



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0140968
(43) 공개일자 2024년09월24일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2018.01)
(52) CPC특허분류
G06F 9/30036 (2023.08)
G06F 9/30043 (2013.01)
(21) 출원번호 10-2024-7029477
(22) 출원일자(국제) 2022년12월20일
심사청구일자 없음
(85) 번역문제출일자 2024년09월02일
(86) 국제출원번호 PCT/GB2022/053313
(87) 국제공개번호 WO 2023/148467
국제공개일자 2023년08월10일
(30) 우선권주장
2201544.0 2022년02월07일 영국(GB)

(71) 출원인
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리턴 폴번로드 110
(72) 발명자
봇맨, 프랑수아 크리스토퍼 자크
영국 씨비1 9엔제이 케임브리지셔 케임브리지 폴본 로드 110 씨/오 에이알엠 리미티드
그로켓, 토마스 크리스토퍼
영국 씨비1 9엔제이 케임브리지셔 케임브리지 폴본 로드 110 씨/오 에이알엠 리미티드
(74) 대리인
남호현

전체 청구항 수 : 총 20 항

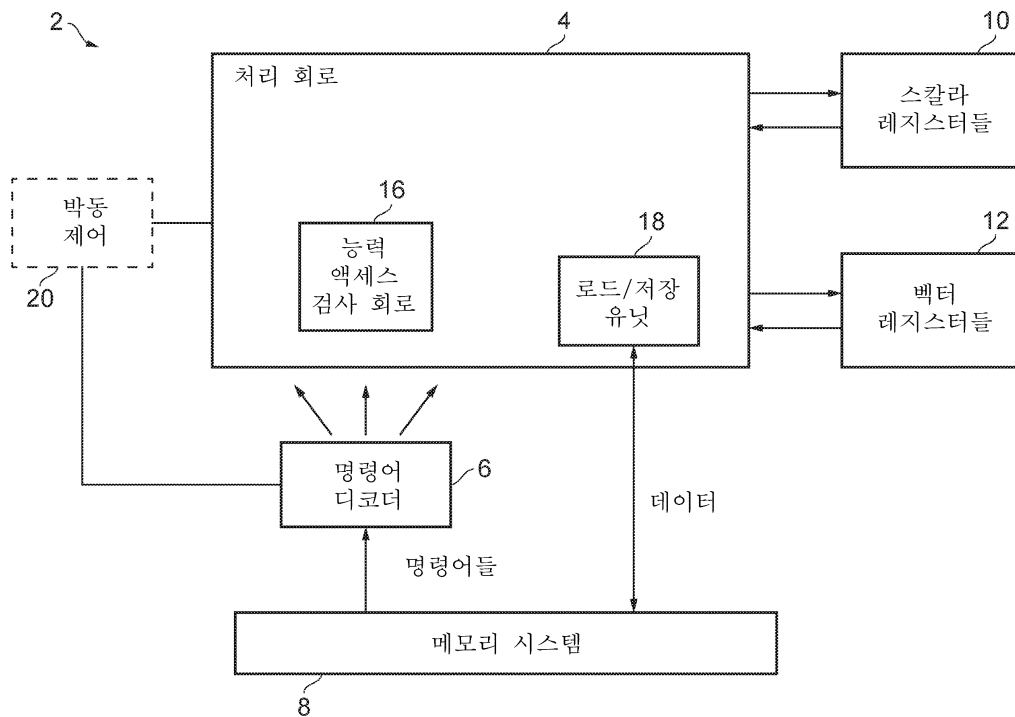
(54) 발명의 명칭 메모리 액세스 동작을 수행하기 위한 기술

(57) 요약

벡터 처리 동작들을 수행하기 위한 처리 회로, 벡터 레지스터들의 세트, 및 벡터 명령어들을 디코딩하여 처리 회로를 제어하여 필요한 동작들을 수행하기 위한 명령어 디코더를 갖는 장치가 설명된다. 명령어 디코더는 주어진 벡터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스

(뒷면에 계속)

대표도 - 도1



동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 주어진 벡터 메모리 액세스 명령어의 데이터 벡터 표시 필드로부터, 복수의 데이터 요소들과 관련된 벡터 레지스터들의 세트 내의 적어도 하나의 벡터 레지스터를 결정하고, 주어진 벡터 메모리 액세스 명령어의 적어도 하나의 능력 벡터 표시 필드로부터, 복수의 능력들을 포함하는 벡터 레지스터들의 세트 내의 복수의 벡터 레지스터들을 결정한다. 각각의 능력은 복수의 데이터 요소들 내의 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 해당 어드레스 표시의 사용을 제약하는 제약 정보를 제공한다. 적어도 하나의 능력 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수는 데이터 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수보다 크다. 명령어 디코더는 처리 회로를 제어하여: 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 관련 능력의 제약 정보와 관련하여 그러한 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고; 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 메모리 액세스 동작의 수행을 가능하게 한다.

