) 압축 패턴을 CMC에 제공할 수 있다. CI에 기초하여, CMC는 메모리 판독 요청에 대해 판독될 메모리 블록들의 수를 결정하고 물리적 어드레스에서 시작하는 결정된 수의 메모리 블록들을 판독한다. 일부 양상들에서, 시스템 메모리의 물리적 어드레스에 대한 메모리 기록 요청의 수신 시에, CMC는 기록 데이터가 기록될 압축 패턴을 결정할 수 있고, 압축 패턴에 기초하여 기록 데이터에 대한 CI를 생성할 수 있다. CMC는 그 후 물리적 어드레스와 연관된 ECC 비트들 및/또는 마스터 디렉토리를 업데이트하고, 생성된 CI에 기초하여 시스템 메모리에 기록 데이터를 기록할 수 있다. CMC의 일부 양상들은 이전에 판독된 CI들을 캐시하기 위한 CI 캐시를 추가로 제공할 수 있다. 이러한 방식으로, CMC는 압축된 그리고 압축되지 않은 데이터를 보다 효율적으로 판독 및 기록할 수 있어

서, 메모리 액세스 레이턴시를 감소시키고 시스템 성능을 개선시킨다.

[0013] [0011] 다른 양상에서, CMC가 제공된다. CMC는 시스템 버스를 통해 시스템 메모리에 액세스하도록 구성된 메모리 인터페이스를 포함한다. CMC는 시스템 메모리의 메모리 라인 내의 액세스될 메모리 블록의 물리적 어드레스를 포함하는 메모리 판독 요청을 수신하도록 구성된다. CMC는 추가로, 상기 시스템 메모리의 마스터 디렉토리로부터 상기 물리적 어드레스에 대응하는 CI를 포함하는 마스터 디렉토리 메모리 블록을 판독하도록 구성된다. CMC는 또한, 상기 CI에 기초하여, 상기 메모리 판독 요청에 대해 판독할, 시스템 메모리의 메모리 라인 내의 메모리 블록들의 수를 결정하도록 구성된다. CMC는 부가적으로, 물리적 어드레스에서 시작하는 시스템 메모리의 메모리 라인 내의 결정된 수의 메모리 블록들을 판독하도록 구성된다.

[0014] [0012] 다른 양상에서, CMC가 제공된다. CMC는 시스템 버스를 통해 시스템 메모리에 액세스하도록 구성된 메모리 인터페이스를 포함한다. CMC는 기록 데이터 및 상기 시스템 메모리의 메모리 라인 내에 기록될 메모리 블록의 물리적 어드레스를 포함하는 메모리 기록 요청을 수신하도록 구성된다. CMC는 추가로 기록 데이터에 대한 압축 패턴을 결정하도록 구성된다. CMC는 또한, 상기 압축 패턴에 기초하여 상기 기록 데이터에 대한 CI를 생성하도록 구성된다. CMC는 부가적으로, 생성된 CI로 마스터 디렉토리의 물리적 어드레스에 대응하는 저장된 CI를 업데이트하도록 구성된다. CMC는 추가로, 상기 생성된 CI에 기초하여 상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들에 상기 기록 데이터를 기록하도록 구성된다. CMC는 또한, 상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들 각각의 하나 또는 그 초과의 ECC 비트들에 상기 생성된 CI를 기록하도록 구성된다.

[0015] [0013] 다른 양상에서, 메모리 판독 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법이 제공된다. 방법은, 시스템 버스를 통해 CMC에 의해, 시스템 메모리의 메모리 라인 내의 액세스될 메모리 블록의 물리적 어드레스를 포함하는 메모리 판독 요청을 수신하는 단계를 포함한다. 방법은 추가로, 상기 시스템 메모리의 마스터 디렉토리로부터 상기 물리적 어드레스에 대응하는 CI를 포함하는 마스터 디렉토리 메모리 블록을 판독하는 단계를 포함한다. 방법은 또한 상기 CI에 기초하여, 상기 메모리 판독 요청에 대해 판독할, 시스템 메모리의 메모리 라인 내의 메모리 블록들의 수를 결정하는 단계를 포함한다. 방법은 부가적으로, 상기 물리적 어드레스에서 시작하는 시스템 메모리의 메모리 라인 내의 결정된 수의 메모리 블록들을 판독하는 단계를 포함한다.

[0016] [0014] 다른 양상에서, 메모리 기록 요청들에 대한 메모리 대역폭 압축을 제공하기 위한 방법이 제공된다. 방법은 시스템 버스를 통해 CMC에 의해, 기록 데이터 및 시스템 메모리의 메모리 라인 내에 기록될 메모리 블록의 물리적 어드레스를 포함하는 메모리 기록 요청을 수신하는 단계를 포함한다. 방법은 추가로, 상기 기록 데이터에 대한 압축 패턴을 결정하는 단계를 포함한다. 방법은 또한, 상기 압축 패턴에 기초하여 상기 기록 데이터에 대한 CI를 생성하는 단계를 포함한다. 방법은 부가적으로, 생성된 CI로 마스터 디렉토리의 물리적 어드레스에 대응하는 저장된 CI를 업데이트하는 단계를 포함한다. 방법은 추가로, 상기 생성된 CI에 기초하여 상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들에 상기 기록 데이터를 기록하는 단계를 포함한다. 방법은 또한, 상기 시스템 메모리의 메모리 라인의 하나 또는 그 초과의 메모리 블록들 각각의 하나 또는 그 초과의 ECC 비트들에 상기 생성된 CI를 기록하는 단계를 포함한다.

[0017] [0015] 다른 양상들에서, 소형 데이터 블록 압축에 매우 적합할 수 있는 압축 방법들 및 포맷이 개시된다. 이 압축 방법들 및 포맷들은 본원에서 개시되는 메모리 대역폭 압축 양상들에 대해 이용될 수 있다.

[0018] [0016] 이들 CMC들 및 압축 매커니즘들의 일부 또는 모든 양상들에 의해, 물리적 메모리 크기의 증가를 경감시키고 시스템 성능에 관한 영향을 최소화하면서, 메모리 액세스 레이턴시를 감소시키고 CPU-기반 시스템의 메모리 대역폭을 효과적으로 증가시키는 것이 가능해질 수 있다.

도면의 간단한 설명

[0019] [0017] 도 1은 중앙 처리 장치(CPU)-기반 시스템을 포함하는 예시적인 SoC(system-on-a-chip)의 개략도이다.

[0018] 도 2는 복수의 CPU들 및 메모리 대역폭 압축을 제공하도록 구성된 CMC(compressed memory controller)를 갖는 예시적인 CPU-기반 시스템을 포함하는 SoC의 개략도이다.

[0019] 도 3은 도 2의 CMC의 보다 상세한 개략도이며, 여기서 CMC는 추가로 메모리 대역폭 압축을 제공하도록 이용될 수 있는 선택적인 내부 메모리에 통신 가능하게 커플링된다.

[0020] 도 4는 도 3의 CMC에 의해 구현될 수 있는 예시적인 메모리 대역폭 압축 매커니즘의 개략도이다.

[0021] 도 5는 CMC의 어드레스 변환으로 인한 성능 손실을 보상하기 위해 선택적인 레벨 4(L4) 캐시를 포함하는 도 1의 SoC의 예를 예시한다.

[0022] 도 6a 및 도 6b는 각각 메모리 판독 동작들 및 메모리 기록 동작들 동안 예시적인 통신 흐름들 및 메모리 대역폭 압축을 제공하기 위해 도 3의 CMC에 의해 액세스될 수 있는 도 4의 시스템 메모리 및 마스터 디렉토리의 예시적인 엘리먼트을 예시하는 도면들이다.

[0023] 도 7은 메모리 대역폭 압축을 이용하여 도 6a의 CMC에 의해 메모리 판독 동작을 수행하기 위한 예시적인 동작들을 예시하는 흐름도이다.

[0024] 도 8은 메모리 대역폭 압축을 이용하여 도 6b의 CMC에 의해 메모리 기록 동작을 수행하기 위한 예시적인 동작들을 예시하는 흐름도이다.

[0025] 도 9a 및 도 9b는 각각 메모리 판독 동작들 및 메모리 기록 동작들 동안 예시적인 통신 흐름들 및 메모리 대역폭 압축을 제공하기 위해 도 3의 CMC의 일부 양상들에 의해 제공되는 CI(compression indicator) 캐시의 예시적인 엘리먼트들을 예시하는 도면들이다.

[0026] 도 10a 내지 도 10c는 CI 캐시를 이용하여 도 9a의 CMC에 의해 메모리 판독 동작을 수행하기 위한 예시적인 동작들을 예시하는 흐름도들이다.

[0027] 도 11a 내지 도 11c는 CI 캐시를 이용하여 도 9b의 CMC에 의해 메모리 기록 동작을 수행하기 위한 예시적인 동작들을 예시하는 흐름도들이다.

[0028] 도 12 내지 도 18은 예시적인 데이터 블록 압축 포맷들 및 매커니즘들을 예시하며, 이들 중 임의의 것이 메모리 데이터 블록들을 압축 및 압축해제하기 위해 도 3의 CMC에 의해 이용될 수 있다.

[0029] 도 19는 도 2의 CMC를 이용할 수 있는 도 1의 SoC를 포함할 수 있는 예시적인 컴퓨팅 디바이스의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0020] [0030] 이제 도면 그림들을 참조하여, 본 개시의 여러 예시적인 양상들이 설명된다. "예시적인"인 이란 단어는, “예, 인스턴스 또는 예시로서 기능하는” 것을 의미하도록 본원에서 사용된다. 본원에서 "예시적인" 것으로 설명되는 임의의 양상은 반드시 다른 양상들에 비해 선호되거나 유리한 것으로 해석될 필요는 없다.

[0021] [0031] 본원에서 개시되는 양상들은 중앙 처리 장치(CPU)-기반 시스템의 CMC(compressed memory controller)들을 이용하여 메모리 대역폭 압축을 제공하는 것을 포함한다. 이와 관련하여, 일부 양상들에서, CMC는 메모리 판독 요청들 및/또는 메모리 기록 요청들을 위한 메모리 대역폭 압축을 제공하도록 구성된다. 일부 양상들에 따라, 시스템 메모리의 물리적 어드레스에 대한 메모리 판독 요청의 수신 시에, CMC는 시스템 메모리의 물리적 어드레스와 연관된 에러 정정 코드(ECC) 비트들 및/또는 마스터 디렉토리로부터 물리적 어드레스에 대한 압축 표시자(CI)를 판독할 수 있다. CI는 일부 양상들에서, (예를 들어, 데이터가 압축되었는지 또는 압축되지 않았는지를 표시하고 및/또는 압축된 데이터가 물리적 어드레스에 저장된 메모리 블록들의 수를 표시하는) 압축 패턴을 CMC에 제공할 수 있다. CI에 기초하여, CMC는 메모리 판독 요청에 대해 판독될 메모리 블록들의 수를 결정하고 물리적 어드레스에서 시작하는 결정된 수의 메모리 블록들을 판독한다. 일부 양상들에서, 시스템 메모리의 물리적 어드레스에 대한 메모리 기록 요청의 수신 시에, CMC는 기록 데이터가 기록될 압축 패턴을 결정할 수 있고, 압축 패턴에 기초하여 기록 데이터에 대한 CI를 생성할 수 있다. CMC는 그 후 물리적 어드레스와 연관된 ECC 비트들 및/또는 마스터 디렉토리를 업데이트하고, 생성된 CI에 기초하여 시스템 메모리에 기록 데이터를 기록할 수 있다. CMC의 일부 양상들은 이전에 판독된 CI들을 캐시하기 위한 CI 캐시를 추가로 제공할 수 있다. 이러한 방식으로, CMC는 압축된 그리고 압축되지 않은 데이터를 보다 효율적으로 판독 및 기록할 수 있어서, 메모리 액세스 레이턴시를 감소시키고 시스템 성능을 개선시킨다.

[0022] [0032] 이와 관련하여, 도 2는 도 1의 CPU-기반 시스템(12)과 유사한, 복수의 CPU 블록들(14(1)-14(N))을 갖는 예시적인 CPU-기반 시스템(12')을 포함하는 SoC(10')의 개략도이다. 도 2의 CPU-기반 시스템(12')은 도 1의 CPU-기반 시스템(12)과 일부 공통적인 컴포넌트들을 포함하며, 이들은 도 1과 도 2 간의 공통 엘리먼트 번호들에 의해 언급된다. 간략함을 위해, 이들 엘리먼트들은 다시 설명되지 않을 것이다. 그러나 도 2의 CPU-기반 시스템(12')에서, CMC(36)가 제공된다. CMC(36)는 시스템 메모리(38)에 대한 액세스를 제어한다. 시스템 메모리(38)는 비-제한적인 예로서, 하나 또는 그 초과의 DDR(double data rate) DRAM들(dynamic random access

memories)(40(1)-40(R))(이하 "DRAM(40(1)-40(R))"으로서 지칭됨)을 포함할 수 있다. CMC(36)는 이 예에서, 아래에서 그리고 본원에서 개시되는 양상들에 따라 메모리 대역폭 압축을 이용한다. 도 1의 CPU-기반 시스템(12)의 메모리 제어기(24)와 유사하게, 도 2의 CPU-기반 시스템(12')의 CMC(36)는 내부 시스템 버스(22)를 통해 CPU 블록들(14(1)-14(N))에 의해 공유된다.

[0023] [0033] 도 2의 CMC(36)의 예시적인 내부 컴포넌트들의 보다 상세한 개략도를 예시하기 위해, 도 3이 제공된다. 이 예에서, CMC(36)는 도 2의 CPU 블록들(14(1)-14(N))을 포함하는 반도체 다이들(46(1), 46(2))과 별개의 반도체 다이(44) 상에 제공된다. 대안적으로, 일부 양상들에서, CMC(36)는 CPU 블록들(14(1)-14(N))을 갖는 공통 반도체 다이(도시되지 않음)에 포함될 수 있다. 다이 구성들과 무관하게, CMC(36)는 CPU 블록들(14(1)-14(N))이 내부 시스템 버스(22)를 통해 CMC(36)에 대해 메모리 액세스 요청들을 행하고, CMC(36)를 통해 메모리로부터 데이터를 수신할 수 있도록 제공된다.

[0024] [0034] 도 3을 계속 참조하여, CMC(36)는 DRAM(40(1)-40(R))을 포함하는 것으로서 도 2 및 도 3에서 도시되는 시스템 메모리(38)에 대한 메모리 액세스를 위한 동작들을 제어한다. CMC(36)는 메모리 액세스 요청들(도시되지 않음)을 서비스하는데 이용되는 복수의 메모리 인터페이스들(MEM I/F들)(48(1)-48(P))(예를 들어, DDR DRAM 인터페이스들)을 포함한다. 이와 관련하여, CMC(36)는 이 예에서 압축 제어기(50)를 포함한다. 압축 제어기(50)는 도 2의 CPU 블록들(14(1)-14(N))로부터의 메모리 액세스 요청들에 대한 응답으로, 시스템 메모리(38)에 저장된 데이터의 압축 및 시스템 메모리(38)로부터 리트리브된 데이터의 압축해제를 제어한다. 이러한 방식으로, CPU 블록들(14(1)-14(N))에는 CMC(36)에 의해 액세스되는 메모리의 실제 용량보다 더 큰 가상 메모리 어드레스 공간이 제공될 수 있다. 압축 제어기(50)는 또한 내부 시스템 버스(22)를 통해 CPU 블록들(14(1)-14(N))에 제공된 정보의 대역폭 압축을 수행하도록 구성될 수 있다.

[0025] [0035] 아래에서 보다 상세히 논의될 바와 같이, 압축 제어기(50)는 메모리 대역폭 압축을 제공하기 위한 임의의 수의 압축 기술들 및 알고리즘들을 수행할 수 있다. 로컬 메모리(52)는 이러한 압축 기술들 및 알고리즘들을 수행하기 위해 압축 제어기(50)에 의해 필요한 데이터 구조들 및 다른 정보를 위해 제공된다. 이와 관련하여, 로컬 메모리(52)는 정적 랜덤 액세스 메모리(SRAM)(54)의 형태로 제공될 수 있다. 로컬 메모리(52)는 압축 제어기(50)가 압축 기술들 및 알고리즘들을 수행하는데 필요할 수 있는 데이터 구조들 및 다른 데이터 저장소들을 위해 이용되기에 충분한 크기로 이루어진다. 로컬 메모리(52)는 또한 CMC(36) 내의 내부 사용을 위해 부가적인 캐시 메모리를 제공하도록 레벨 4(L4) 캐시와 같은 캐시를 포함하기 위해 파티셔닝될 수 있다. 따라서, L4 제어기(55)는 또한 레벨 4 캐시에 대한 액세스를 제공하도록 CMC(36)에 제공될 수 있다. 아래에서 보다 상세히 논의될 바와 같이, 강화된 압축 기술들 및 알고리즘들은 더 큰 내부 메모리를 요구할 수 있다. 예를 들어, 로컬 메모리(52)는 메모리의 128 kB(kilobytes)를 제공할 수 있다.

[0026] [0036] 추가로, 도 3에서 도시된 바와 같이, 그리고 아래에서 보다 상세히 설명될 바와 같이, 선택적인 부가적 내부 메모리(56)가 또한 CMC(36)를 위해 제공될 수 있다. 부가적인 내부 메모리(56)는 예로서 DRAM으로서 제공될 수 있다. 아래에서 보다 상세히 논의될 바와 같이, 부가적인 내부 메모리(56)는 CPU-기반 시스템(12)의 메모리 대역폭 압축을 증가시키기 위해 메모리 압축 및 압축해제 매커니즘들을 제공하는 CMC(36)에 대한 로컬 메모리(52)에서보다 더 크거나 부가적인 양의 데이터 구조들 및 다른 데이터의 저장을 용이하게 할 수 있다. 내부 메모리 제어기(58)는 압축에 이용하기 위한 부가적인 내부 메모리(56)에 대한 메모리 액세스들을 제어하도록 CMC(36)에서 제공된다. 내부 메모리 제어기(58)는 CPU 블록들(14(1)-14(N))에 대해 액세스 가능 또는 보기 가능하지 않을 수 있다.

[0027] [0037] 위에서 언급된 바와 같이, 도 3의 CMC(36)는 일부 양상들에서, 제로-라인 압축(zero-line compression)을 포함하는 메모리 대역폭 압축을 수행할 수 있다. 로컬 메모리(52)는 이러한 압축을 위해 이용되는 더 큰 데이터 구조들을 저장하는데 이용될 수 있다. 아래에서 더 상세히 논의되는 바와 같이, 메모리 대역폭 압축은 메모리 액세스 레이턴시를 감소시키고, CPU들(16(1), 16(2)) 또는 그 각각의 스레드들이 메모리 액세스 레이턴시에 대한 영향을 최소화하면서 동일한 수의 메모리 채널들에 액세스하도록 허용할 수 있다. 일부 양상들에서, 메모리 채널들의 수는, 이러한 압축이 CMC(36)에 의해 수행되지 않는 경우의 훨씬 다수의 메모리 채널들에 비교하여 유사한 레이턴시 결과들을 달성하면서 감소될 수 있으며, 이는 시스템 레벨 전력 소비를 감소시킬 수 있다.

[0028] [0038] 로컬 메모리(52) 및 부가적인 내부 메모리(56)를 비롯해서 도 3의 CMC(36)의 메모리 대역폭 압축을 위해 제공되는 자원들 각각은, 자원들 및 영역, 전력 소비, 메모리 용량 압축을 통한 감소된 메모리 용량 및 메모리 대역폭 압축을 통한 증가된 성능 간의 원하는 밸런스를 달성하도록 개별적으로, 또는 서로 결합하여 이용될 수

있다. 메모리 대역폭 압축은 필요에 따라 인에이블되거나 디스에이블될 수 있다. 추가로, CMC(36)에 의한 이용을 위해 위에서 설명되는 자원들은 메모리 용량 및/또는 대역폭 압축 효율, 전력 소비 및 성능 간의 원하는 트레이드오프들을 달성하기 위해 인에이블 또는 디스에이블될 수 있다. CMC(36)에 대해 이용 가능한 이 자원들을 이용하는 예시적인 메모리 대역폭 압축 기술들이 이제 논의될 것이다.

[0029] [0039] 이와 관련하여, 도 4는 메모리 대역폭 압축을 제공하기 위해 도 3의 CMC(36)에 의해 구현될 수 있는 예시적인 메모리 대역폭 압축 매커니즘(60)의 개략도이다. 메모리 대역폭 압축 매커니즘(60)에서, 시스템 메모리(38)는 복수의 메모리 라인들(62)을 포함하며, 이들 각각은 물리적 어드레스와 연관된다. 복수의 메모리 라인들(62) 각각은 메모리 판독 또는 기록 요청(도시되지 않음)의 물리적 어드레스를 이용하여 CMC(36)에 의해 액세스될 수 있다. 데이터(도시되지 않음)는 압축되거나 압축되지 않은 형태로 시스템 메모리(38)의 메모리 라인들(62) 각각 내에 저장될 수 있다. 일부 양상들에서, CI(64)를 포함하는 하나 또는 그 초과의 ECC 비트들은 메모리 라인(62)이 압축된 또는 그렇지 않은 형태로 저장되는지를 표시하기 위해 각각의 메모리 라인(62)과 연관되게 저장될 수 있다. 이러한 방식으로, 시스템 메모리(38)에 대한 메모리 액세스 요청을 수행할 때, CMC(36)는 메모리 액세스 요청의 프로세싱의 부분으로서 메모리 라인(62)이 압축되는지를 결정하기 위해 어드레싱될 물리적 어드레스에 대응하는 메모리 라인(62)과 연관되는 CI(64)를 검사할 수 있다.

