(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0059106
(43) 공개일자 2008년06월26일

(51) Int. Cl.
G06F 9/06 (2006.01) *G06F 12/00* (2006.01)
(21) 출원번호 10-2007-0136701
(22) 출원일자 2007년12월24일
심사청구일자 2007년12월24일
(30) 우선권주장
11/643,999 2006년12월22일 미국(US)

(71) 출원인
브로드콤 코포레이션
미합중국, 92617 캘리포니아 어빈, 캘리포니아 애비뉴 5300
(72) 발명자
마크, 타우톤
영국, 씨비25 9에프티, 캠브리지, 랜드비치 하이 스트리트 74
(74) 대리인
이수완, 이 성 규, 조진태, 윤종섭, 이재웅

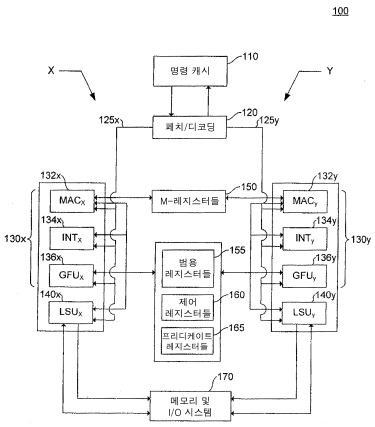
전체 청구항 수 : 총 10 항

(54) 프로세서에서의 마스킹된 저장 동작들을 위한 시스템 및방법

(57) 요 약

프로세서 또는 프로세서-기반 시스템/칩에서 마스킹된 저장 동작들(masked store operations)의 가속된 핸들링을 위한 시스템 및 방법이 설명된다. 프리디케이트 마스크별 아래에서의 저장 동작을 지원하는 명령들의 세트가 제공된다. 본 발명은 ATM 셀들 또는 리드 솔로몬 코드 워드들(Reed Solomon code word)을 다루기 위해 xDSL 모뎀들에 의해 사용될 수 있는 바와 같은 임의의 배열에서 작은 전달들(transfers)의 핸들링을 가속화시킨다.

대표도 - 도1



특허청구의 범위

청구항 1

목적지 메모리에 데이터를 저장하기 위한 방법으로,

마스킹된 저장 명령(masked store instruction)을 프로세서에 발행하는 단계; 및

상기 프로세서에서 상기 마스킹된 저장 명령을 처리하는 단계를 포함하되;

상기 처리 단계는,

상기 명령으로부터 데이터 레지스터, 타겟 주소 및 마스크를 식별하는 단계;

상기 데이터 레지스터의 데이터중에서 어떤 바이트들이 마스킹되어 있지 않은지 식별하는 단계- 여기에서 상기 마스크는 상기 데이터 레지스터의 데이터의 각 바이트가 마스킹되었는지 마스킹되지 않았는지 여부를 가리키는 비트를 포함함;

마스킹되지 않은 데이터 레지스터의 상기 바이트들을 상기 목적지 메모리에 기록하는 단계-여기에서 상기 데이터의 각 바이트는 상기 타겟 주소 및 상기 데이터 레지스터내의 상기 바이트의 상대적인 위치의 합에 상응하는 주소에 기록됨;

를 더 포함하는 데이터 저장 방법.

청구항 2

청구항 1에 있어서, 상기 마스킹된 저장 명령을 상기 프로세서에 발행하는 단계는,

상기 프로세서에 상기 타겟 주소 및 상기 마스크를 구성하는 단계를 포함하는 데이터 저장 방법.

청구항 3

청구항 2에 있어서,

상기 마스크를 프리디케이트 레지스터(predicate register)에 저장하는 단계; 및

상기 프로세서가 상기 프리디케이트 레지스터로부터 상기 마스크를 추출(retrieve)하도록 구성하는 단계를 더 포함하는 데이터 저장 방법.

청구항 4

청구항 3에 있어서,

상기 프리디케이트 레지스터의 위치를 식별하는 단계를 더 포함하는 데이터 저장 방법.

청구항 5

메모리 접속 명령들을 처리하기 위한 집적회로로서,

프로세서;

데이터의 하나 또는 그 이상의 바이트들을 포함하는 데이터 레지스터;

마스크를 홀딩하도록 구성된 프리디케이트 레지스터;

메모리를 포함하며;

상기 프로세서는 상기 데이터 레지스터내에 포함된 데이터를 상기 메모리에 저장하도록 구성되고;

상기 프로세서는 상기 마스크를 읽기 위해 상기 프리디케이트 레지스터를 참고(consult)하고;

상기 프로세서는, 상기 마스크가 데이터의 상기 바이트가 저장되어야 함을 가리키는 경우에만, 상기 데이터 레지스터로부터의 데이터의 하나 또는 그 이상의 바이트들 각각을 상기 메모리에 저장하는 메모리 접속 명령 처리 집적회로.

청구항 6

청구항 5에 있어서,

상기 메모리로의 주소를 포함하는 메모리 포인터를 더 포함하는 메모리 접속 명령 처리 집적회로.

청구항 7

청구항 6에 있어서,

상기 메모리 포인터는 상기 메모리로의 롱워드(long-word) 주소를 포함하는 메모리 접속 명령 처리 집적회로.

청구항 8

청구항 6에서,

상기 메모리 포인터의 상기 주소에 관련된 상기 메모리로의 주소를 포함하는 오프셋 포인터를 더 포함하는 메모리 접속 명령 처리 집적회로.

청구항 9

마스크를 생성하기 위한 방법으로서, 상기 마스크는 바이트들의 시퀀스중에서 어떤 바이트들이 저장되어야 하는지를 정의하되,

세트 마스크 명령(set mask instruction)을 발행하는 단계;

상기 명령으로부터 마스크 결과 위치, 오프셋, 바이트 카운트를 식별하는 단계;

상기 바이트 카운트로부터 상기 오프셋을 감하여 데이터 크기를 결정하는 단계;

상기 데이터 크기가 0 보다 작거나 같으면, 상기 마스크를 모두 0들로 설정하는 단계;

상기 데이터 크기가 비트들에 있어 상기 마스크의 폭보다 크면, 상기 마스크를 모두 1들로 설정하는 단계;

상기 데이터 크기가 0보다 크고, 비트들에 있어 상기 마스크의 폭보다 작거나 같으면, 상기 데이터 크기에 상응하는 상기 마스크의 비트들의 연속된 시퀀스를 1로 설정하고, 상기 마스크의 최하위 비트(least significant bit)로부터 시작하고, 상기 마스크의 잔여 비트들을 0으로 설정하는 단계; 및

상기 마스크를 상기 마스크 결과 위치에 할당하는 단계;를 포함하며,

여기에서, 상기 마스크의 각 비트 값은 타겟 메모리 위치(location)내의 바이트 위치에 해당하는 마스크 생성 방법.

청구항 10

마스크를 생성하기 위한 방법으로서, 상기 마스크는 바이트들의 시퀀스 중에서 어떤 바이트들이 저장되어야 하는지를 정의하되,

세트 마스크 명령을 발행하는 단계;

상기 명령으로부터 마스크 결과 위치, 오프셋, 및 바이트 카운트를 식별하는 단계;

상기 바이트 카운트로부터 상기 오프셋을 감하여 데이터 크기를 결정하는 단계;

상기 데이터 크기가 0보다 작거나 같으면, 상기 마스크를 모두 0들로 설정하는 단계;

상기 데이터 크기가 비트들에 있어 상기 마스크의 폭보다 크면, 상기 마스크를 모두 0들로 설정하는 단계;

상기 데이터 크기가 0보다 크고, 비트들에 있어서 상기 마스크의 폭보다 작거나 같으면, 상기 데이터 크기에 상응하는 상기 마스크의 비트들의 연속된 시퀀스를 0으로 설정하고, 상기 마스크의 최하위 비트로부터 시작하고, 상기 마스크의 잔여 비트들을 1로 설정하는 단계; 및

상기 마스크를 상기 마스크 결과 위치에 할당하는 단계;를 포함하며,

상기 마스크의 각 비트 값은 타겟 메모리 위치내의 바이트 위치에 해당하는 마스크 생성 방법.

명 세 서

발명의 상세한 설명

기 술 분 야

<1> 본 발명은 프로그래머블 디지털 프로세서들에 관한 것으로, 좀더 상세하게는 프로세서 시스템들에서 사용되는 데이터 저장 명령들(data storing instructions)에 관한 것이다.

배 경 기 술

<2> 임의의 배열 또는 임의의 크기를 가지는 데이터 전달(data transfers)은 임의의 소프트웨어 기능들의 수행에 사용된다. 데이터 통신의 영역에서, 그러한 전달들은, 예를 들어 53 바이트 ATM 셀들과 같은 고정된 크기의 셀들의 시퀀스, 3에서 255 바이트 사이의 어떠한 크기를 가질 수 있는 리드 솔로몬 코드 워드(Reed solomon code words)를 형성하는 데이터의 블록들, 또는 가변되는 크기의 패킷들의 스트림을 다루는데서 일어날 수 있으며, 여기에서, 개별 패킷들은 몇 바이트로부터 수천 바이트 또는 그 이상일 수 있다.

<3> 메모리를 포함하는 전형적인 프로그래머블 디지털 프로세서 시스템에서, 상기 메모리의 데이터 스토리지에서 가장 개별적으로 접근가능한 유닛은 제1 크기(예를 들어 8비트를 유지하는 바이트(byte))이며, 한편 상기 메모리에 대한 최우선 접속 메커니즘은 더 큰 제2 크기(예를 들어 32비트의 워드(word), 또는 64비트의 롱워드(long-word))를 가지는 단일 접속 유닛에 있는 메모리로 또는 메모리로부터 데이터를 전달할 수 있다. 그러한 많은 시스템들에서, 상기 프로세서에 의해 바이트보다 더 큰 데이터 유닛들을 메모리에 저장하는 것은 프로세서의 데이터 어드레싱 기법에 의존하는 어떠한 배열들에서만 이루어지거나 효과적으로 이루어질 수 있다.

<4> 예를 들어, 프로세서 및 그에 인터페이스되는 메모리를 포함하는 전형적인 시스템을 고려해보도록 한다. 예증이 되는 목적들(비록 원리들은 특정한 상세사항들을 개별적으로 적용할지라도)을 위하여, 선택된 프로세서는 64비트 머신이다. 즉, 그것이 상기 프로세서의 개별적인 레지스터내에 처리하고 유지하는 데이터 값의 기본 크기는 8개의 8비트 바이트와 같은 64비트 유닛이거나, 메모리의 전체 스토리지이다. 상기 메모리는 각 64비트 유닛이 단일 저장 동작에 기록되는 64비트 폭(롱워드(long-word)) 저장 유닛으로서 구성된다. 상기 메모리는 또한 단일 저장 동작에 개별적인 바이트, 또는 16비트 반워드(half-word), 또는 32비트 워드를 저장하기 위해 접근가능할 수 있다.