(52) CPC특허분류

G06F 9/30101 (2013.01)

G06F 9/30109 (2013.01)

G06F 9/30192 (2013.01)

명세서

청구범위

청구항 1

장치로서,

백터 처리 동작들을 수행하기 위한 처리 회로;

백터 레지스터들의 세트; 및

백터 명령어들을 디코딩하여 상기 처리 회로를 제어하여 상기 백터 명령어들에 의해 지정된 상기 백터 처리 동작들을 수행하기 위한 명령어 디코더를 포함하고;

상기 명령어 디코더는 주어진 백터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스 동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 상기 주어진 백터 메모리 액세스 명령어의 데이터 백터 표시 필드로부터, 복수의 데이터 요소들과 관련된 상기 백터 레지스터들의 세트 내의 적어도 하나의 백터 레지스터를 결정하고, 상기 주어진 백터 메모리 액세스 명령어의 적어도 하나의 능력 백터 표시 필드로부터, 복수의 능력들을 포함하는 상기 백터 레지스터들의 세트 내의 복수의 백터 레지스터들을 결정하며, 각각의 능력은 상기 복수의 데이터 요소들 내의 상기 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 상기 어드레스 표시의 사용을 제약하는 제약 정보를 제공하며, 상기 적어도 하나의 능력 백터 표시 필드로부터 결정된 백터 레지스터들의 수는 상기 데이터 백터 표시 필드로부터 결정된 백터 레지스터들의 수보다 크고;

상기 명령어 디코더는 상기 처리 회로를 제어하여:

상기 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 상기 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 상기 주어진 데이터 요소에 액세스하는 데 사용될 상기 메모리 액세스 동작이 상기 관련 능력의 상기 제약 정보와 관련하여 상기 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고;

상기 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 상기 메모리 액세스 동작의 수행을 가능하게 하도록 추가로 배열되며, 임의의 주어진 데이터 요소에 대한 상기 메모리 액세스 동작의 수행은 상기 주어진 데이터 요소가 상기 메모리 내의 상기 결정된 메모리 어드레스와 상기 적어도 하나의 백터 레지스터 사이에서 이동되게 하는, 장치.

청구항 2

제1항에 있어서, 상기 백터 레지스터들의 세트의 주어진 백터 레지스터들 내의 각각의 능력 크기 블록과 관련하여 유효한 능력 표시 필드를 제공하는 능력 표시 저장소를 추가로 포함하고, 각각의 유효한 능력 표시 필드는 관련 능력 크기 블록이 유효한 능력을 저장하고 그렇지 않으면 소거될 때를 나타내도록 설정되게 배열되는, 장치.

청구항 3

제2항에 있어서, 상기 능력 표시 저장소는 상기 백터 레지스터들의 세트 내에 통합되는, 장치.

청구항 4

제2항 또는 제3항에 있어서, 상기 처리 회로는 상기 장치에 의해 실행 가능한 명령어들의 세트 중 하나 이상의 특정 명령어들의 실행에 응답하여 단지 임의의 유효한 능력 표시 필드가 유효한 능력이 상기 관련 능력 크기 블록에 저장된다는 것을 나타내도록 설정될 수 있게 하도록 배열되는, 장치.

청구항 5

제1항 내지 제4항 중 어느 한 항에 있어서, 상기 적어도 하나의 능력 백터 표시 필드로부터 결정된 상기 복수의

벡터 레지스터들을 형성하는 벡터 레지스터들의 수는 2의 거듭제곱인, 장치.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서, 상기 적어도 하나의 능력 벡터 표시 필드는 하나의 벡터 레지스터를 식별하도록 배열된 단일 능력 표시 필드이고, 상기 명령어 디코더는 결정된 관계에 기초하여 상기 복수의 벡터 레지스터들 중 나머지 벡터 레지스터들을 결정하도록 배열되는, 장치.

청구항 7

제6항에 있어서, 상기 복수의 벡터 레지스터들 내의 벡터 레지스터들의 수는 2^N 이고, 상기 단일 능력 벡터 표시 필드는 상기 하나의 벡터 레지스터를 식별하는 제1 벡터 레지스터 번호를 나타내고, 상기 제1 벡터 레지스터 번호는 논리 0 값에서 그의 N개의 최하위 비트들을 갖도록 제약되고, 상기 명령어 디코더는 상기 제1 벡터 레지스터 번호를 재사용하고 상기 N개의 최하위 비트들 중 적어도 하나를 논리 1 값으로 선택적으로 설정함으로써 상기 나머지 벡터 레지스터들 각각에 대한 벡터 레지스터 번호들을 생성하도록 배열되는, 장치.

청구항 8

제1항 내지 제7항 중 어느 한 항에 있어서, 상기 적어도 하나의 벡터 레지스터 내의 인접 위치들과 관련된 데이터 요소들의 임의의 주어진 쌍에 대해, 상기 관련 능력들은 상기 복수의 벡터 레지스터들 중 상이한 벡터 레지스터들에 저장되는, 장치.

청구항 9

제1항 내지 제8항 중 어느 한 항에 있어서,

상기 데이터 벡터 표시 필드로부터 결정된 상기 적어도 하나의 벡터 레지스터는 단일 벡터 레지스터를 포함하고, 각각의 데이터 요소는 상기 단일 벡터 레지스터의 대응하는 데이터 레인과 관련되고;

각각의 능력은 상기 복수의 벡터 레지스터들 내의 상기 벡터 레지스터들 중 하나 내의 능력 라인 내에 위치되고;

주어진 데이터 요소에 대해, 상기 관련 능력을 포함하는 상기 벡터 레지스터는 상기 대응하는 데이터 레인의 라인 번호의 주어진 수의 최하위 비트들에 따라 결정되고, 상기 관련 능력을 포함하는 상기 능력 레인은 상기 대응하는 데이터 레인의 상기 라인 번호의 나머지 비트들에 따라 결정되는, 장치.

청구항 10

제9항에 있어서,

상기 복수의 능력들을 포함하는 상기 복수의 벡터 레지스터들 내의 벡터 레지스터들의 수는 논리적으로 0 내지 P-1의 값들을 갖는 시퀀스로 간주되는 P이고, 임의의 주어진 벡터 레지스터 내의 능력 라인들의 수는 0 내지 M-1의 값들을 갖는 M이고;

상기 주어진 데이터 요소와 관련된 상기 데이터 레인은 0 내지 X-1의 값들을 갖는 데이터 라인 X이고, 상기 복수의 벡터 레지스터들 내의 상기 관련 능력의 위치는 X를 P로 나누어 몫과 나머지를 생성함으로써 결정되며, 상기 몫은 상기 관련 능력을 포함하는 상기 능력 레인을 식별하고, 상기 나머지는 상기 관련 능력을 포함하는 상기 벡터 레지스터를 식별하는, 장치.