[0030] [0040] 마스터 디렉토리(66)는 시스템 메모리(38)에서 또한 제공된다. 마스터 디렉토리(66)는 물리적 어드레스에 대응하는 시스템 메모리(38)의 메모리 라인(62) 당 하나의 엔트리(68)를 포함한다. 마스터 디렉토리(66)는 또한, 메모리 라인(62)이 메모리 라인(62)에 압축된 채로 저장되었는지 여부 그리고 만약 그렇다면, 다수의 압축 길이들이 지원되는 양상들에서, 데이터의 압축 길이를 표시하는 압축 패턴을 나타내기 위해 엔트리(68) 당 하나(1)의 CI(64)를 포함한다. 예를 들어, 메모리 라인(62)이 길이면에서 128 바이트이고 거기에 저장된 데이터가 64 바이트 또는 그 미만으로 압축될 수 있는 경우, 시스템 메모리(38)에 저장된 데이터에 대응하는 마스터 디렉토리(66)의 CI(64)는, 데이터가 128 바이트 메모리 라인(62) 중 처음 64 바이트들에 저장되었음을 표시하도록 세팅될 수 있다.

[0031] [0041] 도 4를 계속 참조하여, 기록 동작 동안, CMC(36)는 시스템 메모리(38)에 기록될 메모리 블록을 압축할 수 있다. 예를 들어, 데이터(예를 들어, 128 바이트 또는 256 바이트)가 압축된다. 압축된 데이터 블록이 시스템 메모리(38)의 메모리 블록 크기(예를 들어, 64 바이트) 보다 작거나 동일한 경우, 64 바이트가 기록되고, 그렇지 않으면 128 바이트가 기록된다. 256 바이트는 압축된 데이터 크기에 의존하여 64, 128, 192, 또는 256 바이트로서 기록될 수 있다. 시스템 메모리(38)의 메모리 라인(62)과 연관된 하나 또는 그 초과의 ECC 비트들에 저장된 CI(64)는 또한 메모리 라인(62)의 데이터가 압축되었는지 또는 아닌지를 나타내도록 세팅될 수 있다.

[0032] [0042] 판독 동작 예 동안, CMC(36)는 판독될 데이터가 시스템 메모리(38)에서 압축되었는지를 결정하기 위해 마스터 디렉토리(66)로부터 CI(64)를 판독할 수 있다. CI(64)에 기초하여, CMC(36)는 시스템 메모리(38)로부터 액세스될 데이터를 판독할 수 있다. 판독될 데이터가 CI(64)에 의해 표시된 바와 같이 시스템 메모리(38)에서 압축된 경우, CMC(36)는 하나의 메모리 판독 동작에 있어 전체 압축된 데이터 블록을 판독할 수 있다. 판독된 데이터의 부분이 시스템 메모리(38)에서 압축되지 않은 경우, 메모리 액세스 레이턴시는, 판독될 메모리 라인(62)의 부가적인 부분들이 또한 시스템 메모리(38)로부터 판독되어야 하기 때문에 부정적으로 영향을 받을 수 있다. 일부 양상들에서, CMC(36)는 주어진 세트의 상황들에서, 시스템 메모리(38)로부터 2번의 액세스들에서 데이터를 판독하는 것이 양호한지 또는 레이턴시 영향을 방지하기 위해 시스템 메모리(38)로부터 최대량의 데이터를 판독하는 것이 양호한지를 "러닝(learn)"하도록 구성될 수 있는 트레이닝 매커니즘(training mechanism)이 다수의 어드레스 범위들을 위해 이용될 수 있다.

[0033] [0043] 도 4의 예에서, CI 캐시(70)는 또한 시스템 메모리(38) 외부의 별개의 캐시에서 제공될 수 있다. CI 캐시(70)는, 시스템 메모리(38)의 메모리 라인(62)이 압축된 형태로 저장되었는지 여부를 나타내기 위해 시스템 메모리(38)의 메모리 라인(62) 당 하나의 캐시 엔트리(72)를 제공한다. 이러한 방식으로, 시스템 메모리(38)에 대한 메모리 액세스 요청을 수행할 때, CMC(36)는 먼저, 메모리 라인(62)을 판독해야 할 필요 없이, 메모리 액세스 요청의 프로세싱의 부분으로서 시스템 메모리(38)에서 물리적 어드레스의 메모리 라인(62)이 압축되었는지를 결정하기 위해 어드레싱될 물리적 어드레스에 대응하는 CI 캐시(70)의 캐시 엔트리(72)를 검사할 수 있다. 따라서, 메모리 라인(62)이 압축된 채로 저장되었음을 CI 캐시(70)가 표시하는 경우, CMC(36)는 전체 메모리 라인(62)을 판독할 필요가 없고, 이에 따라 레이턴시를 감소시킨다. 메모리 라인(62)이 압축되지 않은 채로 저장되었음을 CI 캐시(70)가 표시하는 경우, CMC(36)는 전체 메모리 라인(62)을 판독할 수 있다. CI 캐시(70)에서 미스(miss)가 발생한 경우, 마스터 디렉토리(66)에 저장된 대응하는 CI(64)가 참고(consult)되고 동일한 물리적

어드레스에 대한 후속 메모리 액세스 요청들에 대해 CI 캐시(70)에 로딩될 수 있다.

[0034] [0044] 일부 양상들에서, CI 캐시(70)는 종래의 캐시로서 구성될 수 있다. CI 캐시(70)는 태그 어레이를 포함할 수 있고 비-제한적인 예로서 n-웨이(way) 연관 캐시로서 구성될 수 있다. CMC(36)는 CI 캐시(70)에 관하여 퇴거 정책(eviction policy)을 구현할 수 있다. 도 4에서 도시된 CI 캐시(70)에서, 각각의 캐시 라인(74)은 다수의 캐시 엔트리들(72)을 저장할 수 있다. 각각의 캐시 엔트리(72)는, 캐시 엔트리(72)와 연관된 시스템 메모리(38)의 메모리 라인(62)이 압축되었는지를 표시하고 그리고/또는 캐시 엔트리(72)에 대응하는 데이터에 대한 압축 크기를 표시하는 압축 패턴을 나타내기 위한 CI(76)를 포함할 수 있다. 예를 들어, CI(76)는 4(four)개의 잠재적인 압축 크기들(예를 들어, 32, 64, 96, 또는 128 바이트)를 나타내는 2(two) 비트들을 포함할 수 있다. 이 예에서, CI(64)는 이 정보가 캐시 엔트리들(72)의 CI(76)에 또한 저장되기 때문에 리던던트(redundant)하다는 것에 주의한다. 예를 들어, 메모리 라인(62)이 길이면에서 128 바이트이고 거기에 저장된 데이터가 64 바이트 또는 그 미만으로 압축될 수 있는 경우, 시스템 메모리(38)의 메모리 라인(62)에 대응하는 CI 캐시(70)의 캐시 엔트리(72)의 CI(76)는, 데이터가 128 바이트 메모리 라인(62) 중 처음 64 바이트들에 저장되었음을 표시하도록 세팅될 수 있다.

[0035] [0045] 메모리 대역폭 압축 매커니즘을 위한 부가적인 캐시를 제공하는 것이 또한 바람직할 수 있다. 이와 관련하여, 도 5는 도 2의 SoC(10')와 같은 대안적인 SoC(10'')의 예를 예시하지만, 도 5의 SoC(10'')는 부가적으로, 이 예에서 L4 캐시인 선택적인 캐시(78)를 포함한다. CMC(36)는 레이턴시를 최소화하기 위해 L4 캐시(78) 및 CI 캐시(70) 둘 다의 물리적 어드레스를 동시에 룩 업(look up)할 수 있다. L4 캐시(78)의 어드레스들은 압축되지 않은 물리적 어드레스이다. L4 캐시(78)의 물리적 어드레스 히트(hit) 시에, CI 캐시(70)의 물리적 어드레스 룩업은 리던던트하다. L4 캐시(78)의 물리적 어드레스 미스 시에, CI 캐시(70)의 물리적 어드레스 룩업은 시스템 메모리(38)로부터 데이터를 획득하기 위해 요구된다. 또한, L4 캐시(78) 및 CI 캐시(70) 둘 다에 액세스하는 CPU(16)의 부가적인 레이턴시를 방지하기 위해, L4 캐시(78) 및 CI 캐시(70)는 준비(prime)될 수 있다.

[0036] [0046] 도 6a 및 도 6b는 예시적인 통신 흐름들 및 메모리 대역폭 압축을 제공하기 위해 도 3의 CMC(36)에 의해 액세스될 수 있는 도 4의 시스템 메모리(38) 및 마스터 디렉토리(66)의 예시적인 엘리먼트들을 예시하도록 제공된다. 특히, 각각, 도 6a는 메모리 판독 동작 동안 예시적인 통신 흐름들을 예시하는 반면에, 도 6b는 메모리 기록 동작 동안 예시적인 통신 흐름들을 예시한다. 도 6a 및 도 6b를 설명하는데 있어, 도 3 및 도 4의 엘리먼트들이 명료함을 위해 참조된다.

[0037] [0047] 도 6a 및 도 6b에서, 시스템 메모리(38)는 압축된 그리고 압축되지 않은 데이터를 저장하기 위해 복수의 메모리 라인들(80(0)-80(X))을 포함한다. 메모리 라인들(80(0)-80(X))은 시스템 메모리(38)의 근본적인 메모리 아키텍처에 의해 결정된 대로 각각의 메모리 블록들(82(0)-82(Z) 및 84(0)-84(Z))로 각각 세분된다. 일부 양상들에서, 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 각각의 크기는 메모리 판독 동작에서 시스템 메모리(38)로부터 판독될 수 있는 데이터의 최소량을 나타낸다. 예를 들어, 일부 예시적인 메모리 아키텍처들에서, 메모리 라인들(80(0)-80(X)) 각각은, 2개의 64바이트 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))로 세분되는 128 바이트의 데이터를 포함할 수 있다. 일부 양상들은, 메모리 라인들(80(0)-80(X)) 각각이 더 많은 또는 더 적은 바이트(예를 들어, 비-제한적인 예로서 256 바이트 또는 64 바이트)의 데이터를 포함할 수 있다는 것을 규정할 수 있다. 유사하게, 일부 양상들에서 따라, 메모리 라인들(80(0)-80(X)) 내의 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))은 더 크거나 더 작을 수 있다(예를 들어, 비-제한적인 예로서 128 바이트 또는 32 바이트). 일부 양상들에서, 메모리 판독 동작은 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 각각의 크기보다 더 적은 바이트를 판독하지만, 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 중 하나와 동일한 양의 메모리 대역폭을 여전히 소비할 수 있다.

[0038] [0048] 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 각각은 하나 또는 그 초과의 대응하는 ECC 비트들(86(0)-86(Z), 88(0)-88(Z))과 연관된다. ECC 비트들(86(0)-86(Z), 88(0)-88(Z))과 같은 ECC 비트들은 통상적으로, 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 내에서 흔히 직면하는 타입들의 내부 데이터 손상을 검출 및 정정하는데 이용된다. 도 6a 및 도 6b의 예에서, ECC 비트들(86(0)-86(Z), 88(0)-88(Z)) 중 하나 또는 그 초과는, 각각의 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))에 대한 CI들(90(0)-90(Z), 92(0)-92(Z))을 저장하도록 용도 변경(repurpose)된다. 도 6a 및 도 6b의 ECC 비트들(86(0)-86(Z), 88(0)-88(Z))이 그 각각의 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))에 인접한 것으로서 도시되지만, ECC 비트들(86(0)-86(Z), 88(0)-88(Z))은 시스템 메모리(38) 내의 다른 곳에 로케이팅될 수 있다는 것이 이해될 것이다.

[0039] [0049] 도 6a 및 도 6b의 예에서, 시스템 메모리(38)는 또한, 다수의 마스터 디렉토리 메모리 블록들(94(0)-94(Y))을 포함하는 마스터 디렉토리(66)를 포함한다. 마스터 디렉토리 메모리 블록들(94(0)-94(Y)) 각각은 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))과 동일한 크기이다. 마스터 디렉토리 메모리 블록들(94(0)-94(Y))은 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))을 저장하며, 이들 각각은 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 중 하나의 물리적 어드레스에 대응한다. 아래에서 보다 상세히 논의되는 바와 같이, 일부 양상들에서, ECC 비트들(86(0)-86(Z), 88(0)-88(Z))에 저장된 CI들(90(0)-90(Z), 92(0)-92(Z)) 및 마스터 디렉토리 메모리 블록들(94(0)-94(Y))에 저장된 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))은 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 중 주어진 하나에 대한 동일한 값을 항상 저장하도록 유지될 수 있다. 일부 양상들은, ECC 비트들(86(0)-86(Z), 88(0)-88(Z))에 저장된 CI들(90(0)-90(Z), 92(0)-92(Z))이 마스터 디렉토리 메모리 블록들(94(0)-94(Y))의 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))보다 더욱 자주 업데이트될 수 있다고 규정한다.

[0040] [0050] CI들(90(0)-90(Z), 92(0)-92(Z)) 및 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))은, 시스템 메모리(38)의 대응하는 메모리 블록(82(0)-82(Z), 84(0)-84(Z))에 저장된 데이터의 압축 상태를 표시하는 하나 또는 그 초과의 비트들을 각각 포함할 수 있다. 일부 양상들에서, CI들(90(0)-90(Z), 92(0)-92(Z)) 및 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W)) 각각은 대응하는 메모리 블록(82(0)-82(Z), 84(0)-84(Z))의 데이터가 압축되었는지 또는 압축되지 않았는지를 표시하는 단일 비트를 포함할 수 있다. 일부 양상들에 따라, CI들(90(0)-90(Z), 92(0)-92(Z)) 및 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))) 각각은 대응하는 메모리 블록들(82(0)-82(Z), 84(0)-84(Z)) 각각에 대한 압축 패턴(예를 들어, 비-제한적인 예로서, 압축된 데이터에 의해 점유되는 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))의 수)을 표시하는데 이용될 수 있는 다수의 비트들을 포함할 수 있다.

[0041] [0051] 도 6a의 예에서, 물리적 어드레스(104)를 특정하는 메모리 판독 요청(102)은 화살표(106)에 의해 표시된 바와 같이 CMC(36)에 의해 수신된다. 예시를 위해, 이 예에서, 물리적 어드레스(104)는 128-바이트 메모리 라인(80(0))의 64-바이트 세분인 메모리 블록(82(0))에 대응한다는 것이 가정된다. 메모리 판독 요청(102)이 수신되는 시간에, CMC(36)는 메모리 블록(82(0))에 저장된 데이터가 압축되었는지 여부를 인식하지 못한다. CMC(36)는 전체 메모리 라인(80(0))의 판독을 진행할 수 있지만, 요청된 데이터가 압축된 형태로 메모리 블록(82(0))에만 저장된 경우, 메모리 블록(82(Z))의 판독은 불필요할 것이고, 메모리 액세스 레이턴시를 증가시킬 것이다. 대안적으로, CMC(36)는 메모리 블록(82(0))만을 판독하고, 그의 콘텐츠에 기초하여 거기에 저장된 데이터가 압축되었는지 여부를 결정하고, 그 후 데이터가 압축되지 않은 경우 메모리 블록(82(Z))을 판독할 수 있다. 그러나 이 접근법은 또한 2개의 별개의 메모리 판독 동작을 발행하는 것으로부터 발생하는 증가된 메모리 액세스 레이턴시를 초래할 수 있다. 따라서, 메모리 블록(82(0))의 압축 상태를 인지함 없이, CMC(36)는 메모리 액세스 레이턴시에 부정적으로 영향을 줄 수 있는 불필요한 메모리 판독 동작의 실행을 감행한다.

[0042] [0052] CI들(90(0)-90(Z), 92(0)-92(Z)) 및/또는 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))을 이용함으로써, CMC(36)는, 주어진 메모리 판독 요청(102)에 대한 모든 데이터(압축되거나 압축되지 않음)가 시스템 메모리(38)로부터 효율적으로 판독될 수 있음을 보장하기 위해, 얼마나 많은 메모리 블록들(82(0)-82(Z))이 판독되어야 하는지를 결정할 수 있다. 따라서, 메모리 판독 요청(102)을 수신한 이후, CMC(36)는 화살표(108)에 의해 표시된 바와 같이, 메모리 블록(82(0))의 물리적 어드레스(104)에 대응하는 CI(96(0))에 액세스하기 위해 마스터 디렉토리(66)의 마스터 디렉토리 메모리 블록(94(0))을 판독할 수 있다. CI(96(0))는 그 후, 메모리 판독 요청(102)에 의해 요청된 데이터가 효율적으로 리트리브된다는 것을 보장하기 위해 메모리 라인(80(0)) 내의 얼마나 많은 메모리 블록들(82(0)-82(Z))을 판독할지 결정하도록 CMC(36)에 의해 이용될 수 있다.

[0043] [0053] 일부 양상들에서, CMC(36)는 CI(96(0))에 의해 표시된 압축 패턴(도시되지 않음)을 결정함으로써 얼마나 많은 메모리 블록들(82(0)-82(X))을 판독할지 결정할 수 있다. 위의 예에서, CI(96(0))는 데이터가 압축되었음(즉, 메모리 블록(82(0))만이 판독되어야 함) 또는 압축되지 않았음(즉, 메모리 블록들(82(0) 및 82(Z)) 둘 다가 판독되어야 함)을 표시하는 단일 비트일 수 있다. 시스템 메모리(38)의 메모리 라인들(80(0)-80(X))이 각각의 2개 초과의 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))을 포함하는 양상들에 따라, CI(96(0))는 얼마나 많은 메모리 블록들(82(0)-82(Z))이 압축된 데이터를 포함하는지를 표시하는 다수의 비트들을 포함할 수 있다. 예를 들어, 메모리 라인들(80(0)-80(X))이 4개의 64-바이트 메모리 블록들(82(0)-82(Z))로 세분되는 256-바이트 메모리 라인들인 경우, CI(96(0))는 압축된 데이터가 64 바이트, 128 바이트 또는 192 바이트(각각 메모리 블록(82(0)), 메모리 블록들(82(0)-82(1)), 및 메모리 블록들(82(0)-82(2))에 대응함)로 저장된다고 표시할 수 있다. CI(96(0))에 기초하여, CMC(36)는 그 후 화살표(110)에 의해 표시된 바와 같이 메모리 라인(80(0)) 내의

결정된 수의 메모리 블록들(82(0)-82(Z))을 판독하도록 적절한 수의 메모리 판독 동작들을 발행할 수 있다. 일부 양상들은, CI(96(0))에 의해 표시된 압축 패턴이, 물리적 어드레스(104)에 저장된 데이터가 전체적으로 0 값들로 이루어졌다고 CMC(36)에 표시하는 제로-라인 표시자를 포함한다고 CMC(36)가 결정할 수 있음을 규정할 수 있다. 이러한 경우에, CMC(36)는 제로 메모리 블록들(82(0)-82(Z))을 판독할 수 있다. 일부 양상들에 따라, CMC(36)는 CI(96(0))에 의해 표시된 압축 패턴이 (비-제한적인 예로서 모든 1(one)들과 같이) 물리적 어드레스(104)에 저장된 다른 고정된 데이터 패턴들의 표시자를 포함한다고 결정할 수 있다.

[0044] [0054] 일부 양상들에서, CMC(36)에 의해 수신된 메모리 판독 요청들(102)의 단지 특정 퍼센티지에 대한 마스터 디렉토리(66)로부터의 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))을 판독하는 것이 바람직할 수 있다. 예를 들어, 일부 양상들에서, 마스터 디렉토리(66)는, CMC(36)에 의해 수신된 메모리 판독 요청들(102)의 10퍼센트(10 %)에 대해서만 액세스될 수 있다. 이러한 양상에서, 메모리 판독 요청(102)의 수신 시에, CMC(36)는 마스터 디렉토리(66)로부터의 메모리 블록(82(0))에 대한 CI(96(0))를 판독할지 또는 대안적으로, ECC 비트들(86(0))로부터의 메모리 블록(82(0))에 대한 CI(90(0))를 판독할지를 (예를 들어, 랜덤 번호 생성기(도시되지 않음)에 기초하여) 확률론적으로 결정할 수 있다. 후자의 경우에, CMC(36)는 점선 화살표(112)에 의해 표시된 바와 같이 CI(90(0))를 판독하고, 메모리 라인(80(0))으로부터 얼마나 많은 메모리 블록들(82(0)-82(Z))을 판독할지 결정하는데 있어 CI(90(0))를 이용할 수 있다.

[0045] [0055] 이제 도 6b를 참조하면, CMC(36)는 화살표(116)에 의해 표시된 바와 같이 메모리 기록 요청(114)을 수신한다. 메모리 기록 요청(114)은 시스템 메모리(38)에 기록될 기록 데이터(118)는 물론, 기록 데이터(118)가 기록되는 시스템 메모리(38)의 물리적 어드레스(104) 둘 다를 포함한다. 예시 목적을 위해, 물리적 어드레스(104)는 메모리 블록(82(0))에 대응한다는 것이 재차 가정된다. 메모리 기록 요청(114)의 수신 시에, CMC(36)는 기록 데이터(118)를 압축하기 위한 동작들을 수행할 수 있고, 기록 데이터(118)의 압축의 결과들에 기초하여, 기록 데이터(118)에 대한 압축 패턴을 결정할 수 있다. 압축 패턴은, 비-제한적인 예로서, 기록 데이터(118)가 성공적으로 압축되었는지 여부, 및/또는 압축된 기록 데이터(118)에 의해 점유된 메모리 블록들(82(0)-82(Z))의 수를 표시할 수 있다. 압축 패턴에 기초하여, CMC(36)는 기록 데이터(118)에 대한 CI(120)를 생성한다.

