(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2010년05월03일
(11) 등록번호 10-0955722
(24) 등록일자 2010년04월23일

(51) Int. Cl.
G06F 12/08 (2006.01)
(21) 출원번호 10-2005-7009466
(22) 출원일자(국제출원일자) 2003년11월06일
심사청구일자 2008년11월05일
(85) 번역문제출일자 2005년05월25일
(65) 공개번호 10-2005-0085150
(43) 공개일자 2005년08월29일
(86) 국제출원번호 PCT/US2003/035280
(87) 국제공개번호 WO 2004/049171
국제공개일자 2004년06월10일
(30) 우선권주장
10/304,605 2002년11월26일 미국(US)
(56) 선행기술조사문헌
KR1019980063475 A
KR1019990071554 A
전체 청구항 수 : 총 10 항

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 원 에이엠디 플레이스 메일 스톱68
(72) 발명자
앨서프 미첼
미국 텍사스 78746 오스틴 캠퍼 코브 2103
(74) 대리인
박장원

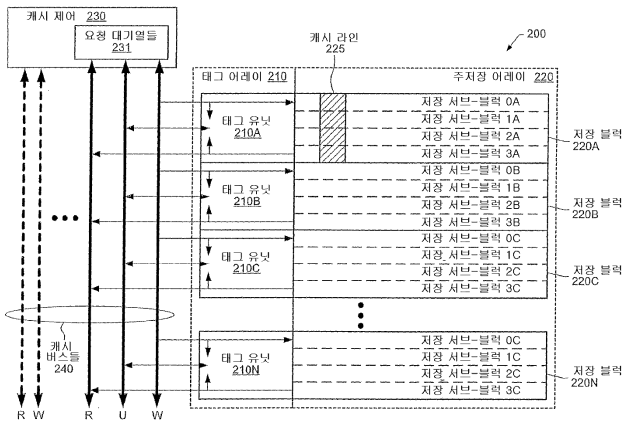
심사관 : 이상헌

(54) 사이클당 복수의 액세스를 지원하는 캐시 메모리를 구비한마이크로프로세서

(57) 요 약

2 레벨 캐시 메모리(200)를 포함하는 마이크로프로세서(100)는 사이클당 복수의 액세스들을 지원한다. 상기 마이크로프로세서(100)는 복수의 버스들(240)에 연결된 캐시 메모리(200)를 포함하는 캐시 메모리 서브시스템에 연결된 실행 유닛(124)을 포함한다. 상기 캐시 메모리(200)는 독립적으로 액세스 가능한 저장 블록들(220)을 포함한다. 상기 버스들(240)은 상기 저장 블록들(220) 각각에 복수의 캐시 액세스 요청들을 전달하도록 연결된다. 서로 다른 저장 블록들(220)은 상기 복수의 캐시 버스들(240)에서 전달된 복수의 캐시 액세스 요청들에 응답하여 동시에 액세스 가능하다.

대 표 도



특허청구의 범위

청구항 1

마이크로 프로세서로서,

명령어들 및 데이터를 연산하도록 구성된 실행 유닛(124)과;

상기 실행 유닛에 연결된 캐시 메모리 서브시스템을 포함하며,

상기 캐시 메모리 서브시스템은,

독립적으로 액세스 가능한 복수의 저장 블록들(220A, 220B,...220N)을 포함하는 캐시 메모리와;

상기 복수의 저장 블록들에 연결되고 그리고 각각 어드레스 태그 값을 포함하는 복수의 태그들을 저장하도록 구성된 독립적으로 액세스 가능한 복수의 태그 유닛들(210A, 210B,...210N)과; 그리고

복수의 캐시 액세스 요청들을 상기 복수의 저장 블록들 각각에 전달하도록 연결된 복수의 캐시 버스들(240)을 포함하며,

상기 복수의 캐시 버스들에 의해 전달되는 상기 복수의 캐시 액세스 요청들에 응답하여, 서로 다른 상기 복수의 저장 블록들이 동시에 액세스 가능하며;

상기 복수의 저장 블록들은 상기 복수의 태그 유닛들에 대하여 비동기식으로 액세스되며; 그리고

상기 복수의 태그 유닛들은 상기 마이크로 프로세서의 나머지 부분과 동기식으로 동작하는 것을 특징으로 하는 마이크로프로세서.

청구항 2

제1항에 있어서,

상기 복수의 저장 블록들 각각은, 독립적으로 액세스 가능한 복수의 저장 서브-블록들을 포함하며,

상기 저장 서브-블록들 각각은, 어써트된(asserted) 비동기 읽기 가능 신호를 수신하는 것에 응답하여 출력 데이터를 제공하도록 구성되며, 그리고

상기 저장 서브-블록들 중 소정의 하나는, 어써트된(asserted) 비동기 쓰기 가능 신호를 수신하는 것에 응답하여 데이터를 저장하도록 구성되는 것을 특징으로 하는 마이크로프로세서.

청구항 3

제1항에 있어서,

상기 복수의 캐시 액세스 요청들을 수신하는 것에 응답하여, 서로 다른 상기 복수의 태그 유닛들은 동시에 액세스 가능한 것을 특징으로 하는 마이크로프로세서.

청구항 4

제1항에 있어서,

상기 복수의 저장 블록들은 독립적으로 액세스 가능한 복수의 저장 서브-블록들을 포함하고,

서로 다른 상기 저장 블록들과 관련된 상기 서브-블록들 중 임의의 두 개는 동시에 액세스 가능하고, 그리고

동일한 저장 블록과 관련된 상기 서브-블록들 중 임의의 두 개는 연속된 일련의 사이클들에서 액세스 가능한 것을 특징으로 하는 마이크로프로세서.

청구항 5

캐시 메모리 서브시스템으로서,

독립적으로 액세스 가능한 복수의 저장 블록들(220A, 220B,...220N)을 포함하는 캐시 메모리와;

상기 복수의 저장 블록들에 연결되고 그리고 각각 어드레스 태그 값을 포함하는 복수의 태그들을 저장하도록 구성된 독립적으로 액세스 가능한 복수의 태그 유닛들(210A, 210B,...210N)과; 그리고

복수의 캐시 액세스 요청들을 상기 복수의 저장 블록들 각각에 전달하도록 연결된 복수의 캐시 버스들(240)

을 포함하며,

상기 복수의 캐시 버스들에 의해 전달된 상기 복수의 캐시 액세스 요청들에 응답하여, 서로 다른 상기 복수의 저장 블록들은 동시에 액세스 가능하며,

상기 복수의 저장 블록들은 상기 복수의 태그 유닛들에 대하여 비동기식으로 액세스되며; 그리고

상기 복수의 태그 유닛들은 CPU 클럭을 통해 동기식으로 액세스되는 것을 특징으로 하는 캐시 메모리 서브시스템.

청구항 6

제5항에 있어서,

상기 복수의 저장 블록들 각각은 독립적으로 액세스 가능한 복수의 저장 서브-블록들을 포함하고,

서로 다른 상기 저장 블록들과 관련된 상기 서브-블록들 중 임의의 두 개는 동시에 액세스 가능하고, 그리고

동일한 저장 블록과 관련된 상기 서브-블록들 중 임의의 두 개는 연속된 일련의 사이클들에서 액세스 가능한 것을 특징으로 하는 캐시 메모리 서브시스템.

청구항 7

컴퓨터 시스템으로서,

명령어들 및 데이터를 저장하도록 구성된 시스템 메모리와;

메모리 버스를 통해 상기 시스템 메모리에 연결된 마이크로프로세서를 포함하며,

상기 마이크로프로세서는,

상기 명령어들 및 데이터를 연산하도록 구성된 실행 유닛(124)과;

상기 실행 유닛에 연결되고 그리고 상기 실행 유닛에 의한 실행을 위해 상기 명령어들 및 데이터를 저장하도록 구성된 캐시 메모리 서브시스템을 포함하며,

상기 캐시 메모리 서브시스템은,

독립적으로 액세스 가능한 복수의 저장 블록들(220A, 220B,...220N)을 포함하는 캐시 메모리와;

상기 복수의 저장 블록들에 연결되고 그리고 각각 어드레스 태그 값을 포함하는 복수의 태그들을 저장하도록 구성된 독립적으로 액세스 가능한 복수의 태그 유닛들(210A, 210B,...210N)과; 그리고

복수의 캐시 액세스 요청들을 상기 복수의 저장 블록들 각각에 전달하도록 연결된 복수의 캐시 버스들(240)을 포함하며,

상기 복수의 캐시 버스들에 의해 전달된 상기 복수의 캐시 액세스 요청들에 응답하여, 서로 다른 상기 복수의 저장 블록들은 동시에 액세스 가능하며,

상기 복수의 저장 블록들은 상기 복수의 태그 유닛들에 대하여 비동기식으로 액세스되며; 그리고

상기 복수의 태그 유닛들은 상기 마이크로 프로세서의 나머지 부분과 동기식으로 동작하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 8

제7항에 있어서,

상기 복수의 저장 블록들 각각은, 독립적으로 액세스 가능한 복수의 저장 서브-블록들을 포함하며,

상기 저장 서브-블록들 각각은, 어써트된(asserted) 비동기 읽기 가능 신호를 수신하는 것에 응답하여 출력 데

이터를 제공하도록 구성되며, 그리고

상기 저장 서브-블록들 중 소정의 하나는, 어써트된(asserted) 비동기 쓰기 가능 신호를 수신하는 것에 응답하여 데이터를 저장하도록 구성되는 것을 특징으로 하는 컴퓨터 시스템.

청구항 9

제7항에 있어서,

상기 복수의 캐시 액세스 요청들을 수신하는 것에 응답하여, 서로 다른 상기 복수의 태그 유닛들이 동시에 액세스 가능한 것을 특징으로 하는 컴퓨터 시스템.

청구항 10

제7항에 있어서,

상기 복수의 저장 블록들 각각은 독립적으로 액세스 가능한 복수의 저장 서브-블록들을 포함하고,

서로 다른 상기 저장 블록들과 관련된 상기 서브-블록들 중 임의의 두 개는 동시에 액세스 가능하고, 그리고

동일한 저장 블록과 관련된 상기 서브-블록들 중 임의의 두 개는 연속된 일련의 사이클들에서 액세스 가능한 것을 특징으로 하는 컴퓨터 시스템.

명 세 서

기 술 분 야

[0001] 본 발명은 마이크로프로세서에 관한 것이고, 더욱 상세하게는 마이크로프로세서에서 캐시 메모리 관리에 관한 것이다.

배 경 기 술

[0002] 일반적인 컴퓨터 시스템들은 하나 이상의 마이크로프로세서들을 포함하며, 상기 마이크로프로세서는 하나 이상의 시스템 메모리에 연결된다. 프로세서들은 시스템 메모리들에 저장되어 있는 데이터를 연산하고 코드를 실행한다. 본 명세서에서 사용되는 용어 "프로세서"는 용어 마이크로프로세서와 동의어로 사용된다. 명령어 및 데이터의 인출(fetching)과 저장을 용이하게 하기 위해, 일반적으로 프로세서는 일부 메모리 시스템 유형을 사용한다. 게다가, 시스템 메모리로의 신속한 액세스 처리를 위해 메모리 시스템은 하나 이상의 캐시 메모리들을 포함한다. 예를 들면, 일부 마이크로프로세서들은 하나 이상의 캐시 메모리 레벨들을 구비하여 구현된다. 일반적인 마이크로프로세서들은 레벨1(L1) 캐시 및 레벨2(L2) 캐시를 사용하고, 일부 새로운 프로세서들은 레벨3(L3) 캐시를 또한 사용한다. 많은 과거 프로세서에서, L1 캐시는 칩내(on-chip)에 존재하고 L2 캐시는 칩 밖(off-chip)에 존재한다. 그러나, 메모리 액세스 타임을 더욱 개량하기 위해, 더 새로운 프로세서들은 칩내에 존재하는 L2 캐시를 사용한다.