<5> 가장 보편적인 접속 패턴들이 논리에서 충돌을 유발하지 않고 사용되기 위해, 이들 타입의 시스템에서 프로세서 및 메모리 사이에서의 인터페이스는 전형적으로 "자연 배열(natural alignment)"를 이용하여 접속을 제공한다. 즉, 바이트는, 주어진 8바이트 롱워드 스토리지 유닛에서 8개의 다른 바이트 위치들, 오프셋들{0, 1, 2, ... 7}에서 자유롭게 저장될 수 있으며, 반면에 반워드(half-word)는 4개의 짝수 오프셋들{0, 2, 4, 6}(즉, 반워드 배열)의 하나에서 저장될 수 있으며, 워드는 워드 배열된 2개의 워드 배열 오프셋들{0, 4}중 하나에서만 기록될 수 있다.

<6> 추가적인 고려사항으로서, 소스 레지스터에서 최하위(least significant) 그런 서브 유닛의 저장을 지원하기 위해, 상기 프로세서의 일반적인 레지스터들-64비트 레지스터로부터의 반워드(16비트들, 두개의 8비트 바이트들) 저장함-의 전체 크보다 더 작은 데이터 유닛을 저장하는 저장 명령에 대하여 전형적이다. 이러한 제한은 종종 상기 프로세서의 메모리 인터페이스 회로에서의 복잡성을 줄이기거나 또는 별개의 명령들의 수가 지나치게 커지는 것을 막는데 또는 두가지 모두의 이유로 강조된다. 상기 예시된 시스템에 대한 후자적인 측면을 고려하면, 15개의 개별 저장 명령들의 전체는 레지스터내에 모든 자연적으로 배열된 서브 유닛들이 모두 4개의 크기들(소스 레지스터에서 자연적으로 배열된 각각 8, 4, 2, 1의 바이트, 반워드, 워드 및 롱워드)로 직접적으로 저장될 수 있다. 그렇지 않을지라도 그것은 상기 소스 레지스터내의 비자연적으로 배열된 유닛들이 반워드 및 워드 크기를 갖는 경우에 대하여 저장되게 하는 것을 허용하지 못한다. 대조적으로 상기 한정된 경우는 저장 동작의 모든 4개의 크기들을 지원하기 위해 단지 4개의 구별된 명령들을 필요로 한다. 프로그래머는 명령들을 결합함으로써 다른 어떠한 경우에라도 구현할 수 있다(예를 들어 저장 명령에 의해 수반되는 우측 쉬프트 명령을 사용함으로써).

<7> 이러한 유형의 시스템에서, 전술된 바와 같이, 단일 바이트를 메모리 내의 임의의 바이트 위치에서 저장할 수 있는 것이 통상적으로 요구된다. 이러한 저장(예를 들면, "바이트 저장" 명령에 의해 구현됨)은 기본 접속 유니트가 64 비트 길이이기 때문에, 메모리 내의 전체 크기의 (64 비트) 저장 위치의 어느 쪽이든지 특정 바이트 위

치를 포함한다는 것을 암시적으로 가리킨다. 그러나, 저장 명령은 동일한 길이-워드 저장 유니트 내의 다른 바이트 위치들에 임의의 데이터를 저장하는 것을 방지하도록 하는 방법으로 구현되어야 한다. 이러한 이유로, 많은 메모리 시스템들은 "바이트-가능" 신호들을 사용하여 구현된다. "바이트-사용" 신호는 메모리의 전체 길이상의 각각의 바이트 레인(lane)에 대해 정의된다(예를 들면 8 바이트 (길이-워드) 넓이의 메모리에 대한 8개의 신호들). 이들 신호들은 선택된 길이-워드 메모리 저장 위치의 그 부분에서의 바이트가 저장 명령이 수행될 때 대응하는 레인내의 접속 경로를 통해 제공되는 새로운 데이터로 겹쳐쓰기될 수 있든지 아니든지 각각의 바이트 레인에 대해 선택된다. 전형적으로, 바이트-가능 신호들은 수행된 각각의 저장 명령의 특정 세부 사항들에 따라 프로세서의 메모리 접속 논리내에서 생성된다. 데이터를 메모리 내에 쓰는 프로세서의 저장 명령의 동작 동안, 하나의 바이트-저장 신호는 각각의 데이터 바이트에 대해 접속 경로 상에서 전송된다.

<8> 이러한 시스템을 사용하여 임의의 정렬에서 데이터 저장을 달성하기 위해(예를 들면, 오프셋 3에서 4 바이트 워드를 저장하기 위해), 또는 기본 저장 유니트 크기까지의 임의의 크기의 유니트를 저장하기 위해(예를 들면, 오프셋 2...6에서 저장되는 데이터의 5 바이트 선택), 프로그래머는 프로세서의 가용 메모리 저장 명령들을 사용하는 알고리즘을 발전시켜야 한다. 이를 행하는 하나의 방법은 목표 메모리 위치에 현재 저장된 데이터를 읽고, 이를 임의의 정렬에 의해 야기된 임의의 간격들에 채우도록 소스 데이터와 병합하는 것이 포함된다. 전체 저장 유니트 크기의 병합 데이터는 목표 메모리 위치에 쓰여질 수 있다. 그러나, 이러한 알고리즘은 목표 위치로부터 데이터를 읽는데 지연이 있을 수 있기 때문에 상대적으로 느릴 수 있다. 또한, 특별히 저장 명령의 관련된 정렬 및 크기가 미리 고정되지 않으면, 알고리즘은 복잡하고, 여러 명령들을 포함할 수 있다. 또한, 이러한 알고리즘은 저장되는 목표 메모리 위치의 부근의 메모리 위치들의 사용상에 특정 제한들을 반영할 수 있다.

<9> 대안적인 접근으로서, 임의의 정렬에서 저장되는 데이터는 여러 작은 부분들로 쪼개질 수 있고, 인터페이스의 제한들을 만족시키기 위해 각각 개별적으로 크기화되고 재정렬되고, 개별적으로 저장된다. 그러나, 이러한 유형의 알고리즘은 또한 일반적으로 정렬된 저장 동작의 경우와 비교할 때 상대적으로 복잡하고 느릴 것이다. 따라서 이러한 알고리즘의 소프트웨어 구현은 고비용으로 인해 일반적인 구조에서 사용하기에 바람직하지 않고, 덜 편의적이다.

<10> 따라서, 임의의 배열들 또는 임의의 크기를 가질 때에 데이터 스토리지 동작들을 수행하는 것의 비용 및 복잡성을 현저하게 줄일 수 있는 시스템 및 방법이 요청된다.

발명의 내용

해결 하고자하는 과제

<11> 본 발명은 이러한 필요성에 의해 안출된 것으로, 프로세서에서의 마스킹된 저장 동작들을 위한 시스템 및 방법을 제공하는데 있다.

과제 해결수단

<12> 본 발명은 목적지 메모리에 데이터를 저장하기 위한 방법을 포함한다. 상기 방법은 마스킹된 저장 명령을 프로세서에 발행(issue)하는 단계, 및 상기 프로세서에서 상기 마스킹된 저장 명령을 처리하는 단계를 포함한다. 본 발명의 일실시예에 따른 상기 마스킹된 저장 명령을 처리하는 단계는 데이터 레지스터, 타겟 주소 및 마스크를 식별하는 단계, 뿐만 아니라, 상기 데이터 레지스터에서의 상기 데이터의 어떤 바이트들이 마스킹되지 않았는지를 식별하는 단계, 및 마스킹되지 않는 상기 데이터 레지스터의 바이트들을 상기 목적지 메모리에 기록하는 단계를 포함한다.

<13> 본 발명은 또한 메모리 접속 명령들을 처리하기 위한 집적회로를 포함한다. 상기 집적회로는 프로세서, 데이터 레지스터, 프리디케이트 레지스터, 및 상기 프로세서가 데이터를 메모리에 저장하기 위한 수단을 포함한다. 상기 메모리는 상기 집적회로의 내부 또는 외부에 포함될 수 있다. 상기 프로세서는 상기 데이터 레지스터내에 포함된 데이터를 상기 메모리에 저장하도록 구성된다. 본 발명의 일실시예에 따라, 상기 프로세서는 또한, 상기 마스크가 데이터의 상기 바이트가 저장되어야 할 것을 가리키는 경우에만, 상기 마스크를 읽기 위해 상기 프리디케이트 레지스터를 참고하고 상기 데이터 레지스터로부터의 데이터의 바이트들의 각각을 상기 메모리에 저장한다.

<14> 본 발명은 마스크를 생성하기 위한 방법을 더 포함하며, 여기에서 상기 마스크는 바이트들의 시퀀스중에서 어떤 바이트들이 저장되어야 하는지를 정의한다. 상기 방법은 세트 마스크 명령(set mask instruction)을 발행하는 단계, 상기 명령으로부터 마스크 결과 위치, 오프셋, 및 바이트 카운트를 식별하는 단계, 및 상기 바이트 카운

트로부터 상기 오프셋을 감함으로써 데이터 크기를 결정하는 단계를 포함한다. 만약 상기 데이터 크기가 0보다 작거나 같으면, 상기 마스크는 모두 0들로 설정된다. 만약 상기 데이터 크기가 비트들에 있어 상기 마스크의 폭보다 크면, 상기 마스크는 모두 1들로 설정된다. 만약 상기 데이터 크기가 0보다 크고 비트들에 있어 상기 마스크의 복보다 작거나 같으면, 상기 데이터 크기에 상응하는 상기 마스크의 비트들의 연속된 시퀀스는 1로 설정되고, 상기 마스크의 최하위 비트로부터 시작하고, 상기 식별된 마스크 결과 위치에 할당되는 상기 마스크 값을 형성하기 위해, 상기 마스크들의 잔여 비트들은 0으로 설정된다. 상기 마스크의 각 비트값은 타겟 메모리 위치내의 바이트 위치에 해당한다. 본 발명의 일실시예에 따라, 상기 마스크에서의 0이라는 값은 상기 해당 바이트 위치에 저장되지 않을 것을 나타내고, 상기 마스크에서의 1이라는 값은 상기 해당 바디트 위치에 저장될 것이라는 것을 나타낸다. 다른 실시예에서는, 상기 0 및 1의 의미는, 0비트는 상기 해당 바이트 위치에 저장되도록, 그리고 1비트는 저장되지 않도록, 바꾸어질 수 있다.

<15> 또한 본 발명은 마스크를 생성하기 위한 제2의 방법을 포함하며, 여기에서 상기 마스크는 바이트들의 시퀀중에서 어떤 바이트들이 저장될 것인지를 정의한다. 상기 제2 방법은 세트 마스크 명령을 발행하는 단계, 상기 명령으로부터 마스크 결과 위치, 오프셋, 및 바이트 카운트를 식별하는 단계, 및 상기 바이트 카운트로부터 상기 오프셋을 감함으로써 데이터 크기를 결정하는 단계를 포함한다. 만약 상기 데이터 크기가 0보다 작거나 같으면, 상기 마스크는 모두 1들로 설정된다. 만약 상기 데이터 크기가 비트들에 있어 상기 마스크의 상기 폭보다 크면, 상기 마스크는 모두 0들로 설정된다. 만약 상기 데이터 크기가 0보다 크고 비트들에서 있어 상기 마스크의 상기 폭보다 작거나 같으면, 상기 데이터 크기에 해당하는 상기 마스크의 비트들의 연속된 시퀀스는 0으로 설정되고, 상기 마스크의 최하위 비트로부터 시작하고, 상기 마스크의 잔여 비트들은 1로 설정된다. 상기 마스크의 각 비트값은 타겟 메모리 위치내의 바이트 위치에 해당한다. 본 발명의 일실시예에 따라, 상기 마스크에서의 0이라는 값은 상기 해당 바이트 위치에 저장되지 않을 것을 가리키고, 상기 마스크에서의 1이라는 값은 상기 해당 바이트 위치에 저장될 것을 가리킨다. 다른 실시예에서, 상기 마스크의 상기 비트들에서의 0 및 1 값들은 바뀔 수 있다.