청구항 11

제1항 내지 제10항 중 어느 한 항에 있어서,

상기 벡터 레지스터들의 세트는 논리적으로 복수의 섹션들로 분할되고, 각각의 섹션은 상기 벡터 레지스터들의 세트 내의 상기 벡터 레지스터들 각각으로부터의 대응하는 부분을 포함하고;

상기 복수의 능력들은 상기 복수의 벡터 레지스터들 내에 위치되어, 각각의 데이터 요소에 대해, 상기 관련 능력이 상기 데이터 요소와 동일한 섹션 내에 저장되게 하며;

상기 주어진 벡터 메모리 액세스 명령어의 실행은 다수의 박동들로 분할되며, 각각의 박동 동안에 상기 주어진

백터 메모리 액세스 명령어를 실행하기 위해 상기 백터 레지스터들의 세트의 하나의 섹션만이 액세스되는, 장치.

청구항 12

제11항에 있어서, 상기 처리 회로는 하나 이상의 박동들에 걸쳐, 주어진 섹션 내의 상기 데이터 요소들에 대한 상기 메모리 액세스 동작들을 수행한 후에, 하나 이상의 박동들에 걸쳐, 다음 섹션 내의 상기 데이터 요소들에 대한 상기 메모리 액세스 동작들을 수행하도록 배열되는, 장치.

청구항 13

제1항 내지 제12항 중 어느 한 항에 있어서,

상기 명령어 디코더는 복수의 백터 능력 메모리 전송 명령어들을 디코딩하도록 배열되며, 상기 복수의 백터 능력 메모리 전송 명령어들은 함께 상기 명령어 디코더로 하여금 상기 처리 회로를 제어하여 상기 메모리와 상기 복수의 백터 레지스터들 사이에서 복수의 능력들을 전송하고, 상기 전송 동안 상기 복수의 능력들을 재배열하여 메모리에 상기 복수의 능력들이 순차적으로 저장되고, 상기 복수의 백터 레지스터들에서 상기 복수의 능력들이 디인터리빙되어 상기 메모리에 순차적으로 저장되는 상기 복수 내의 능력들의 임의의 주어진 쌍이 상기 복수의 백터 레지스터들 중 상이한 백터 레지스터들에 저장되게 하는, 장치.

청구항 14

제13항에 있어서, 각각의 백터 능력 메모리 전송 명령어는 각각의 다른 백터 능력 메모리 전송 명령어에 대한 상이한 능력들을 식별하도록 배열되고, 각각의 백터 능력 메모리 전송 명령어는 상기 처리 회로가 액세스 패턴에 의해 지정된 재배열을 수행하면서 상기 식별된 능력들을 전송하게 하는 상기 액세스 패턴을 식별하도록 배열되는, 장치.

청구항 15

제14항에 있어서,

상기 메모리는 다수의 메모리 뱅크들로 형성되고;

각각의 백터 능력 메모리 전송 명령어에 대해, 상기 액세스 패턴은 상기 백터 능력 메모리 전송 명령어가 상기 처리 회로에 의해 실행될 때 상기 메모리 뱅크들 중 2개 이상이 액세스되게 하도록 정의되는, 장치.

청구항 16

제1항 내지 제15항 중 어느 한 항에 있어서,

상기 주어진 백터 메모리 액세스 명령어의 상기 데이터 백터 표시 필드로부터 결정된 상기 적어도 하나의 백터 레지스터는 단일 백터 레지스터를 포함하고, 상기 능력들은 상기 데이터 요소들의 크기의 2배이며, 상기 적어도 하나의 능력 백터 표시 필드로부터 결정된 상기 복수의 백터 레지스터들은 2개의 백터 레지스터들을 포함하는, 장치.

청구항 17

제1항 내지 제16항 중 어느 한 항에 있어서, 상기 주어진 백터 메모리 액세스 명령어는 어드레스 오프셋을 나타내는 즉시 값을 추가로 포함하고, 상기 처리 회로는 상기 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 상기 어드레스 오프셋을 상기 관련 능력에 의해 제공되는 상기 어드레스 표시와 결합함으로써 상기 주어진 데이터 요소의 상기 메모리 어드레스를 결정하도록 배열되는, 장치.

청구항 18

제1항 내지 제17항 중 어느 한 항에 있어서, 상기 주어진 백터 메모리 액세스 명령어는 어드레스 오프셋을 나타내는 즉시 값을 추가로 포함하고, 각각의 주어진 데이터 요소에 대해, 상기 처리 회로는 상기 어드레스 오프셋에 따라 상기 어드레스 표시를 조정함으로써 상기 복수의 백터 레지스터들 내의 상기 관련 능력의 상기 어드레스 표시를 업데이트하도록 배열되는, 장치.