[0003] 대체적으로, L2 캐시는 L1 캐시보다 크고 느리다. 게다가, L1 캐시가 분리된 명령어 캐시와 데이터 캐시로 구현되는 반면, L2 캐시는 종종 통합 캐시(unified cache)로 구현된다. L1 데이터 캐시는 마이크로프로세서에서 작동하는 소프트웨어에 의해 가장 최근에 '읽기' 혹은 '쓰기'된 데이터를 보유하는데 사용된다. L1 명령어 캐시는 가장 최근에 실행된 명령어들을 보유하는 것을 제외하면 L1 데이터 캐시와 유사하다. 편의를 위해 L1 명령어 캐시와 L1 데이터 캐시는 적절하게 단순히 L1 캐시로 불린다. L2 캐시는 L1 캐시에 적합하지 않은 명령어 및 데이터를 보유하는데 사용된다. L2 캐시는 배타적(exclusive)(예컨대, L1 캐시에 존재하지 않는 정보를 저장)이거나 포함적(inclusive)(예컨대, L1 캐시에 존재하는 정보의 복사본을 저장함)이다.

[0004] 일반적으로 메모리 시스템들은 일부 유형의 캐시 일관성 메커니즘(cache coherence mechanism)을 사용하여 정확한 데이터가 요청자에게 공급되도록 한다. 일반적으로 캐시 일관성 메커니즘은 일관성 유닛으로서 단일 요청에서 이송되는 데이터의 크기를 사용한다. 일반적으로 일관성 유닛은 캐시 라인으로 불린다. 일부 프로세서들에서, 예컨대, 소정의 캐시 라인은 64 바이트일 것이며, 반면에 일부 다른 프로세서들은 32 바이트의 캐시 라인을 이용할 것이다. 또 다른 프로세서들에서, 단일 캐시 라인은 상기와 다른 수의 바이트를 포함할 수 있다. 만약 요청이 L1 및 L2 캐시들에서 미스(miss)하면, 단일 워드만이 요청되었더라도 복수의 워드를 갖는 전체 캐시 라인이 주메모리(main memory)로부터 L2 및 L1 캐시들로 전송된다. 유사하게, 단일 워드에 대한 요청이

L1 캐시에서는 미스했지만 L2에서 히트(hit)한다면, 요청된 워드를 포함한 전체 L2 캐시 라인이 L2 캐시로부터 L1 캐시로 전송된다. 그러므로, 각각의 캐시 라인보다 작은 데이터 유닛에 대한 요청은 전체 캐시 라인이 L2 캐시와 L1 캐시 사이에서 전송되도록 한다. 일반적으로 이러한 이송을 완성하는데 복수의 사이클들이 필요하다.

[0005] 캐시 가능한 메모리를 읽고 혹은 쓰기하는 동안에, 요청된 정보(예컨대, 명령어 혹은 데이터)가 이용가능한지를 검사하기 위해 L1 캐시가 제일 먼저 체크된다. 만약 정보가 이용가능하다면 히트가 발생한다. 만약 정보가 이용불가능하다면, 미스가 발생한다. 만약 미스가 발생하면, L2 캐시가 그 다음 체크된다. 그러므로, L1 캐시에서 미스가 발생하고 L2 캐시에서 히트가 발생하면, 상기 정보는 L2 캐시로부터 L1 캐시로 이송된다. 아래 설명하는 바와 같이, L2와 L1 캐시 사이에서 이송되는 정보의 양은 일반적으로 하나의 캐시 라인이다. 게다가, L1 캐시에서 이용가능한 공간에 따라, 새로운 캐시 라인을 위한 공간을 만들기 위해 L1 캐시로부터 캐시 라인이 축출되어 L2 캐시 라인에 저장될 것이다. 현대 프로세서들은 이러한 복수의 데이터 이동을 수행함으로써 캐시 데이터 이동을 최적화한다. 종래의 일부 프로세서들에서, 이러한 캐시 라인 "교체(swap)" 동안에는 L1, L2 캐시 모두에 대한 다른 액세스들이 처리될 수 없다.

발명의 상세한 설명

[0006] 사이클당 복수의 액세스를 지원하는 레벨2 캐시 메모리를 구비하는 마이크로프로세서의 다양한 실시예들이 개시된다. 일 실시예에서, 마이크로프로세서는 실행 유닛(execution unit)을 포함하며, 여기서 상기 실행 유닛은 복수의 버스들에 연결된 캐시 메모리를 포함한 캐시 메모리 서브시스템에 연결된다. 상기 캐시 메모리는 독립적으로 액세스 가능한 복수의 저장 블록들을 구비한다. 상기 버스들은 복수의 캐시 액세스 요청들을 각 저장 블록들에 전달하기 위해 연결된다. 복수의 캐시 버스들에 의해 전달된 복수의 캐시 액세스 요청들에 응답하여, 서로 다른 저장 블록들은 동시에 액세스 가능하다.

[0007] 일 실시예에서, 상기 캐시 메모리는 복수의 저장 블록들에 연결된 독립적으로 액세스 가능한 복수의 태그 유닛(tag unit)을 구비하고, 상기 태그 유닛은 각각 어드레스 태그 값을 갖는 복수의 태그들을 저장하도록 구성된다. 복수의 캐시 액세스 요청들의 수신에 응답하여, 서로 다른 복수의 태그 유닛은 동시에 액세스 가능하다.

[0008] 다른 실시예에서, 복수의 저장 블록 각각은 독립적으로 액세스 가능한 복수의 저장 서브-블록들을 구비한다. 서로 다른 저장 블록들에 관련된 임의의 두 개의 서브-블록들은 동시에 액세스 가능하다. 게다가, 동일한 저장 블록에 관련된 임의의 두 개의 서브-블록들은 연속적인 일련의 사이클들에서 액세스 가능하다.

실 시 예

[0017] 도 1에서, 예시적인 마이크로프로세서(100)의 일 실시예에 대한 블록 다이어그램이 도시된다. 마이크로프로세서(100)는 시스템 메모리(도시되지 않음)에 저장된 명령어들을 실행하도록 구성된다. 이러한 명령어들 다수는 시스템 메모리에 저장된 데이터를 연산한다. 시스템 메모리는 컴퓨터 시스템 전반에 분포되어 있고, 그리고 예컨대 마이크로프로세서(100)와 같은 하나 이상의 마이크로프로세서들에 의해 액세스 된다. 일 실시예에서, 마이크로프로세서(100)는 예컨대 Athlon™ 프로세서와 같은 x86 아키텍처를 구현하는 마이크로프로세서의 예시이다. 그러나, 다른 실시예들은 다른 유형의 마이크로프로세서들을 포함할 수 있다.

[0018] 상술된 실시예에서, 마이크로프로세서(100)는 제 1 레벨 1(L1) 캐시 및 제 2 L1 캐시: 명령어 캐시(101A) 및 데이터 캐시(101B)를 포함한다. 실시에 따라, L1 캐시는 통합 캐시 혹은 분기(bifurcate) 캐시이다. 단순함을 위해 어느 경우라도, 적절하게 명령어 캐시(101A) 및 데이터 캐시(101B)는 집합적으로 L1 캐시로 불린다. 마이크로프로세서(100)는 명령어 캐시(101A)에 밀접하게 연결된 프리디코드 유닛(102)(predecode unit)과 분기 예측 로직(103, branch prediction logic)을 또한 포함한다. 마이크로프로세서(100)는 명령어 디코더(104)에 연결된 인출 및 디코드 제어 유닛(105)을 또한 포함하며, 상기 명령어 디코더와 인출 및 디코드 제어 유닛은 모두 명령어 캐시(101A)에 연결되어있다. 명령어 제어 유닛(106)은 명령어 디코더(104)로부터 명령어를 수신하고 스케줄러(118)에 연산을 디스패치(dispatch)하기 위해 연결된다. 스케줄러(118)는 명령어 제어 유닛(106)으로부터 디스패치된 연산들을 수신하고 실행 유닛(124)에 연산들을 제공하기 위해 연결된다. 실행 유닛(124)은 데이터 캐시(101B)에 액세스를 수행하도록 구성된 로드/저장 유닛(126)(load/store unit)을 포함한다. 실행 유닛(124)에서 생성된 결과는 후속으로 제공된 명령어들에 대한 오퍼랜드(operand) 값으로 사용되고 및/또는 기록 파일에 저장된다. 게다가, 마이크로프로세서(100)는 명령어 캐시(101A), 데이터 캐시(101B) 그리고 시스템 메모리 사이에 연결된 칩내의 L2 캐시(130)를 포함한다.

[0019] 명령어 캐시(101A)는 실행 전에 명령어들을 저장한다. 명령어 캐시(101A)에 관련된 기능들은 명령어 인출(읽기), 명령어 선-인출(pre-fetching), 명령어 프리디코딩 및 분기 예측이다. 명령어 코드는 버스 인터페이스 유닛(140)을 통해 시스템 메모리로부터, 혹은 아래서 설명되는 바와 같이 L2 캐시(130)로부터 코드를 선-인출함으로써 명령어 캐시(106)로 제공된다. 명령어 캐시(101A)는 다양한 구성(예컨대, 세트-연관(set-associaive), 완전-연관(fully-associative), 혹은 직접-매핑(direct-mapped))으로 구현된다. 일 실시예에서, 명령어 캐시(101A)는 복수의 캐시 라인들을 저장하도록 구성되고, 명령어 캐시(101A)의 소정의 캐시 라인 내의 바이트 숫자는 구현예에 따라 달라질 수 있다. 게다가, 일 실시예에서 명령어 캐시(101A)는 정적 기억 장치(SRAM)로 구현될 수 있는 반면, 다른 실시예들은 다른 유형의 메모리를 포함할 수 있다. 일 실시예에서, 명령어 캐시(101A)는 예컨대 캐시 라인 삽입(fill), 대체, 그리고 일관성(coherency)을 제어하기 위한 제어 회로 소자(도시되지 않음)를 포함할 수 있다.

[0020] 명령어 디코더(104)는 명령어들을 연산들로 디코드하고, 이들은 일반적으로 마이크로코드 ROM 혹은 MROM(도시되지 않음)으로 불리는 칩내의 읽기-전용 메모리(ROM)내에 저장된 연산들을 사용하여 직접 혹은 간접적으로 디코드된다. 명령어 디코더(104)는 특정 명령어들을 실행 유닛(124)에서 실행될 수 있는 연산들로 디코드한다. 단순 명령어들은 단일 연산에 대응한다. 일부 실시예들에서, 더욱 복잡한 명령어들은 복수의 연산들에 대응한다.

[0021] 명령어 제어 유닛(106)은 실행 유닛(124)에 연산들을 디스패치하는 것을 제어한다. 일 실시예에서, 명령어 제어 유닛(106)은 명령어 디코더(104)로부터 수신된 연산들을 보유하기 위해 재순서 버퍼(reorder buffer)를 포함한다. 명령어 제어 유닛(106)은 연산들의 폐기(retirement)를 제어하도록 또한 구성된다.

[0022] 명령어 제어 유닛(106)의 출력에서 제공되는 연산들 및 즉시 데이터(immediate data)는 스케줄러(118)로 라우팅된다. 스케줄러(118)는 하나 이상의 스케줄러 유닛(예컨대, 정수(integer) 스케줄러 유닛 및 부동 소수점 스케줄러 유닛)을 포함한다. 본 명세서에 사용되는 바와 같이, 스케줄러는 연산들이 실행을 위한 준비가 되고 하나 이상의 실행 유닛들에 연산을 제공할 준비가 된 때를 검출하는 디바이스이다. 예를 들면, 예약 스테이션(reservation station)은 스케줄러이다. 각 스케줄러(118)는 실행 유닛(124)으로 제공되기를 대기하는 수 개의 연산들에 대한 연산 정보(예컨대, 오퍼랜드 값, 오퍼랜드 태그, 및/또는 즉시 데이터 뿐 아니라 비트 엔코딩된 실행 비트들)를 보유할 수 있다. 일부 실시예들에서, 각 스케줄러(118)는 오퍼랜드 값을 저장하지 않는다. 대신에, 각 스케줄러는 오퍼랜드 값들이 실행 유닛(124)에서 읽기 될 수 있을 때를 결정하기 위해 기록 파일에서 이용가능한 결과들 및 제공된 연산들을 모니터한다. 일부 실시예들에서, 각 스케줄러(118)는 실행 유닛(124) 중 하나의 전담 유닛에 관련한다. 다른 실시예들에서, 단일 스케줄러(118)는 하나 이상의 실행 유닛(124)에 연산들을 제공한다.