<16> 본 발명의 일측면에 의하면, 목적지 메모리에 데이터를 저장하기 위한 방법이 제공되며, 상기 방법은,

<17> 마스킹된 저장 명령(masked store instruction)을 프로세서에 발행하는 단계; 및

<18> 상기 프로세서에서 상기 마스킹된 저장 명령을 처리하는 단계를 포함하되;

<19> 상기 처리 단계는,

<20> 상기 명령으로부터 데이터 레지스터, 타겟 주소 및 마스크를 식별하는 단계;

<21> 상기 데이터 레지스터의 데이터중에서 어떤 바이트들이 마스킹되어 있지 않은지 식별하는 단계- 여기에서 상기 마스크는 상기 데이터 레지스터의 데이터의 각 바이트가 마스킹되었는지 마스킹되지 않았는지 여부를 가리키는 비트를 포함함;

<22> 마스킹되지 않은 데이터 레지스터의 상기 바이트들을 상기 목적지 메모리에 기록하는 단계-여기에서 상기 데이터의 각 바이트는 상기 타겟 주소 및 상기 데이터 레지스터내의 상기 바이트의 상대적인 위치의 합에 상응하는 주소에 기록됨;

<23> 를 더 포함한다.

<24> 바람직하게는, 상기 마스킹된 저장 명령을 상기 프로세서에 발행하는 단계는,

<25> 상기 프로세서에 상기 타겟 주소 및 상기 마스크를 구성하는 단계를 포함한다.

<26> 바람직하게는, 상기 방법은,

<27> 상기 마스크를 프리디케이트 레지스터(predicate register)에 저장하는 단계; 및

<28> 상기 프로세서가 상기 프리디케이트 레지스터로부터 상기 마스크를 추출(retrieve)하도록 구성하는 단계를 더 포함한다.

<29> 바람직하게는, 상기 방법은,

<30> 상기 프리디케이트 레지스터의 위치를 식별하는 단계를 더 포함한다.

<31> 바람직하게는, 상기 마스킹된 저장 명령을 상기 프로세서에 발행하는 단계는,

<32> 상기 프로세서에 상기 데이터 레지스터의 위치를 구성하는 단계를 포함한다.

<33> 바람직하게는, 상기 마스킹된 저장 명령을 처리하는 단계는,

<34> 상기 데이터 레지스터의 위치를 식별하는 단계를 더 포함한다.

<35> 본 발명의 일측면에 의하면, 메모리 접속 명령들을 처리하기 위한 집적회로로가 제공되며, 상기 집적회로는,

<36> 프로세서;

<37> 데이터의 하나 또는 그 이상의 바이트들을 포함하는 데이터 레지스터;

<38> 마스크를 홀딩하도록 구성된 프리디케이트 레지스터;

<39> 메모리를 포함하며;

<40> 상기 프로세서는 상기 데이터 레지스터내에 포함된 데이터를 상기 메모리에 저장하도록 구성되고;

<41> 상기 프로세서는 상기 마스크를 읽기 위해 상기 프리디케이트 레지스터를 참고(consult)하고;

<42> 상기 프로세서는, 상기 마스크가 데이터의 상기 바이트가 저장되어야 함을 가리키는 경우에만, 상기 데이터 레지스터로부터의 데이터의 하나 또는 그 이상의 바이트들 각각을 상기 메모리에 저장한다.

<43> 바람직하게는, 상기 집적회로는,

<44> 상기 메모리로의 주소를 포함하는 메모리 포인터를 더 포함한다.

<45> 바람직하게는, 상기 메모리 포인터는 상기 메모리로의 롱워드(long-word) 주소를 포함한다.

<46> 바람직하게는, 상기 집적회로는,

<47> 상기 메모리 포인터의 상기 주소에 관련된 상기 메모리로의 주소를 포함하는 오프셋 포인터를 더 포함한다.

<48> 바람직하게는, 상기 오프셋 포인터는 상기 메모리로의 롱워드(long-word) 주소를 포함한다.

<49> 바람직하게는, 상기 집적회로는,

<50> 상기 데이터 레지스터로부터 상기 메모리에 복사되어야할 데이터의 바이트드의 수를 저장하도록 구성된 바이트 카운트 레지스터를 더 포함한다.

<51> 바람직하게는, 상기 프로세서는 저장 동작에서 데이터의 얼마나 많은 바이트들이 복사되어야 하는 지를 결정함으로써 상기 마스크를 생성하기 위해 상기 바이트 카운트 레지스터를 참고하도록 구성된다.

<52> 바람직하게는, 상기 프로세서는 상기 마스크를 상기 프리디케이트 레지스터에 저장하도록 구성된다.

<53> 본 발명의 일측면에 의하면, 마스크를 생성하기 위한 방법이 제공되며, 상기 마스크는 바이트들의 시퀀스중에서 어떤 바이트들이 저장되어야 하는지를 정의하되, 상기 방법은,

<54> 세트 마스크 명령(set mask instruction)을 발행하는 단계;

<55> 상기 명령으로부터 마스크 결과 위치, 오프셋, 바이트 카운트를 식별하는 단계;

<56> 상기 바이트 카운트로부터 상기 오프셋을 감하여 데이터 크기를 결정하는 단계;

<57> 상기 데이터 크기가 0 보다 작거나 같으면, 상기 마스크를 모두 0들로 설정하는 단계;

<58> 상기 데이터 크기가 비트들에 있어 상기 마스크의 폭보다 크면, 상기 마스크를 모두 1들로 설정하는 단계;

<59> 상기 데이터 크기가 0보다 크고, 비트들에 있어 상기 마스크의 폭보다 작거나 같으면, 상기 데이터 크기에 상응하는 상기 마스크의 비트들의 연속된 시퀀스를 1로 설정하고, 상기 마스크의 최하위 비트(least significant bit)로부터 시작하고, 상기 마스크의 잔여 비트들을 0으로 설정하는 단계; 및

<60> 상기 마스크를 상기 마스크 결과 위치에 할당하는 단계;를 포함하며,

<61> 여기에서, 상기 마스크의 각 비트 값은 타겟 메모리 위치(location)내의 바이트 위치에 해당한다.

<62> 바람직하게는, 상기 마스크에서의 0이라는 값은 상기 해당 바이트 위치에 저장되지 않을 것을 가리키며, 여기에서 상기 마스크에서의 1이라는 값은 상기 해당 바이트 위치에 저장될 것을 가리킨다.

<63> 바람직하게는, 상기 마스크에서의 0이라는 값은 상기 해당 바이트 위치에 저장될 것을 가리키며, 여기에서 상기 마스크에서의 1이라는 값은 상기 해당 바이트 위치에 저장되지 않을 것을 가리킨다.

<64> 바람직하게는, 상기 데이터 크기에 상응하는 상기 마스크의 비트들의 양을 1로 설정하는 단계는,

<65> 상기 마스크를 모두 1들로 설정하는 단계; 및

<66> 상기 마스크를 상기 마스크의 폭에서 상기 데이터 크기를 뺀 것에 일치하는 배수의 양만큼 오른쪽으로 이동시키는 단계를 더 포함한다.

<67> 바람직하게는, 상기 방법은

<68> 상기 명령으로부터 프리디케이트를 식별하는 단계; 및

<69> 상기 세트 마스크 명령을 수행할 것인지 여부를 결정하기 위해 상기 프리디케이트를 참고하는 단계를 더 포함한다.

<70> 바람직하게는, 상기 방법은,

<71> 상기 바이트 카운트로부터 데이터의 라인에서의 상기 바이트들의 상기 양을 감하는 단계를 더 포함한다.

<72> 바람직하게는, 상기 방법은,

<73> 상기 바이트 카운트가 복사되기 위해 남아있는 바이트들중 양의 수를 가리키면 각 단계를 반복하는 단계를 더 포함한다.

<74> 바람직하게는, 상기 방법은,

<75> 어떤 바이트들이 복사되어야 하는지를 결정하기 위해 상기 마스크를 사용하여, 바이트들의 상기 시퀀스에의 상기 각 바이트들을 복사하는 단계를 더 포함한다.

<76> 본 발명의 일측면에 의하면, 마스크를 생성하기 위한 방법이 제공되며, 상기 마스크는 바이트들의 시퀀스 중에서 어떤 바이트들이 저장되어야 하는지를 정의하되, 상기 방법은,

<77> 세트 마스크 명령을 발행하는 단계;

<78> 상기 명령으로부터 마스크 결과 위치, 오프셋, 및 바이트 카운트를 식별하는 단계;

<79> 상기 바이트 카운트로부터 상기 오프셋을 감하여 데이터 크기를 결정하는 단계;

<80> 상기 데이터 크기가 0보다 작거나 같으면, 상기 마스크를 모두 0들로 설정하는 단계;

<81> 상기 데이터 크기가 비트들에 있어 상기 마스크의 폭보다 크면, 상기 마스크를 모두 0들로 설정하는 단계;

<82> 상기 데이터 크기가 0보다 크고, 비트들에 있어서 상기 마스크의 폭보다 작거나 같으면, 상기 데이터 크기에 상응하는 상기 마스크의 비트들의 연속된 시퀀스를 0으로 설정하고, 상기 마스크의 최하위 비트로부터 시작하고, 상기 마스크의 잔여 비트들을 1로 설정하는 단계; 및

<83> 상기 마스크를 상기 마스크 결과 위치에 할당하는 단계;를 포함하며,

<84> 상기 마스크의 각 비트 값은 타겟 메모리 위치내의 바이트 위치에 해당한다.

<85> 바람직하게는, 상기 마스크에서의 0이라는 값은 상기 해당 바이트 위치에 저장되지 않을 것을 가리키며, 여기에서 상기 마스크에서의 1이라는 값은 상기 해당 바이트 위치에 저장될 것을 가리킨다.

<86> 바람직하게는, 상기 마스크에서의 0이라는 값은 상기 해당 바이트 위치에 저장될 것을 가리키며, 여기에서 상기 마스크에서의 1이라는 값은 상기 해당 바이트 위치에 저장되지 않을 것을 가리킨다.

<87> 바람직하게는, 상기 데이터 크기에 상응하는 상기 마스크의 비트들의 양을 1로 설정하는 단계는,

<88> 상기 마스크를 모두 1들로 설정하는 단계; 및

<89> 상기 마스크를 상기 마스크의 폭에서 상기 데이터 크기를 뺀 것에 일치하는 배수의 양만큼 오른쪽으로 이동시키는 단계를 더 포함한다.

<90> 바람직하게는, 상기 방법은

<91> 상기 명령으로부터 프리디케이트를 식별하는 단계; 및

<92> 상기 세트 마스크 명령을 수행할 것인지 여부를 결정하기 위해 상기 프리디케이트를 참고하는 단계를 더 포함한다.

<93> 바람직하게는, 상기 방법은,

<94> 상기 바이트 카운트로부터 데이터의 라인에서의 상기 바이트들의 상기 양을 감하는 단계를 더 포함한다.