청구항 19

백터 처리 동작들을 수행하기 위한 처리 회로 및 백터 레지스터들의 세트를 제공하는 장치 내에서 메모리 액세스 동작들을 수행하는 방법으로서,

명령어 디코더를 이용하여, 주어진 백터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스 동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 상기 주어진 백터 메모리 액세스 명령어의 데이터 백터 표시 필드로부터, 복수의 데이터 요소들과 관련된 상기 백터 레지스터들의 세트 내의 적어도 하나의 백터 레지스터를 결정하고, 상기 주어진 백터 메모리 액세스 명령어의 적어도 하나의 능력 백터 표시 필드로부터, 복수의 능력들을 포함하는 상기 백터 레지스터들의 세트 내의 복수의 백터 레지스터들을 결정하는 단계 - 각각의 능력은 상기 복수의 데이터 요소들 내의 상기 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 상기 어드레스 표시의 사용을 제약하는 제약 정보를 제공하며, 상기 적어도 하나의 능력 백터 표시 필드로부터 결정된 백터 레지스터들의 수는 상기 데이터 백터 표시 필드로부터 결정된 백터 레지스터들의 수보다 큼 -;

상기 처리 회로를 제어하여:

상기 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 상기 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 상기 주어진 데이터 요소에 액세스하는 데 사용될 상기 메모리 액세스 동작이 상기 관련 능력의 상기 제약 정보와 관련하여 상기 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고;

상기 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 상기 메모리 액세스 동작의 수행을 가능하게 하는 단계를 포함하며, 임의의 주어진 데이터 요소에 대한 상기 메모리 액세스 동작의 수행은 상기 주어진 데이터 요소가 상기 메모리 내의 상기 결정된 메모리 어드레스와 상기 적어도 하나의 백터 레지스터 사이에서 이동되게 하는, 방법.

청구항 20

명령어 실행 환경을 제공하도록 호스트 데이터 처리 장치를 제어하기 위한 컴퓨터 프로그램으로서,

백터 처리 동작들을 수행하기 위한 처리 프로그램 논리;

백터 레지스터들의 세트를 에뮬레이팅하기 위한 백터 레지스터 에뮬레이팅 프로그램 논리; 및

백터 명령어들을 디코딩하여 상기 처리 프로그램 논리를 제어하여 상기 백터 명령어들에 의해 지정된 상기 백터 처리 동작들을 수행하기 위한 명령어 디코드 프로그램 논리를 포함하고;

상기 명령어 디코드 프로그램 논리는 주어진 백터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스 동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 상기 주어진 백터 메모리 액세스 명령어의 데이터 백터 표시 필드로부터, 복수의 데이터 요소들과 관련된 상기 백터 레지스터들의 세트 내의 적어도 하나의 백터 레지스터를 결정하고, 상기 주어진 백터 메모리 액세스 명령어의 적어도 하나의 능력 백터 표시 필드로부터, 복수의 능력들을 포함하는 상기 백터 레지스터들의 세트 내의 복수의 백터 레지스터들을 결정하며, 각각의 능력은 상기 복수의 데이터 요소들 내의 상기 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 상기 어드레스 표시의 사용을 제약하는 제약 정보를 제공하며, 상기 적어도 하나의 능력 백터 표시 필드로부터 결정된 백터 레지스터들의 수는 상기 데이터 백터 표시 필드로부터 결정된 백터 레지스터들의 수보다 크고;

상기 명령어 디코드 프로그램 논리는 상기 처리 프로그램 논리를 제어하여:

상기 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 상기 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 상기 주어진 데이터 요소에 액세스하는 데 사용될 상기 메모리 액세스 동작이 상기 관련 능력의 상기 제약 정보와 관련하여 상기 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고;

상기 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 상기 메모리 액세스 동작의 수행을 가능하게 하도록 추가로 배열되며, 임의의 주어진 데이터 요소에 대한 상기 메모리 액세스 동작의 수행은 상기 주어진 데이터 요소가 상기 메모리 내의 상기 결정된 메모리 어드레스와 상기 적어도 하나의 백터 레지스터 사이에서 이

동되게 하는, 컴퓨터 프로그램.

발명의 설명

배경 기술

- [0001] 본 기술은 데이터 처리 분야에 관한 것이며, 더 상세하게는 메모리 액세스 동작의 처리에 관한 것이다.
- [0002] 주어진 벡터 명령어에 의해 정의된 동작이 데이터 요소들의 벡터 내의 다수의 데이터 요소에 대해 독립적으로 수행되도록 하기 위해 그러한 주어진 벡터 명령어가 실행될 수 있게 함으로써 코드 밀도 그리고 종종 성능을 개선하려고 시도하는 벡터 처리 시스템이 개발되었다. 따라서, 메모리 액세스 동작의 맥락에서, 벡터 로드 명령어에 응답하여 메모리로부터 지정된 벡터 레지스터로 복수의 연속 데이터 요소들을 로드하거나, 벡터 저장 명령어에 응답하여 지정된 벡터 레지스터로부터 복수의 연속 데이터 요소들을 메모리에 저장하는 것이 가능하다. 이러한 벡터 로드 또는 저장 명령어의 벡터 수집 또는 벡터 분산 변형을 제공하여, 처리된 데이터 요소가 메모리 내의 임의의 위치에 상주할 수 있도록 하는 것이 가능하다. 이러한 벡터 수집 또는 벡터 분산 명령어를 사용할 때, 벡터가 처리될 복수의 데이터 요소에 대해 식별되는 것에 더하여, 각각의 데이터 요소의 메모리 어드레스를 결정하기 위해 사용되는 복수의 어드레스 표시를 제공하기 위해 벡터가 식별될 수도 있다.
- [0003] 주어진 프로세스에 대해 특정 능력이 정의되는 능력 기반 아키텍처에 대한 관심이 증가하고 있으며, 정의된 능력 밖에서 동작을 수행하려는 시도가 있을 경우 예러가 트리거될 수 있다. 능력은 다양한 형태를 취할 수 있지만, 능력의 한 가지 유형은 ("팻 포인터(fat pointer)"로도 지칭될 수 있는) 유계 포인터(bounded pointer)이다.
- [0004] 각각의 능력은 해당 능력을 사용할 때 수행될 수 있는 동작들을 제한하는 데 사용되는 제약 정보를 포함할 수 있다. 예를 들어, 유계 포인터를 고려하면, 이는 관련 허가들을 식별하는 하나 이상의 허가 플래그와 함께, 해당 능력을 사용할 때 처리 회로에 의해 액세스 가능한 메모리 어드레스들의 확장 불가능한 범위를 식별하는 데 사용되는 정보를 제공할 수 있다.
- [0005] 벡터 수집 또는 벡터 분산 명령어의 실행을 지원하는 것이 바람직하지만, 능력의 사용을 통해 제공되는 보안 이익을 활용하기 위해 다양한 어드레스 표시를 능력에 의해 지정될 수 있도록 하는 것이 바람직하다. 그러나, 어드레스 표시를 제공하는 능력은 본질적으로, 능력을 형성하기 위해 어드레스 표시와 관련하여 제공되는 제약 정보로 인해, 동등한 표준 어드레스 표시보다 크다.