[0023] 일 실시예에서, 실행 유닛(124)은 예컨대 정수 실행 유닛과 같은 실행 유닛을 포함한다. 그러나 다른 실시예들에서, 마이크로프로세서(100)는 수퍼스칼라 프로세서(superscalar processor)이고, 이런 경우에 실행 유닛(124)은 이동(shift), 회전(rotation), 논리 연산 및 분기 연산들뿐 아니라 덧셈 그리고 뺄셈의 정수 산술 연산을 수행하도록 구성된 복수의 실행 유닛(예컨대, 복수의 정수 실행 유닛(도시되지 않음))들을 포함한다. 게다가, 부동 소수점 연산들을 수행하기 위해 하나 이상의 부동 소수점 유닛(도시되지 않음)들이 또한 포함된다. 하나 이상의 실행 유닛들은 로드/저장 유닛(126)에 의해 수행되는 메모리 연산들의 로드 및 저장을 위해 어드레스를 생성하도록 구성된다.

[0024] 로드/저장 유닛(126)은 실행 유닛(124)과 데이터 캐시(101B) 사이에 인터페이스를 제공하도록 구성된다. 일 실시예에서, 로드/저장 유닛(126)은 로드 혹은 저장들을 보류하기 위해 데이터 및 어드레스 정보를 위한 몇 개의 저장 위치들을 구비한 로드/저장 버퍼(도시되지 않음)와 함께 구성된다. 로드/저장 유닛(126)은 데이터 일관성을 유지하기 위해 새로운 저장 명령어들에 대한 오래된 로드 명령어들의 의존도(dependency)를 또한 체크한다.

[0025] 데이터 캐시(101B)는 로드/저장 유닛(126)과 시스템 메모리 사이에 이송되는 데이터를 저장하도록 제공된 캐시 메모리이다. 상술한 명령어 캐시(101A)에 유사하게, 데이터 캐시(101B)는 세트-연관 구성을 포함하는 다양한 특정 메모리 구성으로 구현된다. 일 실시예에서, 데이터 캐시(101B) 및 명령어 캐시(101A)는 별개의 캐시 유닛들로 구현된다. 상술한 바와 같이, 대체 실시예들에서는 데이터 캐시(101B) 및 명령어 캐시(101A)는 통합 캐시로 구현될 수도 있다. 일 실시예에서, 데이터 캐시(101B)는 복수의 캐시 라인들을 저장하며, 데이터 캐시(101B)의 소정의 캐시 라인 내의 바이트 숫자는 구현예별로 달라질 수 있다. 명령어 캐시(101A)와 유사하게, 일 실시예에서 데이터 캐시(101B)는 정적 기억 장치(SRAM)로 구현될 수 있는 반면, 다른 실시예들은 다른 유형의 메모리를 포함할 수 있다. 일 실시예에서, 데이터 캐시(101B)는 예컨대 캐시 라인 삽입, 대체, 그리고 일관성을 제어하기 위한 제어 회로 소자(도시되지 않음)를 포함할 수 있다.

[0026] L2 캐시(130)는 또한 캐시 메모리로서 명령어 및/또는 데이터를 저장하도록 구성된다. 도시된 실시예에서, L2 캐시(130)는 칩내의 캐시이고 완전 연관, 세트 연관, 혹은 이들 양자의 결합으로 구성된다. 일 실시예에서, L2 캐시(130)는 복수의 캐시 라인들을 저장한다. L2 캐시(130)는 예컨대 캐시 라인 삽입, 대체, 그리고 일관성을 제어하기 위한 제어 회로 소자(도시되지 않음)를 포함할 수 있다.

[0027] 도 4의 도면과 관련하여 아래서 더욱 자세하게 설명되는 바와 같이, 일 실시예에서, L2 캐시(103)의 일부분(예컨대, L2 캐시(130)의 주 캐시 저장 어레이)은 비동기적으로 액세스되는 반면에, L2 캐시(130)의 다른 부분(예컨대, 주 어레이에 대응하는 L2 캐시(130) 태그 어레이)은 완전 파이프라인 방식에서 동기식으로 액세스된다. 게다가, 주 저장 어레이 셀들은 독립적으로 액세스 가능한 복수의 저장 블록들로 배열되거나 혹은 다른 저장 블록들과 동시에 액세스되는 메모리 유닛들로 배열된다.

[0028] 버스 인터페이스 유닛(140)은 시스템 메모리와 L2 캐시(130) 사이, 그리고 시스템 메모리와 L1 명령어 캐시(101A) 및 L1 데이터 캐시(101B) 사이에서 명령어들 및 데이터를 이송하도록 구성된다. 일 실시예에서, 버스 인터페이스 유닛(140)은 쓰기 사이클 스트림라이닝(streamlining) 동안에 쓰기 트랜잭션들(transaction)을 버퍼링하기 위해 버퍼들(도시되지 않음)을 포함한다.

[0029] 도 2에서, 도 1의 마이크로프로세서의 캐시 서브 시스템의 일 실시예의 블록 다이어그램이 도시된다. 캐시 서브시스템(200)은 태그 어레이(210)에 연결된 주 저장 어레이(220)를 포함하는 캐시 메모리를 구비한다. 게다가, 캐시 서브시스템(200)은 캐시 버스들(240)로 표시된 복수의 버스들을 통해 태그 어레이(210)에 연결된 캐시 제어(230)를 포함한다. 일 실시예에서, 주 저장 어레이(220) 및 태그 어레이(210)는 예컨대 도 1의 L2 캐시(130)와 같은 L2 캐시 서브시스템에서 사용된다. 그러나, 다른 실시예에서, 주 저장 어레이(220) 및 태그 어레이(210)는 다른 캐시 서브시스템에서 사용될 수 있다.

[0030] 주 저장 어레이(220)는 독립적으로 액세스 가능한 저장 블록들로 배열되는 복수의 메모리 유닛들을 포함한다. 도시된 실시예에서, 저장 블록들은 220A-220N로 표시되고, 여기서 N은 임의의 블록 개수이다. 일 실시예에서는 8개의 저장 블록들이 존재하는 반면, 다른 실시예들에서는 이와 다른 수의 블록들이 존재할 수 있다. 게다가, 각 저장 블록들(220A-N)은 서브 블록들(0-3)로 표시되는 독립적으로 액세스 가능한 네 개의 저장 서브-블록들을 포함한다. 각각의 저장 블록(220A-N)은 네 개의 서브-블록들을 포함하고 있지만, 다른 실시예들에서는 각 저장 블록들(220A-N)은 다른 수의 서브-블록들을 포함할 수 있음을 인식해야 한다.

[0031] 태그 어레이(210)는 캐시 라인 태그 정보를 저장하도록 구성된 저장 어레이이다. 태그 내의 어드레스 정보는 소정의 데이터 일부가 메모리 요청 동안에 캐시내에 존재하는지를 결정하는데 사용된다. 예를 들면, 메모리 요청은 요청된 데이터의 어드레스를 포함한다. 태그 어레이(210) 내의 비교 로직(도시되지 않음)은 요청된 어드레스를 태그 어레이(210)의 소정의 태그 유닛(210A-N) 내에 저장된 각 태그의 어드레스 정보와 비교한다. 만약, 요청된 어드레스와 소정의 태그와 관련된 어드레스 사이에 매칭(match)이 발생하면, 상술한 바와 같이 히트가 표시된다. 만약 매칭 태그가 없으면, 미스가 표시된다. 태그 어레이(210)는 210A-210N으로 표시된 복수의 태그 유닛들로 배열되며, 여기서 N은 태그 유닛들의 임의의 개수이다. 각 태그 유닛(210A-N)은 독립적으로 액세스 가능한 저장 블록들 중 하나에 대응하고 그리고 복수의 태그 요소들을 포함한다. 예를 들면, 도시된 실시예에서, 태그 유닛(210A)은 저장 블록(220A)에 대응한다. 게다가, 소정의 태그 유닛 내의 각 태그 요소는 소정의 저장 블록에서 모든 저장 서브-블록들에 대응한다. 태그 엔트리는 저장 어레이(220)에 저장된 데이터 피스(piece)의 어드레스 부분를 저장한다. 어드레스의 다른 부분은 액세스될 저장 어레이(220) 내의 위치를 특정할 것이다. 캐시 라인이 소정의 저장 블록 내에 저장되어 상기 캐시 라인은 네 개의 모든 저장 서브-블록들에 존재한다. 예를 들면, 도시된 실시예에서, 캐시 라인(225)은 각 서브-블록에 저장된 하나의 서브 라인을 갖는 저장 블록(220A)의 저장 서브-블록(0-3) 전반에 저장된다.

[0032] 캐시 제어(230)는 하나 이상의 요청 대기열(231) 및 제어 로직(도시되지 않음)을 포함하며, 이는 들어오는 캐시 요청들을 저장하고 그리고 캐시 버스들(240)로 전달하기 위한 요청들을 선택하고 스케줄하도록 구성된다. 요청 대기열(231)은 수신된 캐시 요청들의 유형 및/또는 소스(source)에 대응한다(아래서 설명됨). 일 실시예에서, 캐시 제어(230)는 캐시가 도 1의 L1 명령어 캐시(101A) 또는 L1 데이터 캐시(101B) 모두에서 미스되면 이러한 소스로부터 요청들을 수신한다. 게다가, 캐시 제어(23)는 버스 인터페이스 유닛(140)으로부터 혹은 스눕 요청의 형태로 다른 프로세서(도시되지 않음)로부터 요청들을 수신한다.

[0033] 대체로, 캐시 요청들은 읽기 요청, 쓰기 요청 혹은 업데이트 요청의 형태로 들어온다. 도시된 실시예에서, 이러한 요청들을 수용하기 위해 캐시 버스들(240)은 읽기 버스(R), 쓰기 버스(W) 그리고 업데이트 버스(U)를 포함한다. 반면, 다른 실시예들은 임의의 개수의 각 버스 유형을 포함하도록 한다. 예를 들면, 추가의 버스들(점선에

의해 표시됨)이 대체 실시예들에서 사용될 수 있다. 캐시 제어(230)는 캐시 버스들(240)을 통해 태그 어레이(210) 및 주 저장 어레이(220)에 연결된다. 업데이트 버스(U)가 태그 어레이(210)에만 연결되는데 반해, 읽기 버스(R) 및 쓰기 버스(W)는 주 저장 어레이(220)에도 연결된다. 일 실시예에서, 버스(R)는 어드레스 및 제어 정보를 태그 어레이(210)와 주 저장 어레이(220)로 전달하는 신호 경로와, 그리고 주 저장 어레이(220)로부터 데이터를 전달하는 신호 경로를 포함한다. 버스(W)는 어드레스 및 제어 정보를 태그 어레이(210)와 주 저장 어레이(220)로 전달하는 신호 경로와, 그리고 주 저장 어레이(220)로 데이터를 전달하는 신호 경로를 포함한다. 버스(U)는 어드레스 및 제어 정보를 태그 어레이(210)로 전달하는 신호 경로를 포함한다.