<95> 바람직하게는, 상기 방법은,

<96> 상기 바이트 카운트가 복사되기 위해 남아있는 바이트들중 양의 수를 가리키면 각 단계를 반복하는 단계를 더 포함한다.

<97> 바람직하게는, 상기 방법은,

<98> 어떤 바이트들이 복사되어야 하는지를 결정하기 위해 상기 마스크를 사용하여, 바이트들의 상기 시퀀스에의 상기 각 바이트들을 복사하는 단계를 더 포함한다.

효 과

<99> 본 발명에 의하면, 프로세서에서의 마스킹된 저장 동작들을 위한 시스템 및 방법을 효과적으로 제공함으로써, 임의의 배열들 또는 임의의 크기를 가질 때에 데이터 스토리지 동작들을 수행하는 것의 비용 및 복잡성을 현저하게 줄일 수 있다.

발명의 실시를 위한 구체적인 내용

<100> 여기에 첨부되고 본 명세서의 일부를 구성하는 첨부된 도면들은, 설명과 함께, 본 발명의 원리들을 설명하는 것을 더 돕기 위해, 그리고 관련 분야에서 숙련된 사람이 본 발명을 만들고 이용할 수 있도록 하기 위하여 본 발명을 예시한다. 본 발명은 첨부된 도면들을 참조하여 설명될 것이다. 도면에서, 동일 참조번호들은 동일하거나 기능적으로 유사한 구성요소들을 나타낼 수 있다. 또한 참조번호의 레프트 모스트 디지트(left-most digits)는 도면에서 처음 나타나는 참조번호를 식별할 수 있다.

<101> 1. 구조 개요(Architecture overview)

<102> 도 1은 본 발명에서의 사용을 위한 대표적인 프로세서 시스템을 예시한다. 실시예에서, 상기 프로세서 시스템은 참조 문자 X 및 Y에 의해 지정된 두 평행 SIMD(single instruction multiple data) 실행 유닛들을 포함하는 64 비트 롱 명령 워드 머신이다. 해당 분야에서 숙련된 사람에 의해 인식되는 바와 같이, 프로세서 시스템(100)에 대한 다른 구성들은 본 발명과 함께 사용될 수 있다.

<103> 프로세서 시스템(100)은 프로그램 메모리(미도시됨)로부터 명령들(instructions)을 수신하고 유지하기 위한 명령 캐시(instruction cache)를 포함한다. 명령 캐시(100)는 페치(fetch)/디코딩 회로(120)에 결합되어 있다. 페치/디코딩 회로(120)는 명령들이 페치될 출처가 되는 상기 프로그램 메모리에 있는 주소들을 발행하며, 상기 캐시(110)(또는 프로그램 메모리)로부터 각 페치 동작에 대하여 64비트 명령들을 수신한다. 또한 페치/디코딩 회로(120)는 지정된 레지스터들과 MAC(multiplier accumulator)(132), 정수 유닛(integer unit : INT)(134), GFU(Galois Field Unit)(136), 및 LSU(load/store)(140) 기능 유닛들 사이에서 데이터의 이동을 제어하기 위해 명령에 있는 오피코드(opcode)를 계산하고 채널들(125X, 125Y)을 따라 제어 신호들을 송신한다. 다른 형태들의 기능 유닛들(미도시됨)이 상기 프로세서내에 또한 존재할 수 있으며, 상기 페치/디코딩 회로 및 상기 다양한 레지스트들 중에 몇몇 또는 모두에 마찬가지로 결합될 수 있다.

<104> 프로세서 시스템(100)은 두 SIMD 실행 유닛들(130x, 130y), 상기 머신의 X측에 대하여 하나 및 상기 머신의 Y측에 대하여 하나를 포함한다. SIMD 실행 유닛들(130x, 130y)의 각각은 MAC(132), INT(134), 및 GPU(316)를 포함한다. 승산기 누산 유닛(132x, 132y)은 많은 디지털 신호 처리 알고리즘들에서 일반적으로 사용되는 생성물들의 승산 및 가산의 처리를 수행한다. 정수 유닛들(134x, 134y)은 일반 계산 및 신호 처리에서 사용되는 정수값에 대한 많은 일반 동작들을 수행한다. 갈로이스(Galois) 필드 유닛들(136x, 136y)은 리드 솔로몬 오류 보호 코딩 기법의 구현에서 실행될 수 있는 바와 같은 갈로이스 필드 연산을 사용하는 특별 연산을 수행한다.

<105> 또한, 본 발명의 일실시예에 따른 로딩/저장 유닛(LSU)(140x, 140y)은 X 및 Y 측 SIMD 유닛들에 대하여 제공된다. 로딩/저장 유닛(140x, 140y)은 데이터 메모리 및 I/O 시스템(170)에 접속하여 그것으로부터 범용 레지스터

(155)로 데이터값들을 로딩하거나 범용 레지스터(155)로부터 그것에 값들을 저장한다. 로딩/저장 유닛들(140x, 140y)은 상기 프로세서 시스템(100)의 상기 명령 메모리 또는 다른 메모리들에 대한 그리고 그로부터의 데이터 접속을 허용하기 위하여 (미도시된 수단들에 의해) 또한 연결될 수 있다.

<106> 본 발명의 일실시예에 따라, 프로세서 시스템(100)은 데이터 메모리(170)의 값들에 대한 더 빠른 접속을 제공하기 위하여 데이터 캐시를 포함한다. 관련 분야에서 숙련된 사람은 다른 스토리지 구현들(storage implementations)이 또한 본 발명과 사용될 수 있음을 인식할 수 있을 것이다.

<107> 프로세서 시스템(100)은 승산 누적 결과들을 유지하기 위한 소정 개수의 승산 누적 레지스터들(M-레지스터들)과 복수의 범용 레지스터들(155)을 포함한다. 본 발명의 일실시예에 따라, 프로세서 시스템(100)은 4개의 M-레지스터들과 64개의 범용 레지스터들을 포함한다. 본 발명의 더 상세한 실시예에 따라, 프로세서 시스템(100)은 복수개의 제어 레지스터들(160)과 복수의 프리디케이트 레지스터들(165)을 또한 포함한다.

<108> 2. 마스킹된 저장 명령들

<109> 2.1 STLM(store long-word under mask)

<110> 도 2는 본 발명의 일실시예에 따른 STLM(store long-word under mask) 명령(200)에 대한 대표적인 포맷을 예시한다. STLM 명령(200)은 프리디케이트 레지스터의 제어하에 레지스터로부터의 0 및 8 바이트의 데이터를 메모리 위치(location)에 저장한다.

<111> STLM 명령(200)은 프리디케이트 오퍼랜드(predicate operand)(202), opcode(204), 소스(source) 오퍼랜드(206), 베이스(base) 오퍼랜드(208), 오프셋(offset) 오퍼랜드(210)를 포함한다. 타겟(target) 주소는 베이스 오퍼랜드(208)과 오프셋 오퍼랜드(210)의 합으로 계산된다. 상기 타겟 주소는 상기 저장 동작이 수행될 메모리의 주소를 나타낸다. 관련 분야들에서 숙련된 사람은 STLM 명령(200)의 오퍼랜드들이 도 2에 나타낸 순서로 표시될 필요는 없으며, 추가 오퍼랜드들이 선택적으로 동반될 수 있음을 인식할 것이다. 더욱이 관련 분야들에서 숙련된 사람은 상기 오퍼랜드들이 STLM 명령(200)에 설명된 바와 같은 오퍼랜드들의 값이 손쉽게 결정되는 것을 허용할 수 있는 어떠한 방식의 소프트웨어 구현내에서 정의될 수 있음을 인식할 것이다.

<112> 본 발명의 일실시예에 따라, 베이스 오퍼랜드(208)는 그 내용이 베이스 값들로 읽혀지는 범용 레지스터들(155) 중 하나를 식별하며, 오프셋 오퍼랜드(210)는 상기 명령의 필드내의 어떤 폼(예를 들어 2의 보수 바이너리 넘버와 같이)으로 표현된 일정한 정수이다. 관련 분야들에서 숙련된 사람은 타겟 주소를 나타내는 추가적인 방법들이 예를 들어 스케일링 인자의 배수가 되는 오프셋 값들의 커다란 범위를 허용하기 위해 상기 일정한 정수 오프셋 값을 스케일링함으로써 사용될 수 있다.

<113> 본 발명의 더 상세한 실시예에 따라, 오프셋 오퍼랜드(210)는 상수 정수(constant integer)를 나타내지 않고 그 대신에 상기 오프셋 값을 제공하기 위하여 그 내용들이 읽어들여지는 제2 레지스터들을 범용 레지스터들(155)로부터 식별한다. 본 발명의 추가 실시예에 따라, 오프셋 오퍼랜드(210)는 상기 명령에 무시되거나 존재하지 않으며 상기 베이스 오퍼랜드(208)는 직접적으로 상기 타겟 주소를 형성한다.

<114> 본 발명의 일실시예에 따라, 롱 워드(64비트) 폭이며 잘못배열된(misaligned) 저장 동작을 지원하지 않는 메모리 공간들에 대한 저장을 위하여, STLM 명령(200)의 타겟 주소는 롱 워드 경계에 배열되어야 한다. 본 발명의 상세한 실시예에서, 미배열 저장들을 지원하는 메모리 공간들에 대한 저장들을 위해, STLM 명령(200)의 타겟 주소는 롱워드 경계에 배열될 필요가 없다. 상기 타겟 주소는 데이터를 저장하기 위한 메모리 위치를 특정화(specify)한다.

<115> 소스 오퍼랜드(206)는 범용 레지스터들(155) 중 하나와 같은 레지스터를 특정화하는데, 이 레지스터로부터 롱 워드 데이터가 획득된다. 본 발명의 일실시예에서, 롱워드는 8비트로 정의된다. 프리디케이트(202)는 STLM 명령(200)의 동작에서 저장되기 위하여 바이트마다 하나의 비트를 가지는 프리디케이트 레지스터들(165) 중의 하나와 같은 프리디케이트 레지스터를 식별한다. 본 발명의 일실시예에 따라, 프리디케이트(202)는 8비트 레지스터를 식별한다.

<116> 관련 분야에서 숙련된 사람은 STLM 명령(200)이 현저한 변경없이 유사한 어플리케이션으로 확장될 수 있다는 것을 인식할 수 있을 것이다. 예를 들어, STLM 명령(200)의 균등물들이 임의의 워드폭 및 데이터 어드레싱을 위한 임의의 모양(granularity)을 가지고 시스템들에서 사용될 수 있다.

<117> 다음은 도 2에 설명된 포맷을 사용하는 대표적인 STLM 명령(200)이다. 예를 들어, 'pM'은 프리디케이트 오퍼랜드(202)를 나타내며, 'STLM'은 opcode(204)에 대한 상징적인 표현이며, 'src'는 소스 오퍼랜드(206)를

나타내며, 'base는 베이스 오퍼랜드(208)를 나타내며, '#offset'은 상수 오프셋 오퍼랜드(210)를 나타내며, 여기에서 '오프셋'은 이 예에서 오프셋의 특정 값을 상징한다.