발명의 내용

- [0006] 제1 예시적인 배열에서, 벡터 처리 동작들을 수행하기 위한 처리 회로; 벡터 레지스터들의 세트; 및 벡터 명령어들을 디코딩하여 처리 회로를 제어하여 벡터 명령어들에 의해 지정된 벡터 처리 동작들을 수행하기 위한 명령어 디코더를 포함하는 장치가 제공되며; 명령어 디코더는 주어진 벡터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스 동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 주어진 벡터 메모리 액세스 명령어의 데이터 벡터 표시 필드로부터, 복수의 데이터 요소들과 관련된 벡터 레지스터들의 세트 내의 적어도 하나의 벡터 레지스터를 결정하고, 주어진 벡터 메모리 액세스 명령어의 적어도 하나의 능력 벡터 표시 필드로부터, 복수의 능력들을 포함하는 벡터 레지스터들의 세트 내의 복수의 벡터 레지스터들을 결정하며, 각각의 능력은 복수의 데이터 요소들 내의 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 해당 어드레스 표시의 사용을 제약하는 제약 정보를 제공하며, 적어도 하나의 능력 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수는 데이터 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수보다 크고; 명령어 디코더는 처리 회로를 제어하여: 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 관련 능력의 제약 정보와 관련하여 그러한 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고; 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 메모리 액세스 동작의 수행을 가능하게 하도록 추가로 배열되며, 임의의 주어진 데이터 요소에 대한 메모리 액세스 동작의 수행은 그러한 주어진 데이터 요소가 메모리 내의 결정된 메모리 어드레스와 적어도 하나의 벡터 레지스터 사이에서 이동되게 한다.
- [0007] 추가의 예시적인 배열에서, 벡터 처리 동작들을 수행하기 위한 처리 회로 및 벡터 레지스터들의 세트를 제공하는 장치 내에서 메모리 액세스 동작들을 수행하는 방법이 제공되며, 방법은 명령어 디코더를 이용하여, 주어진 벡터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스

동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 주어진 벡터 메모리 액세스 명령어의 데이터 벡터 표시 필드로부터, 복수의 데이터 요소들과 관련된 벡터 레지스터들의 세트 내의 적어도 하나의 벡터 레지스터를 결정하고, 주어진 벡터 메모리 액세스 명령어의 적어도 하나의 능력 벡터 표시 필드로부터, 복수의 능력들을 포함하는 벡터 레지스터들의 세트 내의 복수의 벡터 레지스터들을 결정하는 단계 - 각각의 능력은 복수의 데이터 요소들 내의 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 해당 어드레스 표시의 사용을 제약하는 제약 정보를 제공하며, 적어도 하나의 능력 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수는 데이터 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수보다 큼 -; 처리 회로를 제어하여: 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 관련 능력의 제약 정보와 관련하여 그러한 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고; 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 메모리 액세스 동작의 수행을 가능하게 하는 단계를 포함하며, 임의의 주어진 데이터 요소에 대한 메모리 액세스 동작의 수행은 그러한 주어진 데이터 요소가 메모리 내의 결정된 메모리 어드레스와 적어도 하나의 벡터 레지스터 사이에서 이동되게 한다.

[0008]

또 다른 추가의 예시적인 배열에서, 명령어 실행 환경을 제공하도록 호스트 데이터 처리 장치를 제어하기 위한 컴퓨터 프로그램이 제공되며, 컴퓨터 프로그램은 벡터 처리 동작들을 수행하기 위한 처리 프로그램 논리; 벡터 레지스터들의 세트를 에뮬레이팅하기 위한 벡터 레지스터 에뮬레이팅 프로그램 논리; 및 벡터 명령어들을 디코딩하여 처리 프로그램 논리를 제어하여 벡터 명령어들에 의해 지정된 벡터 처리 동작들을 수행하기 위한 명령어 디코드 프로그램 논리를 포함하고; 명령어 디코드 프로그램 논리는 주어진 벡터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스 동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 주어진 벡터 메모리 액세스 명령어의 데이터 벡터 표시 필드로부터, 복수의 데이터 요소들과 관련된 벡터 레지스터들의 세트 내의 적어도 하나의 벡터 레지스터를 결정하고, 주어진 벡터 메모리 액세스 명령어의 적어도 하나의 능력 벡터 표시 필드로부터, 복수의 능력들을 포함하는 벡터 레지스터들의 세트 내의 복수의 벡터 레지스터들을 결정하며, 각각의 능력은 복수의 데이터 요소들 내의 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 해당 어드레스 표시의 사용을 제약하는 제약 정보를 제공하며, 적어도 하나의 능력 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수는 데이터 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수보다 크고; 명령어 디코드 프로그램 논리는 처리 프로그램 논리를 제어하여: 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 관련 능력의 제약 정보와 관련하여 그러한 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고; 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 메모리 액세스 동작의 수행을 가능하게 하도록 추가로 배열되며, 임의의 주어진 데이터 요소에 대한 메모리 액세스 동작의 수행은 그러한 주어진 데이터 요소가 메모리 내의 결정된 메모리 어드레스와 적어도 하나의 벡터 레지스터 사이에서 이동되게 한다.