[0046] [0056] CMC(36)는 그 후, 화살표(122)에 의해 표시된 바와 같이, 생성된 CI(120)로 메모리 블록(82(0))의 물리적 어드레스(104)에 대해 마스터 디렉토리(66)에 저장된 CI(96(0))를 업데이트할 수 있다. 일부 양상들에 따라, 마스터 디렉토리(66)는 CMC(36)에 의해 수신된 메모리 기록 요청들(114)의 특정 퍼센티지에 대해서만 업데이트될 수 있다. 예를 들어, 일부 양상들에서, 마스터 디렉토리(66)는, CMC(36)에 의해 수신된 메모리 기록 요청들(114)의 10퍼센트(10 %)에 대해서만 업데이트될 수 있다. 이에 따라, 이러한 양상들에서, CMC(36)는 마스터 디렉토리(66)에서 메모리 블록(82(0))에 대해 저장된 CI(96(0))를 업데이트할지를 (예를 들어, 랜덤 번호 생성기(도시되지 않음)에 기초하여) 확률론적으로 결정할 수 있다. 아닌 경우, 동작은 생략될 수 있다. CMC(36)는 그 후, 화살표(124)에 의해 표시된 바와 같이, 기록 데이터(118)를 메모리 블록들(82(0)-82(Z)) 중 하나 또는 그 초과에 기록한다. CMC(36)는 또한, 화살표(126)에 의해 표시된 바와 같이 기록 데이터(118)가 기록된 하나 또는 그 초과의 메모리 블록들(82(0)-82(Z))의 ECC 비트들(86(0)-86(Z))에 저장된 CI들(90(0)-90(Z))에 생성된 CI(120)를 기록한다.

[0047] [0057] 도 7은 도 6a에 관하여 논의된 바와 같은 메모리 대역폭 압축을 이용하여 도 3의 CMC(36)에 의해 메모리 판독 동작을 수행하기 위한 예시적인 동작들을 예시하는 흐름도이다. 명료함을 위해, 도 2, 도 3 및 도 6a의 엘리먼트들은 도 7을 설명하는데 있어 참조된다. 도 7에서, 동작들은, CMC(36)가 내부 시스템 버스(22)를 통해, 시스템 메모리(38)의 메모리 라인(80(0)) 내의 액세스될 메모리 블록(82(0))의 물리적 어드레스(104)를 포함하는 메모리 판독 요청(102)을 수신하는 것(블록 128)으로 시작한다. 일부 양상들에서, CMC(36)는 선택적으로, CI(96(0))가 마스터 디렉토리(66)로부터 판독되어야 하는지를 확률론적으로 결정할 수 있다(블록 130). CI(96(0))가 마스터 디렉토리(66)로부터 판독되지 않아야 한다고 CMC(36)가 결정한 경우, CMC(36)는 메모리 블록(82(0))의 판독과 함께 하나 또는 그 초과의 ECC 비트들(86(0))로부터 CI(90(0))를 판독하고, CI(96(0))로서 CI(90(0))를 이용한다(블록 132). 그러나 CMC(36)가 확률론적 결정을 이용하지 않는 경우, 또는 CI(96(0))가 마스터 디렉토리(66)로부터 판독되어야 한다는 것을 CMC(36)가 판단 블록(130)에서 확률론적으로 결정하는 경우, CMC(36)는 시스템 메모리(38)의 마스터 디렉토리(66)로부터 물리적 어드레스(104)에 대응하는 CI(96(0))를 포함하는 마스터 디렉토리 메모리 블록(94(0))을 판독한다(블록 134).

[0048] [0058] CMC(36)는 그 후, CI(96(0))에 기초하여, 메모리 판독 요청(102)에 대해 판독할 시스템 메모리(38)의 메모리 라인(80(0)) 내의 메모리 블록들(82(0)-82(Z))의 수를 결정한다(블록 136). 일부 양상들에서, 판독할 메

모리 블록들(82(0)-82(Z))의 수를 결정하기 위한 블록(136)의 동작들은 CI(96(0))에 의해 표시된 압축 패턴을 결정하는 것을 포함할 수 있다(블록 138). 예를 들어, CMC(36)는, CI(96(0))가 제로 메모리 블록들(82(0)-82(Z))이 판독되어야 한다고 표시하는 제로-라인 표시자를 포함한다고 결정할 수 있다(블록 140). CMC(36)는 그 후 물리적 어드레스(104)에서 시작하는 시스템 메모리(38)의 메모리 라인(80(0)) 내의 결정된 수의 메모리 블록들(82(0)-82(Z))을 판독한다(블록 142).

[0049] [0059] 도 6b에 관하여 위에서 논의된 바와 같은 메모리 대역폭 압축을 이용하여 도 3의 CMC(36)에 의해 메모리 기록 동작을 수행하기 위한 예시적인 동작들을 예시하기 위해 도 8이 제공된다. 이와 관련하여, 도 2, 도 3 및 도 6b의 엘리먼트들은 명료함을 위해 도 8을 설명하는데 있어 참조된다. 도 8의 동작들은, CMC(36)가 내부 시스템 버스(22)를 통해, 기록 데이터(118) 및 시스템 메모리(38)의 메모리 라인(80(0)) 내에 기록될 메모리 블록(82(0))의 물리적 어드레스(104)를 포함하는 메모리 기록 요청(114)을 수신하는 것(블록 144)으로 시작한다. 기록 데이터(118)를 압축하도록 시도한 이후, CMC(36)는 기록 데이터(118)에 대한 압축 패턴을 결정한다(예를 들어, 기록 데이터(118)가 압축되었는지 및/또는 얼마나 많은 메모리 블록들(82(0)-82(Z))이 기록 데이터(118)를 저장하기 위해 요구되는지)(블록 146). 압축 패턴에 기초하여, CMC(36)는 압축 패턴에 기초하여 기록 데이터(118)에 대한 CI(120)를 생성한다(블록 148).

[0050] [0060] 일부 양상들에 따라, CMC(36)는 선택적으로, 마스터 디렉토리(66)에 저장된 CI(96(0))가 업데이트되어야 하는지를 확률론적으로 결정할 수 있다(블록 150). 마스터 디렉토리(66)의 저장된 CI(96(0))가 업데이트되지 않아야 한다고 CMC(36)가 결정하는 경우, 프로세싱은 도 8의 블록(152)에서 재개된다. 그러나 마스터 디렉토리(66)의 저장된 CI(96(0))가 업데이트되어야 한다는 것을 CMC(36)가 판단 블록(150)에서 결정하는 경우, 또는 CMC(36)가 확률론적 결정을 이용하지 않는 경우, CMC(36)는 생성된 CI(120)로 마스터 디렉토리(66)의 물리적 어드레스(104)에 대응하는 저장된 CI(96(0))를 업데이트한다(블록 154).

[0051] [0061] CMC(36)는 다음으로, 생성된 CI(120)에 기초하여 시스템 메모리(38)의 메모리 라인(80(0))의 메모리 블록들(82(0)-82(Z)) 중 하나 또는 그 초과에 기록 데이터(118)를 기록한다(블록 152). CMC(36)는 또한, 시스템 메모리(38)의 메모리 라인(80(0))의 메모리 블록들(82(0)-82(Z)) 각각의 하나 또는 그 초과의 ECC 비트들(86(0)-86(Z)) 내에 생성된 CI(120)를 기록한다(블록 156). 일부 양상들에서, 생성된 CI(120)는 기록 데이터(118)를 제 1 메모리 블록(82(0))에 기록하는 것과 동일한 메모리 기록 동작의 부분으로서 ECC 비트들(86(0))에 기록된다.

[0052] [0062] 도 4에 관하여 위에서 언급된 바와 같이, CMC(36)에 의해 제공된 메모리 대역폭 압축은 일부 양상들에서, 도 4의 CI 캐시(70)의 이용을 통해 추가로 강화될 수 있다. 이와 관련하여, 도 9a 및 도 9b는 예시적인 통신 흐름들 및 메모리 대역폭 압축을 제공하기 위해 도 3의 CMC(36)에 의해 액세스될 수 있는 CI 캐시(70)의 예시적인 엘리먼트들을 예시하도록 제공된다. 특히, 도 9a 및 도 9b는 CI 캐시(70)를 이용한 메모리 판독 동작 및 메모리 기록 동작 동안 각각 예시적인 통신 흐름들을 예시한다. 도 9a 및 도 9b를 설명하는데 있어, 도 3, 도 4, 도 6a 및 도 6b의 엘리먼트들이 명료함을 위해 참조된다.

[0053] [0063] 도 9a에서 알 수 있는 바와 같이, CMC(36)의 CI 캐시(70)는 개시된 CI들(160(0)-160(W), 162(0)-162(W), 및 164(0)-164(W))을 저장하는 다수의 CI 캐시 엔트리들(158(0)-158(T))을 포함한다. 캐시 엔트리들(158(0)-158(T)) 각각은 마스터 디렉토리 메모리 블록들(94(0)-94(Y))과 동일한 크기이고, 일부 양상들에서, CMC(36)에 의한 메모리 기록 동작 또는 메모리 판독 동작 과정에서 판독되는 마스터 디렉토리 메모리 블록들(94(0)-94(Y)) 중 하나를 저장하도록 구성된다. 이러한 방식으로, CMC(36)는 CI들(90(0)-90(Z), 92(0)-92(Z)) 및/또는 CI들(96(0)-96(W), 98(0)-98(W), 100(0)-100(W))을 판독하기 위해 시스템 메모리(38)에 액세스해야 하기 보단, 후속 메모리 판독 동작들 및 메모리 기록 동작들 상에서 캐시된 CI들(160(0)-160(W), 162(0)-162(W), 및 164(0)-164(W))에 액세스할 수 있다. CMC(36)가 시스템 메모리(38)로의 기록 및 이로부터의 판독보다 훨씬 더 빨리 CI 캐시(70)에 기록하고 이로부터 판독하기 때문에, CI 캐시(70)의 이용은 CMC(36)가 메모리 액세스 레이턴시를 추가로 감소시키는 것을 가능케 한다.

[0054] [0064] 도 9a의 예에서, CI 캐시(70)를 제공하는 CMC(36)는 화살표(166)에 의해 표시된 바와 같이 물리적 어드레스(104)를 포함하는 메모리 판독 요청(102)을 수신한다. 물리적 어드레스(104)는 시스템 메모리(38)의 메모리 라인(80(0))의 메모리 블록(82(0))에 대응한다고 가정된다. CMC(36)는 그 후 물리적 어드레스(104)가 CI 캐시 엔트리들(158(0)-158(T)) 중 하나에 대응하는지를 결정할 수 있다. CI 캐시 엔트리들(158(0)-158(T)) 중 하나, 예컨대 CI 캐시 엔트리(158(0))가 메모리 블록(82(0))의 물리적 어드레스(104)에 대응하는 경우(즉, 캐시 "히트"), CMC(36)는 메모리 블록(82(0))에 대한 CI 표시자, 예컨대, CI(160(0))를 포함하는 CI 캐시 엔트리

(158(0))를 식별한다. CMC(36)는 그 후, CI 캐시(70)의 CI 캐시 엔트리(158(0))로부터 CI(160(0))를 판독하고도 6a에 관하여 위에서 논의된 바와 같이, CI(160(0))에 기초하여 판독할 메모리 블록들(82(0)-82(Z))의 수를 결정한다.

[0055] [0065] 그러나 물리적 어드레스(104)가 CI 캐시 엔트리들(158(0)-158(T)) 중 어느 것에도 대응하지 않는다고 CMC(36)가 결정한 경우, CMC(36)는 화살표(168)에 의해 표시된 바와 같이, 메모리 블록(82(0))의 물리적 어드레스(104)에 대응하는 CI(96(0))에 액세스하도록 마스터 디렉토리(66)의 마스터 디렉토리 메모리 블록(94(0))을 판독할 수 있다. 위에서 언급된 바와 같이, 일부 양상들에서, CMC(36)에 의해 수신된 메모리 판독 요청(102)의 단지 특정 퍼센티지에 대해 CMC(36)가 마스터 디렉토리(66)를 판독하는 것이 바람직할 수 있다. 이에 따라, 이러한 양상에서, CMC(36)는 마스터 디렉토리(66)로부터의 마스터 디렉토리 메모리 블록(94(0))을 판독할지 또는 대안적으로, ECC 비트들(86(0))로부터의 메모리 블록(82(0))에 대한 CI(90(0))를 판독할지를 (예를 들어, 랜덤 번호 생성기(도시되지 않음)에 기초하여) 확률론적으로 결정할 수 있다. 후자의 경우에, CMC(36)는 점선 화살표(170)에 의해 표시된 바와 같이 CI(90(0))을 판독할 수 있다. CMC(36)는 그 후 CI 캐시(70)를 추가로 업데이트함 없이 CI(90(0))를 이용할 수 있다.

[0056] [0066] CMC(36)는 그 후 새로운 CI 캐시 엔트리(158(0))로서 CI 캐시(70)에 마스터 디렉토리 메모리 블록(94(0))을 기록할 수 있다. 일부 양상들에서, 마스터 디렉토리 메모리 블록(94(0))을 새로운 CI 캐시 엔트리(158(0))로서 기록하기 이전에, CI 캐시(70)는 먼저 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 하는지를 결정할 수 있다. 만약 그렇다면, CI 캐시(70)는 추가로, 현재 CI 캐시 엔트리(158(0))가 CI 캐시(70)에 기록된 이래로 수정되었는지를(예를 들어, 마스터 디렉토리(66)로부터 판독된 이래로 적어도 하나의 캐시된 CI(160(0)-160(W))가 변경되었는지를 결정함으로써) 결정할 수 있다. 일부 양상들에서, CI 캐시(70)는, 현재 CI 캐시 엔트리(158(0))와 연관된 더티(dirty) 비트(도시되지 않음)가 세팅되었는지를 확인하도록 검사함으로써 현재 CI 캐시 엔트리(158(0))가 수정되었는지를 결정할 수 있다. 현재 CI 캐시 엔트리(158(0))가 수정된 경우, CMC(36)는 마스터 디렉토리 메모리 블록들(94(0)-94(Y)) 중 대응하는 하나에 현재 CI 캐시 엔트리(158(0))를 기록한다. CMC(36)는 그 후, CI 캐시(70)의 CI 캐시 엔트리(158(0))로부터 CI(160(0))를 판독하고, 위에서 논의된 바와 같이, CI(160(0))에 기초하여 판독할 메모리 블록들(82(0)-82(Z))의 수를 결정한다. 결정된 수의 메모리 블록들(82(0)-82(Z))은 그 후 화살표(172)에 의해 표시된 바와 같이 CMC(36)에 의해 판독된다.

[0057] [0067] 일부 양상들에서, CMC(36)는, 물리적 어드레스(104)가 CI 캐시 엔트리들(158(0)-158(T)) 중 하나에 대응한다는 결정과 동시에, 메모리 판독 요청(102)에 기초하여, 얼리(early) 메모리 판독 요청(174)을 시스템 메모리(38)에 발행할 수 있다. 얼리 메모리 판독 요청(174)은, CMC(36)가 CI(160(0))에 기초하여 판독할 메모리 블록들(82(0)-82(Z))의 수를 결정하는 것과 병렬로 프로세싱될 수 있다. 판독할 메모리 블록들(82(0)-82(Z))의 수의 결정 시에, 얼리 메모리 판독 요청(174)은 CI(160(0))에 기초하여 수정될 수 있다. 비-제한적인 예로서, 얼리 메모리 판독 요청(174)은 결정된 수의 메모리 블록들(82(0)-82(Z))을 판독하도록 수정될 수 있다. 이러한 방식으로, 부가적인 메모리 대역폭 압축 및 감소된 메모리 액세스 레이턴시는, 얼리 메모리 판독 요청(174) 및 CI 캐시(70) 판독들 및/또는 기록들의 병렬 프로세싱을 통해 제공될 수 있다.

[0058] [0068] 또한, 도 5에 관하여 위에서 논의된 일부 양상들에 따라, CMC(36)는 또한, 물리적 어드레스(104)가 CI 캐시(70)의 CI 캐시 엔트리들(158(0)-158(T)) 중 하나에 대응하는지를 결정하는 것과 동시에 L4 캐시(78)의 물리적 어드레스(104)를 룩업하기 위한 캐시 판독 동작(도시되지 않음)을 수행할 수 있다. 이를 행함으로써, CMC(36)는 메모리 액세스 레이턴시를 추가로 감소시킬 수 있다.

[0059] [0069] 이제 도 9b를 참조하면, CMC(36)는 화살표(176)에 의해 표시된 바와 같이 메모리 기록 요청(114)을 수신한다. 메모리 기록 요청(114)은 시스템 메모리(38)에 기록될 기록 데이터(118)는 물론, 기록 데이터(118)가 기록되는 시스템 메모리(38)의 물리적 어드레스(104) 둘 다를 포함한다. 예시 목적을 위해, 물리적 어드레스(104)는 메모리 블록(82(0))에 대응한다는 것이 재차 가정된다. 메모리 기록 요청(114)의 수신 시에, CMC(36)는 기록 데이터(118)를 압축하기 위한 동작들을 수행할 수 있고, 기록 데이터(118)의 압축의 결과들에 기초하여, 기록 데이터(118)에 대한 압축 패턴을 결정할 수 있다. 압축 패턴은, 비-제한적인 예로서, 기록 데이터(118)가 성공적으로 압축되었는지 여부, 및/또는 압축된 기록 데이터(118)에 의해 점유된 메모리 블록들(82(0)-82(Z))의 수를 표시할 수 있다. 압축 패턴에 기초하여, CMC(36)는 기록 데이터(118)에 대한 CI(120)를 생성한다.

[0060] [0070] CMC(36)는 그 후 물리적 어드레스(104)가 CI 캐시 엔트리들(158(0)-158(T)) 중 하나에 대응하는지를 결정할 수 있다. CI 캐시 엔트리들(158(0)-158(T)) 중 하나, 예컨대 CI 캐시 엔트리(158(0))가 메모리 블록

(82(0))의 물리적 어드레스(104)에 대응하는 경우(즉, 캐시 "히트"), CMC(36)는 메모리 블록(82(0))에 대한 CI 표시자, 예컨대, CI(160(0))를 포함하는 CI 캐시 엔트리(158(0))를 식별한다. 이에 따라, CMC(36)는 화살표(178)에 의해 표시된 바와 같이 생성된 CI(120)로 CI 캐시 엔트리(158(0))의 CI(160(0))를 업데이트한다. CMC(36)는 다음으로, 화살표(180)에 의해 표시된 바와 같이, 메모리 라인(80(0))의 메모리 블록들(82(0)-82(Z)) 중 하나 또는 그 초과에 기록 데이터(118)를 기록한다. CMC(36)는 또한, 화살표(182)에 의해 표시된 바와 같이 기록 데이터(118)가 기록된 메모리 블록들(82(0)-82(Z))의 하나 또는 그 초과의 메모리 블록의 ECC 비트들(86(0)-86(Z))의 하나 또는 그 초과에 생성된 CI(120)를 기록한다. 일부 양상들에 따라, 생성된 CI(120)는 기록 데이터(118)를 제 1 메모리 블록(82(0))에 기록하는 것과 동일한 메모리 기록 동작의 부분으로서 ECC 비트들(86(0))에 기록된다.

[0061] [0071] 그러나 물리적 어드레스(104)가 CI 캐시 엔트리들(158(0)-158(T)) 중 어느 것에도 대응하지 않는다고 CMC(36)가 결정한 경우(즉, 캐시 미스), CMC(36)는 화살표(184)에 의해 표시된 바와 같이 생성된 CI(120)로 마스터 디렉토리(66)의 물리적 어드레스(104)에 대응하는 저장된 CI(96(0))를 업데이트할 수 있다. CMC(36)는 그 후 화살표(186)에 의해 표시된 바와 같이, 새로운 CI 캐시 엔트리(158(0))로서 CI 캐시(70)에 마스터 디렉토리 메모리 블록(94(0))을 기록할 수 있다. 일부 양상들에서, 마스터 디렉토리 메모리 블록(94(0))을 새로운 CI 캐시 엔트리(158(0))로서 기록하기 이전에, CI 캐시(70)는 먼저 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 하는지를 결정할 수 있다. 만약 그렇다면, CI 캐시(70)는, 현재 CI 캐시 엔트리(158(0))가 CI 캐시(70)에 기록된 이래로 수정되었는지를 추가로 결정할 수 있다. 일부 양상들에서, CI 캐시(70)는, CI 캐시 엔트리(158(0))와 연관된 더티 비트(도시되지 않음)가 세팅되었는지를 확인하도록 검사함으로써 현재 CI 캐시 엔트리(158(0))가 수정되었는지를 결정할 수 있다. 현재 CI 캐시 엔트리(158(0))가 수정된 경우, 새로운 CI 캐시 엔트리(158(0))로서 CI 캐시(70)에 마스터 디렉토리 메모리 블록(94(0))을 기록하기 이전에, CMC(36)는 화살표(188)에 의해 표시된 바와 같이, 마스터 디렉토리 메모리 블록들(94(0)-94(Y)) 중 대응하는 하나에 현재 CI 캐시 엔트리(158(0))를 기록한다.

[0062] [0072] 일부 양상들에서, 마스터 디렉토리(66)는 CMC(36)에 의해 수신된 메모리 기록 요청들(114)의 단지 특정 퍼센티지에 대해 CI 캐시(70)의 캐시 미스에 대한 응답으로 업데이트될 수 있다. 이에 따라, 이러한 양상들에서, CMC(36)는 마스터 디렉토리(66)에서 메모리 블록(82(0))에 대해 저장된 CI(96(0))를 업데이트할지를 (예를 들어, 랜덤 번호 생성기(도시되지 않음)에 기초하여) 확률론적으로 결정할 수 있다. 아닌 경우, 캐시 미스에 관하여 위에서 설명된 동작들은 생략된다(즉, 마스터 디렉토리(66) 또는 CI 캐시(70) 어느 것도 업데이트되지 않고 생성된 CI(120)는 메모리 블록들(82(0)-82(Z))의 하나 또는 그 초과의 메모리 블록의 ECC 비트들(86(0)-86(Z))에 저장됨).