<118> pM.STLM src,[base, #offset]

<119> 도 3은 프리디케이트 레지스터들(165)중의 하나 일 수 있으며, 프리디케이트 오퍼랜드(165)에 의해 식별될 수 있는 프리디케이트 마스크 레지스터(300)를 예시한다. 본 발명의 실시예에 따라, 프리디케이트 마스크 레지스터(300)는 프리디케이트 마스크 비트들(pM_0 - pM_7(302a - 302h), 소스 오퍼랜드(206)에 의해 특정되는 레지스터에서 각 바이트에 상응하는 하나의 비트를 포함한다.

<120> 도 4는 STLM 명령(200)을 사용하는 위치들(D_0 - D_7)(404a - 404h)에서 목적지 메모리(410)에 저장되어 있는 8바이트의 데이터(R_0 - R_7)(402a - 404h)를 포함하는 소스 레지스터(400)를 예시한다. 적용된 프리디케이트 레지스터 마스크(300)와 함께, 마스크 비트들(302a - 302h)은 목적지 메모리(410)에 저장될 레지스터(400)로부터의 바이트들을 제어한다. 발명의 일실시예에 따라, 마스크 제어 비트들(302a - 302h)의 값들을 메모리 시스템에 대한 바이트-인에이블 신호들로서 보냄으로써 제어가 구현된다. 관련 분야에서 숙련된 사람은 목적지 메모리(410)에 저장될 레지스터(400)로부터의 바이트들을 제어하기 다른 방법들이 사용될 수 있음을 알 수 있을 것이다.

<121> 관련 분야에서 숙련된 사람은 STLM 명령을 사용하여 데이터를 저장하는 프로세스는 저장될 데이터의 시퀀스를 획득하기 위한 방법으로부터 독립적이라는 것을 알 것이다. 본 발명의 일실시예에 따라, 그러한 한가지 방법은 메모리의 소스 주소로부터 데이터 바이트들의 시퀀스를 로딩함으로써 수행된다. 상기 로딩 방법 및 하나 또는 그 이상의 STLM 명령의 예들을 사용하는 저장 프로세스의 결합된 효과는 소스 메모리로부터의 데이터 바이트들의 시퀀스를 목적지 메모리에 복사하는 것이다. 이들 단계들의 과정에서, STLM의 사용은 복사된 데이터에 의해 덮어씌일 상기 목적지의 정확한 바이트 위치가 실질적으로 쓰여지는 한편, 모든 저장 동작들은 단일 동작에서 저장 요청되는 만큼의 데이터 바이트들의 수만큼 정확하게 그들의 각 소스 레지스터에 저장한다라는 점에서 최대의 효율성을 허용한다. 따라서, 상기 목적지 메모리의 접근 폭에 의해 요청되는 저장 동작의 최소 수치를 이용하고, 수행될 상기 목적지 메모리로부터 아무런 로딩 동작들을 요구하지 않는 복사 동작이 수행될 수 있다.

<122> 2.2 TSTCM(test count to set for STLM)

<123> 도 5는 본 발명의 일실시예에 따른 TSTCM(test count to set for STLM) 명령(500)에 대한 대표적인 포맷을 예시한다. TSTCM 명령(500)은 바이트들의 시퀀스를 저장할 때, 도 1에 도시된 프로세서(100)의 프리디케이트 레지스터들(165) 중 하나 일 수 있으며, 도 3으로부터 프리디케이트 레지스터(300)와 같은, 프리디케이트 마스크 레지스터의 비트들을 설정함으로써, STLM의 사용을 돕기 위해 설계된다. 연속된 바이트들의 시퀀스가 저장될 수 있는 연속 전달(continuous transfer)를 위하여, TSTCM 명령(500)은 어떤 오프셋에서 소스 레지스터 홀딩 바이트들 중 어떤 바이트들이 저장되어야 하는지를 결정한다. TSTCM 명령(500)은 오피코드(opcode)(504), 프리디케이트 마스크 오퍼랜드(506), 카운트 오퍼랜드(508), 및 오프셋 오퍼랜드(510)를 포함한다. 프리디케이트 마스크 오퍼랜드(506)는 TSTCM 명령에 의해 발생된 마스크 값이 기록될 프리디케이트 마스크 레지스터를 식별하기 위해 사용될 수 있다. 카운트 오퍼랜드(508)는 읽어질 카운트의 값으로부터, 기록될 바이트들의 카운트를 지시하며, 특별히 범용 레지스터들(510) 중 하나를 지시한다. 오프셋 오퍼랜드(510)는 본 발명의 일실시예에 따라 특별히 상수 정수로 표현되는 주소 오프셋의 값을 정의한다. 관련 분야들에서 숙련된 사람은 TSTCM 명령(500)의 오퍼랜드들이 도 5에 보여진 순서대로 나타날 필요가 없으며, 추가 오퍼랜드를 선택으로 동반할 수 있음을 알 수 있을 것이다. 또한, 관련 분야들에서 숙련된 사람은 상기 오퍼랜드들이, TSTCM 명령(500)에 설명된 바와 같은 오퍼랜드들의 값들이 손쉽게 결정되는 것을 허용할 수 있는 어떠한 방식의 소프트웨어 구현내에서 정의될 수 있음을 알 것이다.

<124> TSTCM 명령(500)의 대표적인 활용에서, 전체 길이 L를 가지는 데이터 바이트들의 연속된 시퀀스가 저장되며, 여기에서 길이 L은 저장 동작이 수행되기 시작하기까지는 알 수 없다(이를 테면 L은 일정한 값). 상기 L 바이트들은 목적지 메모리내에 저장될 것이다. 상기 길이 L이 일반적으로 알려져 있지 않기 때문에, 저장 동작은 편의상 모든 L 바이트들이 저장될 때까지 명령들의 시퀀스에 대한 반복적인 실행을 유발하기 위한 소프트웨어 루프를 이용하여 구현될 수 있다. 일반화된 함수, L=K*M+N에서,(여기에서 M은 루프의 한 반복에서 저장될 수 있는 데이터 바이트들의 최대 수를 나타내는 0보다 큰 상수 정수이며, K 및 N은 각각 0 이상의 정수들이며, L의 값이 일단 알려지면 계산된다.) 상기 저장은 N 바이트들이 저장되는 마지막 저장 시퀀스의 실행이 수반되는 루프의 K 반복들, 각 반복에 대하여 M 바이트들을 저장하는 것을 이용하여 구현될 수 있다. 상기 마지막 N 바이트들의 저장은 동일 루프의 (K+1)번째 반복 또는 환경들에 따른 명령들의 분리된 시퀀스에 따라 구현될 수 있다. 적어도

상기 마지막 N 바이트들을 저장할 때, 하나 또는 그 이상의 소스 레지스터들(SR0, SR1, 등)의 순서 세트는 상기 시퀀스에서 M 데이터 바이트들의 전체와 함께 준비될 수 있으며, 상기 시퀀스의 N 데이터 바이트들(N≤M)은 타겟 주소 T에서 시작하는 상기 목적지 메모리내로 저장될 수 있다. 상기 세트내의 개별 소스 레지스터(SRn)는 상기 전달의 잠재적인 부분인 데이터 바이트들을 유지한다. N 바이트들을 포함하는 상기 전달의 선택이 충분한 길이라면, 즉, N>R이라면, SRn에서 적어도 하나의 데이터 바이트들은 T에 대하여 상대적인 오프셋 R에서 시작하는 목적지 메모리내에 저장될 수 있다. N의 주어진 값 및 상기 세트에서 소스 레지스터 SRn 중 가장 낮게 어드레싱된 바이트와 관련된 오프셋 R에 대하여, TSTCM 명령(500)은 본 발명의 일실시예에 따라 주소 T+R에서 시작하는 타겟 메모리에서 그들의 각 위치들에 저장될 수 있는 SRn에서 그 바이트들을 확인하기 위하여 사용된다. 그렇게 함으로써 그것은 SRn을 그것의 소스 오퍼랜드(206)로서 또한 사용하는 연관 STLM 명령의 프리디케이트 오퍼랜드(202)를 위하여 사용되기에 적절한 마스크를 생성하고, 따라서 그들 식별된 바이트들을 상기 타겟 메모리에 저장하게 한다.

<125> 본 발명의 일실시예에 따라, TSTCM 명령(500)은 선택적 프리디케이트(502)를 더 포함하며, 여기에서 프리디케이트(502)는 프리디케이트 마스크 레지스터(300)의 비트마다 하나의 비트를 가지는, 프리디케이트 레지스터들(165)중 하나와 같은 프리디케이트 레지스터를 식별한다. 만약 프리디케이트(502)가 특정화되면, 상기 식별된 프리디케이트 레지스터의 비트들은 프리디케이트 마스크 레지스터(300)의 연관된 비트가 본 발명의 더 상세한 일실시예에 따른 TSTCM 명령(500)에 의해 설정될 것인지 여부를 가리킨다.

<126> 다음은 도 5에서 상술된 포맷을 이용하는 대표적인 TSTCM 명령문이다.

<127> pC.TSTCM pM,count, #offset

<128> TSTCM 명령(500)의 이 예는 카운트, 상수 값, 오프셋, 및 추가 프리디케이트 pC로서 식별된 오퍼랜드의 값에 따라, 이 예에서 pM으로서 식별된 프리디케이트 마스크 레지스터(300)를 설정하는 효과를 가진다. 계산은 다음의 알고리즘에 근거하거나 대등하며, 알고리즘에서 크기는 국부 임시 정수 값이며, tm은 폭에서 8비트의 국부 임시 마스크 값이다.

<129> size=count-offset;

<130> if(size<=0) then tm=00000000_2;

<131> else if(size>7) then tm=11111111_2;

<132> else tm=11111111_2>>(8-size);

<133> pM=(tm&pC)|(pM&(~pC));

<134> 상기 알고리즘에서, "pM"은 프리디케이트 마스크 레지스터이고, 아래첨자 "2"는 방금전에 상술된 이진 수들을 가리킨다. ">>"연산자는 그것들의 최상위종점에 0들을 삽입하는 논리적인(비트관련된) 우측 쉬프트이다. 상기 바이너리(binary)"&" 연산자는 비트관련 논리적인 AND이고, 바이너리"|"연산자는 비트관련된 논리적인 OR이고, 유너리(unary) "~" 연산자는 논리적인 비트관련 보수(반전)이다. 상기 카운트 및 오프셋은 모두 바이트들의 유닛들로 측정된다. 만약 TSTCM 명령(500)의 또 다른 예에서, 선택적인 프리디케이트(502)(상기 예에서 pC)는 명확하게 제공되지 않고, 상기 알고리즘의 실행은 값 11111111_2을 가지는 바와 같이 취급된 pC로 진행한다. 이 알고리즘의 특정한 어플리케이션은 아래에서 더 설명될 것이다.

<135> 2.3 TSTIM(test count to set inverted mask for TSTIM)

<136> 도 6은 본 발명의 일실시예에 따른, TSTIM(test count to set inverted mask for TSTIM)을 위한 대표적인 포맷을 예시한다. TSTIM 명령(600)은 도 5에 도시된 TSTCM 명령(500)과 유사한 목적으로 설계된다. 그러나, TSTIM 명령(600)은 저장되지 않을 일련의 바이트들의 연속된 전달의 시작점(끝점이 아닌)에서 소정 개수의 바이트들이 있는 특별한 환경을 위한 것이다.