[0009]

또 다른 추가의 예시적인 배열에서, 벡터 처리 동작들을 수행하기 위한 처리 수단; 벡터 레지스터 수단의 세트; 및 벡터 명령어들을 디코딩하여 처리 수단을 제어하여 벡터 명령어들에 의해 지정된 벡터 처리 동작들을 수행하기 위한 명령어 디코드 수단을 포함하는 장치가 제공되며; 명령어 디코드 수단은 주어진 벡터 메모리 액세스 명령어가 복수의 메모리 액세스 동작들을 지정하는 것에 응답하여 - 각각의 메모리 액세스 동작은 관련 데이터 요소에 액세스하기 위해 수행됨 -, 주어진 벡터 메모리 액세스 명령어의 데이터 벡터 표시 필드로부터, 복수의 데이터 요소들과 관련된 벡터 레지스터 수단의 세트 내의 적어도 하나의 벡터 레지스터 수단을 결정하고, 주어진 벡터 메모리 액세스 명령어의 적어도 하나의 능력 벡터 표시 필드로부터, 복수의 능력들을 포함하는 벡터 레지스터 수단의 세트 내의 복수의 벡터 레지스터 수단을 결정하며, 각각의 능력은 복수의 데이터 요소들 내의 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 해당 어드레스 표시의 사용을 제약하는 제약 정보를 제공하며, 적어도 하나의 능력 벡터 표시 필드로부터 결정된 벡터 레지스터 수단의 수는 데이터 벡터 표시 필드로부터 결정된 벡터 레지스터 수단의 수보다 크고; 명령어 디코드 수단은 처리 수단을 제어하여: 복수의 데이터 요소들 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 어드레스 표시에 기초하여 메모리 어드레스를 결정하고, 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 관련 능력의 제약 정보와 관련하여 그러한 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하고; 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 메모리 액세스 동작의 수행을 가능하게 하도록 추가로 배열되며, 임의의 주어진 데이터 요소에 대한 메모리 액세스 동작의 수행은 그러한 주어진 데이터 요소가 메모리 내의 결정된 메모리 어드레스와 적어도 하나의 벡터 레지스터 사이에서 이동되게 한다.

도면의 간단한 설명

[0010]

본 기술은 첨부 도면에 예시된 바와 같은 그의 예들을 참조하여, 단지 예시로서, 추가로 설명될 것이다.

도 1은 하나의 예시적인 구현에 따른 장치의 블록도이다.

도 2는 하나의 예시적인 구현에 따른, 능력들과 관련하여 태그 비트의 사용을 예시한다.

도 3a 및 도 3b는 하나의 예시적인 구현에 따른, 해당 능력 크기 블록(capability sized block)이 유효한 능력을 저장하는지 여부를 나타내기 위해 벡터 레지스터의 각각의 능력 크기 블록과 관련하여 유효한 능력 표시(이는 일례에서 태그 비트의 형태를 취함)가 저장될 수 있는 상이한 방법들을 예시한다.

도 4a 및 도 4b는 하나의 예시적인 구현에 따른, 벡터 레지스터의 각각의 능력 크기 블록과 관련하여 유지되는 태그 비트가 관리될 수 있는 방법을 예시하는 흐름도이다.

도 5a는 하나의 예시적인 구현에 따른, 벡터 메모리 액세스 명령어 내에 제공될 수 있는 필드를 예시하는 반면, 도 5b는 하나의 예시적인 구현에 따른, 이러한 벡터 메모리 액세스 명령어를 실행할 때 수행되는 단계를 예시하는 흐름도이다.

도 6a 및 도 6b는 하나의 예시적인 구현에 따른, 수집 및 분산 동작을 수행할 때 사용되는 필요한 능력을 보유하는 다수의 벡터 레지스터를 결정하는 데 사용될 수 있는 기술들을 예시하는 흐름도이다.

도 7은 하나의 예시적인 구현에 따른, 벡터 레지스터 세트가 다수의 섹션으로 논리적으로 분할될 수 있는 방법을 개략적으로 예시한다.

도 8a 내지 도 8c는 본 명세서에 설명된 유형의 수집 또는 분산 동작을 수행할 때 사용될 수 있는 데이터 요소 및 관련 능력의 특정한 예시적인 배열을 예시한다.

도 9는 하나의 예시적인 구현에 따른, 각각의 데이터 요소에 대한 관련 능력이 결정될 수 있는 방법을 예시하는 흐름도이다.

도 10은 벡터 명령어들의 중첩 실행의 예를 도시한다.

도 11은 상이한 프로세서 구현들 사이에서 또는 명령어 실행의 상이한 인스턴스들 사이에 런타임에서 연속적인 벡터 명령어들 간의 중첩의 양을 스케일링하는 세 가지 예를 도시한다.

도 12는 능력들이 본 명세서에서 설명된 방식으로 수집 및 분산 동작을 수행할 때 그들의 사용을 허용하는 배열로 복수의 벡터 레지스터 내에 저장되는 것을 보장하는 방식으로 메모리와 벡터 레지스터 사이에서 능력들을 이동시키기 위해 벡터 능력 메모리 전송 명령어 시퀀스가 하나의 예시적인 구현에서 사용될 수 있는 방법을 예시하는 흐름도이다.