[0063] [0073] 도 10a 내지 도 10c는 도 9a 및 도 9b의 CI 캐시(70)를 이용하여 도 3의 CMC(36)에 의해 메모리 판독 동작을 수행하기 위한 예시적인 동작들을 예시하는 흐름도들이다. 특히, 도 10a는 메모리 판독 요청(102)을 수신하고 메모리 판독 요청(102)의 물리적 어드레스(104)에 대응하는 CI(160(0))에 대한 CI 캐시(70)의 검색이 히트 또는 미스를 초래하는지를 결정하는 동작들을 예시한다. 도 10b는 CI 캐시(70) 상의 캐시 미스의 결과로서 수행되는 동작들을 예시하는 반면에, 도 10c는 CI 캐시(70) 상의 캐시 히트의 결과로서 수행되는 동작들을 예시한다. 도 2, 도 3 및 도 9a의 엘리먼트들은 명료함을 위해 도 10a 내지 도 10c를 설명하는데 있어 참조된다.

[0064] [0074] 도 10a의 동작들은, CMC(36)가 내부 시스템 버스(22)를 통해, 시스템 메모리(38)의 메모리 라인(80(0)) 내의 액세스될 메모리 블록(82(0))의 물리적 어드레스(104)를 포함하는 메모리 판독 요청(102)을 수신하는 것(블록 190)으로 시작한다. 일부 양상들에서, CMC(36)는 후속 동작들과 동시에 얼리 메모리 판독 요청(174)을 시스템 메모리(38)에 송신할 수 있다(블록 192). CMC(36)는, 일부 양상들에 따라, 후속 동작들과 동시에, 도 5의 L4 캐시(78)와 같은 L4 캐시 상에서 캐시 판독 동작을 수행할 수 있다(블록 194). 그 후 L4 캐시(78) 상의 캐시 판독 동작이 캐시 히트를 초래하는지에 관한 결정이 내려진다(블록 195). 만약 그렇다면, 캐시 판독 동작의 결과들이 리턴되고, 시스템 메모리(38)에 대한 판독은 무의미해진다(블록 196).

[0065] [0075] CMC(36)는 다음으로, 물리적 어드레스(104)가 CI 캐시(70)의 복수의 CI 캐시 엔트리들(158(0)-158(T))의 CI 캐시 엔트리(158(0))에 대응하는지를 결정한다(블록 197). 물리적 어드레스(104)가 복수의 CI 캐시 엔트리들(158(0)-158(T))의 CI 캐시 엔트리(158(0))에 대응하지 않는다고 CMC(36)가 판단 블록(196)에서 결정하는 경우(즉, 캐시 미스), 프로세싱은 도 10b의 블록(198)에서 재개된다. 물리적 어드레스(104)가 CI 캐시 엔트리(158(0))에 대응하는 것으로 결정된 경우(즉, 캐시 히트), 프로세싱은 도 10c의 블록(200)에서 재개된다.

[0066] [0076] 이제 도 10b를 참조하면, CMC(36)는 일부 양상들에서, 캐시 미스에 대한 응답으로 CI(96(0))가 마스터

디렉토리(66)로부터 판독되어야 하는지를 확률론적으로 결정할 수 있다(블록 198). 이러한 양상들에서, 메모리 블록들(82(0)-82(Z), 84(0)-84(Z))의 하나 또는 그 초과의 ECC 비트들(86(0)-86(Z), 88(0)-88(Z))은 CI들(90(0)-90(Z), 92(0)-92(Z))을 저장할 것이란 점에 주의한다. 확률론적 결정은, 비-제한적인 예로서 랜덤 번호 생성기를 이용하여 내려질 수 있다. CI(96(0))가 마스터 디렉토리(66)로부터 판독되어야 한다는 것이 판단 블록(198)에서 결정된 경우, CMC(36)는 시스템 메모리(38)의 마스터 디렉토리(66)로부터 물리적 어드레스(104)에 대응하는 CI(96(0))를 포함하는 마스터 디렉토리 메모리 블록(94(0))을 판독한다(블록 202). CI(96(0))가 마스터 디렉토리(66)로부터 판독되지 않아야 한다고 판단 블록(198)에서 CMC(36)가 결정한 경우, CMC(36)는 메모리 블록(82(0))의 판독과 함께 하나 또는 그 초과의 ECC 비트들(86(0))로부터 CI(90(0))를 판독한다(블록 204). 프로세싱은 그 후 도 10c의 블록(214)에서 재개된다.

[0067] [0077] 도 10b를 계속 참조하면, CMC(36)는 다음으로, 일부 양상들에 따라, CI 캐시(70)의 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 하는지를 결정할 수 있다(블록 206). 아닌 경우, 프로세싱은 도 10b의 블록(208)에서 재개된다. 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 한다고 CMC(36)가 판단 블록(206)에서 결정한 경우, CMC(36)는 다음으로, 현재 CI 캐시 엔트리(158(0))가 수정되었는지를 결정한다(블록 210). 이 결정은 비-제한적인 예로서, CI 캐시 엔트리(158(0))의 더티 비트가 세팅되었는지를 결정하는 것에 기초할 수 있다. 현재 CI 캐시 엔트리(158(0))가 변경되지 않은 것으로 판단 블록(210)에서 결정된 경우, 프로세싱은 도 10b의 블록(208)에서 재개된다. 그렇지 않으면, CMC(36)는 현재 CI 캐시 엔트리(158(0))를 마스터 디렉토리(66)에 기록한다(블록 212). 마스터 디렉토리 메모리 블록(94(0))은 그 후 CI 캐시(70)의 CI 캐시 엔트리(158(0))에 기록된다(블록 208). 프로세싱은 그 후 도 10c의 블록(200)에서 재개된다.

[0068] [0078] 이제 도 10c로 넘어가서, CMC(36)는 CI 캐시(70)의 CI 캐시 엔트리(158(0))로부터 CI(160(0)를 판독한다(블록 200). CMC(36)는 다음으로, CI(160(0))에 기초하여, 메모리 판독 요청(102)에 대해 판독할, 시스템 메모리(38)의 메모리 라인(80(0)) 내의 메모리 블록들(82(0)-82(Z))의 수를 결정한다(블록 214). CMC(36)는 그 후 물리적 어드레스(104)에서 시작하는 시스템 메모리(38)의 메모리 라인(80(0)) 내의 결정된 수의 메모리 블록들(82(0)-82(Z))을 판독한다(블록 216). 얼리 메모리 판독 요청(174)이 발행된 양상들에서, 결정된 수의 메모리 블록들(82(0)-82(Z))을 판독하기 위한 블록(216)의 동작들은 CI(160(0))에 기초하여 얼리 메모리 판독 요청(174)을 수정하는 것을 포함할 수 있다(블록 218).

[0069] [0079] 도 8의 CI 캐시(70)를 이용하여 도 3의 CMC(36)에 의해 메모리 기록 동작을 수행하기 위한 예시적인 동작들을 예시하기 위해, 도 11a 내지 도 11c가 제공된다. 도 11a는, 메모리 기록 요청(114)을 수신하고, 메모리 기록 요청(114)의 물리적 어드레스(104)에 대응하는 CI 캐시 엔트리(158(0)-158(T))에 대한 CI 캐시(70)의 검색이 히트 또는 미스를 초래하는지를 결정하는 동작들을 예시한다. 도 11b는 CI 캐시(70) 상의 캐시 미스의 결과로서 수행되는 동작들을 예시하는 반면에, 도 11c는 CI 캐시(70) 상의 캐시 히트의 결과로서 수행되는 동작들을 예시한다. 도 2, 도 3 및 도 9b의 엘리먼트들은 명료함을 위해 도 11a 내지 도 11c를 설명하는데 있어 참조된다.

[0070] [0080] 도 11a에서, 동작들은, CMC(36)가 내부 시스템 버스(22)를 통해, 기록 데이터(118) 및 시스템 메모리(38)의 메모리 라인(80(0)) 내의 기록될 메모리 블록(82(0))의 물리적 어드레스(104)를 포함하는 메모리 기록 요청(114)을 수신하는 것(블록 220)으로 시작한다. CMC(36)는 기록 데이터(118)에 대한 압축 패턴을 결정한다(블록 222). 일부 양상들에서, 압축 패턴은, 기록 데이터(118)가 압축되었는지 또는 압축되지 않았는지, 및/또는 기록 데이터(118)에 의해 점유되는 메모리 블록들(82(0)-82(Z))의 수를 표시할 수 있다. 압축 패턴에 기초하여, CMC(36)는 압축 패턴에 기초하여 기록 데이터(118)에 대한 CI(120)를 생성한다(블록 224).

[0071] [0081] CMC(36)는 다음으로, 물리적 어드레스(104)가 CI 캐시(70)의 복수의 CI 캐시 엔트리들(158(0)-158(T))의 CI 캐시 엔트리(158(0))에 대응하는지를 결정한다(블록 226). 물리적 어드레스(104)가 복수의 CI 캐시 엔트리들(158(0)-158(T))의 CI 캐시 엔트리(158(0))에 대응하지 않는 것으로 판단 블록(226)에서 결정되는 경우(즉, 캐시 미스), 프로세싱은 도 11b의 블록(228)에서 재개된다. 그러나 물리적 어드레스(104)가 CI 캐시 엔트리(158(0))에 대응한다고 CMC(36)가 판단 블록(226)에서 결정한 경우(즉, 캐시 히트), 프로세싱은 도 11c의 블록(230)에서 재개된다.

[0072] [0082] 이제 도 11b를 참조하면, CMC(36)의 일부 양상들은 CI(96(0))가 마스터 디렉토리(66)에서 업데이트되어야 하는지를 확률론적으로 결정할 수 있다(블록 228). CI(96(0))가 업데이트되지 않아야 한다고 판단 블록(228)에서 결정되는 경우, 프로세싱은 도 11c의 블록(242)에서 재개된다. 그러나 CMC(36)가 CI(96(0))를 업데이트하도록 판단 블록(228)에서 결정하거나 또는 CMC(36)가 확률론적 결정을 이용하도록 구성되지 않는 경우,

CMC(36)는 생성된 CI(120)로 마스터 디렉토리(66)의 물리적 어드레스(104)에 대응하는 저장된 CI(96(0))를 업데이트한다(블록 232).

[0073] [0083] 일부 양상들에서, CMC(36)는 다음으로, CI 캐시(70)의 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 하는지를 결정할 수 있다(블록 234). 아닌 경우, 프로세싱은 도 11b의 블록(236)에서 재개된다. 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 한다고 CMC(36)가 판단 블록(234)에서 결정한 경우, CMC(36)는 현재 CI 캐시 엔트리(158(0))가 수정되었는지를 결정한다(블록 238). CI 캐시 엔트리(158(0))가 수정되었는지를 결정하는 것은, 비-제한적인 예로서, CI 캐시 엔트리(158(0))의 더티 비트가 세팅되었는지를 결정하는 것을 포함한다. 현재 CI 캐시 엔트리(158(0))가 변경되지 않은 것으로 판단 블록(238)에서 결정된 경우, 프로세싱은 도 11b의 블록(236)에서 재개된다. 그러나 현재 CI 캐시 엔트리(158(0))가 수정되었다고 판단 블록(238)에서 CMC(36)가 결정하는 경우, CMC(36)는 현재 CI 캐시 엔트리(158(0))를 마스터 디렉토리(66)에 기록한다(블록 240). CMC(36)는 그 후, CI 캐시(70)의 CI 캐시 엔트리(158(0))에 저장된 CI(96(0))를 기록한다(블록 236). 프로세싱은 도 11c의 블록(242)에서 재개된다.

[0074] [0084] 도 11c에서, CMC(36)는 생성된 CI(120)으로 CI 캐시(70)의 CI 캐시 엔트리(158(0))를 업데이트한다(블록 230). CMC(36)는 그 후 생성된 CI(120)에 기초하여 시스템 메모리(38)의 메모리 라인(80(0))의 하나 또는 그 초과의 메모리 블록들(82(0)-82(Z))에 기록 데이터(118)를 기록한다(블록 242). 생성된 CI(120)는 시스템 메모리(38)의 메모리 라인(80(0))의 하나 또는 그 초과의 메모리 블록들(82(0)-82(Z)) 각각의 하나 또는 그 초과의 ECC 비트들(86(0)-86(Z)) 내에 기록된다(블록 244). 일부 양상들에서, 생성된 CI(120)는 기록 데이터(118)를 제 1 메모리 블록(82(0))에 기록하는 것과 동일한 메모리 기록 동작의 부분으로서 ECC 비트들(86(0))에 기록된다.

[0075] [0085] 위에서 논의된 바와 같이, 압축을 최적화하기 위한 특수 경우로서 구성 가능한 고정된 데이터 패턴을 갖는 메모리 데이터 블록(82(0)-82(Z))의 압축을 제공하는 것이 바람직할 수 있다. 예를 들어, 메모리 데이터 블록(82(0)-82(Z))(예를 들어, 128 바이트)이 모두 0들인 경우, 데이터는 판독 또는 기록되지 않지만, 모든 0들로서 별개의 데이터 구조에 표시한다. 제로-비트는, 메모리 라인(80(0)-80(X))이 압축되어 메모리 데이터 블록(82(0)-82(Z)) 당 1(one) 비트를 부가하는지를 나타내기 위해 시스템 메모리(38)에 제공될 수 있다. CMC(36)는 예를 들어, 블록이 모두 0인 경우 128 바이트 메모리 블록들(82(0)-82(Z))의 수를 기억하는 캐시를 유지할 수 있다. 기록 동작에 대해, 메모리 데이터 블록(82(0)-82(Z))의 라인이 모두 0들인 경우, CMC(36)는 시스템 메모리(38)에 라인을 기록하는 것이 아니라 CI 캐시(70)가 업데이트된다. 판독 동작에 대해, CI 캐시(70)의 제로-비트가 라인에 대해 검사된다. 라인이 CI 캐시(70)에 있는 경우, 제로-비트에 의존하여, 라인이 시스템 메모리(38)로부터 판독되거나, 또는 모든 0들이 리턴된다. 라인이 CI 캐시(70)에 있지 않은 경우, 그것은 시스템 메모리(38)로부터 판독된다. 라인이 0인 경우, CI 캐시(70)는 업데이트될 수 있다.

[0076] [0086] CI 캐시(70)는 알려진 기술들(LRU, 의사(pseudo)-LRU 등)에 따라 캐시 라인(74)을 퇴거하도록 업데이트될 수 있다. 라인이 퇴거되는 경우, 그의 제로-비트 세트를 갖는 모든 라인들이 메인 메모리에 기록될 필요는 없다. 이는 라이트-올-제로 큐(write-all-zeros queue)로서 구성될 수 있다. 가능한 최적화는 2(two) 비트들을 이용할 것이고, 여기서 1비트는 라인 모두 0인 경우를 나타내고, 다른 비트는 라인이 더티(즉, 메인 메모리에 아직 기록되어 있지 않음)인 경우를 나타낸다. CMC(36)의 백그라운드 작업은 CI 캐시(70)를 검사하고, "라이트-올-제로" 큐에 더티-비트들을 갖는 라인들을 큐잉한다. 위에서 설명된 메모리 대역폭 압축 매커니즘을 통해, 상이한 메모리 압축 매커니즘들은 원하는 대로 CMC(36)에 의해 이용될 수 있다. 예를 들어, 64, 128, 및 256 바이트와 같은 작은 데이터 블록들에 대해 최적화되는 메모리 압축 매커니즘들을 이용하는 것이 바람직할 수 있는데, 그 이유는 위의 예들에서 설명된 캐시 라인들(74)이 예로서 이 크기들을 포함하기 때문이다.

[0077] [0087] 일부 양상들에서, 다수의 비트들을 포함하는 CI의 값은, 메모리 블록들(82(0)-82(Z)) 중 하나와 같은 메모리 블록에 저장된 고정된 데이터 패턴 및/또는 압축 상태를 표시할 수 있다. 비-제한적인 예로서, 2(two) 비트들의 CI에 대해, "00"의 값은, 대응하는 메모리 블록이 압축되지 않았음을 표시할 수 있는 반면에, "01"의 값은 대응하는 메모리 블록이 압축되었음을 표시할 수 있다. "11"의 값은 고정된 패턴(예를 들어, 모두 0(zero)들, 또는 모두 1(one)들)이 대응하는 메모리 블록에 저장되었음을 표시할 수 있다. 이러한 양상들에서, 마스터 디렉토리(66)는 확률론적 결정을 조건으로 업데이트되기 보단, 항상 업데이트될 것이다.

[0078] [0088] 이와 관련하여, 도 12는 빈발 패턴 압축 데이터 압축 매커니즘(246)을 예시한다. 이와 관련하여, 압축될 소스 데이터 포맷(248)의 소스 데이터는 예로서 128바이트로 도시된다. 압축 데이터 포맷(250)이 아래에서 도시된다. 압축된 데이터 포맷(250)은 프리픽스 코드들(Px) 및 DaTax로서 프리픽스 뒤의 데이터의 포맷으로 제

공된다. 프리픽스는 3-비트이다. 프리픽스 코드들은, 빈발 패턴 인코딩 표(254)의 프리픽스 코드 열(252)에서 도시되며, 이 표는 프리픽스 코드 열(252)에서의 주어진 프리픽스 코드에 대해 인코딩된 패턴 열(256)에서의 인코딩된 패턴을 도시한다. 인코딩된 패턴에 대한 데이터 크기는, 빈발 패턴 인코딩 표(254)의 데이터 크기 열(258)에서 제공된다.

[0079] [0089] 도 13은 32-비트 빈발 패턴 압축 데이터 압축 매커니즘(260)을 예시한다. 이와 관련하여, 압축될 소스 데이터 포맷(262)의 소스 데이터는 예로서 128바이트로 도시된다. 압축 데이터 포맷(264)이 아래에서 도시된다. 압축된 데이터 포맷(264)은 프리픽스(Px) 및 DaTax로서 프리픽스 바로 뒤의 데이터의 포맷으로 제공된다. 새로운 압축된 데이터 포맷(266)이 프리픽스 코드들(Px), 데이터(Datax), 플래그들 및 패턴들의 상이한 포맷으로 제공되며, 이는 효율성 목적으로 함께 그룹핑되도록 구성된다. 프리픽스 코드는 3-비트이다. 프리픽스 코드들은 빈발 패턴 인코딩 표(270)의 프리픽스 코드 열(268)에서 도시되며, 이 표는 프리픽스 코드 열(268)에서의 주어진 프리픽스 코드에 대해 패턴 인코딩된 열(272)에서의 인코딩된 패턴을 도시한다. 인코딩된 패턴에 대한 데이터 크기는, 빈발 패턴 인코딩 표(270)의 데이터 크기 열(274)에서 제공된다. 프리픽스 코드 000은, 새로운 압축된 데이터 포맷(266)에서 32-비트의 최대 크기의 데이터일 수 있는 압축되지 않은 패턴을 나타낸다. 프리픽스 코드 001은 새로운 압축된 데이터 포맷(266)의 데이터에서 0비트로서 제공될 수 있는 모두 0의 데이터 블록을 나타낸다. 3-비트 프리픽스를 통해, 프리픽스 코드들(010-111)은, 이 예에서 각각 0, 4, 8, 12, 16, 및 24 비트의 패턴들인, 소스 데이터에서 인지되는 다른 특정 패턴들을 인코딩하는데 이용될 수 있다.

[0080] [0090] 도 14는 32-비트 빈발 패턴 압축 데이터 압축 매커니즘(276)의 예를 예시한다. 이와 관련하여, 압축될 소스 데이터 포맷(278)의 소스 데이터는 예로서 128바이트로 도시된다. 압축 데이터 포맷(280)이 아래에서 도시된다. 압축된 데이터 포맷(280)은 프리픽스(Px) 및 DaTax로서 프리픽스 바로 뒤의 데이터의 포맷으로 제공된다. 새로운 압축된 데이터 포맷(282)이 프리픽스 코드들(Px), 데이터(Datax), 플래그들 및 패턴들의 상이한 포맷으로 제공되며, 이는 효율성 목적으로 함께 그룹핑되도록 구성된다. 프리픽스 코드는 3-비트이다. 프리픽스 코드들은, 빈발 패턴 인코딩 표(286)의 프리픽스 코드 열(284)에서 도시되며, 이 표는 프리픽스 코드 열(284)에서의 주어진 프리픽스 코드에 대해 패턴 인코딩된 열(288)에서의 인코딩된 패턴을 도시한다. 인코딩된 패턴에 대한 데이터 크기는, 빈발 패턴 인코딩 표(286)의 데이터 크기 열(290)에서 제공된다. 프리픽스 코드 000은, 새로운 압축된 데이터 포맷(282)에서 32-비트의 최대 크기의 데이터일 수 있는 압축되지 않은 패턴을 나타낸다. 프리픽스 코드 001은 새로운 압축된 데이터 포맷(282)의 데이터에서 0비트로서 제공될 수 있는 모두 0의 데이터 블록을 나타낸다. 프리픽스 코드 010은 특정 패턴인 패턴 0xFFFFFFFF를 나타내며, 이에 따라 새로운 압축된 데이터 포맷(282)에 따라 압축된 데이터에서 0-비트 데이터 크기를 요구한다. 다른 패턴들은 프리픽스 코드(011-111)에 대한 빈발 패턴 인코딩 표(286)에서 도시된다. 새로운 압축된 데이터 포맷(282)에서 플래그 필드는, 프리픽스 코드들(001-111)에 대한 패턴들 중 어느 것이 압축된 데이터의 데이터 부분들(즉, DataX)에 존재하는지를 표시한다. 패턴이 압축된 데이터에 존재하는 경우, 패턴들은, 추후에 압축되지 않은 데이터를 재생하기 위해 참고될 수 있는 새로운 압축된 데이터 포맷(282)에 저장된다. 데이터 필드는 새로운 압축된 데이터 포맷(282)의 데이터 필드와 연관되는 프리픽스 코드에 따라 압축된 데이터를 포함한다.