<137> TSTIM 명령(600)은 오피코드(604), 프리디케이트 마스크 오퍼랜드(606), 카운트 오퍼랜드(608), 및 오프셋 오퍼랜드(610)를 포함한다. 프리디케이트 마스크 오퍼랜드(606)는 TSTIM 명령으로부터 기인되는 마스크 값이 기록될, 도 3으로부터의 프리디케이트 마스크 레지스터(300)와 같은 프리디케이트 마스크 레지스터를 식별하기 위해 사용된다. 본 발명의 일실시예에 따른, 카운트 오퍼랜드(608)는 얼마나 많은 바이트들이 기록될 것인지를 가리키며, 오프셋 오퍼랜드(610)는 현재 롱워드 오프셋을 가리킨다. 관련 분야들에서 숙련된 사람은 TSTIM 명령

(600)의 오퍼랜드들이 도 6에 나타낸 순서로 표시될 필요는 없으며, 추가 오퍼랜드들이 선택적으로 동반될 수 있음을 인식할 것이다. 더욱이 관련 분야들에서 숙련된 사람은 상기 오퍼랜드들이 TSTIM 명령(600)에 설명된 바와 같은 오퍼랜드들의 값이 손쉽게 결정되는 것을 허용할 수 있는 어떠한 방식의 소프트웨어 구현내에서 정의될 수 있음을 인식할 것이다.

<138> 본 발명의 일실시예에 따라, TSTIM 명령(600)은 선택적 프리디케이트(602)를 더 포함하며, 여기에서 프리디케이트(602)는 프리디케이트 마스크 레지스터(300)의 비트마다 하나의 비트를 가지는, 프리디케이트 레지스터들(165)중 하나와 같은 프리디케이트 레지스터를 식별한다. 만약 프리디케이트(602)가 특정화되면, 상기 식별된 프리디케이트 레지스터의 비트들은 프리디케이트 마스크 레지스터(300)의 연관된 비트가 본 발명의 더 상세한 일실시예에 따른 TSTIM 명령(600)에 의해 설정될 것인지 여부를 가리킨다.

<139> 다음은 도 6과 관련하여 상술된 포맷을 이용하는 대표적인 TSTIM 명령문이다.

<140> pC.TSTIM pM,count,#offset

<141> TSTIM 명령(600)은 다음의 알고리즘에 근거하여, 여기에서는 pM으로서 식별된, 프리디케이트 마스크 레지스터(300)의 출력을 설정하는 효과를 가지며, 알고리즘에서 크기는 국부 임시 정수 값이며, tm은 폭에서 8비트의 국부 임시 마스크 값이다.

<142> size=count-offset;

<143> if(size<=0)then tm=11111111_2;

<144> else if(size>7)then tm=00000000_2;

<145> else tm=11111111_2<<size;

<146> pM=(tm&pC)|(pM&(~pC));

<147> 선택적 프리디케이트 pC를 포함하여, TSTCM에 대한 알고리즘을 설명하기 위해 앞서 사용된 모음들은 이 알고리즘에 대하여 또한 적용가능하다. 추가적으로 이 알고리즘에서, "<<" 연산자는 그 결과의 최하위종점(least significant end)에 0을 삽입하는 논리적(비트관련)좌측 쉬프트이다. 이 알고리즘의 상세한 어플리케이션은 아래에서 더 설명될 것이다.

<148> 3. STLM 명령의 대표적인 어플리케이션

<149> 도 7은 본 발명의 일실시예에 따른 8바이트 폭 롱워드 어드레스 메모리(700)섹션을 보여준다. 이 예에서, 도시된 메모리 섹션은 ATM 셀들이 ADSL 모뎀의 동작의 일부에 따라 기록될 버퍼내에 포함되어 있다. 상기 버퍼는 소프트웨어 데이터 처리가 전방향(주소를 증가시키는) 순서로 상기 버퍼를 진행하는 바와 같이, 상기 버퍼에서 마지막 주소에서의 데이터가 처리된 후에, 처리될 다음 데이터는 다시 상기 버퍼의 시작에 있게 되는 "서큘러(circular)" 방식으로 사용된다. 상기 처리는 두 개의 다른 페이즈들(phases)에서 반복적으로 일어난다. 제1 페이즈에서, ATM 셀들(길이에서 각 각 53바이트들)이 임의의 수단에 의해 생성되어 상기 버퍼에 저장된다. 제2 페이즈에서, 상기 셀들은 다시 읽혀진다. 그 다음 제1 페이즈는 더 많은 셀 데이터, 등등을 기록하기 위해 실행된다. 상기 셀들은 정확하게 53바이트의 완전한 ATM 셀 유닛들로서 기록되거나 읽혀질 필요는 없다. 예를 들어 읽기 페이즈는 상기 버퍼를 단순히 바이트들의 시퀀스로서 취급할 수 있으며, 그것의 임의의 수(전체 개수에 까지 가능함)는 어떠한 특정 시간에라도 추출될 수 있다. 그러므로 읽기 페이즈는 임의의 시점에라도 멈출 수 있기 때문에, 기록 페이즈는 데이터가 기록될 셀들을 위한 여유가 있는 만큼 상기 셀들로부터 데이터의 바이트만큼만을 저장해야 한다.

<150> 얼마나 많은 ATM 셀들이 상기 버퍼로의 기록동안 생성될 수 있는가의 한 예는 아무런 액티브 데이터 셀들도 진행될 수 없는 경우이며, 대신에 "아이들 셀들(idle cells)"이 생성되야 한다. 아이들 셀들은 48 "페이로드" 바이트가 수반되며, 그것의 각각은 동일 고정값을 가지는 고정된 패턴을 가지는 5바이트의 "헤더"를 포함한다. 아이들 셀을 생성하는 것은, 그것이 포함하는 값들이 고정되어 있지 않기 때문에 계산적으로 집중적인 프로세스가 아니다. 상기 페이로드 바이트들 모두는 동일한 값들을 포함하고 있으며; ADSL 모뎀의 내부 동작들의 문맥에서, 이 값은 16진수 56(86 십진수)이다. 그러므로 상기 기록 프로세스가 이용가능한 액티브 데이터 셀들을 가지지 않을 때, 그것은 5 아이들 셀 헤더 바이트들, 그 각각이 16진수 56인 48 아이들 페이로드 바이트들, 또 다른 5 헤더 바이트들, 그 다음 48 아이들 페이로드 바이트들, 등등으로 이루어지는 고정된 패턴을 포함하는 바이트들

의 시퀀스를 기록할 것이다.

<151> 또 다른 예에서, 기록 페이즈는 그것이 아이들 셀 데이터를 버퍼내로 기록함에 따라, 읽기 페이즈와 함께 "캐치 업(catch up)"한다. 이 예에서, 아이들 셀 페이로드 데이터의 상기 48바이트들은 기록되는 프로세스에 있다. 그러나, 상기 소프트웨어 기능의 수행되는 동안, 모든 48 바이트들이 기록될 수는 없다고 결정되는데, 이는 상기 버퍼가 거의 차있고, 상기 제1 아이들 셀 페이로드 바이트들이 갈 곳들 사이에 단지 27 가용 바이트 위치들의 갭과, 상기 읽기 페이즈에 의해 읽어내지 못하고 덮어쓰지(overwritten) 못한 상기 버퍼의 상기 제1 바이트가 있기 때문이다. 따라서, 상기 기록 페이즈는 단지 아이들 셀 페이로드 바이트들중 27바이트만을 상기 버퍼에 기록할 수 있고 그 다음 정지해야 한다. 도 7에서, 상기 27 가용 바이트 위치들은 "W"로 마킹되어 있다. 상기 읽기 페이즈에 의해 읽혀지는 바이트들의 수는 일정할 필요가 없으며, ATM 셀들은 8바이트 롱워드 저장 유닛 크기의 배수에 한정되지 않고 어떠한 경우의 53바이트 길이면 됨에 따라, 상기 27 바이트 가용 공간의 시작 위치 및 끝나는 위치는 일반적으로 아주 임의의 배열들을 가질 수 있다. 7에 도시된 바와 같이, 기록될 상기 제1 아이들 페이로드 바이트는 롱워드 스토리지 유닛(710)(즉 바이트 주소 WRITE_BASE+2에서)의 오프셋 2에 있으며, 상기 마지막은 그것의 포함하는 롱워드 스토리지 유닛(716)(바이트 어드레스 WRITE_BASE+24+4)의 오프셋 4에 있다.

<152> 본 발명의 일실시예에 따른, 2 인터리빙 롱워드 (712 및 714)의 모든 바이트들로의 아이들 셀 페이로드 바이트 값들을 저장하는 프로세스는 전통적인 저장 방법들을 이용하여 수행될 수 있다. 각 스토리지 유닛의 모든 바이트 위치들은 기록도록 요청되며, 따라서 각각이 각 롱워드 스토리지 유닛에 아이들 셀 페이로드 바이트 값의 전체 8바이트들을 기록하고 있는 두 개의 전통적인 "저장 롱워드(store long-word)" 명령들이 요청된 기능을 수행할 것이다. 8바이트들의 각각, 16진수 56의 모두를 위해 저장될 상기 데이터는 단일 64비트 값, 즉 16진수 5656565656565656로 표현된다.

<153> 메모리(700)의 버퍼의 해당 섹션에 기록되어야 할 27 바이트 시퀀스의 시작 위치와 끝나는 위치에서 상기 짧은(서브-롱 워드(sub-long-word) 크기의) 섹션들을 어떻게 처리할 것인지는 고려할 사항이다. 전체 쓰기 페이즈 기능중 이들 부분들의 효과는, 특히 쓰기 동작들이 이 특정한 예에서와 같이 거의 작은 크기라면, 민감한 것이 될 수 있다.

<154> 추가의 예에서, 이들 두 짧은 섹션들(short sections)의 특정한 크기들 및 오프셋들은 상기 소프트웨어 기능이 실제적으로 수행될 때가 아니라 프로그래밍의 시점에서 미리 알 수 있다. STLM 명령(200)이 없는 경우에, 롱워드(712)에 상기 6개 시작 아이들 레이로드 바이트들을 저장하는 것은 두개의 노멀 저장 명령들, 2바이트값 16진수값 5656을 롱워드 위치(712)의 오프셋 2에서 상기 배열된 반워드(halfword)에 기록하는 저장 반-워드(store half-word) 명령, 및 값 16진수 56565656을 동일 롱워드(712)의 오프셋 4에 배열된 워드에 저장하는 저장 워드 명령을 이용하여 수행될 수 있다. 마찬가지로, 단일 저장 워드 명령(오프셋 0에서 워드에 대한 기록) 및 단일 저장 바이트 명령(바이트 오프셋4에서)은 상기 마지막 5바이트들을 롱워드 (716)에 저장하기 위해 사용될 수 있다. 이 접근방법에서는 각각 7바이트들을 포함하는 기록될 시작 및 마지막 짧은 섹션들에 대하여, 4개의 저장 명령 모두 및 6개까지 필요할 수 있다.

<155> ADSL 모뎀 어플리케이션에서의 소프트웨어 처리의 이 대표적인 구성요소에 대하여, STLM 명령(200)은 이들 간단한 섹션들을 저장해 나가는 비용을 줄이기 위해 사용될 수 있다. 관련 분야들에서의 숙련된 사람은 STLM 명령(200)이 많은 다른 유사 어플리케이션에도 사용될 수 있으며, 상기 ADSL 모뎀 어플리케이션은 예시의 방법으로 설명되었으며 제한되지 않는다는 것을 알 것이다.