도 13은 본 명세서에 설명된 기술들에 따른, 메모리와 벡터 레지스터 사이에서 능력들을 전송하기 위해 벡터 능력 메모리 전송 명령어 시퀀스를 사용할 때 상이한 메모리 뱅크가 액세스될 수 있는 방법을 개략적으로 예시한다.

도 14는 사용될 수 있는 시뮬레이터 예를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0011]

본 명세서에 설명된 기술들에 따르면, 벡터 처리 동작을 수행하기 위한 처리 회로, 벡터 레지스터 세트, 및 벡터 명령어에 의해 지정된 벡터 처리 동작을 수행하도록 처리 회로를 제어하기 위해 벡터 명령어를 디코딩하기 위한 명령어 디코더를 갖는 장치가 제공된다. 벡터 명령어에 의해 지정된 벡터 처리 동작은 벡터 내의 복수의 데이터 요소 각각에 대해 독립적으로 필요한 동작을 수행함으로써 구현될 수 있고, 이러한 필요한 동작들은 병렬로, 순차적으로 하나씩 또는 그룹으로 수행될 수 있다(예를 들어, 그룹 내의 동작들은 병렬로 수행될 수 있고, 각각의 그룹은 순차적으로 수행될 수 있음).

[0012]

명령어 디코더는 복수의 메모리 액세스 동작을 지정하는 주어진 벡터 메모리 액세스 명령어를 처리하도록 배열될 수 있으며, 각각의 메모리 액세스 동작은 관련 데이터 요소에 액세스하기 위해 수행되고, 따라서 복수의 메모리 액세스 동작은 집합적으로 벡터 메모리 액세스 명령어에 의해 지정된 벡터 메모리 액세스 동작을 구현하는 것으로 간주될 수 있다. 특히, 이러한 주어진 벡터 메모리 액세스 명령어에 응답하여, 명령어 디코더는 주어진

벡터 메모리 액세스 명령어의 데이터 벡터 표시 필드로부터, 복수의 데이터 요소와 관련된 벡터 레지스터 세트 내의 적어도 하나의 벡터 레지스터를 결정하도록 배열될 수 있다. 따라서, 데이터 벡터 표시 필드로부터 결정된 각각의 벡터 레지스터는 예를 들어, 해당 소스 레지스터로부터의 데이터 요소들을 메모리의 다양한 위치들에 저장하려고 시도하는 벡터 분산 동작을 위한 소스 레지스터를 형성할 수 있거나, 해당 벡터 레지스터 내의 저장을 위해 메모리의 다양한 위치들로부터 데이터 요소들을 로드하려고 시도하는 벡터 수집 동작을 위한 목적지 레지스터로서 작용할 수 있다.

[0013] 명령어 디코더는 또한 주어진 벡터 메모리 액세스 명령어의 적어도 하나의 능력 벡터 표시 필드로부터, 복수의 능력을 포함하는 벡터 레지스터 세트 내의 복수의 벡터 레지스터를 결정하도록 배열된다. 하나의 예시적인 구현에서, 단일 능력 벡터 표시 필드가 사용되고, 복수의 벡터 레지스터는 해당 단일 능력 벡터 표시 필드 내의 정보로부터 결정된다. 그러나, 대안적인 구현에서는, 다수의 능력 벡터 표시 필드가 제공되어, 예를 들어 각각의 능력 벡터 표시 필드가 대응하는 벡터 레지스터를 식별하는 것을 가능하게 할 수 있다. 하나의 예시적인 구현에서, 복수의 벡터 레지스터의 각각의 벡터 레지스터는 복수의 능력을 포함하는 반면에, 다른 예에서 복수의 벡터 레지스터의 각각의 벡터 레지스터는 단일 능력을 포함한다.

[0014] 결정된 복수의 벡터 레지스터 내의 각각의 능력은 복수의 데이터 요소 내의 데이터 요소들 중 하나와 관련되고, 어드레스 표시 및 메모리에 액세스할 때 해당 어드레스 표시의 사용을 제약하는 제약 정보를 제공한다. 제약 정보는 다양한 형태를 취할 수 있지만, 예를 들어 능력에 의해 제공되는 어드레스 표시를 사용할 때 액세스될 수 있는 메모리 어드레스들의 허용 범위를 결정하는 데 사용되는 범위 정보, 및/또는 어드레스 표시를 사용하여 수행될 수 있는 액세스 유형들(예를 들어, 판독 액세스가 허용되는지 여부, 기입 액세스가 허용되는지 여부, 능력을 사용하여 인출 및 실행할 명령어의 메모리 어드레스를 생성할 수 있는지 여부, 특정 보안 또는 특권 레벨에서 액세스가 허용되는지 여부 등)을 지정하는 하나 이상의 허가 속성을 식별할 수 있다. 추가적인 예에서, 제약 정보는 제약 정보 세트 내의 엔트리를 나타내는 값을 식별하는 제약일 수 있다. 제약 정보 세트 내의 각각의 엔트리는 다양한 형태를 취할 수 있지만, 예를 들어 능력에 의해 제공되는 어드레스 표시를 사용할 때 액세스될 수 있는 메모리 어드레스들의 허용 범위를 결정하는 데 사용되는 범위 정보, 및/또는 어드레스 표시를 사용하여 수행될 수 있는 액세스 유형들(예를 들어, 판독 액세스가 허용되는지 여부, 기입 액세스가 허용되는지 여부, 능력을 사용하여 인출 및 실행할 명령어의 메모리 어드레스를 생성할 수 있는지 여부, 특정 보안 또는 특권 레벨에서 액세스가 허용되는지 여부 등)을 지정하는 하나 이상의 허가 속성을 식별할 수 있다. 일부 구현들에서, 생성된 메모리 어드레스는 메모리 시스템 내의 위치에 직접 대응하는 물리적 메모리 어드레스일 수 있는 반면, 다른 구현들에서, 생성된 메모리 어드레스는 액세스할 물리적 메모리 어드레스를 결정하기 위해 어드레스 변환이 수행될 필요가 있을 수 있는 가상 어드레스일 수 있다.