[0034] 일 실시예에서, 캐시 업데이트는 예를 들면 태그 유닛(210A)과 같은 소정의 태그 유닛의 태그 요소 내의 정보를 수정한다. 상기 정보는 특정 캐시 라인의 상태 변경을 포함한다. 예를 들면, 일 실시예에서, 메모리 서브시스템은 수정된, 소유된, 배타적인, 분배된, 무효의(modified, owned, exclusive, shared, invalid)(MOESI) 일관성 프로토콜을 이용한다. 이러한 실시예에서, 요청자는 대응하는 주 저장 어레이(220) 위치에 저장된 데이터를 변경함이 없이 태그 어레이(210)에 새로운 캐시 라인 상태를 쓰기 할 것이다. 태그 어레이(210)에서 히트하는 캐시 라인 읽기 요청은 대응하는 주 저장 어레이(220) 위치에 저장된 데이터가 읽기 버스(R)로 전달되도록 한다. 유사하게, 태그 어레이(210)에서 히트하는 캐시 라인 쓰기 요청은 쓰기 데이터가 쓰기 버스(W)로 전달되도록 하고 주 저장 어레이(220)에 쓰기되도록 한다.

[0035] 상술된 바와 같이, 각 태그 유닛들(210A-N) 및 각 저장 블록들(220A-N)은 독립적으로 액세스 가능하다. 게다가, 캐시 버스들(240)의 버스들(R,W 및 U)은 각 태그 유닛(210A-N)에 연결되고, 그리고 버스들(R 및 W)은 각 저장 블록(220A-N)들에 연결되기 때문에, 캐시 요청(예컨대, 읽기, 쓰기 혹은 업데이트)의 유형에 따라 태그 유닛들(210A-N) 및 저장 블록들(220A-N)은 동시에 액세스 가능하다(예컨대 동일 사이클 동안에). 따라서 도시된 실시예에서, 서로 다른 태그 유닛들에 대한 세 개의 독립적인 액세스 요청들은 동시에 서비스될 수 있다. 예를 들면, 태그 유닛(210B)에 대한 읽기 요청은 태그 유닛(210A)에 대한 쓰기 요청과 동시에 서비스될 수 있고, 상기 쓰기 요청은 태그 유닛(210C)에 대한 업데이트 요청과 동시에 서비스될 수 있다. 태그 어레이(210) 및 주 저장 어레이(220)에 연결된 더 많은 버스들을 구비한 실시예들에서는 더 많은 요청이 동시에 서비스된다.

[0036] 각 태그 유닛들(210A-N)은 유일한 어드레스 값이 할당된다. 유일한 어드레스 값은 어드레스의 일부분에 대응하고 그리고 어드레스의 블록 비트 부분으로 불린다. 들어오는 요청이 소정의 태그 유닛에 할당된 유일한 값과 매칭하는 블록 비트들을 갖는 어드레스 값을 포함하면, 상기 요청은 상기 태그 유닛에 대한 것이다. 일 실시예에서, 각 태그 유닛들(210A-N)은 캐시 버스들(240)의 버스들(R,W 및 U)을 모니터한다. 만약 복수의 요청들(예컨대, R,W,U)이 소정의 사이클에서 캐시 버스들(240)의 버스들(R,W 및 U) 상에 제공되고 그리고 2 개 이상의 요청들이 동일한 태그 유닛에 대한 것이라면, 상기 태그 유닛은 동일 사이클 내의 요청들 중 하나를 선택할 것이다. 상기 태그 유닛에 대한 다른 요청들은 동일 사이클 내에서는 거절된다. 그러나, 상기 거절된 대립(conflicting) 요청들은 후속 사이클에서 선택된다. 일 실시예에서, 태그 선택 로직(도시되지 않음)은 업데이트를 최우선 순위로 하고 그 다음 읽기, 쓰기 순서로 요청들을 선택한다. 복수의 요청들이 소정의 사이클에서 캐시 버스들(240)의 버스들(R,W 및 U)에 존재하고 그리고 2 개 이상의 요청들이 동일 태그 유닛에 대한 것일 때, 대안적인 실시예들은 읽기, 쓰기 및 업데이트에 대해 상기와 다른 우선순위를 사용한다.

[0037] 상술한 바와 같이, 캐시 제어(230)는 들어오는 캐시 요청들을 저장하는 하나 이상의 요청 대기열들(231)을 포함한다. 일 실시예에서, 캐시 제어(230)는 로직(도시되지 않음)을 포함하며, 여기서 상기 로직은 요청 대기열들(231)로부터 3 개의 요청들(예컨대, 읽기, 쓰기 및 업데이트)을 선택하고 그리고 상기 선택된 요청들이 각 캐시 액세스 사이클 동안에 캐시 버스들(240)의 각 버스들(R,W 및 U)에 의해 전달되도록 한다. 캐시 제어(230)는 하나의 읽기 및 하나의 쓰기 요청을 그들이 동일 태그 유닛에서 대립할 것인지를 체크하지 않고 선택한다. 이러한 실시예에서, 만약 요청들이 대립하면, 상기 태그 유닛은 상술한 바와 같이 상기 요청들을 선택하고 거절한다. 다른 실시예에서, 캐시 제어(230)는 선택된 읽기 요청 어드레스 태그에 기초하여 쓰기 요청을 선택하여 소정의 태그 유닛에서 대립이 없도록 한다. 만약 업데이트 요청이 제공되면, 태그 유닛 대립을 체크함이 없이 선택될 것이다. 다른 실시예에서, 캐시 제어(230)는 읽기 및 쓰기 요청들을 선택하여 만약 대립하지 않는 읽기 및 쓰기 요청 쌍이 존재하면, 상기 쌍은 언제나 선택된다. 예를 들면, 만약 읽기 및 쓰기 요청 모두가 동일한 블록 비트들을 구비하면, 캐시 제어(230)는 서로 다른 사이클 동안에 두 개의 대립 요청들을 전달하여, 전달된 요청들 간에 대립을 제거한다. 게다가, 일 실시예에서는 읽기 혹은 쓰기 요청의 선택 전에 소정의 저장 블록 혹은 서브-블록이 바쁘지 않은지를 또한 캐시 제어(230)가 확인한다.

[0038] 따라서, 태그 어레이(210)에 대한 캐시 액세스의 유형 및 숫자에 따라, 서로 다른 저장 블록들에 대한 복수의 액세스들이 동시에 서비스될 수 있다. 게다가 도 3과 관련하여 아래서 더욱 상세하게 설명되는 바와 같이, 태그

어레이(210)에 대한 액세스는 완전 파이프라인되어 주 저장 어레이(220)의 임의의 저장 블록(그러나, 서로 다른 서브블록)에 대한 복수의 액세스들은 연속적인 일련의 사이클들에서 서비스된다. 주 저장 어레이(220)의 바쁜 서브-블록에 대한 후속 액세스는 서브-블록이 더 이상 바쁘지 않을 때까지 지연된다.

[0039] 도 3에서, 도 2의 캐시 서브시스템의 저장 블록의 일 실시예에 대한 액세스 시퀀스를 도시하는 다이어그램이 도시된다. 도 2에 도시된 것에 대응하는 요소들은 단순성 및 명확성을 위해 동일 번호로 지칭된다. 저장 블록(220A)은 태그 유닛(210A)에 연결된다. 도 2의 태그 어레이(210)와 같은 태그 어레이에 대한 예시적인 입력 요청 시퀀스가 도시된다. 상기 입력 요청들은 어드레스 A, A+32, A+16, 그 다음 A+48에 대한 읽기 혹은 쓰기이다. 이번 예시에서, 네 개 입력 요청들 모두는 태그 유닛(210A)에 히트한다고 가정한다. 저장 블록(220A)에 도시된 바와 같이, 소정의 어드레스들에 대응하는 각 데이터 세그먼트(segment)들이 서로 다른 서브-블록에 저장된다. 저장 블록(220A)의 세로 열은 서브-블록들(0-3)을 나타내고, 가로 행은 소정의 데이터 세그먼트가 액세스되는 사이클을 도시한다.

[0040] 상술한 바와 같이, 도 2의 어레이(210)는 완전 파이프라인 된다. 예를 들면, 만약 복수의 요청들이 소정의 태그 유닛(그러나, 서로 다른 서브-블록에)에 히트한다면, 각 연속된 요청은 순차적으로 그리고 앞선 요청이 완료되기 전에 처리된다. 그러나, 일련의 액세스들이 동일 서브-블록에 대한 것이면 주 저장 어레이(220)에 대한 액세스들은 파이프라인 될 필요가 없다. 이는 메모리 셀 혹은 셀들 그룹으로부터의 데이터를 쓰기 혹은 읽기 하는데 관련된 지연 때문이다. 따라서, 소정의 셀 혹은 셀들 그룹(예컨대, 서브-블록)에 대한 어떤 요청도 상기 셀 혹은 셀들 그룹이 다시 액세스 될 때까지의 지연 후에 처리될 수 있다.

[0041] 소정의 입력 요청 시퀀스에 대해서, 사이클 0에서 어드레스 'A'에 대응하는 데이터가 리턴(return)되거나 쓰기된다. 사이클 1에서 어드레스 'A+32'에 대응하는 데이터가 리턴되거나 쓰기된다. 사이클 2에서 어드레스 'A+16'에 대응하는 데이터가 리턴되거나 쓰기된다. 사이클 3에서 어드레스 'A+48'에 대응하는 데이터가 리턴되거나 쓰기된다. 따라서, 요청된 어드레스들에 대응하는 데이터는 4개의 서로 다른 서브-블록들에 저장되기 때문에, 상기 액세스들은 대기 사이클의 개입 없이 4 개의 일련의 사이클들에서 리턴된다. 결과 출력 시퀀스들에서 어드레스들 사이의 대기 사이클들은 존재하지 않는다.

[0042] 상술된 바와 같이, 동일한 서브-블록에 저장된 데이터를 갖는 하나의 태그 유닛에 히트하는 일련의 요청들은, 서브-블록의 액세스에 관련한 지연 때문에 대기 사이클의 개입 없는 연속적인 순차적 액세스 사이클로 서비스될 수 없다.

[0043] 도 4에서, 도 1의 마이크로프로세서의 캐시 서브시스템의 다른 실시예의 다이어그램이 도시된다. 도 1 내지 도 3에 도시된 것에 대응하는 요소는 단순성 및 명확성을 위해 동일 번호로 지칭된다. 캐시 서브시스템(400)은 태그 어레이(210)에 연결된 주 저장 어레이(220)를 구비하는 캐시 메모리를 포함한다. 게다가, 캐시 서브시스템(400)은 비동기 읽기 가능(read enable) 신호(451)와 비동기 쓰기 가능(write enable) 신호(455)를 통해 주 저장 어레이(220)에 연결된 캐시 제어(430)를 포함한다. 캐시 제어(430)는 또한 복수의 캐시 액세스 요청 버스들(240)을 통해 태그 어레이(210)에 연결된다. 캐시 제어(430) 및 태그 어레이(210) 각각은 마이크로프로세서 클럭 트리에 의해 공급되는 CPU 클럭 신호(415)를 수신한다. 일 실시예에서, 주 저장 어레이(220) 및 태그 어레이(210)는 예컨대 도 1의 L2 캐시(130)와 같은 L2 캐시 서브시스템에서 사용된다. 그러나, 다른 실시예들에서, 주 저장 어레이(220) 및 태그 어레이(210)는 임의의 캐시 서브시스템에서 사용된다.