<156> 본 발명의 일실시예에 따라, 여기에서 각각 pI(시작) 및 pF(마지막)로 식별된, 상기 프리디케이트 레지스터들(165)의 각 둘중 하나인 상기 시작 및 마지막 짧은 섹션 크기들이 미리 알려지는 경우, 두 각 마스크 값은 미리 구성될 수 있다. 상기 환경들에 따라, 이들 구성된 마스크 값들은 그들의 구성비용이 감가상각되는 것을 허용하는 STLM 명령(200)의 다수의 연속적인 예들을 통해 사용될 수 있다. 제1 프리디케이트 레지스터 pI에서 구성된, 상기 시작 짧은 섹션을 위한 상기 마스크 값은 11111100_2가 될 것이다. 프리디케이트 레지스터 pF의 제2 마스크 값은 00011111_2가 될 것이다. 여기에서는 rICP로 식별된 64비트 범용 레지스터들(155)중 하나는 상기 8바이트 반복된 아이들 셀 페이로드 값, 16진수 5656565656565656을 포함하기 위해 준비될 수 있다. 여기에서 rWB로 식별되는 또 하나의 범용 레지스터(155)는 상기 롱워드 스토리지 유닛(710)(즉, 주소 WRITE_BASE)의 베이스 주소를 포함하기 위해 준비될 수 있다. 상기 시작 6바이트 시퀀스, 단일 STLM 명령(200)의 저장을 구현하는 것이 실행된다. 요청된 동작을 수행하기 위해, 아래와 같이 pI는 프리디케이트 오퍼랜드(202)로, rICP는 소스 오퍼랜드(206)으로, rWB는 베이스 레지스터 오퍼랜드(208)로, 0은 오프셋 오퍼랜드(210)로 사용될 수 있다.

<157> pI.STLM rICP,[rWB,#0]

<158> 따라서, 본 발명의 일실시예에 따라, 단일 단계로, 롱워드 스토리지(710)의 모든 적절 바이트(pertinent byte) 위치들은 필요에 따라 아이들 셀 페이로드 바이트 값과 함께 기록될 수 있다. 대조적으로, 만약 STLM 명령(200)이 이용가능하지 않으면, 두 개의 전통적인 저장 명령들, 더 일반적으로는 3개의 전통적인 저장 명령들만큼의 많은 수가 동일 효과를 달성하기 위해 요청될 것이다. STLM 명령(200)의 제2 예는 pI 대신에 pF를 그것의 제어 마스크 값 프리디케이트 오퍼랜드(202) 및 적절한 어드레싱 오프셋으로 사용하여 단일 단계로 모든 5바이트 마지막 짧은 섹션을 저장하는 것에 의해 상기 프로세스를 완료한다.

<159> pF.STLM rICP[rWB, #24]

<160> 3. TSTCM 명령의 대표적인 어플리케이션

<161> STLM 명령(200)의 상술한 예는 데이터의 임의의 시작 및/또는 마지막 짧은 섹션들의 크기들 및 오프셋들이 코드 실행의 시점 뿐만 아니라 프로그래밍 시점에서 알려져 있는 경우에 대한 전개되었다. 이 조건이 유지될 수 있는 몇몇 어플리케이션에서, 상술된 예에서 사용된 바와 같은 접근방식이 충분할 것이다. 그러나, 좀더 일반적으로 이들 짧은 데이터 섹션들의 크기들 및 배열들은 지금까지 미리 알려지지 않는다. 대신에 종종 해당 정보를 실행하는 특정 소프트웨어가 가능하게 됨에 따라, 상술된 버퍼의 ATM 셀들의 처리중 기록 페이즈와 같은 기능의 경우에 대체적으로 맞게 된다.

<162> 도 5에 있는 TSTCM 명령(500)은 임의의 크기 및 배열을 가지는 짧은 섹션들을 메모리(700)에 저장하는 것을 효과적으로 지원한다. 메모리에 대하여 바이트들의 시퀀스중 임의의 크기 및 배열된 짧은 마지막(또는 단독) 섹션을 저장할 때, 오직 요청된 바이트들의 세트가 STLM 명령(200)의 실행에 의해 저장되게 하는 것은 마스크 값을 생성하는 것에 의해 이루어진다. STLM 예의 전술한 전개는 이 마지막 섹션이 하나의 스토리지 유닛, 즉 크기에 있어 단지 0 .. 7바이트임,에 한정되었다는 것을 주목하라. 사실, 저장 시퀀스의 확장된 섹션을 처리하기 위해 TSTCM 및 STLM 명령들을 조합하여 사용하는 것이 가능하다. 그 측면에 관련된 프레임워크(framework)는 위에서 이미 상술되었다.

<163> 특별한 사용예로, 바이트들중 임의의 수(카운트로서, 전형적으로는 동적 변수로 언급됨)가 상기 시퀀스에서 저장되기 위해 남아있는 환경에서, 상기 STLM 및 TSTCM 명령들은 조합하여 사용된다. 본 발명의 일실시예에 따라, 카운트의 현재 값은 범용 레지스터 rCount에서 유지된다. 본 발명의 또 다른 실시예에서, 이 예에서 16바이트들까지 저장가능한 데이터 값들은, SR0 및 SR1로 식별된 두 범용 레지스터들의 세트의 각각에서 8바이트들로 유지된다. 관련 분야에서 숙련된 사람은, 값들이 레지스터들(SR0 및 SR1)로의 유입되게 하는 방법은 쉽게 드러나며, 나아가 사용된 특정 방법이 상기 STLM 및 TSTCM 명령의 행동에 영향을 미치지 않는다는 것을 인식할 것이다. 본 발명의 더 상세한 실시예에서는 상기 시퀀스에서 현재 저장 위치의 주소(즉, 기록될 다음 바이트가 저장될 곳)는 또 다른 레지스터 rWP에 유지된다. 데이터 바이트들중 상기 요청된 수(카운트, 또는 많아야 16)의 기록을 구현하기 위해, 명령들의 다음 시퀀스가 사용될 수 있다.

<164> TSTCM p0,rCount, #0

<165> p0.STLM SR0,[rWP, #0]

<166> TSTCM p1,rCount,#8

<167> p1.STLM SR1,[rWP,#8]

<168> 프로세서 시스템(100)의 실시예에서, 개선된 성능(명령 실행의 더 적은 사이클로 정의됨)을 위해, TSTCM 및 STLM 동작 쌍이 아래와 같이 병행하여 실행될 수 있음을 주목하라. 본 발명의 일실시예에 따라, ":"의 앞에 있는 제1 명령의 각 라인은 X 측 실행 유닛(예: TSTCM 명령에 대하여는 INTx 유닛(134x)에서, 다음의 STLM 명령에 대하여는 LSUx(140x))상에 실행되고, ":"의 뒤에 있는 상기 제2 명령은 적절한 Y측 실행 유닛에서 동시에 실행될 수 있다.

<169> TSTCM p0,rCount,#0 : TSTCM p1,rCount,#8

<170> p0.STLM SR0,[rWP,#0] : p1.STLM SR1,[rWP,#8]

<171> 4. TSTIM 명령의 대표적인 어플리케이션

<172> 도 6에 도시된 TSTIM 명령(600)은, 상기 저장 시퀀스를 가속화하기 위해, 저장 시퀀스의 시작 짧은 섹션에서 시

작 데이터 바이트들의 임의의 배열의 특정 문맥에 대하여, 가능성을 제공한다. 이 예에서, 전술된 롱워드에 대한 상기 짧은 섹션에서의 제1 기록된 위치의 상대적인 배열은 저장될 상기 제1 데이터 바이트보다 더 낮은 주소에서 어떠한 위치에서라도의 덮어쓰기를 막기 위하여 STLM 명령(200)을 사용하는데 있어 즉각적인 중요성의 위치를 배열했다. 이전의 예(도 7)에서, 주소 WRITE_BASE에서 롱워드 위치(710)에서 W 마킹된 제2(right-most)바이트 위치에 대한 이 오프셋은 2이다. 이러한 맥락에서 rOffset으로 명명된 레지스터로 말해지는 그 오프셋에 대한 값이 계산될 수 있다(수단은 미도시됨). 본 발명의 일실시예에 따르면, 동작에서 TSTIM 명령(600)은 상기 타겟 롱워드 스토리지 유닛의 이 시작 위치에서 또는 이후에 대한 데이터 바이트들만이 STLM 명령(200)에 의해 저장되게 하는 마스크 값 및 베이스 주소(WRITE_BASE)를 포함하는 하기에서 rWB로 식별된 레지스터를, 그 마스크를 그것의 프리디케이트 오퍼랜드(202)로 사용하여 생성한다. 예에서, 그것은 다음의 방식으로 나타낼 수 있다.

<173> TSTLM p0, rOffet, #0

<174> p0.STLM rData,[rWP,#0]

<175> 5. 메모리 시스템 설계 측면들

<176> 메모리 시스템들의 다양한 설계들은 비배열된(non-aligned) 주소에서 메모리에 대한 접속을 위한 서로 다른 레벨의 지원을 제공한다. 기록 접속(저장 동작들)에 관련하여, 상술한 예들 및 STLM 명령(200)의 가능한 어플리케이션들의 설명은 메모리의 관련 클래스들의 최소가능성 모델(즉, 가장 일반적임), 즉 개별 바이트의 레벨쪽으로의 서브유닛들 다운에 대한 저장을 지원하는 동안, 단지 일반 배열된 저장 동작들을 허용하는 것을 가정한다. 이런 가장 한정된 경우에서, 모든 STLM, TSTCM 및 TSTIM 명령들은 임의의 크기들 또는 임의의 배열들에서의 저장 동작의 소프트웨어 효율성을 증가시키는 것에 관련하여 특화된 이익 목적들을 담당한다.

<177> 좀더 가능성 있고, 그렇지만 좀더 복잡한 메모리(700)(또는 프로세서 시스템(100)내에서 그것에 대한 인터페이스)의 구현을 사용하는 또 다른 실시예에서, 비자연 경계들(홀수 바이트 오프셋에서 반워드(half-word)와 같음)에서 데이터의 저장이 가능하다. 본 발명의 추가적인 일실시예에 따른 좀더 가능성 있는 구현예에서, 하나의 롱워드 스토리지 유닛에서의 가장 높은 오프셋 바이트 위치(들)에서 하나 또는 그 이상의 바이트들을 저장하는 것과 같은, 이어지는(다음으로 더 높게 어드레싱된) 롱워드 스토리지 유닛에서의 가장 낮은 오프셋 위치(들)에서 하나 또는 그 이상의 바이트들을 저장하는 그리고, 동일 저장 동작의 일부와 같은 롱워드 경계를 가로지르는 저장 동작들을 수행하는 것이 가능하다. 이러한 설계의 메모리에 대하여, TSTIM 및 STLM의 조합의 단점들은 롱워드 스토리지 유닛 경계들에 관련된 것과 상관없이 기록될 (임의로 배열된) 제1 바이트 주소에서 다른 저장 동작을 개시하는 것이 가능할 수 있기 때문에, 감소된다. 그럼에도 TSTCM 및 STLM 명령들의 조합의 사용은, 마지막 저장 동작이 여전히 임의의 길이로 있을 수 있기 때문에, 저장 동작 또는 저장 동작들의 시퀀스에 관련된 장점을 남길 수 있으며, 따라서 기록될 마지막 바이트를 넘는 바이트들의 마스킹을 필요로 한다. 상술된 바와 같은 STLM 명령(200), 또는 TSTCM 명령(500) 및 STLM 명령(200)의 조합은 메모리 시스템의 좀더 가능성 있는 설계와 함께 사용되어지는 경우에도 저장 동작들의 효율성을 증가시킬 수 있다. 관련 분야의 숙련된 사람은 여기에서 개시된 동작들이 추가적인 메모리 시스템들과 함께 사용되는 경우 마찬가지의 이점이 유도될 수 있다는 것을 인식할 것이다. 상기 개시된 메모리 시스템들은 단지 예시의 방법으로 표현된 것이며 제한되지 않는다.