[0081] [0091] 도 15는 64-비트 빈발 패턴 압축 데이터 압축 매커니즘(292)의 다른 예를 예시한다. 이와 관련하여, 압축될 소스 데이터 포맷(294)의 소스 데이터는 예로서 128바이트로 도시된다. 새로운 압축된 데이터 포맷(296)이 프리픽스 코드들(Px), 데이터(Datax), 플래그들 및 패턴들의 상이한 포맷으로 제공되며, 이는 효율성 목적으로 함께 그룹핑되도록 구성된다. 프리픽스 코드는 4비트이다. 프리픽스 코드들은, 빈발 패턴 인코딩 표(302)의 프리픽스 코드 열들(298, 300)에서 도시되며, 이 표는 프리픽스 코드 열들(298, 300)에서 주어진 프리픽스 코드에 대해 패턴 인코딩된 열들(304, 306)에서 인코딩된 패턴을 도시한다. 인코딩된 패턴에 대한 데이터 크기는, 빈발 패턴 인코딩 표(302)의 데이터 크기 열들(308, 310)에서 제공된다. 프리픽스 코드 0000은 새로운 압축된 데이터 포맷(296)의 데이터에서 0비트로서 제공될 수 있는 모두 0의 데이터 블록을 나타낸다. 다른 패턴들은 ASCII 패턴들을 빈번하게 발생시키기 위해 ASCII 패턴들을 포함하는 프리픽스 코드들 0001-1111에 대한 빈발 패턴 인코딩 표(302)에서 도시된다. 새로운 압축된 데이터 포맷(296)에서 플래그 필드는, 프리픽스 코드들(0001-1111)에 대한 패턴들 중 어느 것이 압축된 데이터의 데이터 부분들(즉, DataX)에 존재하는지를 표시한다. 패턴이 압축된 데이터에 존재하는 경우, 패턴들은, 추후에 압축되지 않은 데이터를 재생하기 위해 참고될 수 있는 새로운 압축된 데이터 포맷(296)에 저장된다. 데이터 필드는 새로운 압축된 데이터 포맷(296)의 데이터 필드와 연관되는 프리픽스 코드에 따라 압축된 데이터를 포함한다.

[0082] [0092] 도 16은 64-비트 빈발 패턴 압축 데이터 압축 매커니즘(312)의 다른 예를 예시한다. 이와 관련하여, 압축될 소스 데이터 포맷(314)의 소스 데이터는 예로서 128바이트로 도시된다. 새로운 압축된 데이터 포맷(316)

이 프리픽스 코드들(Px), 데이터(Datax), 플래그들 및 패턴들의 상이한 포맷으로 제공되며, 이는 효율성 목적으로 함께 그룹핑되도록 구성된다. 프리픽스 코드는 4비트이다. 프리픽스 코드들은, 빈발 패턴 인코딩 표(322)의 프리픽스 코드 열들(318, 320)에서 도시되며, 이 표는 프리픽스 코드 열들(318, 320)에서 주어진 프리픽스 코드에 대해 패턴 인코딩된 열들(324, 326)에서 인코딩된 패턴을 도시한다. 인코딩된 패턴에 대한 데이터 크기는, 빈발 패턴 인코딩 표(322)의 데이터 크기 열들(328, 330)에서 제공된다. 프리픽스 코드 0000은 새로운 압축된 데이터 포맷(316)의 데이터에서 0비트로서 제공될 수 있는 모두 0의 데이터 블록을 나타낸다. 다른 패턴들은 고정된 패턴들의 결합들을 포함할 수 있는 프리픽스 코드들 0001-1111에 대한 빈발 패턴 인코딩 표(322)에서 도시된다. 새로운 압축된 데이터 포맷(316)에서 플래그 필드는, 프리픽스 코드들(0001-1111)에 대한 패턴들 중 어느 것이 압축된 데이터의 데이터 부분들(즉, DataX)에 존재하는지를 표시한다. 패턴이 압축된 데이터에 존재하는 경우, 패턴들은, 추후에 압축되지 않은 데이터를 재생하기 위해 데이터 압축 동안 참고될 수 있는 새로운 압축된 데이터 포맷(316)에 저장된다. 프리픽스 코드(P0-P31)는 압축되지 않는 포맷의 최대 길이 데이터를 재생하기 위해 대응하는 데이터(Datax)와 함께 이용되는 패턴들에 링크될 수 있다. 데이터 필드는 새로운 압축된 데이터 포맷(316)의 데이터 필드와 연관되는 프리픽스 코드에 따라 압축된 데이터를 포함한다.

[0083] [0093] 도 16의 빈발 패턴 압축 데이터 압축 매커니즘(312)에서 이용될 수 있는 고정된 패턴들의 예들은, 고정된 패턴들이 패턴 열(334)에 있어서, 그 길이가 길이 열(336)에서 제공되고, 패턴의 정의가 패턴 정의 열(338)에서 제공되는 도 17의 표(332)에서 도시된다. 플래그 정의들은, CMC(36)가 압축되지 않은 데이터를 재생하는데 이용되는 정의에 대해 프리픽스 코드에 링크된 주어진 패턴을 상관시키도록 허용하기 위해 플래그 정의 표(340)에서 도시된다. 플래그 정의 표(340)는, 플래그 열(342)에서 주어진 플래그에 대한 비트들, 플래그 값 열(344)에서 주어진 플래그에 대한 비트들의 값 및 플래그 정의 열(346)에서 주어진 플래그에 대한 플래그 정의를 포함한다.

[0084] [0094] 도 18은 64-비트 빈발 패턴 압축 데이터 압축 매커니즘(348)의 다른 예를 예시한다. 이와 관련하여, 압축될 소스 데이터 포맷(350)의 소스 데이터는 예로서 128바이트로 도시된다. 새로운 압축된 데이터 포맷(352)이 프리픽스 코드들(Px), 데이터(Datax), 플래그들 및 패턴들의 상이한 포맷으로 제공되며, 이는 효율성 목적으로 함께 그룹핑되도록 구성된다. 프리픽스 코드는 4비트이다. 프리픽스 코드들은, 빈발 패턴 인코딩 표(358)의 프리픽스 코드 열들(354, 356)에서 도시되며, 이 표는 프리픽스 코드 열들(354, 356)에서 주어진 프리픽스 코드에 대해 패턴 인코딩된 열들(360, 362)에서 인코딩된 패턴을 도시한다. 인코딩된 패턴에 대한 데이터 크기는, 빈발 패턴 인코딩 표(358)의 데이터 크기 열들(364, 366)에서 제공된다. 프리픽스 코드 0000은 새로운 압축된 데이터 포맷(352)의 데이터에서 0비트로서 제공될 수 있는 모두 0의 데이터 블록을 나타낸다. 프리픽스 코드 1111은 새로운 압축된 데이터 포맷(352)에서 압축되지 않은 데이터 블록을 나타낸다. 다른 패턴들은 거기서 도시된 정의된 패턴들의 결합들을 포함할 수 있는, 프리픽스 코드들 0001-1110에 대한 빈발 패턴 인코딩 표(358)에서 도시된다. 새로운 압축된 데이터 포맷(352)에서 플래그 필드는, 프리픽스 코드들(0000-1110)에 대한 패턴들 중 어느 것이 압축된 데이터의 데이터 부분들(즉, DataX)에 존재하는지를 표시한다. 패턴이 압축된 데이터에 존재하는 경우, 패턴들은, 추후에 압축되지 않은 데이터를 재생하기 위해 참고될 수 있는 새로운 압축된 데이터 포맷(352)에 저장된다. 새로운 압축된 데이터 포맷(352)은 패턴들 0-5만을 포함하는 것으로서 도시되는데, 그 이유는 이들이 이 예에서, 소스 데이터에 존재하는 프리픽스 코드들 0000-1110에 대해 참작되는 유일한 패턴들이기 때문이다. 데이터 필드는 새로운 압축된 데이터 포맷(352)의 데이터 필드와 연관되는 프리픽스 코드에 따라 압축된 데이터를 포함한다.

[0085] [0095] 본원에서 개시되는 양상들에 따라 CPU-기반 시스템에서 CMC들을 이용하여 메모리 대역폭 압축을 제공하는 것은 임의의 프로세서-기반 디바이스에서 제공되거나 이에 통합될 수 있다. 예들은, 제한 없이, 셋톱 박스, 엔터테인먼트 유닛, 내비게이션 디바이스, 통신 디바이스, 고정 위치 데이터 유닛, 모바일 위치 데이터 유닛, 모바일 전화, 셀룰러 전화, 컴퓨터, 휴대용 컴퓨터 데스크톱 컴퓨터, 개인용 디지털 보조기기(PDA), 모니터, 컴퓨터 모니터, 텔레비전, 튜너, 라디오, 위성 라디오, 음악 플레이어, 디지털 음악 플레이어, 휴대용 음악 플레이어, 디지털 비디오 플레이어, 비디오 플레이어, 디지털 비디오 디스크(DVD) 플레이어, 휴대용 디지털 비디오 플레이어를 포함한다.

[0086] [0096] 이와 관련하여, 도 19는 도 2의 CMC(36)와 더불어, 도 2의 SoC(10')를 이용할 수 있는 프로세서-기반 시스템(368)의 예를 예시한다. 이 예에서, 프로세서-기반 시스템(368)은 하나 또는 그 초과의 CPU들(370)을 포함하며, 각각은 하나 또는 그 초과의 프로세서들(372)을 포함한다. CPU(들)(370)는 일시적으로 저장된 데이터에 대한 빠른 액세스를 위해 프로세서(들)(372)에 커플링되는 캐시 메모리(374)를 가질 수 있다. CPU(들)(370)는 시스템 버스(376)에 커플링되고 프로세서-기반 시스템(368)에 포함되는 디바이스들을 상호커플링할 수 있다.

잘 알려진 바와 같이, CPU(들)(370)는 시스템 버스(376) 상에서 어드레스, 제어 및 데이터 정보를 교환함으로써 이들 다른 디바이스들과 통신한다. 예를 들어, CPU(들)(370)는 슬래이브 디바이스의 예로서 메모리 제어기들(378)에 버스 트랜잭션 요청들을 통신한다. 도 19에서 예시되지 않지만, 다수의 시스템 버스들(376)이 제공될 수 있다.

[0087] [0097] 다른 디바이스들이 시스템 버스(376)에 연결될 수 있다. 도 19에서 예시된 바와 같이, 이들 디바이스들은 예들로서, 메모리 시스템(380), 하나 또는 그 초과의 입력 디바이스들(382), 하나 또는 그 초과의 출력 디바이스들(384), 하나 또는 그 초과의 네트워크 인터페이스 디바이스들(386), 및 하나 또는 그 초과의 디스플레이 제어기들(388)을 포함할 수 있다. 입력 디바이스(들)(382)는 입력 키들, 스위치들, 음성 프로세서들 등을 포함(그러나 이것으로 제한되지 않음)하는 임의의 타입의 입력 디바이스들을 포함할 수 있다. 출력 디바이스(들)(384)는 오디오, 비디오, 다른 시각적 표시기들 등을 포함(그러나 이것으로 제한되지 않음)하는 임의의 타입의 출력 디바이스를 포함할 수 있다. 네트워크 인터페이스 디바이스(들)(386)는 네트워크(390)로의 그리고 이로부터의 데이터의 교환을 허용하도록 구성되는 임의의 디바이스일 수 있다. 네트워크(390)는 유선 또는 무선 네트워크, 사설 또는 공중 네트워크, LAN(local area network), 와이드 로컬 영역 네트워크, 무선 로컬 영역 네트워크, 블루투스(BT) 및 인터넷을 포함(그러나 이것으로 제한되지 않음)하는 임의의 타입의 네트워크일 수 있다. 네트워크 인터페이스 디바이스(들)(386)는 원하는 임의의 타입의 통신 프로토콜을 지원하도록 구성될 수 있다. 메모리 시스템(380)은 하나 또는 그 초과의 메모리 유닛들(392(0)-392(N))을 포함할 수 있다.

[0088] [0098] CPU(들)(370)는 또한 하나 또는 그 초과의 디스플레이들(394)에 전송되는 정보를 제어하기 위해 시스템 버스(376) 상에서 디스플레이 제어기(들)(388)에 액세스하도록 구성될 수 있다. 디스플레이 제어기(들)(388)는 디스플레이(들)(394)에 대해 적합한 포맷으로 디스플레이될 정보를 프로세싱하는 하나 또는 그 초과의 비디오 프로세서들(396)을 통해 디스플레이될 정보를 디스플레이(들)(394)에 전송한다. 디스플레이(들)(394)는 음극선관(CRT), 액정 디스플레이(LCD), 발광 다이오드(LED) 디스플레이, 플라스마 디스플레이 등을 포함(그러나 이것으로 제한되지 않음)하는 임의의 타입의 디스플레이를 포함할 수 있다.

[0089] [0099] 당업자들은 추가로, 본원에서 개시된 양상과 관련하여 설명되는 다양한 예시적인 로직 블록들, 모듈들, 회로들 및 알고리즘들이 전자 하드웨어, 메모리에 또는 다른 컴퓨터-판독 가능 매체에 저장되고 프로세서 또는 다른 프로세싱 디바이스에 의해 실행되는 명령들, 또는 이 둘의 조합으로서 구현될 수 있다는 것을 인지할 것이다. 본원에서 설명되는 디바이스들은 예들로서 임의의 회로, 하드웨어 컴포넌트, 집적 회로(IC), 또는 IC 칩에서 이용될 수 있다. 본원에서 개시된 메모리는 임의의 타입 및 크기의 메모리일 수 있고 원하는 임의의 타입의 정보를 저장하도록 구성될 수 있다. 이러한 상호 교환성을 명확하게 예시하기 위해, 다양한 예시적인 컴포넌트, 블록, 모듈, 회로 및 단계는 그의 기능성의 견지에서 대체로 위에서 설명되었다. 이러한 기능성이 어떻게 구현되는지는 특정 애플리케이션, 설계 선택들 및/또는 전체 시스템에 부과되는 설계 제약들에 의존한다. 당업자는 각각의 특정 애플리케이션마다 다양한 방식들로 설명된 기능을 구현할 수 있지만, 이러한 구현 결정은 본 개시의 범위를 벗어나게 하는 것으로 해석되어서는 안 된다.

[0090] [00100] 본원에서 개시된 양상들과 관련하여 설명된 다양한 예시적인 로직 블록, 모듈, 및 회로들은, 프로세서, 디지털 신호 프로세서(DSP), 주문형 집적 회로(ASIC), 필드 프로그래밍 가능 게이트 어레이(FPGA), 또는 기타 프로그래밍 가능 로직 디바이스, 이산 게이트 또는 트랜지스터 로직, 이산 하드웨어 컴포넌트, 또는 본원에서 설명된 기능을 수행하도록 설계된 이들의 임의의 조합으로 구현 또는 수행될 수 있다. 프로세서는 마이크로프로세서일 수 있지만, 대안적으로, 프로세서는 임의의 종래의 프로세서, 제어기, 마이크로 제어기, 또는 상태 머신일 수 있다. 프로세서는 또한 컴퓨팅 디바이스들의 조합, 예를 들어, DSP와 마이크로프로세서의 조합, 복수의 마이크로프로세서들, DSP 코어와 연결된 하나 또는 그 초과의 마이크로프로세서들 또는 임의의 다른 이러한 구성으로서 구현될 수 있다.

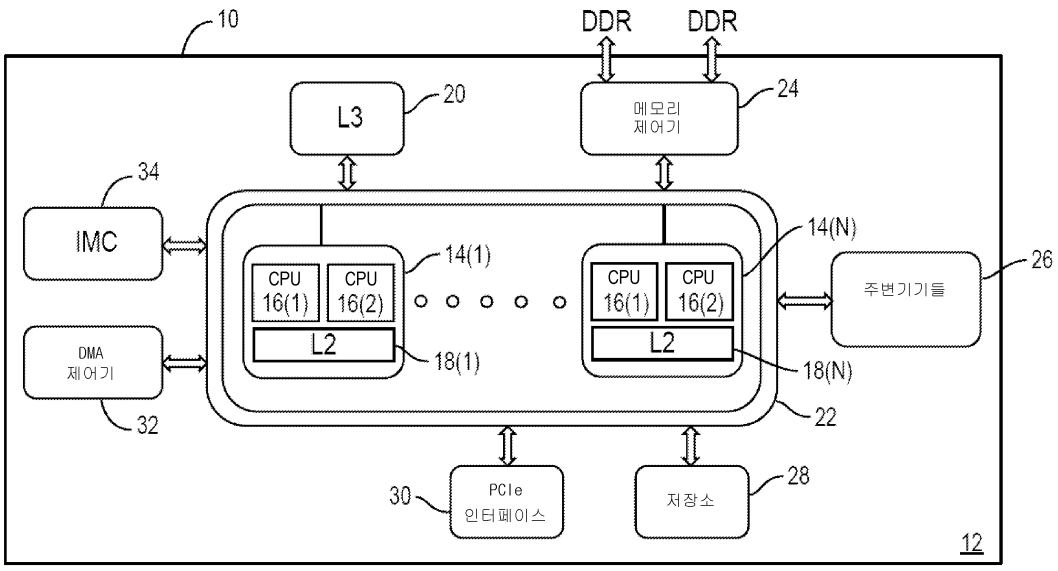
[0091] [00101] 본원에서 개시되는 양상들은 하드웨어 및 하드웨어에 저장된 명령들로 구현될 수 있으며, 예를 들어, RAM(Random Access Memory), 플래시 메모리, ROM(Read Only Memory), EPROM(Electrically Programmable ROM), EEPROM(Electrically Erasable Programmable ROM), 레지스터들, 하드 디스크, 제거 가능 디스크, CD-ROM, 또는 당 업계에 알려진 임의의 다른 형태의 컴퓨터 판독 가능 매체에 상주할 수 있다. 예시적인 저장 매체는 프로세서에 커플링되어, 프로세서는 저장 매체로부터 정보를 판독하고, 저장 매체에 정보를 기록할 수 있다. 대안적으로, 저장 매체는 프로세서에 통합될 수 있다. 프로세서 및 저장 매체는 ASIC에 상주할 수 있다. ASIC는 원격 스테이션에 상주할 수 있다. 대안적으로, 프로세서 및 저장 매체는 원격 스테이션, 기지국 또는 서버에 이산 컴포넌트로서 상주할 수 있다.

[0092] [00102] 또한, 본원의 예시적인 양상에서 설명된 동작 단계들은 예들 및 논의를 제공하기 위해 설명된다는 것이 주의된다. 설명된 동작들은 예시된 시퀀스 이외의 다른 수많은 다른 시퀀스들에서 수행될 수 있다. 또한, 단일 동작 단계에서 설명된 동작들은 실제로 다수의 다른 단계들에서 수행될 수 있다. 부가적으로, 예시적인 양상들에서 논의된 하나 또는 그 초과의 동작 단계들은 결합될 수 있다. 흐름도 다이어그램에 예시되는 동작 단계들은 당업자에게 쉽게 자명하게 될 바와 같이 수많은 다른 변형들이 가해질 수 있다는 것이 이해될 것이다. 당업자는, 정보 및 신호들이 다양한 상이한 기술들 및 기법들을 이용하여 표현될 수 있다는 것을 또한 이해할 것이다. 예를 들어, 위의 설명 전반에 걸쳐 참조될 수 있는 데이터, 명령들, 커맨드들, 정보, 신호들, 비트들, 심볼들 및 칩들은 전압들, 전류들, 전자기파들, 자기 필드들 또는 자기 입자들, 광 필드들 또는 광 입자들, 또는 이들의 임의의 조합으로 표현될 수 있다.

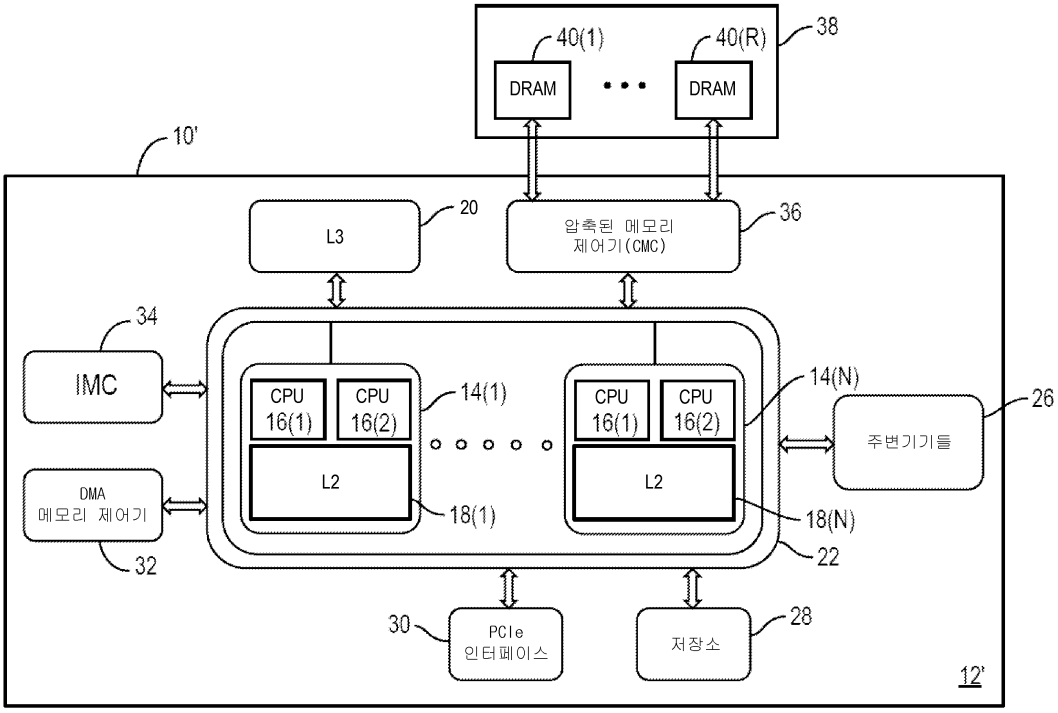
[0093] 본 개시의 이전 설명은 임의의 당업자가 본 개시를 실시하거나 또는 이용할 수 있도록 제공된다. 본 개시에 대한 다양한 변형은 당업자에게 쉽게 명백할 것이며, 여기에 정의된 일반적인 원리들은 본 개시의 사상 또는 범위를 벗어나지 않고 다른 변동들에 적용될 수도 있다. 따라서, 본 개시는 본원에서 설명된 예들 및 설계들로 제한되도록 의도된 것이 아니라, 본원에서 개시된 원리들 및 신규한 특징들에 부합하는 최광의의 범위로 허여될 것이다.