[0044] 주 저장 어레이(220)는 독립적으로 액세스 가능한 저장 블록들로 배열되는 복수의 메모리 셀들을 포함한다. 도시된 실시예에서, 저장 블록들은 220A-220N으로 표시되고, 여기서 N은 블록들의 임의의 개수이다. 일 실시예에서는 8개의 저장 블록들이 존재하는 반면, 다른 실시예들에서는 이와 다른 수의 블록들이 존재할 수 있다. 게다가, 각 저장 블록들(220A-N)은 서브 블록들(0-3)로 표시된 독립적으로 액세스 가능한 네 개의 저장 서브-블록들을 포함한다. 각각의 저장 블록(220A-N)은 네 개의 서브-블록들을 포함하고 있지만, 다른 실시예들에서는 각 저장 블록들(220A-N)은 다른 수의 서브-블록들을 포함할 수 있음을 인식해야 한다. CPU 혹은 시스템 클럭을 사용하여 동기식으로 액세스되는 일부 캐시 어레이들에 대조적으로, 도시된 실시예에서는 주 저장 어레이(220)로 라우팅되는 클럭 공급 네트워크가 존재하지 않는다. 주 저장 어레이(220)는 태그 유닛(210) 및 마이크로프로세서(100) 내의 다른 로직에 관해서 비동기식으로 액세스 된다. 도 5 및 도 6의 도면과 관련하여 아래서 더욱 자세하게 설명하는 바와 같이, 독립적으로 액세스 가능한 저장 서브-블록(0-3) 각각은 어써트된 읽기 가능 신호(asserted read enable signal)(451)의 수신에 응답하여 출력 데이터를 제공한다. 또한, 어써트된(asserted) 쓰기 가능 신호(455)의 수신에 응답하여 소정의 서브-블록에 데이터가 쓰기된다. 주 저장 어레이(220)로 라우팅되는 클럭 공급 네트워크가 존재하지 않기 때문에, 각 사이클 동안에 캐시 어레이에서 사용되지 않는 메모리 유

닛의 클럭에 관련된 불필요한 전력 소모가 없다.

[0045] 태그 어레이(210)는 캐시 라인 태그 정보를 저장하도록 구성된 저장 어레이이다. 태그 어레이(210)는 210A-210N으로 표시된 복수의 태그 유닛들로 배열되며, 여기서 N은 태그 유닛들의 임의의 개수이다. 각 태그 유닛(210A-N)은 저장 블록에 대응하고 복수의 태그들 혹은 '태그 요소'들을 포함한다. 예를 들면, 도시된 실시예에서, 태그 유닛(210A)은 저장 블록(220A)에 대응한다. 게다가, 소정의 태그 유닛 내의 각 태그 요소는 소정의 저장 블록 내의 모든 저장 서브-블록들에 대응한다. 따라서, 캐시 라인은 소정의 저장 블록 내에 저장되어 캐시라인은 모든 저장 서브-블록들에 존재한다. 예를 들면, 도시된 실시예에서, 캐시 라인(225)은 저장 블록(220A)의 저장 서브-블록들(0-3)들 전반에 저장된다. 도시된 실시예에서, 태그 어레이(210)는 CPU 클럭(415)을 통해 동기식으로 액세스 된다.

[0046] 캐시 제어(430)는 다양한 소스로부터 캐시 액세스 요청들을 수신하도록 구성된다. 일 실시예에서, 캐시 제어(430) 및 태그 어레이(210)는 CPU 클럭 신호(415)에 따라 마이크로프로세서의 나머지 부분과 동기식으로 연산한다. 그러므로, 캐시 제어(430) 및 태그 어레이(210)는 CPU 클럭 신호(415)를 수신하도록 구성된다.

[0047] 상기 도시된 실시예에서, 캐시 제어(430)는 읽기 가능 신호(451) 및 쓰기 가능 신호(455)를 생성하도록 구성되며, 상기 신호들은 주 저장 어레이(220)의 각 저장 블록 내의 각 저장 서브 블록들(0-3)로 라우팅된다.

[0048] 일 실시예에서, 캐시 제어(430)는 태그 어레이(210) 내의 특정 어드레스에 대한 캐시 히트 혹은 캐시 삽입 요청에 기초하여 어써트된(asserted) 읽기 가능 신호(451) 및 어써트된(asserted) 쓰기 가능 신호(455)를 제공하도록 구성된다. 예를 들면, 소정의 읽기 요청은 태그 유닛(210A)에서 히트할 것이다. 태그 유닛(210A)은 캐시 제어(430)에 히트 표시(도시되지 않음)를 제공한다. 캐시 제어(430)는 읽기 가능 신호(451)를 출력한다. 읽기 가능 신호(451) 및 태그 유닛(210A)으로부터의 어드레스 정보에 응답하여, 대응하는 서브-블록은 소정의 지연 후에 요청된 데이터를 출력한다. 유사하게, 만약 태그 유닛(210A)에서의 히트하는 쓰기 요청이 수신되면, 태그 유닛(210A)은 캐시 제어(430)로 표시(indication)(도시되지 않음)를 제공할 것이다. 캐시 제어(430)는 쓰기 가능 신호(455)를 출력한다. 쓰기 가능 신호(455) 및 태그 유닛(210A)으로부터의 어드레스 정보에 응답하여, 대응하는 서브-블록은 상기 데이터를 가지고 쓰기 될 것이다. 일 실시예에서, 읽기 및 쓰기 가능 신호들은 소정의 지속시간을 갖는 펄스들이다.

[0049] 게다가, 상기 캐시 제어(430)는 프로그램가능하다. 프로그램가능 특징은 각 연속적인 읽기 가능 신호(451) 사이 및 각 연속적인 쓰기 가능 신호(455) 사이에서 펄스 반복 시간들(pulse repetition times)이 가변될 수 있도록 한다. 그러므로, 연속적인 펄스들 사이의 시간을 프로그램할 수 있는 능력은 프로세스 변경 및 서로 다른 캐시 어레이 크기를 갖는 캐시 제어 회로들의 재사용을 가능하게 한다. 캐시 제어(430)를 프로그램하는 다양한 방법들이 존재한다. 예를 들면, 일 실시예에서, 프로그램가능 레지스터들(registers)이 사용될 수 있다. 다른 실시예에서, 다양한 펄스 지연 시간 선택은, 와이어 점퍼(wire jumper)를 이용하여 다이 상에 배선접속(hard wired)될 수도 있다. 제조되는 동안에, 상기 와이어 점퍼 연결들은 필요한 지연을 제공하도록 레이저 식각된다.

[0050] 대안적인 실시예들은 태그 어레이(210) 내의 로직(도시되지 않음)이 읽기 가능 신호(451) 및 쓰기 가능 신호(455)를 생성하도록 한다.

[0051] 도 5에서, 도 2 및 도 4의 캐시 서브시스템의 일 실시예의 연산을 설명하는 타이밍 다이어그램이 도시된다. 이 타이밍 다이어그램은 8개의 신호를 포함하며, 위에서부터 마지막까지 다음과 같다: CPU 클럭, 어드레스 읽기 0A, 읽기 가능 펄스 0A, 데이터 출력 0A, 데이터 출력 래치, 어드레스 쓰기 1B, 데이터 입력 1B, 및 쓰기 가능 펄스 1B. 도 1 내지 도 4에서, CPU 클럭 신호는 캐시 서브시스템에 대한 기준 클럭이고 도 4에서 CPU 클럭(415)으로 도시된다. 상술한 바와 같이, CPU 클럭 신호는 태그 유닛(210)에 제공되고 그리고 주 어레이(220)에 통하는 공급 네트워크와 같이 라우팅되지 않는다.

[0052] 타이밍 다이어그램은, 동시 읽기와 동시에 수신된 동시 쓰기 연산(simultaneous read and a concurrently received simultaneous write operation)의 일 실시예를 도시한다. 상술한 바와 같이, 태그 어레이(210)는 완전 파이프라인되고, 그리고 주 저장 어레이(220)는 액세스들이 서로 다른 서브-블록에 대한 것일 때 파이프라인된다. 그러나, 만약 동시 읽기 혹은 쓰기 요청들이 동일 서브-블록에 대한 소정의 태그 유닛에 의해 수신된다면, 캐시 제어(430)는 정확한 시간에 읽기 및 쓰기 펄스들을 출력하도록 프로그램된다. 또한, 주 어레이(220)의 서로 다른 저장 블록들(220A-N)에 대한 액세스 요청들은 동시에 서비스된다. 다음의 예시에서, 저장 블록(220A)의 동일 서브-블록(0A)에서의 어드레스에 대한 두 개의 읽기들 및 저장 블록(220B)의 동일 서브-블록(1B)에서의 어드레스에 대한 두 개의 쓰기들이 수신된다. 그러므로, 상기 읽기들은 상기 쓰기들과 서로 다른 태

그 유닛에 대한 것이다. 도 5에서 도시된 특정 타이밍은 설명을 목적으로 하고 다른 실시예들은 다른 타이밍을 포함할 수 있다.

[0053] 타이밍 마크(t0)에서, 어드레스(0A1)는 저장 블록(220A)(주 저장 어레이(220)의 서브-블록(0))에 제공된다. 동일 CPU 클럭 펄스의 하강 에지(falling edge)에서, 읽기 가능 펄스(R1)가 1 클럭 사이클 동안 어써트된다(asserted). 이러한 제 1 읽기 가능 펄스에 응답하여, 저장 블록(220A)(서브-블록(0))의 메모리 셀들의 블록은 약간의 지연 시간 후에 데이터(DOA1)를 출력한다. 이번 예시에서, 상기 지연은 R1이 어써트된(asserted) 후 5 개의 CPU 클럭 사이클들과 동일하다. 따라서, 타이밍 마크(t1)에서, 데이터 출력 래치 신호가 래치 데이터(DOA1)에 어써트될 수 있다. 또한, 타이밍 마크(t0)에서, 어드레스(1B 1) 및 데이터 입력(DI1)은 저장 블록(220B)(주 저장 어레이(220)의 서브-블록(1))에 제공된다. 동일한 CPU 클럭 펄스의 하강 에지에서, 쓰기 가능 펄스(W1)가 1 클럭 사이클 동안 어써트된다. 이러한 제 1 쓰기 가능 펄스에 응답하여, 데이터 입력(DI1)은 저장 블록(220B)(서브-블록(1)) 내에 있는 메모리 셀들의 블록으로 쓰기된다.

[0054] 게다가, 타이밍 마크(t1)에서, 제 2 어드레스(0A 2)는 저장 블록(220A)(주 저장 어레이(220)의 서브-블록(0))에 제공된다. t1 후의 CPU 클럭 펄스의 다음 하강 에지에서, 제 2 읽기 가능 펄스(R2)가 1 클럭 사이클 동안 어써트된다. 이러한 제 2 읽기 가능 펄스에 응답하여, 저장 블록(220A)(서브-블록(0))의 메모리 셀들의 블록은 5 개의 CPU 클럭 사이클과 동일한 지연 시간 후에 데이터(DOA2)를 출력한다. 타이밍 마크(t2)에서, 제 2 데이터 출력 래치 신호가 래치 데이터에 어써트된다.

[0055] 타이밍 마크(t1)에서, 어드레스(1B 2) 및 데이터 입력(DI2)은 저장 블록(220B)(주 저장 어레이(220)의 서브-블록(1))에 제공된다. t1 후의 CPU 클럭 펄스의 다음 하강 에지에서, 제 2 쓰기 가능 펄스(W2)가 1 클럭 사이클 동안 어써트된다. 이러한 제 2 쓰기 가능 펄스에 응답하여, 저장 블록(220B)(서브-블록(1)) 내에 있는 메모리 셀들의 블록은 DI2를 가지고 쓰기된다.

[0056] 상술한 바와 같이, 동일한 서브-블록에 대한 읽기 사이클들 및 쓰기 사이클들은 대기 사이클들을 야기하지만, 서로 다른 저장 블록들에 대한 액세스 사이클들은 동시에 혹은 연속적인 일련의 사이클들에서 서비스된다. 이러한 대기 사이클들을 수용하기 위해, 연속적인 읽기 펄스들 사이의 지연이 프로그램된다. 예를 들면, 상기 예시에서, 캐시 제어(430)는 동일한 서브-블록에 대한 읽기 가능 펄스들 및 쓰기 가능 펄스들을 매 6 CPU 클럭 사이클보다 느리게 출력하도록 프로그램된다. 일 실시예에서, 어드레스 및 데이터 신호들은 유효한 데이터가 읽기 및 쓰기 되도록 충분히 오랫동안 유지되어야 한다.