<178> 상술된 설명은 메모리 어드레싱 중 일반적으로 "리틀 엔디안(little endian)"모드로 명명되는 것에 근거하여 설명되었으며, 여기에서 롱워드 스토리지 유닛의 주소는 최하위 바이트(least significant byte)의 주소와 같은 것으로 정의된다. 본 발명의 또 다른 실시예에서, 동일한 원리들이 다른 "빅 엔디안(big-endian)" 모드를 사용하는 메모리 시스템에 적용될 수 있으며, 여기에서 스토리지 롱워드의 상기 주소는 최상위 바이트(most significant byte)의 주소와 동일하다. 이것은 관련 분야들에서 숙련된 사람에게 손쉽게 자명할 수 있는 상기 설명에서 사용된 구조들과 정의들에 대한 조절들에 의해 이루어질 수 있다.

<179> 6. 결론

<180> 본 발명의 다양한 실시예들이 상술되었으나, 그것은 예로서 표현될 것일 뿐이며 제한하지 않는다. 첨부된 청구항들에 한정된 본 발명의 정신이나 범위에서 벗어나지 않으면서 여기에서 설명된 폼(form) 및 상세한 사항들에서 다양한 변형이 이루어질 수 있음은 관련분야에 숙련된 사람에 이해될 것이다. 따라서, 본 발명의 폭 및 범주는 대표적인 실시예들로 설명된 것들에 한정되지 않아야 하며, 다음의 청구항들 및 그 균등물에 따라 한정되어야 할 것이다.

도면의 간단한 설명

<181> 도 1은 본 발명에서의 사용을 위한 대표적인 프로세서 시스템을 예시한다.

<182> 도 2는 본 발명의 일실시예에 따른 STLM(store long-word under mask) 명령에 대한 대표적인 포맷을 예시한다.

<183> 도 3은 본 발명의 일실시예에 따른 대표적인 프리디케이트 레지스터를 예시한다.

<184> 도 4는 본 발명의 일실시예에 따른 대표적인 STLM 구현을 예시한다.

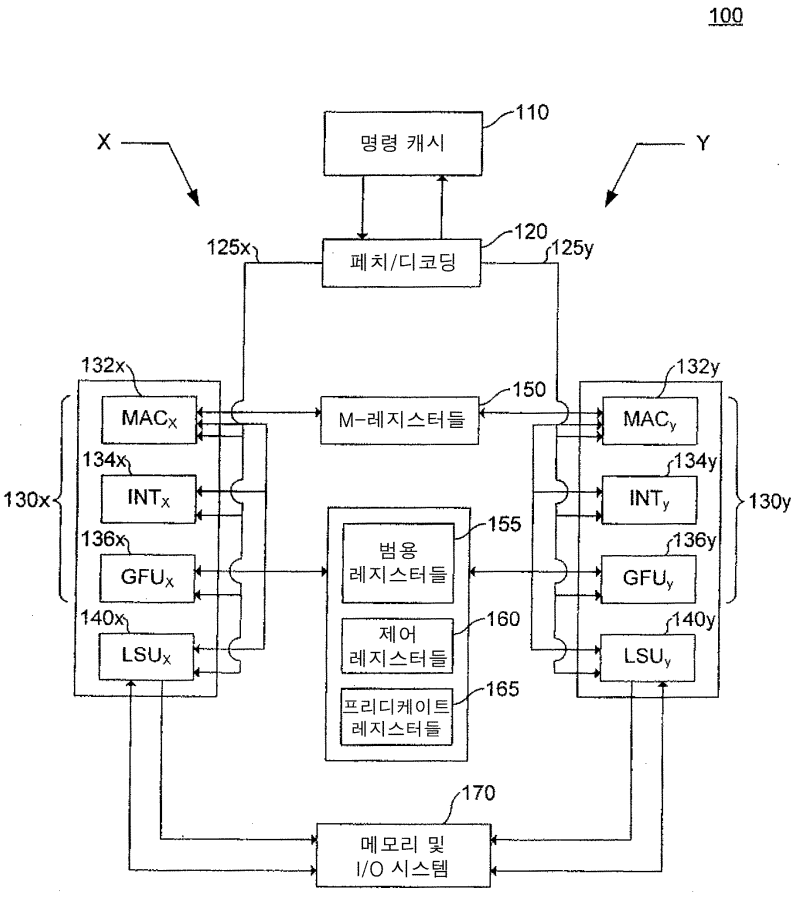
<185> 도 5는 본 발명의 일실시예에 따른 TSTCM(test count to set for STLM) 명령에 대한 대표적인 포맷을 예시한다.

<186> 도 6은 본 발명의 일실시예에 따른, TSTIM(test count to set inverted mask for TSTIM)을 위한 대표적인 포맷을 예시한다.

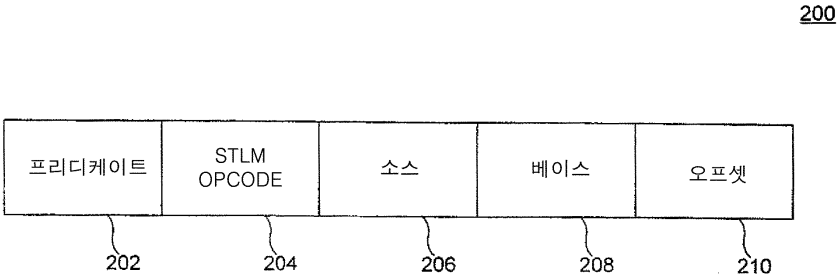
<187> 도 7은 본 발명의 일실시예에 따른 8바이트 폭 롱워드 어드레스 메모리(700)부를 보여준다.

도면

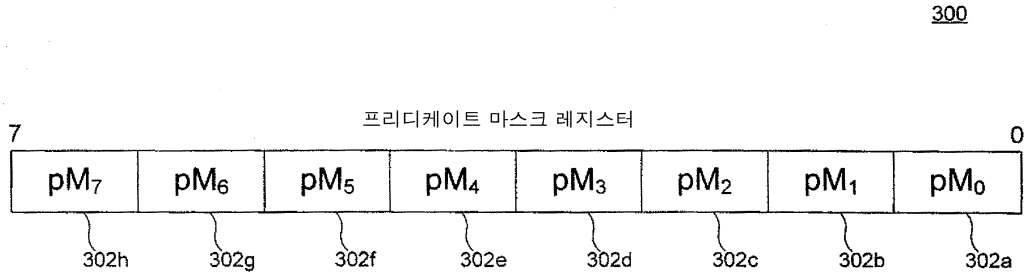
도면1



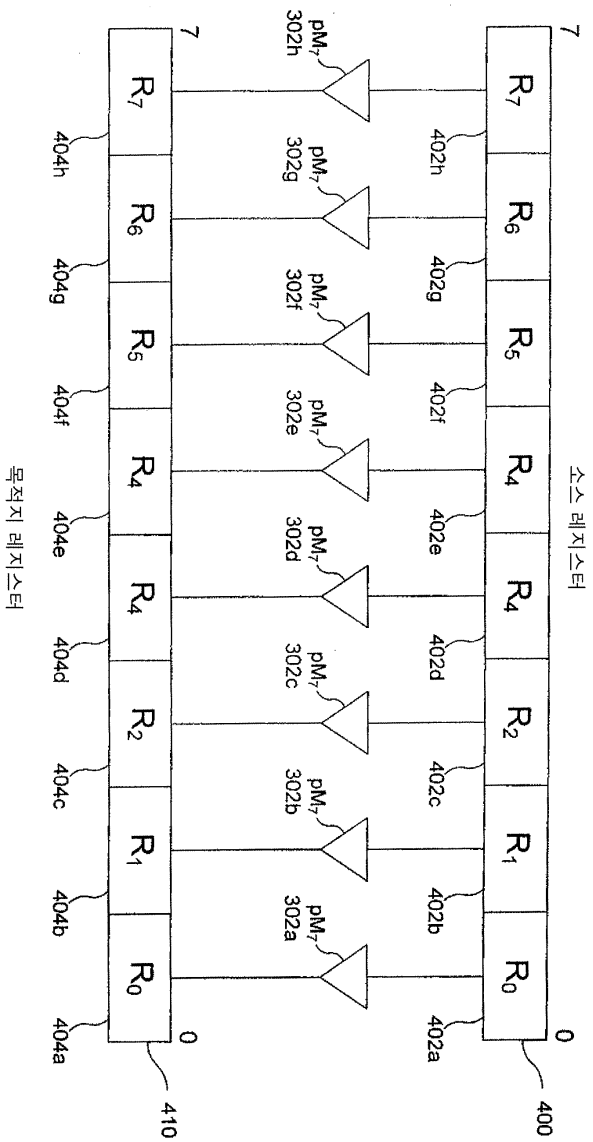
도면2


200
프리디케이트
STLM OPCODE
소스
베이스
오프셋
202
204
206
208
210

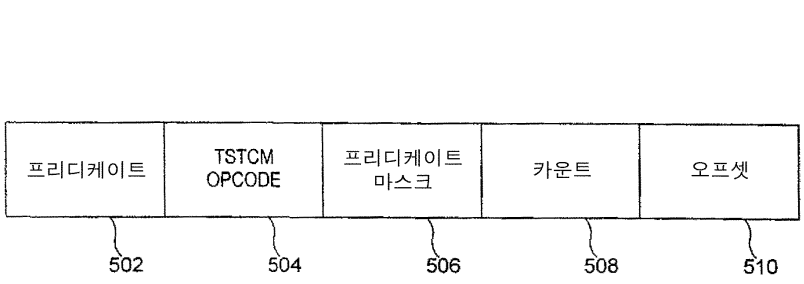
도면3


300
프리디케이트 마스크 레지스터
7
0
pM7
pM6
pM5
pM4
pM3
pM2
pM1
pM0
302h
302g
302f
302e
302d
302c
302b
302a

도면4


소스 레지스터
목적지 레지스터
400
410
402a
402b
402c
402d
402e
402f
402g
402h
404a
404b
404c
404d
404e
404f
404g
404h
302a
302b
302c
302d
302e
302f
302g
302h
pM7
R0
R1
R2
R4
R5
R6
R7
0
7

도면5


500
프리디케이트
TSTCM OPCODE
프리디케이트 마스크
카운트
오프셋
502
504
506
508
510

도면6

600

도면7

700