도면

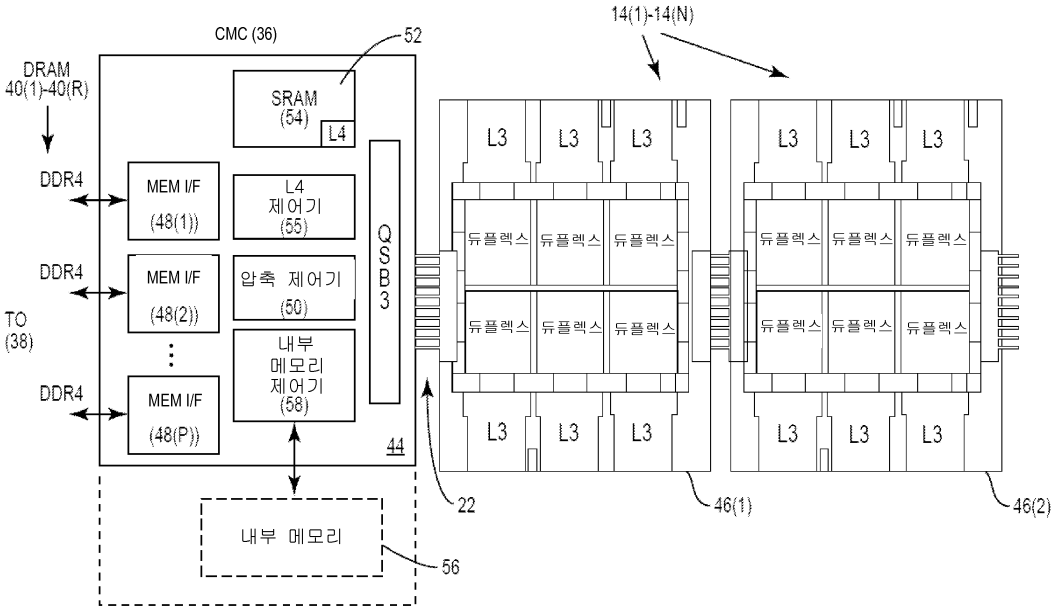
도면1



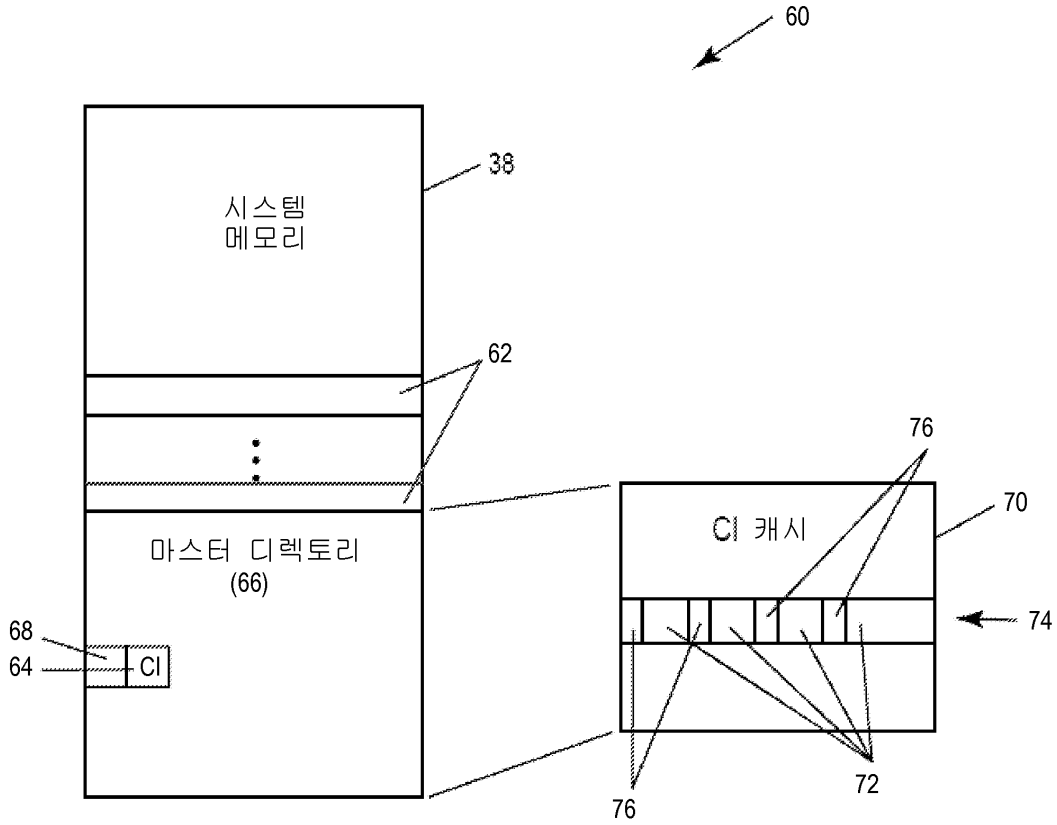
도면2


40(1)
40(R)
38
DRAM
DRAM
10'
L3
20
압축된 메모리
제어기(CMC)
36
34
IMC
CPU
16(1)
CPU
16(2)
14(1)
CPU
16(1)
CPU
16(2)
14(N)
26
주변기기들
L2
L2
DMA
메모리 제어기
18(1)
18(N)
22
32
30
PCIe
인터페이스
저장소
28
12'

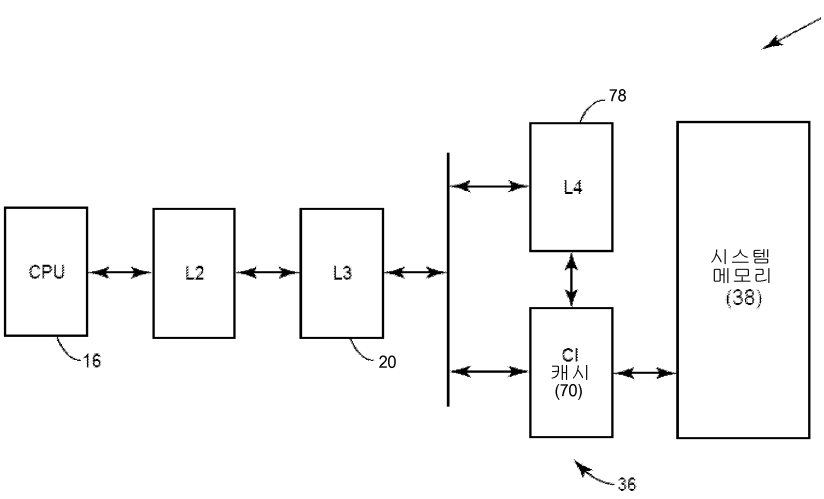
도면3


CMC (36)
52
14(1)-14(N)
DRAM
40(1)-40(R)
SRAM
(54)
L4
L3
L3
L3
L3
L3
L3
DDR4
MEM I/F
(48(1))
L4
제어기
(55)
듀플렉스
듀플렉스
듀플렉스
듀플렉스
듀플렉스
듀플렉스
DDR4
MEM I/F
(48(2))
압축 제어기
(50)
QSB3
TO
(38)
내부
메모리
제어기
(58)
DDR4
MEM I/F
(48(P))
듀플렉스
듀플렉스
듀플렉스
듀플렉스
듀플렉스
듀플렉스
44
L3
L3
L3
L3
L3
L3
22
46(1)
46(2)
내부 메모리
56

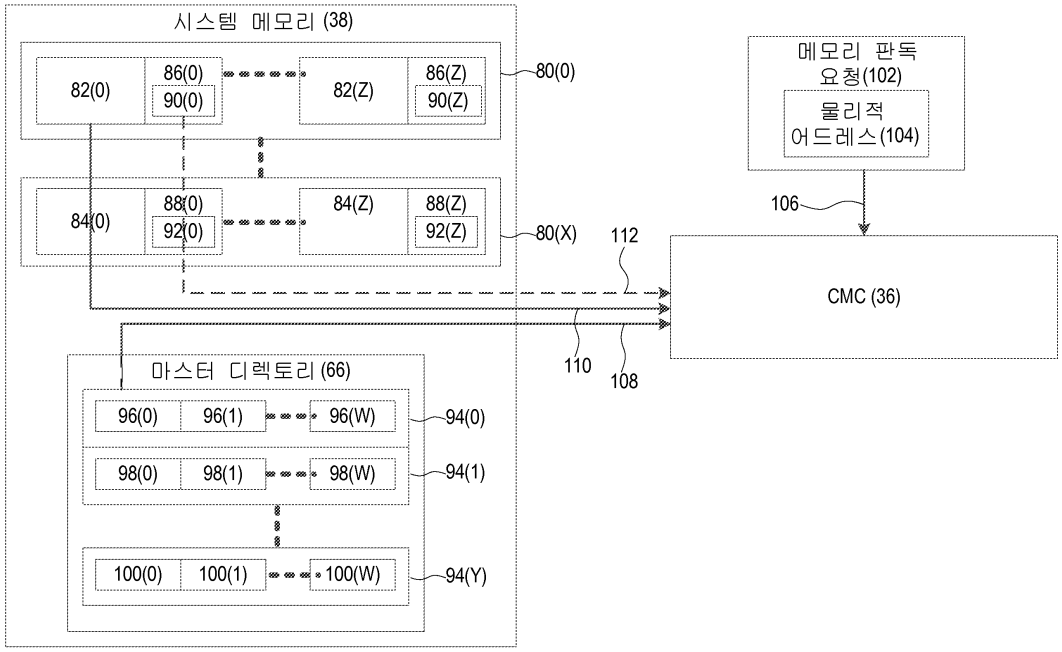
도면4


60
38
시스템
메모리
62
76
70
CI 캐시
74
마스터 디렉토리
(66)
68
64
CI
76
72

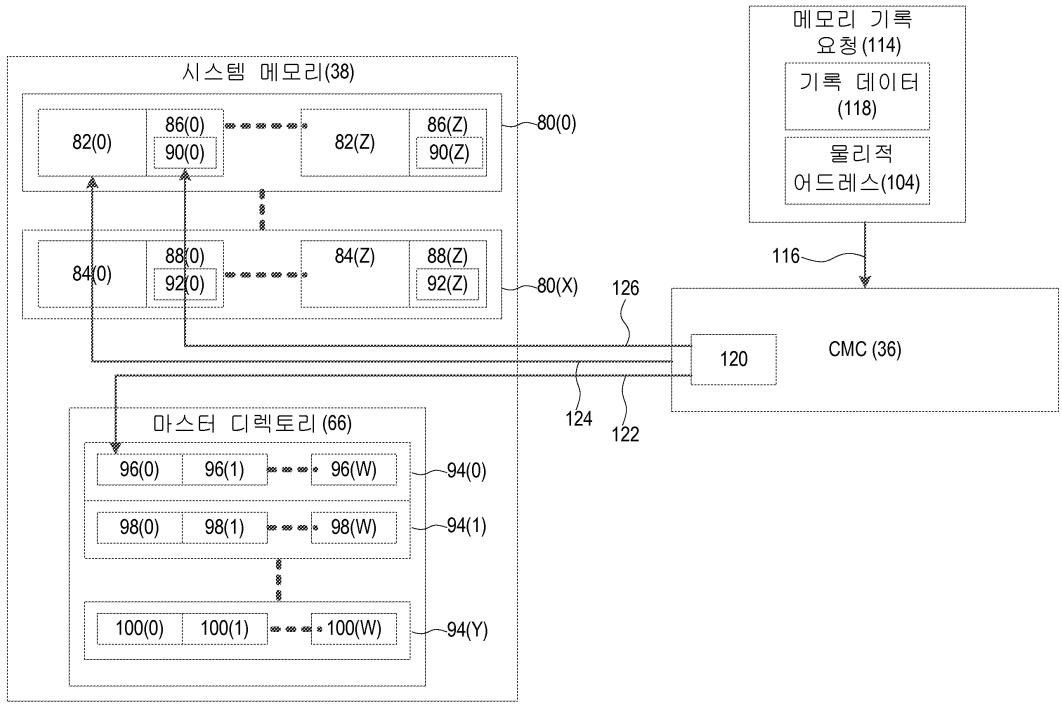
도면5


10"
78
L4
CPU
L2
L3
시스템
메모리
(38)
CI
캐시
(70)
16
20
36

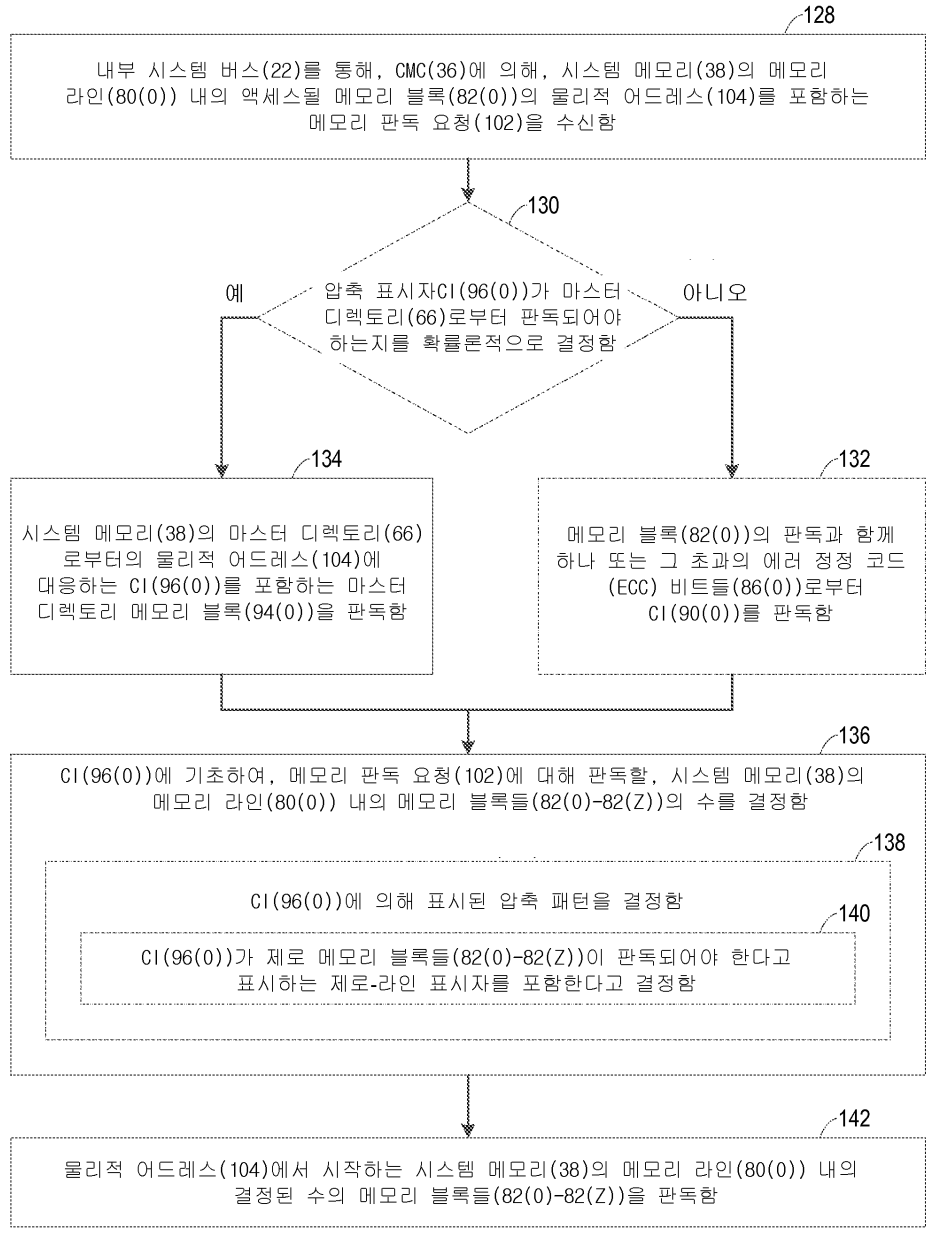
도면6a


시스템 메모리 (38)
82(0)
86(0)
90(0)
82(Z)
86(Z)
90(Z)
80(0)
84(0)
88(0)
92(0)
84(Z)
88(Z)
92(Z)
80(X)
112
메모리 판독 요청(102)
물리적 어드레스(104)
106
CMC (36)
110
108
마스터 디렉토리 (66)
96(0)
96(1)
96(W)
94(0)
98(0)
98(1)
98(W)
94(1)
100(0)
100(1)
100(W)
94(Y)

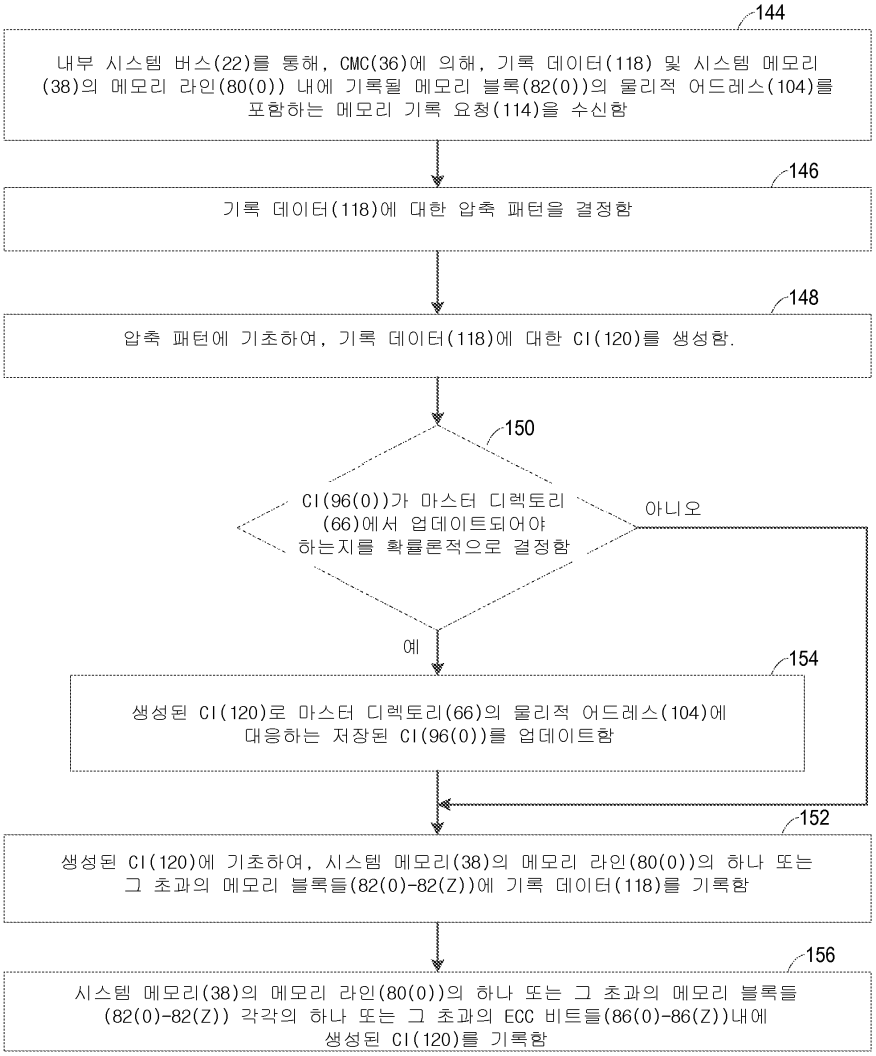
도면6b


시스템 메모리(38)
82(0)
86(0)
90(0)
82(Z)
86(Z)
90(Z)
80(0)
84(0)
88(0)
92(0)
84(Z)
88(Z)
92(Z)
80(X)
126
메모리 기록 요청(114)
기록 데이터 (118)
물리적 어드레스(104)
116
120
CMC (36)
124
122
마스터 디렉토리 (66)
96(0)
96(1)
96(W)
94(0)
98(0)
98(1)
98(W)
94(1)
100(0)
100(1)
100(W)
94(Y)