[0057] 도 6에서, 도 2 및 도 4의 캐시 서브시스템들의 파이프라인 연산을 설명하는 타이밍 다이어그램이 도시된다. 이 타이밍 다이어그램은 위에서부터 바닥까지 다음의 10 개의 신호를 포함한다: CPU 클럭, 서브-블록(0A) 어드레스, 읽기 가능 펄스(0A), 데이터 출력(0A), 서브-블록(2B) 어드레스, 읽기 가능 펄스(2B), 데이터 출력(2B), 서브-블록(3B) 어드레서, 읽기 가능 펄스(3B) 및 데이터 출력(3B). 도 1 내지 도 4에서, CPU 클럭 신호는 캐시 서브시스템에 대한 기준 클럭이고 도 4에서 CPU 클럭(415)으로 도시된다. 상술한 바와 같이, CPU 클럭 신호는 태그 유닛(210)에 제공되고 그리고 주 어레이(220)에 통하는 공급 네트워크와 같이 라우팅되지 않는다.

[0058] 상기 타이밍 다이어그램은 주 저장 어레이(220)의 서로 다른 3 개의 서브-블록들에 대한 3 개의 백투백(back-to-back) 읽기 연산의 예시를 도시한다. 상술한 바와 같이, 태그 어레이(210)는 완전 파이프라인된다. 예를 들면, 복수의 백투백 읽기 혹은 쓰기 요청들이 하나 이상의 서로 다른 태그 유닛에 의해 수신되면, 캐시 서브시스템은 각각의 연속적인 클럭 사이클 동안 서로 다른 서브-블록들에 읽기 데이터 혹은 쓰기 데이터를 출력하여, 거의 파이프라인된 특성을 나타낸다. 연속적인 사이클에서 동일 태그 유닛 및 동일 서브-블록에 대한 백투백 요청들이 수신되면, 캐시 서브시스템은 파이프라인 특성을 나타내지 않는다. 다음의 예시에서, 저장 블록(220A)의 서브-블록(0A), 저장 블록(220B)의 서브-블록들 (2B 및 3B)의 어드레스들 각각에 두 개의 읽기들이 백투백 수신된다. 도 6에서 도시된 특정 타이밍은 설명을 목적으로 하며 다른 실시예들은 다른 타이밍을 포함할 수 있다.

[0059] 타이밍 마크(t0)에서, 어드레스(0A 1)는 주 저장 어레이(220)의 서브-블록(0A)에 제공된다. 제 1 CPU 클럭 펄스의 하강 에지에서, 읽기 가능 펄스(ROA 1)는 1 클럭 사이클 동안 어써트된다. 태그 어레이 파이프라인은 추가적인 읽기 요청들이 로드(load)되기 때문에, 타이밍 마크(t1)에서 어드레스(2B 1)가 주 저장 어레이(220)의 서브-블록(2B)에 제공된다. t1후에 CPU 클럭의 다음 하강 에지에서, 읽기 가능 펄스(R2B 1)는 1 클럭 사이클 동안 어써트된다. 게다가, 타이밍 마크(t2)에서, 어드레스(2B 1)는 주 저장 어레이(220)의 서브-블록(2B)에 제공되고, 그리고 t2 후에 CPU 클럭의 다음 하강 에지에서 읽기 가능 펄스(R3B 1)가 1 클럭 사이클 동안에 어써트된다. 그러므로, 3 개의 서로 다른 서브-블록들에 대해서 3 개의 읽기 요청들이 발행되었다.

[0060] 읽기 가능 펄스(ROA 1)에 응답하여, 저장 서브-블록(0A)은 일정 시간 지연 후에 데이터(DOA 1)를 출력한다. 이번 예시에서, ROA 1 후에 5 개의 CPU 클럭 사이클들이 시간 지연으로 인가된다. 읽기 가능 펄스(R2B 1)에 응답하여, 저장 서브-블록(2B)은 일정 시간 지연 후에 데이터(D2B 1)를 출력한다. 이번 예시에서, R2B 1 후에 5 개의 CPU 클럭 사이클들이 시간 지연으로 인가된다. 타이밍 마크(t4)에서, 제 2 어드레스가 주 저장 어레이(220)의 서브-블록(0A)에 제공된다. 동일한 CPU 클럭 펄스의 하강 에지에서, 읽기 가능 펄스(ROA 2)는 1 클럭 사이클 동안 어써트된다. 읽기 가능 펄스(R3B 1)에 응답하여, 저장 서브-블록(3B)은 일정 시간 지연 후에 데이터(D3B 1)를 출력한다. 이번 예시에서, R3B 1 후에 5 개의 CPU 클럭 사이클들이 시간 지연으로 인가된다. 타이밍 마크(t5)에서, 제 2 어드레스가 주 저장 어레이(220)의 서브-블록(2B)에 제공된다. 동일한 CPU 클럭 펄스의 하강 에지에서, 읽기 가능 펄스(R2B 2)는 1 클럭 사이클 동안 어써트된다. 타이밍 마크(t6)에서, 제 2 어드레스는 주 저장 어레이(220)의 서브-블록(3B)에 제공된다. 동일한 CPU 클럭 펄스의 하강 에지에서, 읽기 가능 펄스(R3B 2)는 1 클럭 사이클 동안 어써트된다. 따라서, 타이밍 마크(t4,t5 및 t6)에서 3 개의 독립적인 서브-블록들(0A,2B 및 3B) 각각에 대한 제 1 세 개의 읽기 연산들로부터의 데이터가 순차적으로 이용가능하다.

[0061] 읽기 가능 펄스(ROA 2)에 응답하여, 저장 서브-블록(0A)은 일정 시간 지연 후에 데이터(DOA 2)를 출력한다. 이번 예시에서, ROA 2 후에 5 개의 CPU 클럭 사이클들이 시간 지연으로 인가된다. 읽기 가능 펄스(R2B 2)에 응답하여, 저장 서브-블록(2B)은 일정 시간 지연 후에 데이터(D2B 2)를 출력한다. 이번 예시에서, R2B 2 후에 5 개의 CPU 클럭 사이클들이 시간 지연으로 인가된다. 읽기 가능 펄스(R3B 2)에 응답하여, 저장 서브-블록(3B)은 일정 시간 지연 후에 데이터(D3B 2)를 출력한다. 이번 예시에서, R3B 2 후에 5 개의 CPU 클럭 사이클들이 시간 지연으로 인가된다. 그러므로, 타이밍 마크(t7,t8 및 t9)에서, 동일한 3 개의 독립적인 서브-블록들(0A,2B 및 3B) 각각에 대한 제 2 세 개의 읽기 연산으로부터의 데이터가 순차적으로 이용가능하다. 각 서브-블록으로부터 출력 데이터는 래칭 신호(도시되지 않음)에 의해 래칭될 것이다.

[0062] 도 5 및 도 6과 관련하여 설명된 읽기 및 쓰기 연산이 분리되어 설명되었지만, 상기 설명된 연산들은 동시에 수행될 수 있다. 예를 들면, 일련의 읽기들 및 일련의 쓰기들은 대립하지 않는 블록들에 동시에 파이프라인 될 수 있다.

[0063] 도 7에서, 도 1의 마이크로프로세서를 포함하는 컴퓨터 시스템의 일 실시예의 블록 다이어그램이 도시된다. 도 1 내지 도 4에 도시된 것과 대응하는 요소들은 명확성 및 단순성을 위해 동일 번호로 지칭된다. 컴퓨터 시스템(700)은 메모리 버스(715)를 통해 시스템 메모리(710)로 연결된 마이크로프로세서(100)를 포함한다. 마이크로프로세서(100)는 시스템 버스(725)를 통해 I/O 노드(720)에 또한 연결된다. I/O 노드(720)는 그래픽 버스(735)를 통해 그래픽 어댑터(730)에 연결된다. I/O 노드(720)는 주변장치 버스를 통해 주변장치 디바이스(740)에 또한 연결된다.

[0064] 도시된 실시예에서, 마이크로프로세서(100)는 메모리 버스(715)를 통해 직접 시스템 메모리(710)에 연결된다. 그러므로, 마이크로프로세서는 시스템 메모리(710)로의 액세스들을 제어하도록 메모리 인터페이스(도 7에 도시되지 않음)를 포함한다. 그러나, 다른 실시예에서, 시스템 메모리(710)는 I/O 노드(720)를 통해 마이크로프로세서(100)에 연결된다. 이러한 실시예에서, I/O 노드(720)는 메모리 인터페이스(도시되지 않음)를 포함한다. 또한 일 실시예에서, 마이크로프로세서(100)는 도 2의 캐시 서브시스템(200)과 같은 캐시 서브시스템을 포함한다. 다른 실시예에서, 마이크로프로세서(100)는 도 4의 캐시 서브시스템(400)과 같은 캐시 서브시스템을 포함한다.

[0065] 시스템 메모리(710)는 임의의 적당한 메모리 디바이스들을 포함한다. 예를 들면, 일 실시예에서, 시스템 메모리는 다이내믹 램(DRAM) 디바이스들의 하나 이상의 뱅크를 포함한다. 다른 실시예들은 다른 메모리 디바이스들 및 구성들을 포함할 수 있다.

[0066] 도시된 실시예에서, I/O 노드(720)는 그래픽 버스(735), 주변장치 버스(740) 및 시스템 버스(725)에 연결된다. 따라서, I/O 노드(720)는 버퍼들 및 다양한 버스들 사이의 프랜잭션 흐름을 관리하는 제어 로직을 포함하는 다양한 버스 인터페이스 로직(도시되지 않음)을 포함한다. 일 실시예에서, 시스템 버스(725)는 HyperTransport™ 기술과 양립 가능한 패킷 기반 상호연결이다. 이러한 실시예에서, I/O 노드(720)는 패킷 트랜잭션을 처리하도록 구성된다. 대안적인 실시예에서, 시스템 버스(725)는 예컨대 전면 버스(FSB)와 같은 일반적인 분할 버스 아키텍처이다.

[0067] 게다가, 그래픽 버스(735)는 가속 그래픽 포트(AGP) 버스 기술과 양립 가능하다. 일 실시예에서, 그래픽 어댑터(730)는 디스플레이를 위한 그래픽 영상을 생성하고 디스플레이하도록 구성된 다양한 그래픽 디바이스들 중 임의의 것이다. 주변장치 버스(745)는 예컨대 주변장치 상호연결(PCI) 버스와 같은 일반적인 주변장치 버스의 예

시이다. 주변장치 디바이스(740)는 예컨대 모뎀 혹은 사운드 카드와 같은 주변장치 디바이스 유형 중 임의의 것이다.

[0068] 상기 실시예들은 매우 자세하게 설명되었지만, 본 명세서를 숙지한 당업자에게 수많은 변경 및 수정들은 명백하다. 첨부된 청구항들은 이러한 모든 변경들 및 수정들을 포함하도록 설명된다.

산업상 이용 가능성

[0069] 본 발명은 일반적으로 마이크로프로세서에 적절하다.

도면의 간단한 설명

[0009] 도 1은 마이크로프로세서의 일 실시예의 블록 다이어그램이다.

[0010] 도 2는 도 1의 마이크로프로세서의 캐시 서브시스템의 일 실시예의 블록 다이어그램이다.

[0011] 도 3은 도 2의 캐시 서브시스템의 저장 블록의 일 실시예에 대한 액세스 시퀀스를 도시하는 다이어그램이다.

[0012] 도 4는 도 1의 마이크로프로세서의 캐시 서브시스템의 다른 실시예의 다이어그램이다.

[0013] 도 5는 도 2 및 도 4의 캐시 서브시스템의 일 실시예의 작동을 도시하는 타이밍 다이어그램이다.

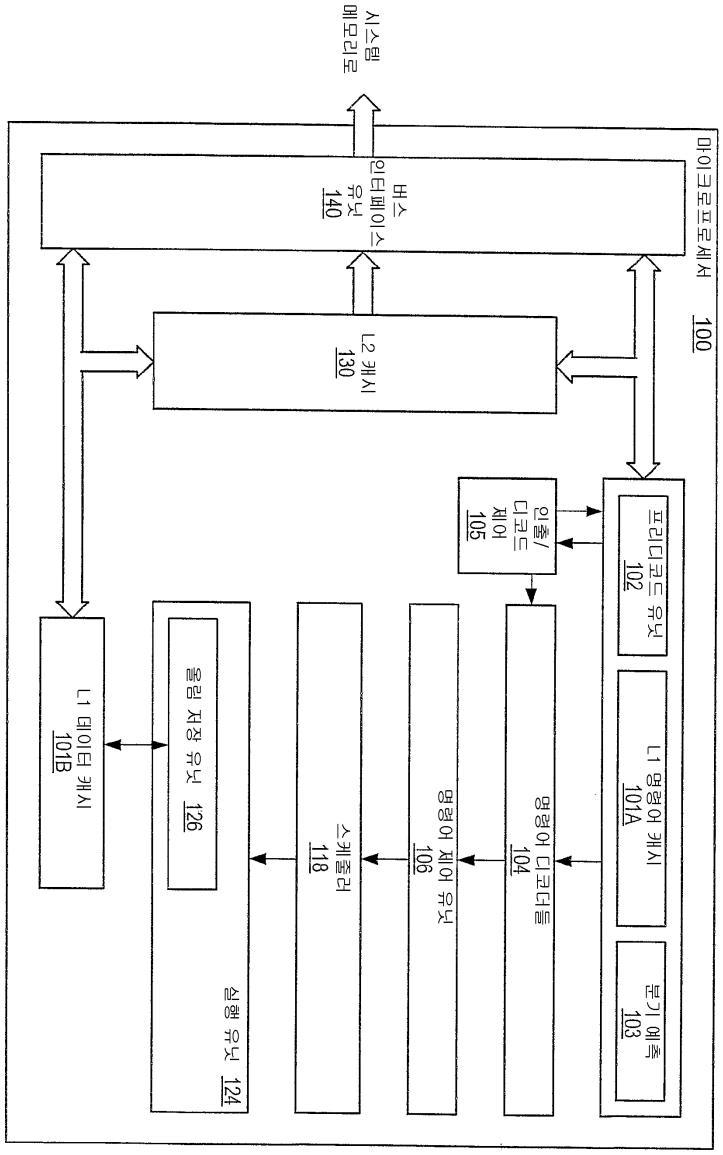
[0014] 도 6은 도 2 및 도 4의 캐시 서브시스템의 파이프라인 동작을 도시하는 타이밍 다이어그램이다.

[0015] 도 7은 도 1의 마이크로프로세서를 포함하는 컴퓨터 시스템의 일 실시예의 블록 다이어그램이다.

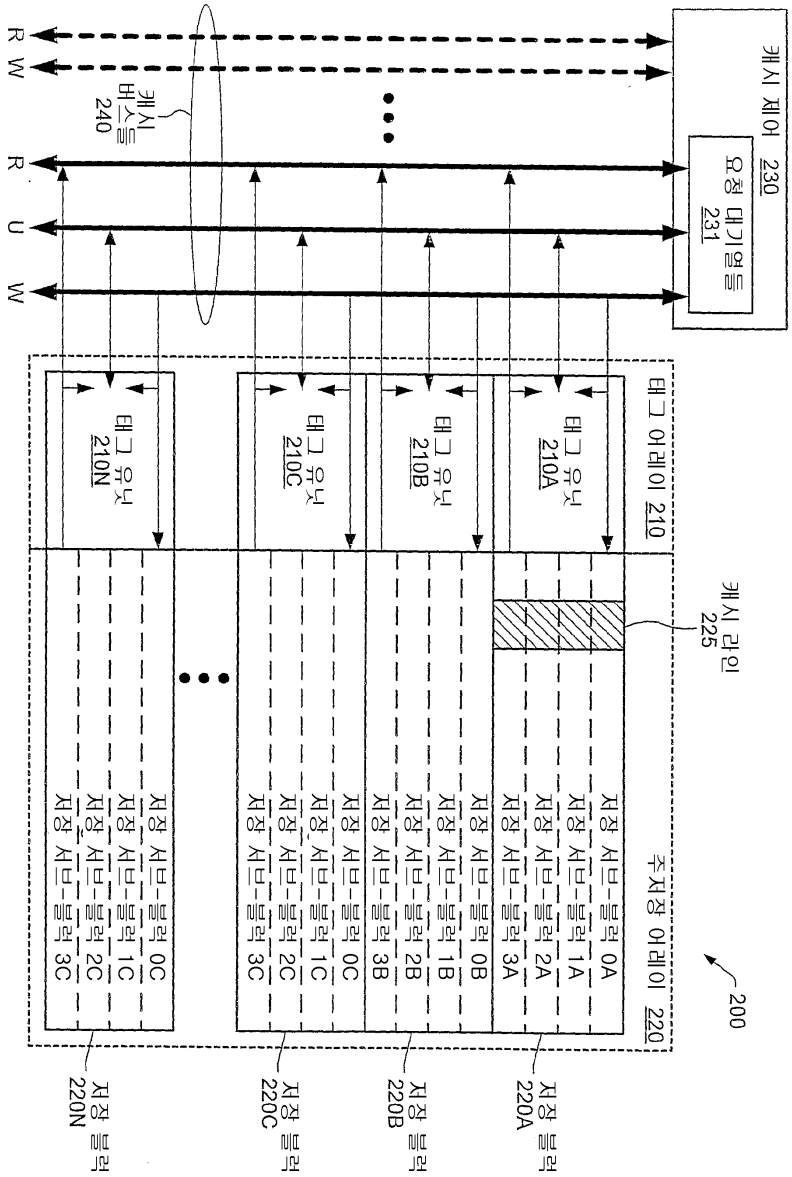
[0016] 본 발명은 다양한 수정들 및 대체 형상들이 가능하지만, 이들의 특정 실시예들이 도면에서 예시로서 도시되고 본 명세서에서 상세하게 설명될 것이다. 그러나 이들에 대한 도면들 및 상세한 설명은 본 발명을 개시된 특정 형태로 제한하기 위함이 아니고, 첨부된 청구항들에 의해 정의된 본 발명의 정신 및 범위 내에 있는 모든 수정물, 균등물 및 대체물들을 포함하도록 의도된다. .

도면

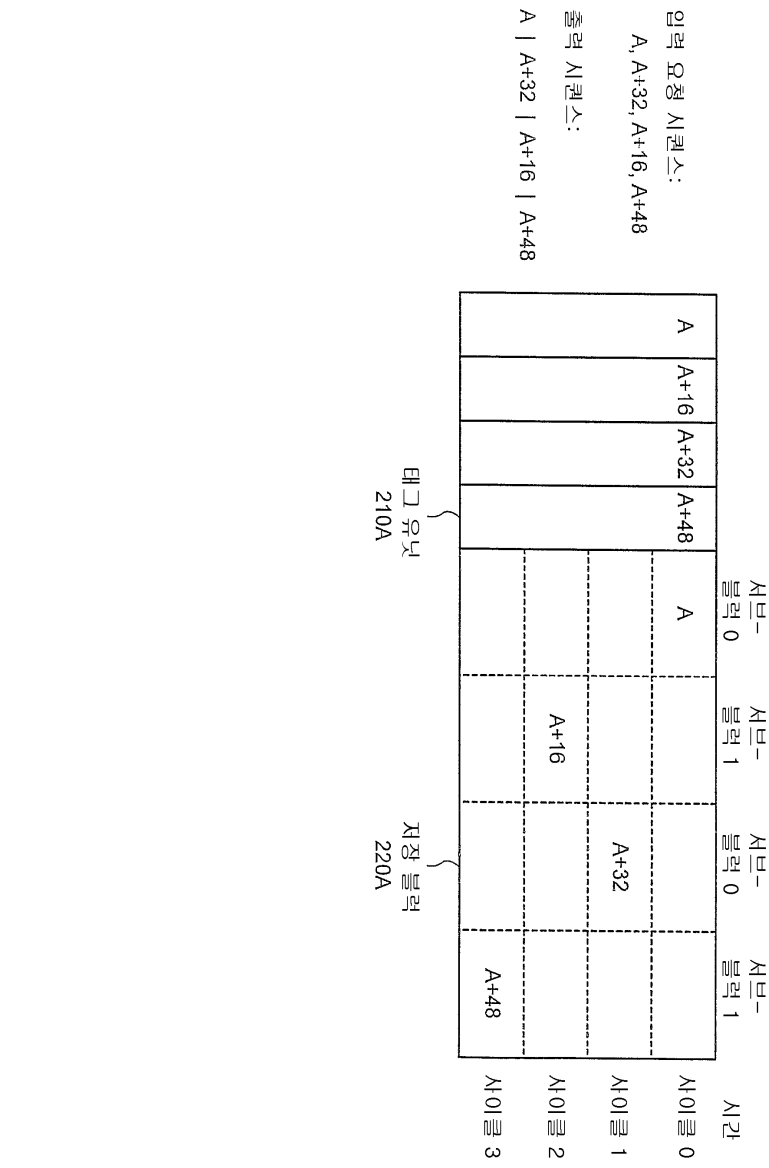
도면1


시스템 메모리로
마이크로프로세서 100
버스 인터페이스 유닛 140
L2 캐시 130
인출/디코드 제어 105
프리디코드 유닛 102
L1 명령어 캐시 101A
분기 예측 103
명령어 디코더들 104
명령어 제어 유닛 106
스케줄러 118
실행 유닛 124
올림 저장 유닛 126
L1 데이터 캐시 101B

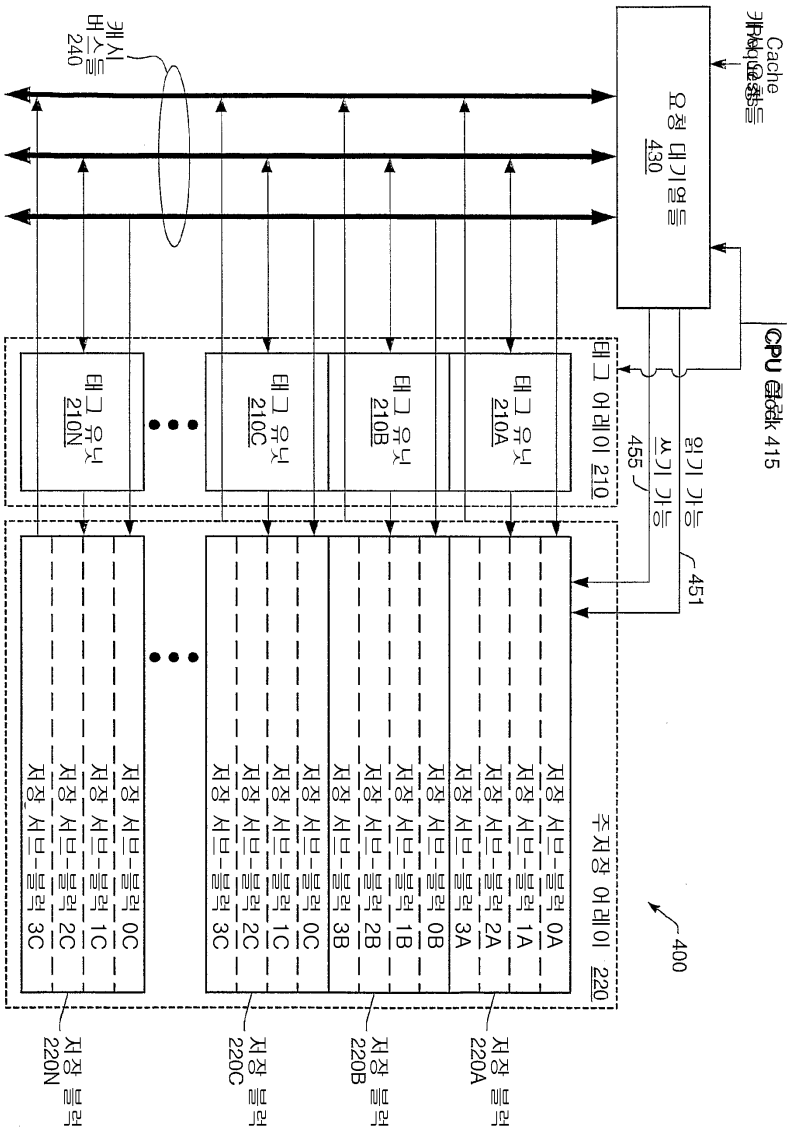
도면2


캐시 제어 230
요청 대기열들 231
R W
R U W
캐시 버스들 240
태그 어레이 210
태그 유닛 210A
태그 유닛 210B
태그 유닛 210C
태그 유닛 210N
캐시 라인 225
주저장 어레이 220
200
저장 서브-블럭 0A
저장 서브-블럭 1A
저장 서브-블럭 2A
저장 서브-블럭 3A
저장 서브-블럭 0B
저장 서브-블럭 1B
저장 서브-블럭 2B
저장 서브-블럭 3B
저장 서브-블럭 0C
저장 서브-블럭 1C
저장 서브-블럭 2C
저장 서브-블럭 3C
저장 블럭 220A
저장 블럭 220B
저장 블럭 220C
저장 블럭 220N