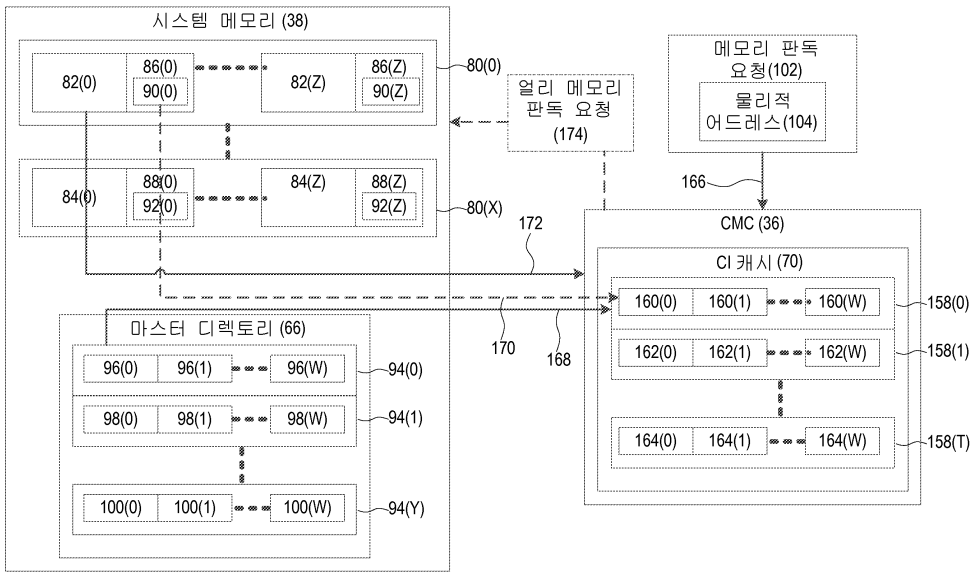
도면7


128
내부 시스템 버스(22)를 통해, CMC(36)에 의해, 시스템 메모리(38)의 메모리 라인(80(0)) 내의 액세스될 메모리 블록(82(0))의 물리적 어드레스(104)를 포함하는 메모리 판독 요청(102)을 수신함
130
예
압축 표시자CI(96(0))가 마스터 디렉토리(66)로부터 판독되어야 하는지를 확률론적으로 결정함
아니오
134
시스템 메모리(38)의 마스터 디렉토리(66)로부터의 물리적 어드레스(104)에 대응하는 CI(96(0))를 포함하는 마스터 디렉토리 메모리 블록(94(0))을 판독함
132
메모리 블록(82(0))의 판독과 함께 하나 또는 그 초과의 에러 정정 코드(ECC) 비트들(86(0))로부터 CI(90(0))를 판독함
136
CI(96(0))에 기초하여, 메모리 판독 요청(102)에 대해 판독할, 시스템 메모리(38)의 메모리 라인(80(0)) 내의 메모리 블록들(82(0)-82(Z))의 수를 결정함
138
CI(96(0))에 의해 표시된 압축 패턴을 결정함
140
CI(96(0))가 제로 메모리 블록들(82(0)-82(Z))이 판독되어야 한다고 표시하는 제로-라인 표시자를 포함한다고 결정함
142
물리적 어드레스(104)에서 시작하는 시스템 메모리(38)의 메모리 라인(80(0)) 내의 결정된 수의 메모리 블록들(82(0)-82(Z))을 판독함

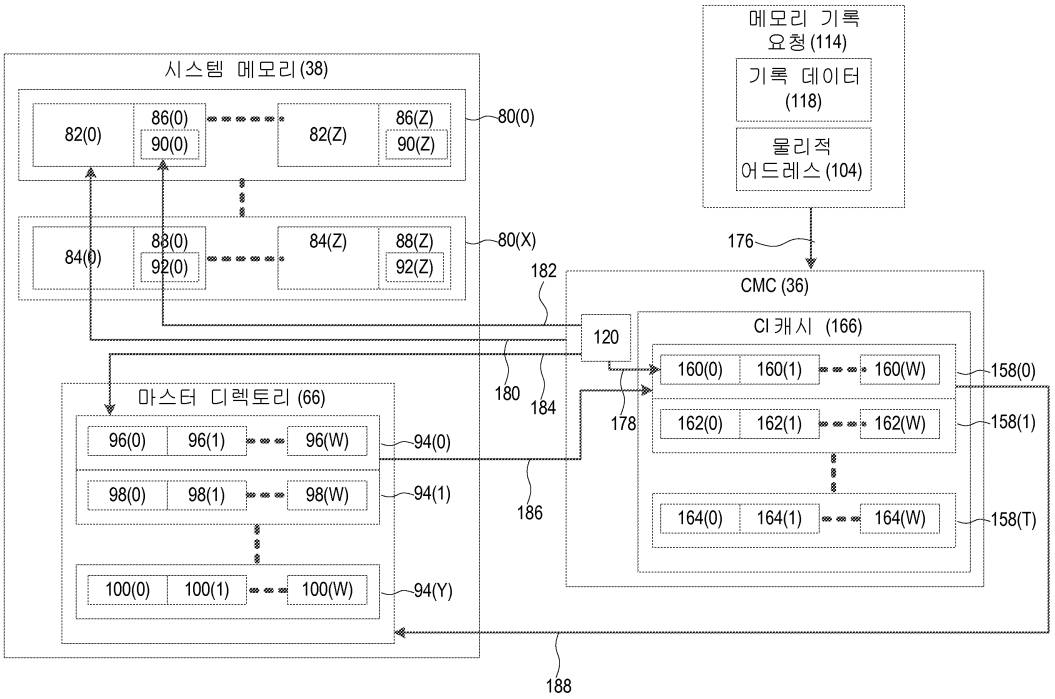
도면8


144
내부 시스템 버스(22)를 통해, CMC(36)에 의해, 기록 데이터(118) 및 시스템 메모리(38)의 메모리 라인(80(0)) 내에 기록될 메모리 블록(82(0))의 물리적 어드레스(104)를 포함하는 메모리 기록 요청(114)을 수신함
146
기록 데이터(118)에 대한 압축 패턴을 결정함
148
압축 패턴에 기초하여, 기록 데이터(118)에 대한 CI(120)를 생성함.
150
CI(96(0))가 마스터 디렉토리(66)에서 업데이트되어야 하는지를 확률론적으로 결정함
아니오
예
154
생성된 CI(120)로 마스터 디렉토리(66)의 물리적 어드레스(104)에 대응하는 저장된 CI(96(0))를 업데이트함
152
생성된 CI(120)에 기초하여, 시스템 메모리(38)의 메모리 라인(80(0))의 하나 또는 그 초과의 메모리 블록들(82(0)-82(Z))에 기록 데이터(118)를 기록함
156
시스템 메모리(38)의 메모리 라인(80(0))의 하나 또는 그 초과의 메모리 블록들(82(0)-82(Z)) 각각의 하나 또는 그 초과의 ECC 비트들(86(0)-86(Z))내에 생성된 CI(120)를 기록함

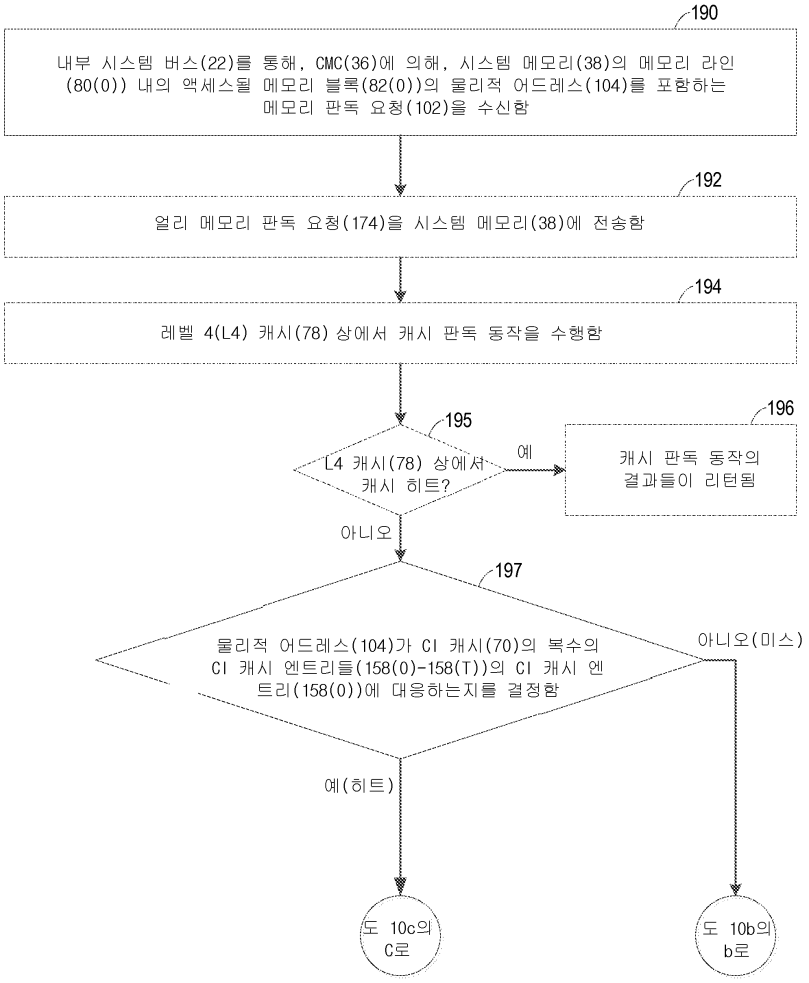
도면9a


시스템 메모리 (38)
82(0)
86(0)
90(0)
82(Z)
86(Z)
90(Z)
80(0)
84(0)
88(0)
92(0)
84(Z)
88(Z)
92(Z)
80(X)
얼리 메모리 판독 요청 (174)
메모리 판독 요청(102)
물리적 어드레스(104)
166
CMC (36)
CI 캐시 (70)
160(0)
160(1)
160(W)
158(0)
162(0)
162(1)
162(W)
158(1)
164(0)
164(1)
164(W)
158(T)
172
170
168
마스터 디렉토리 (66)
96(0)
96(1)
96(W)
94(0)
98(0)
98(1)
98(W)
94(1)
100(0)
100(1)
100(W)
94(Y)

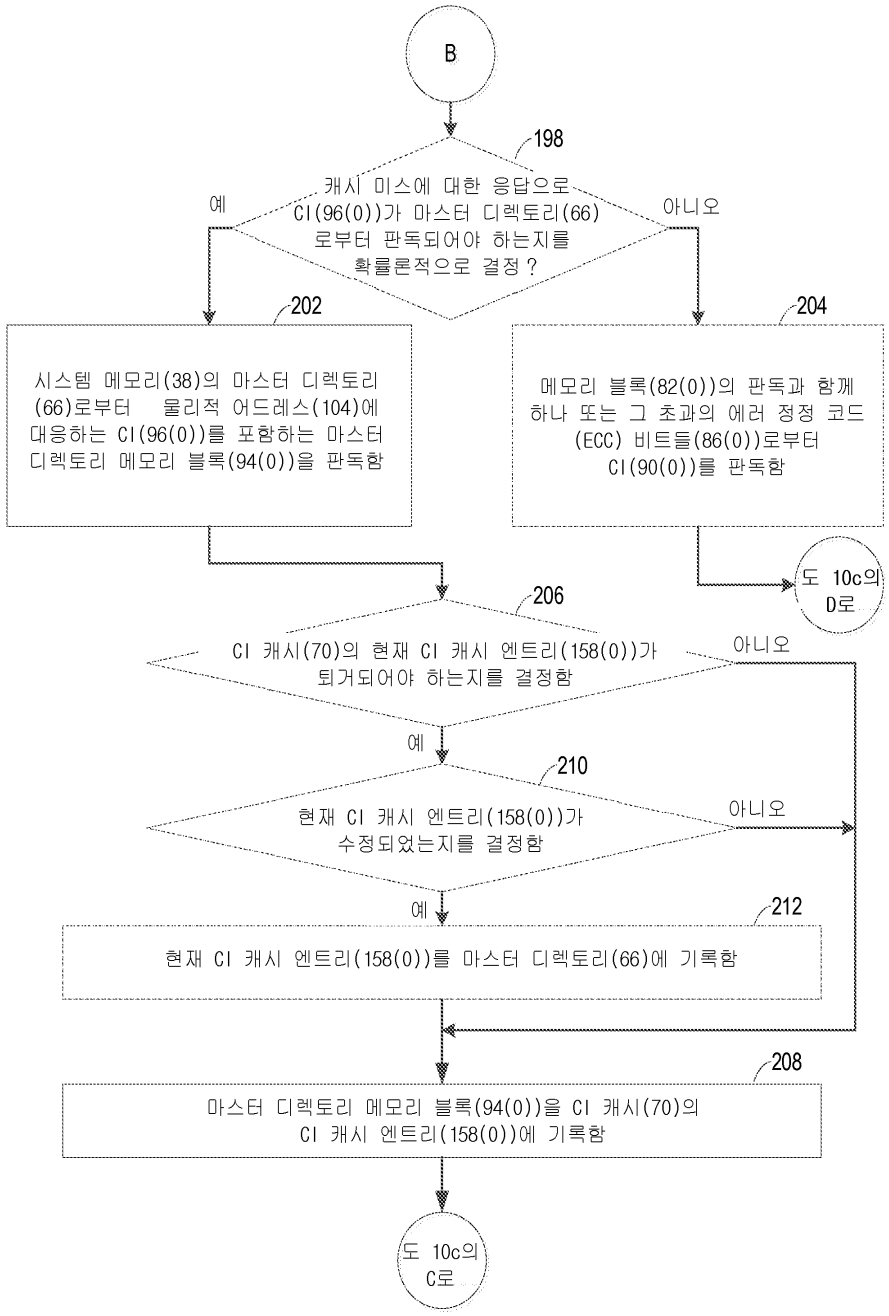
도면9b


메모리 기록 요청(114)
기록 데이터(118)
물리적 어드레스(104)
176
시스템 메모리(38)
82(0)
86(0)
90(0)
82(Z)
86(Z)
90(Z)
80(0)
84(0)
88(0)
92(0)
84(Z)
88(Z)
92(Z)
80(X)
182
CMC (36)
120
CI 캐시 (166)
160(0)
160(1)
160(W)
158(0)
162(0)
162(1)
162(W)
158(1)
164(0)
164(1)
164(W)
158(T)
178
180
184
186
마스터 디렉토리(66)
96(0)
96(1)
96(W)
94(0)
98(0)
98(1)
98(W)
94(1)
100(0)
100(1)
100(W)
94(Y)
188

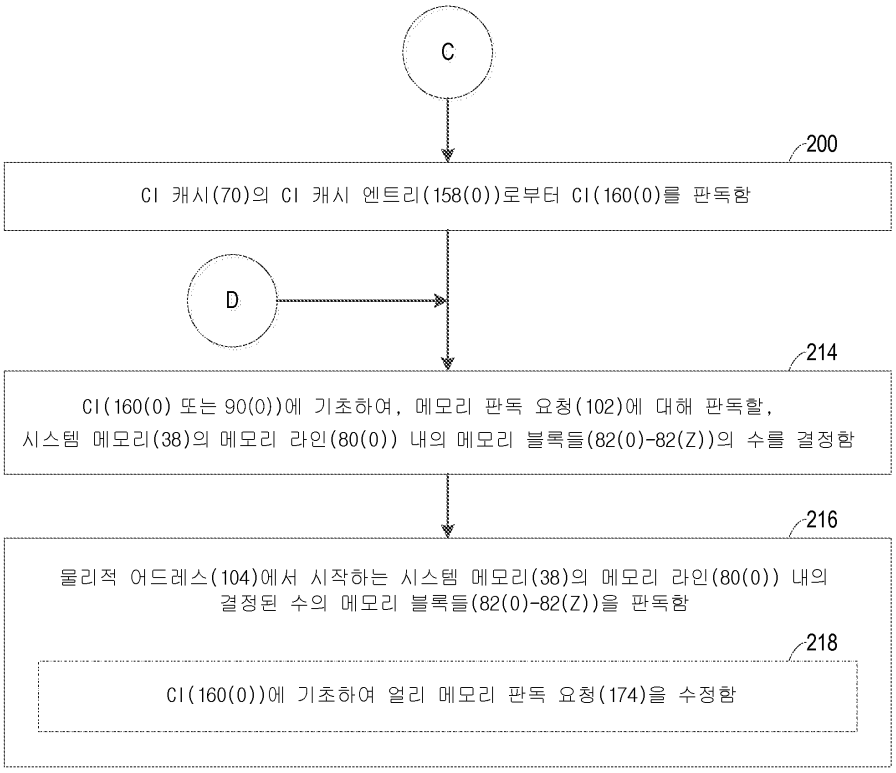
도면10a


190
내부 시스템 버스(22)를 통해, CMC(36)에 의해, 시스템 메모리(38)의 메모리 라인(80(0)) 내의 액세스될 메모리 블록(82(0))의 물리적 어드레스(104)를 포함하는 메모리 판독 요청(102)을 수신함
192
얼리 메모리 판독 요청(174)을 시스템 메모리(38)에 전송함
194
레벨 4(L4) 캐시(78) 상에서 캐시 판독 동작을 수행함
195
L4 캐시(78) 상에서 캐시 히트?
예
196
캐시 판독 동작의 결과들이 리턴됨
아니오
197
물리적 어드레스(104)가 CI 캐시(70)의 복수의 CI 캐시 엔트리들(158(0)-158(T))의 CI 캐시 엔트리(158(0))에 대응하는지를 결정함
아니오(미스)
예(히트)
도 10c의 C로
도 10b의 b로

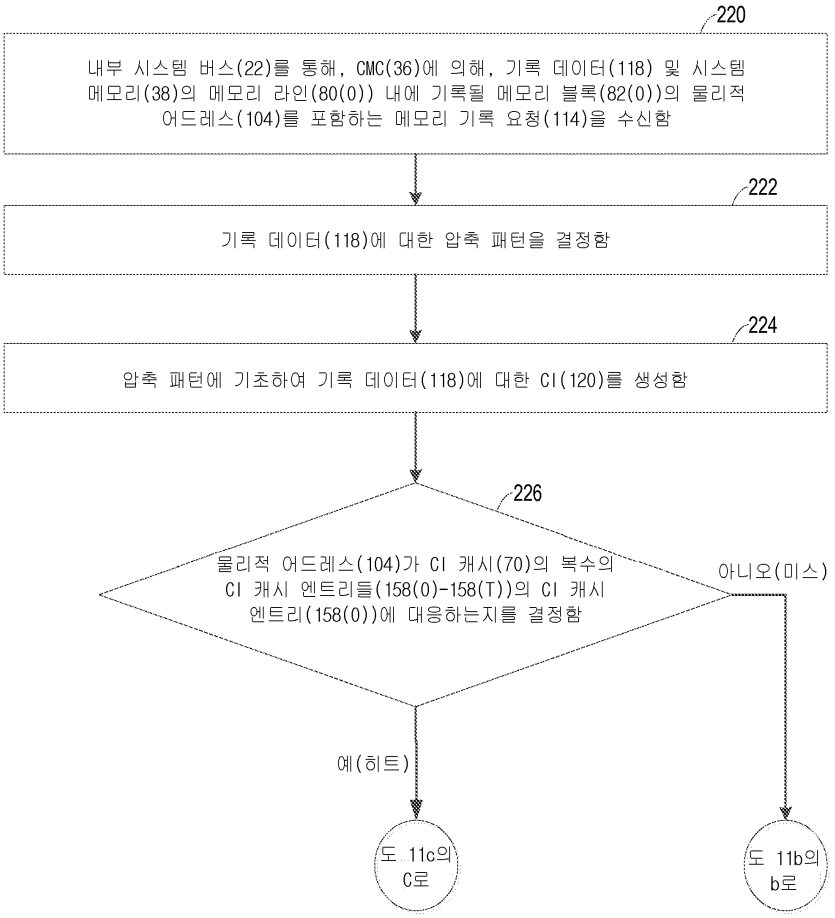
도면10b


B
198
캐시 미스에 대한 응답으로 CI(96(0))가 마스터 디렉토리(66)로부터 판독되어야 하는지를 확률론적으로 결정?
예
아니오
202
시스템 메모리(38)의 마스터 디렉토리(66)로부터 물리적 어드레스(104)에 대응하는 CI(96(0))를 포함하는 마스터 디렉토리 메모리 블록(94(0))을 판독함
204
메모리 블록(82(0))의 판독과 함께 하나 또는 그 초과의 에러 정정 코드(ECC) 비트들(86(0))로부터 CI(90(0))를 판독함
도 10c의 D로
206
CI 캐시(70)의 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 하는지를 결정함
아니오
예
210
현재 CI 캐시 엔트리(158(0))가 수정되었는지를 결정함
아니오
예
212
현재 CI 캐시 엔트리(158(0))를 마스터 디렉토리(66)에 기록함
208
마스터 디렉토리 메모리 블록(94(0))을 CI 캐시(70)의 CI 캐시 엔트리(158(0))에 기록함
도 10c의 C로