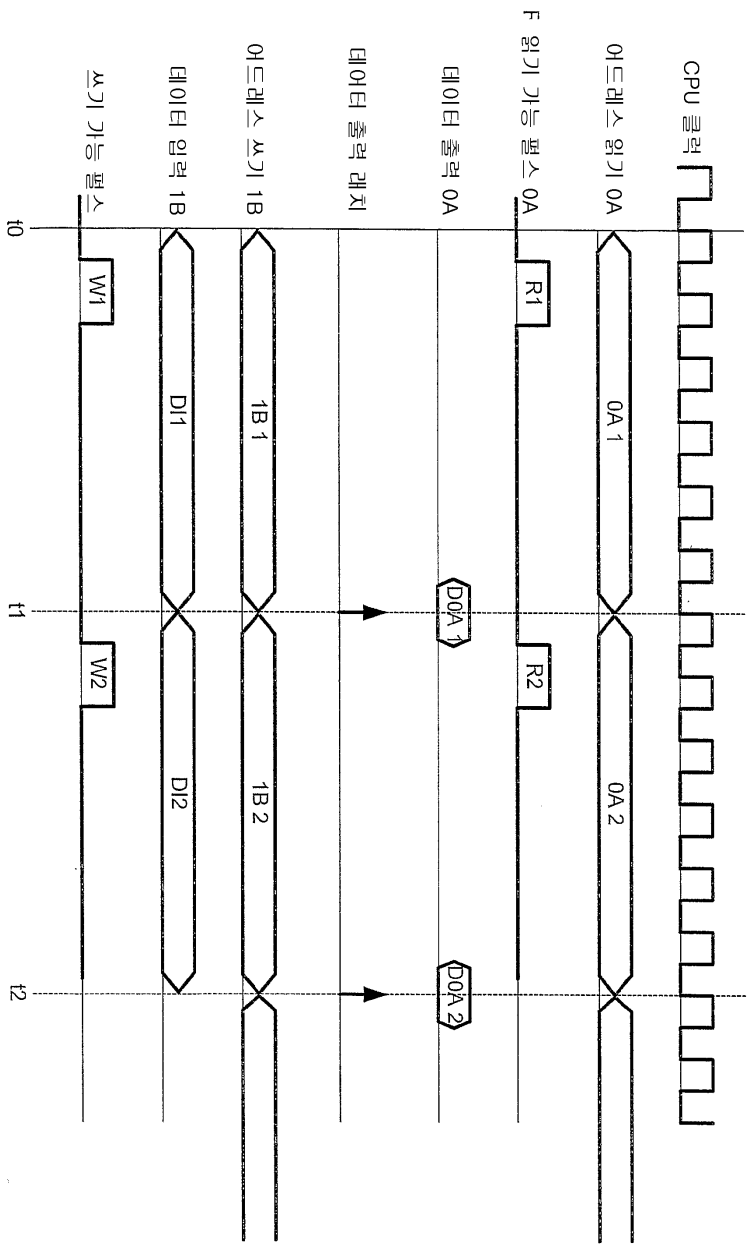
도면3


입력 요청 시퀀스:
A, A+32, A+16, A+48
출력 시퀀스:
A | A+32 | A+16 | A+48
서브-블럭 0
서브-블럭 1
서브-블럭 0
서브-블럭 1
시간
사이클 0
사이클 1
사이클 2
사이클 3
A
A+32
A+16
A+48
A
A+16
A+32
A+48
태그 유닛
210A
저장 블럭
220A

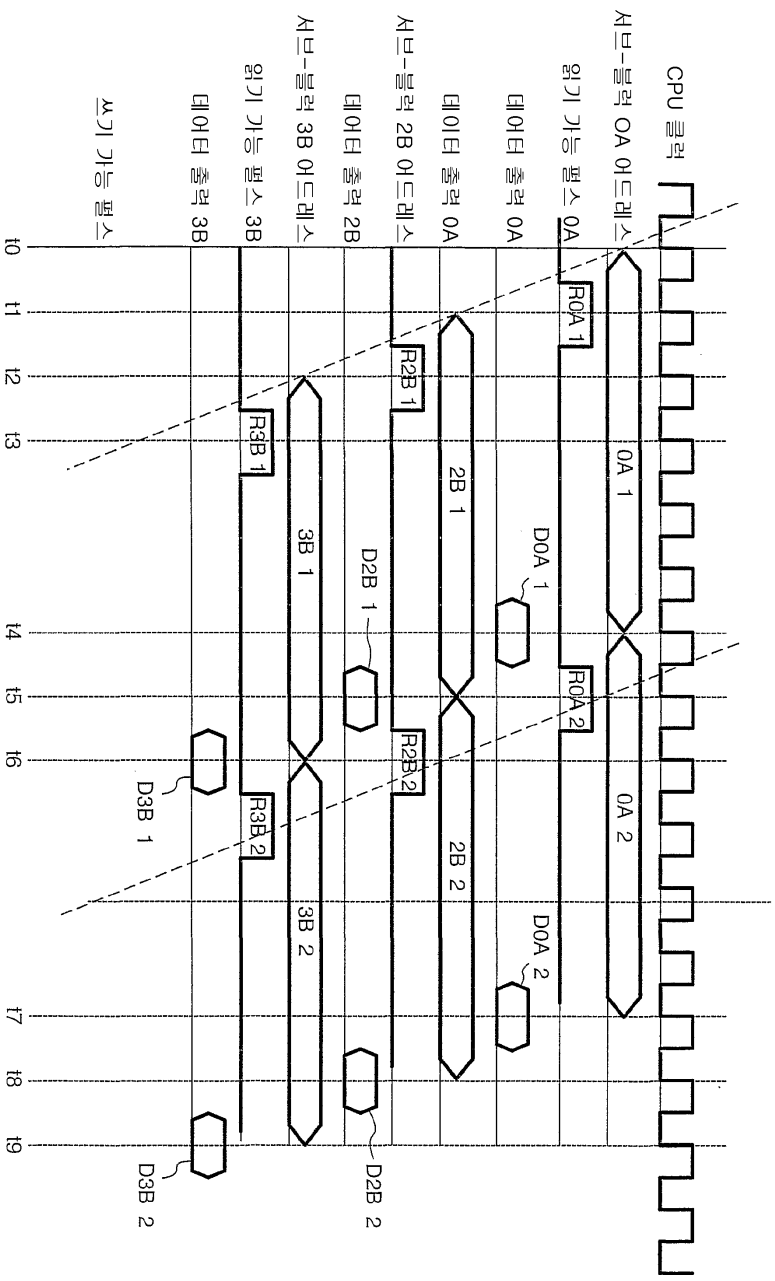
도면4


캐시 버스들 240
Cache 캐시 요청들
요청 대기열들 430
CPU 클럭 415
태그 어레이 210
태그 유닛 210A
태그 유닛 210B
태그 유닛 210C
태그 유닛 210N
읽기 기능
쓰기 기능
451
455
주저장 어레이 220
400
저장_서브-블럭 0A
저장_서브-블럭 1A
저장_서브-블럭 2A
저장_서브-블럭 3A
저장_서브-블럭 0B
저장_서브-블럭 1B
저장_서브-블럭 2B
저장_서브-블럭 3B
저장_서브-블럭 0C
저장_서브-블럭 1C
저장_서브-블럭 2C
저장_서브-블럭 3C
저장 블럭 220A
저장 블럭 220B
저장 블럭 220C
저장 블럭 220N

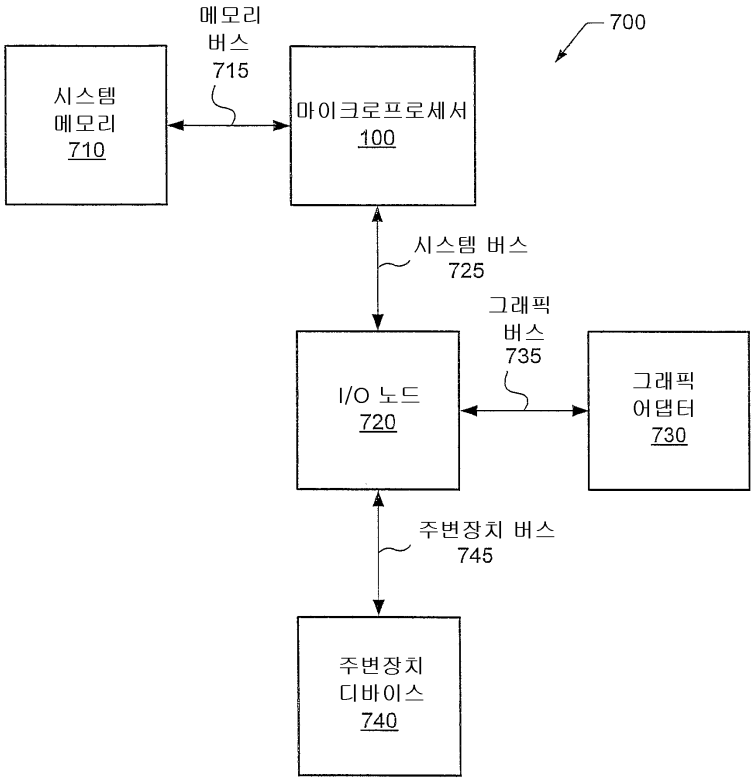
도면5


CPU 클럭
어드레스 읽기 0A
0A 1
0A 2
F 읽기 가능 펄스 0A
R1
R2
데이터 출력 0A
D0A 1
D0A 2
데이터 출력 래치
어드레스 쓰기 1B
1B 1
1B 2
데이터 입력 1B
DI1
DI2
쓰기 가능 펄스
W1
W2
t0
t1
t2

도면6


CPU 클럭
서브-블럭 0A 어드레스
읽기 가능 펄스 0A
데이터 출력 0A
데이터 출력 0A
서브-블럭 2B 어드레스
데이터 출력 2B
서브-블럭 3B 어드레스
읽기 가능 펄스 3B
데이터 출력 3B
쓰기 가능 펄스
t0
t1
t2
t3
t4
t5
t6
t7
t8
t9
R0A 1
0A 1
D0A 1
R0A 2
0A 2
D0A 2
R2B 1
2B 1
D2B 1
R2B 2
2B 2
D2B 2
R3B 1
3B 1
D3B 1
R3B 2
3B 2
D3B 2

도면7


700
메모리
버스
715
시스템
메모리
710
마이크로프로세서
100
시스템 버스
725
그래픽
버스
735
I/O 노드
720
그래픽
어댑터
730
주변장치 버스
745
주변장치
디바이스
740