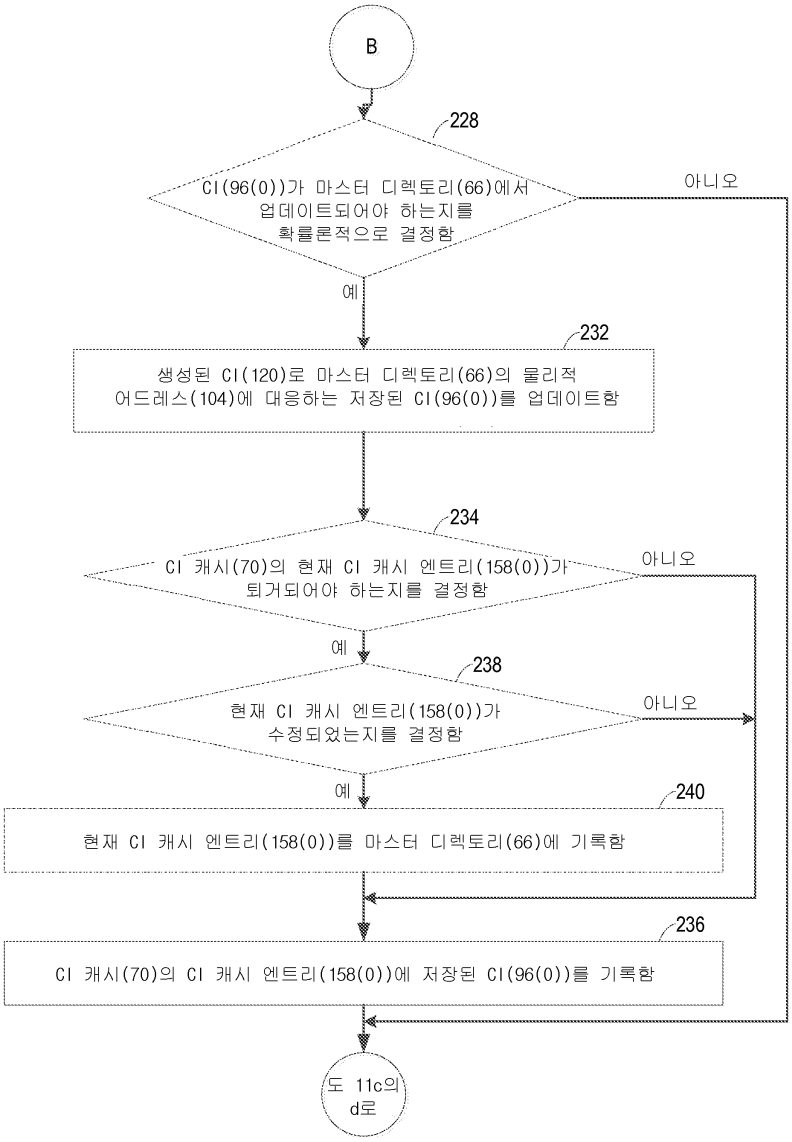
도면10c


C
200
CI 캐시(70)의 CI 캐시 엔트리(158(0))로부터 CI(160(0)를 판독함
D
214
CI(160(0) 또는 90(0))에 기초하여, 메모리 판독 요청(102)에 대해 판독할,
시스템 메모리(38)의 메모리 라인(80(0)) 내의 메모리 블록들(82(0)-82(Z))의 수를 결정함
216
물리적 어드레스(104)에서 시작하는 시스템 메모리(38)의 메모리 라인(80(0)) 내의
결정된 수의 메모리 블록들(82(0)-82(Z))을 판독함
218
CI(160(0))에 기초하여 얼리 메모리 판독 요청(174)을 수정함

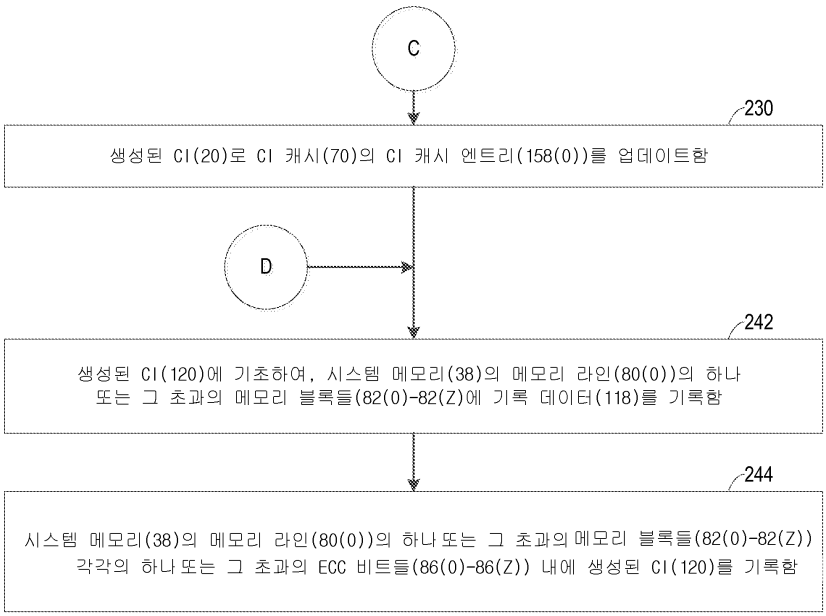
도면11a


220
내부 시스템 버스(22)를 통해, CMC(36)에 의해, 기록 데이터(118) 및 시스템
메모리(38)의 메모리 라인(80(0)) 내에 기록될 메모리 블록(82(0))의 물리적
어드레스(104)를 포함하는 메모리 기록 요청(114)을 수신함
222
기록 데이터(118)에 대한 압축 패턴을 결정함
224
압축 패턴에 기초하여 기록 데이터(118)에 대한 CI(120)를 생성함
226
물리적 어드레스(104)가 CI 캐시(70)의 복수의
CI 캐시 엔트리들(158(0)-158(T))의 CI 캐시
엔트리(158(0))에 대응하는지를 결정함
아니오(미스)
예(히트)
도 11c의
C로
도 11b의
b로

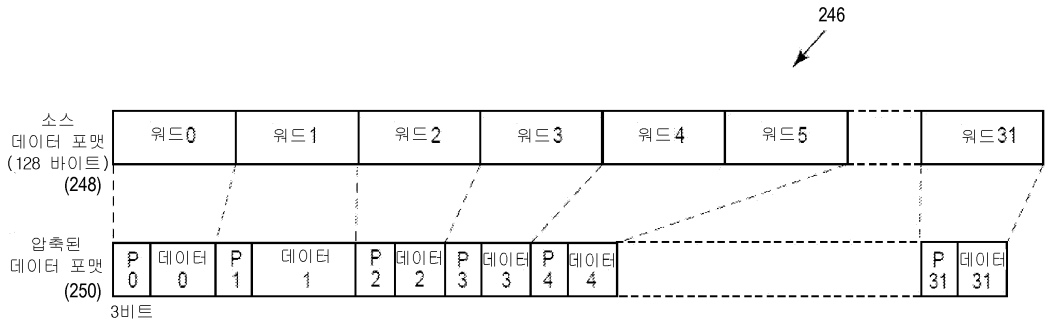
도면11b


B
228
CI(96(0))가 마스터 디렉토리(66)에서 업데이트되어야 하는지를 확률론적으로 결정함
아니오
예
232
생성된 CI(120)로 마스터 디렉토리(66)의 물리적 어드레스(104)에 대응하는 저장된 CI(96(0))를 업데이트함
234
CI 캐시(70)의 현재 CI 캐시 엔트리(158(0))가 퇴거되어야 하는지를 결정함
아니오
예
238
현재 CI 캐시 엔트리(158(0))가 수정되었는지를 결정함
아니오
예
240
현재 CI 캐시 엔트리(158(0))를 마스터 디렉토리(66)에 기록함
236
CI 캐시(70)의 CI 캐시 엔트리(158(0))에 저장된 CI(96(0))를 기록함
도 11c의 d로

도면11c



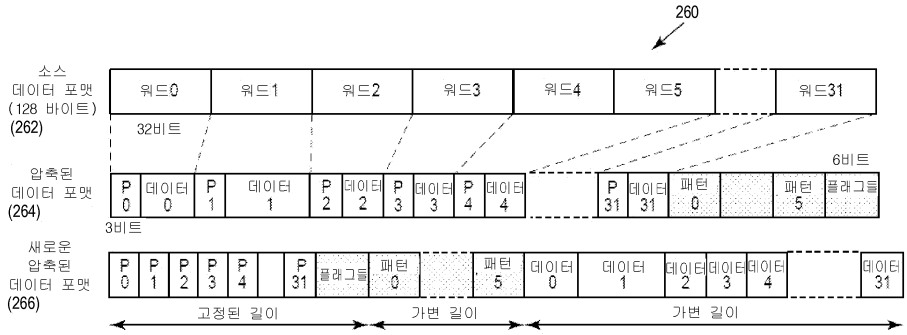
도면12



빈발 패턴 인코딩 (254)

프리픽스 (252)	인코딩된 패턴 (256)	데이터 크기(비트) (258)
000	제로런	3비트(8개의 0들까지 연장을 위해)
001	4-비트 사인-확장	4비트
010	1-바이트 사인-확장	8비트
011	하프워드 사인-확장	16비트
100	제로 하프워드로 패딩된 하프워드	비제로 하프워드(16비트)
101	2 하프워드들, 각각은 1바이트 사인-확장	2바이트(16비트)
110	반복 바이트들로 구성된 워드	8비트
111	압축되지 않은 워드	오리지널 워드(32비트)

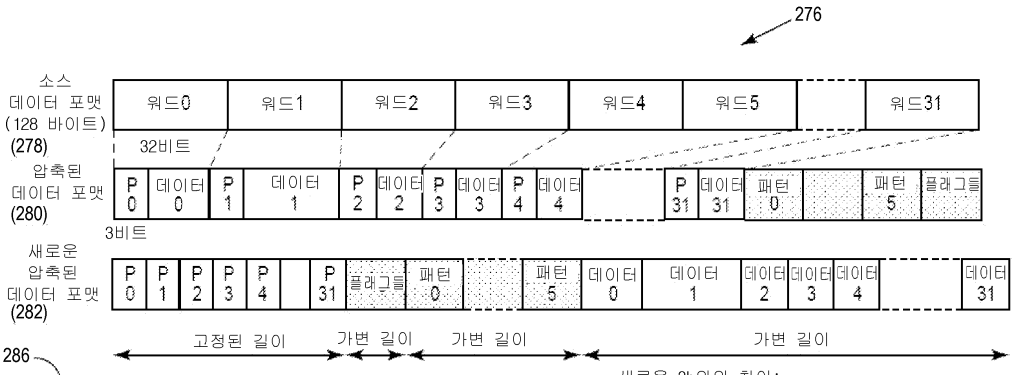
도면13



프리픽스 (268)	인코딩된 패턴 (272)	데이터 크기(비트) (274)
000	압축되지 않음	32
001	0x00000000	0
010	패턴 0	0
011	패턴 1	4
100	패턴 2	8
101	패턴 3	12
110	패턴 4	16
111	패턴 5	24

270

도면14



286

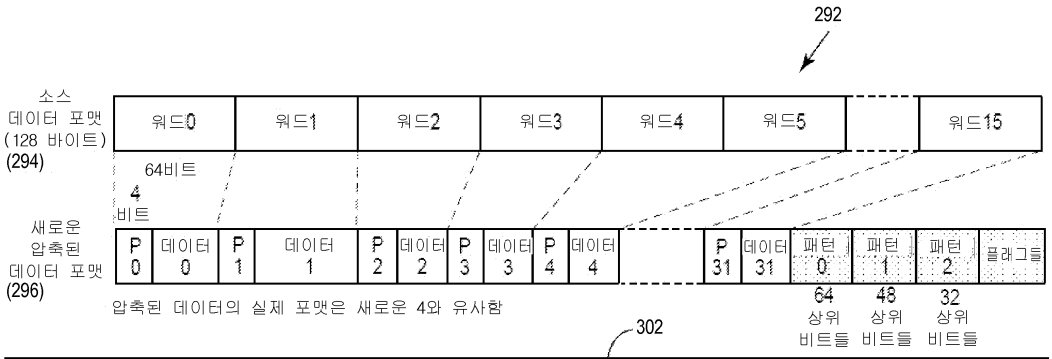
프리픽스 (284)	인코딩된 패턴 (288)	데이터 크기 (비트) (290)
000	압축되지 않음	32
001	0x00000000	0
010	패턴 0 (0xFFFFFFFF)	0
011	패턴 1 (0xFFFFFFF0)	4
100	패턴 2 (0xFFFFFF00)	8
101	패턴 3 (0xFFFFF000)	12
110	패턴 4 (0xFFFF0000) 또는 (0x0000FFFF)	16
111	패턴 5 (0xFF000000) 또는 (0x000000FF)	24

새로운 3b와의 차이:

- 패턴들이 또한 압축/인코딩됨(다음 슬라이드 참조)
- 패턴 4 및 패턴 5는 32개의 서브 패턴들(상위/하위 비트들을 반영함)을 가짐

패턴 0 : 32비트
패턴 1 : 28비트
패턴 2 : 24비트
패턴 3 : 20비트
패턴 4 : 16비트(0xFFFF0000 또는 0x0000FFFF 를 나타내도록 하나의 추가 비트를 가짐)
패턴 5 : 8비트(0xFF000000 또는0x000000FF를 위해 하나의 추가 비트를 가짐)
압축기에 대해: 패턴은 후보가 되기 위해 적어도 2번 발생해야 하고- 그렇지 않으면, 0이라 가정될 것임

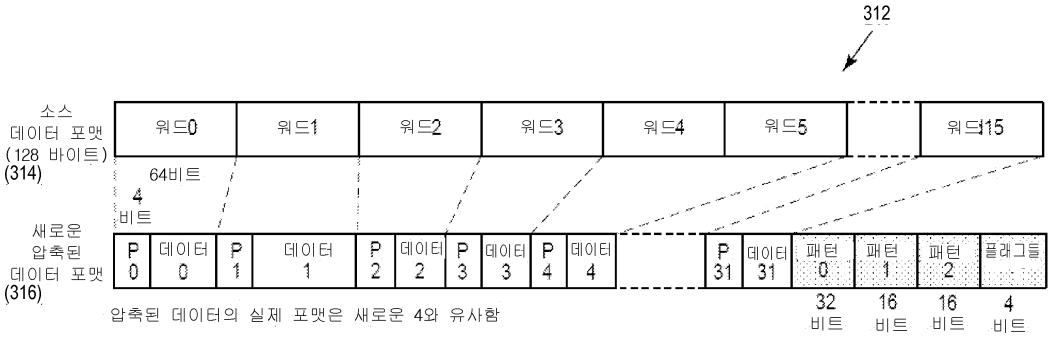
도면15



302

프리픽스 (298)	인코딩된 패턴 (304)	데이터 크기 (비트) (308)	프리픽스 (300)	인코딩된 패턴 (306)	데이터 크기 (비트) (310)
0000	0x0000000000000000	0	1000	0x20-0x3F 범위의 7 ASCII 문자	44
0001	0x000000000000yyyy	16	1001	0x00-0x7F 범위의 1 ASCII 문자	
0010	0x00000000yyyyyyyy	40	1010	0x60-0x7F 범위의 7 ASCII 문자	44
0011	패턴0	0	1011	0x00-0x7F 범위의 1 ASCII 문자	
0100	패턴1	16	1100	64의 미리 정의된 사전으로부터 8 ASCII 문자 (소문자에 포커싱함)	48
0101	패턴2	32	1101	64의 미리 정의된 사전으로부터 8 ASCII 문자 (대문자에 포커싱함)	48
0110	0x20-0x3F 범위의 8 ASCII 문자	40	1110	0x00-0x7F 범위의 8 ASCII 문자	56
0111	0x60-0x7F 범위의 ASCII	40	1111	압축되지 않음	64

도면16



322

프리픽스 (318)	인코딩된 패턴 (324)	데이터 크기 (비트) (328)	프리픽스 (320)	인코딩된 패턴 (326)	데이터 크기 (비트) (330)
0000	0x0000000000000000	0	1000	압축되지 않음	64
0001	0x0000000000000001	0	1001	패턴 0을 이용하는 워드	32
0010	0x000000000000000y	4	1010	패턴 1을 이용하는 워드	48
0011	0x00000000000000yy	8	1011	패턴 1 + 패턴0 을 이용하는 워드	16
0100	0x0000000000000yyy	12	1100	패턴 2를 이용하는 워드	48
0101	0x000000000000yyyy	16	1101	패턴 2 + 패턴0을 이용하는 워드	16
0110	0x00000000000yyyyy	20	1110	패턴 2 + 패턴1을 이용하는 워드	32
0111	0x00000000000yyyyy	20	1111	패턴 2 + 패턴1 + 패턴0 을 이용하는 워드	0

도면17

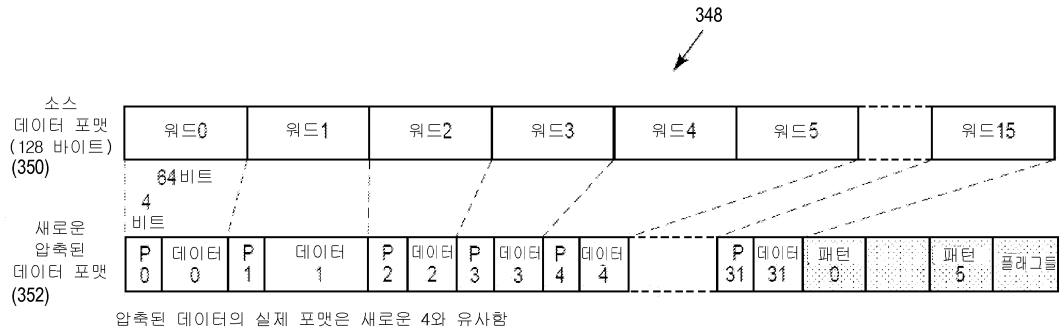
334 336 332

패턴	길이	정의 (338)
패턴0	32	64-비트 워드의 비트 63..32를 커버함
패턴1	16	64-비트 워드의 비트 31..16를 커버함
패턴2	16	64-비트 워드의 비트 15..0를 커버함

342 344 340

플래그들	값	정의 (346)
비트 0, 1	00	패턴0=0, 가변 길이 베이스 블록에서 코딩되지 않음
	01	-
	10	패턴0 = 0x000000yy 패턴0은 베이스 블록에서 8-비트 값으로서 코딩됨
	11	패턴0은 베이스 블록에서 32-비트 값으로서 코딩됨
비트 2	1	패턴1은 베이스 블록에서 16-비트 값으로서 코딩됨
비트 3	1	패턴2는 베이스 블록에서 8-비트 값으로서 코딩됨

도면18



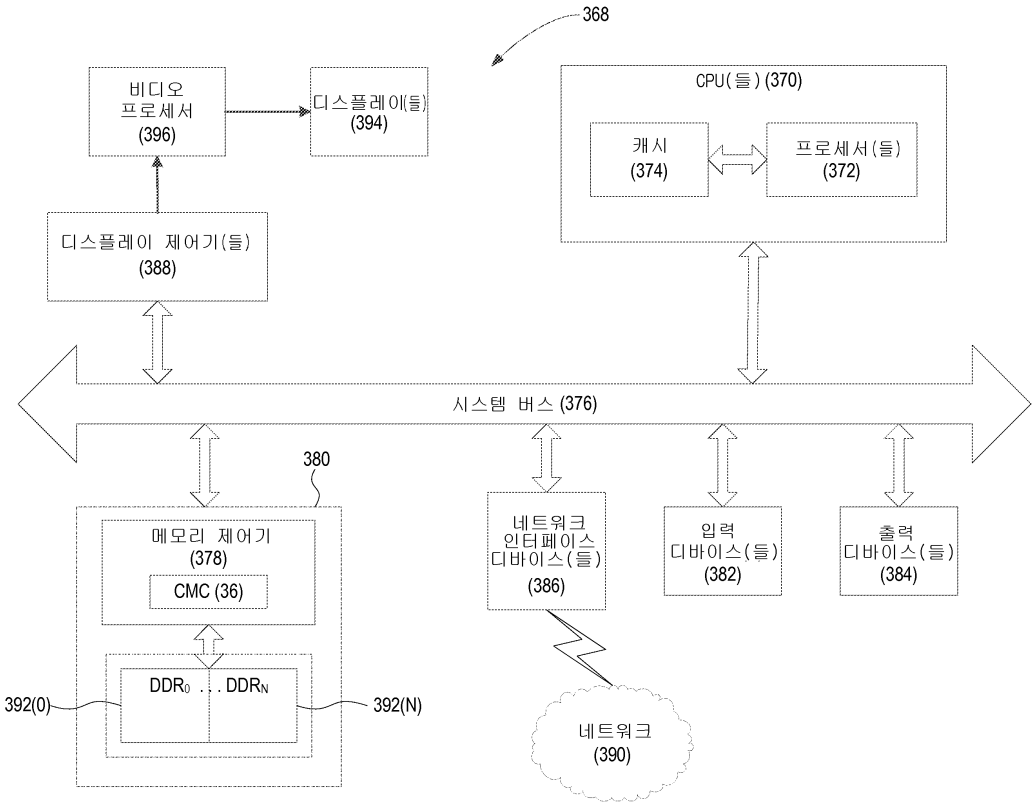
압축된 데이터의 실제 포맷은 새로운 4와 유사함

358

프리픽스 (354)	인코딩된 패턴 (360)	데이터 크기 (비트) (364)	프리픽스 (356)	인코딩된 패턴 (362)	데이터 크기 (비트) (366)
0000	0x0000000000000000	0	1000	0x00000000000yyyyyy	24
0001	0x000000000000000y	4	1001	패턴0	0
0010	0x000000000000000y	8	1010	패턴1	8
0011	0x000000000000000yy	12	1011	패턴2	16
0100	0x000000000000000yyy	16	1100	패턴3	24
0101	0x000000yy000000yy	16	1101	패턴4	32
0110	0x00000000000yyyyy	20	1110	패턴5	40
0111	0x0000yyyy0000yyyy	32	1111	압축되지 않음	64

패턴 0 : 64 상위 비트들
패턴 1 : 56 상위
패턴 2 : 48 상위
패턴 3 : 40 상위
패턴 4 : 32 상위
패턴 5 : 24 상위

도면19


368
비디오 프로세서 (396)
디스플레이(들) (394)
CPU(들) (370)
캐시 (374)
프로세서(들) (372)
디스플레이 제어기(들) (388)
시스템 버스 (376)
380
메모리 제어기 (378)
CMC (36)
DDR0 . . . DDRN
392(0)
392(N)
네트워크 인터페이스 디바이스(들) (386)
입력 디바이스(들) (382)
출력 디바이스(들) (384)
네트워크 (390)