[0015] 본 명세서에 설명된 기술들에 따르면, 적어도 하나의 능력 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수는 데이터 벡터 표시 필드로부터 결정된 벡터 레지스터들의 수보다 크다.

[0016] 명령어 디코더는 복수의 데이터 요소 내의 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 어드레스 표시에 기초하여 메모리 어드레스(가상 어드레스 또는 물리적 어드레스일 수 있음)를 결정하고, 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 관련 능력의 제약 정보와 관련하여 그러한 결정된 메모리 어드레스에 대해 허용되는지 여부를 결정하기 위해 처리 회로를 제어하도록 추가로 배열된다. 앞서 언급한 바와 같이, 제약 정보는 다양한 형태를 취할 수 있으며, 따라서 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 허용되는지 여부를 결정하기 위해 여기서 수행되는 검사는 다양한 형태를 취할 수 있다. 따라서, 이러한 검사는 예를 들어, 능력 내의 임의의 범위 제약 정보가 주어지면, 결정된 메모리 어드레스가 액세스될 수 있는지 여부를 식별할 수 있을 뿐만 아니라, 액세스 유형이 허용되는지 여부(예를 들어, 액세스 동작이 메모리에 기입을 수행하는 경우, 능력 내의 제약 정보가 이러한 기입의 수행을 허용하는지 여부)를 결정할 수도 있다.

[0017] 그 다음, 처리 회로는 메모리 액세스 동작이 허용되는 각각의 데이터 요소에 대한 메모리 액세스 동작의 수행을 가능하게 하도록 배열될 수 있으며, 임의의 주어진 데이터 요소에 대한 메모리 액세스 동작의 수행은 그러한 주어진 데이터 요소가 메모리 내의 결정된 메모리 어드레스와 적어도 하나의 벡터 레지스터 사이에서 이동되게 한다(이동 방향은 데이터가 메모리에서 레지스터로 로드되거나 레지스터에서 메모리로 저장되는지에 의존한다는 것을 이해한다). 하나의 예시적인 구현에서, 원래 위치에 있는 주어진 데이터 요소는 이 프로세스 동안 그대로 남겨질 수 있으며, 따라서 이 경우 이동 동작은 주어진 데이터 요소를 복사함으로써 수행될 수 있다. 예를 들어, 이것은 전형적으로 적어도 벡터 레지스터 내의 저장을 위해 메모리로부터 데이터 요소를 로드하는 경우일

수 있으며, 그러면 벡터 레지스터 내에 저장된 데이터 요소는 메모리에 저장된 데이터 요소의 사본이다.

[0018] 하나의 예시적인 구현에서는 메모리 액세스 동작들이 이러한 메모리 액세스 동작들이 허용되는 각각의 데이터 요소에 대해 수행될 수 있지만, 다른 구현들에서는 다른 메모리 액세스 동작들이 허용되지 않는 경우들에서 하나 이상의 허용된 메모리 액세스 동작의 수행을 억제하도록 결정될 수 있다. 이러한 상황에서 정확히 어떤 허용 가능한 액세스가 억제되는지는 구현, 및 관련 액세스가 허용되지 않는 데이터 요소의 데이터 요소 벡터 내의 위치에 의존할 수 있다. 순전히 예시적인 예로서, 다양한 액세스들이 순차적으로 수행될 수 있고, 따라서 허용되지 않는 하나의 액세스가 검출될 때, 후속 액세스들을 이들의 허용 여부에 관계없이, 그러나 이전 액세스들이 이미 수행된 경우에는 억제하기로 결정될 수 있다.

[0019] 하나의 예시적인 구현에서, 벡터 레지스터 내에 저장된 유효한 능력을 추적하기 위한 메커니즘이 제공된다. 특히, 하나의 예시적인 구현에서, 장치는 벡터 레지스터 세트의 주어진 벡터 레지스터 내의 각각의 능력 크기 블록과 관련하여 유효한 능력 표시 필드를 제공하는 능력 표시 저장소를 추가로 포함하고, 각각의 유효한 능력 표시 필드는 관련 능력 크기 블록이 유효한 능력을 저장하고 그렇지 않으면 소거될 때를 나타내도록 설정되게 배열된다. 하나의 예시적인 구현에서는 벡터 레지스터 세트 내의 벡터 레지스터들 중 어느 하나가 능력을 저장하는 것이 가능할 수 있지만, 다른 예시적인 구현에서는 능력을 저장하기 위한 능력이 세트 내의 벡터 레지스터 서브세트로 제한될 수 있으며, 후자의 경우, 능력 표시 저장소는 해당 벡터 레지스터 서브세트 내의 각각의 능력 크기 블록에 대한 유효한 능력 표시 필드를 제공하는 것만이 필요할 것이다.

[0020] 하나의 예시적인 구현에서는 능력 표시 저장소가 벡터 레지스터 세트에 개별적으로 제공될 수 있지만, 대안적인 예시적인 구현에서는 능력 표시 저장소가 벡터 레지스터 세트 내에 통합될 수 있다.

[0021] 유효한 능력 표시 필드들이 설정되는 방법을 제약하기 위해, 처리 회로는 장치에 의해 실행 가능한 명령어 세트 중 하나 이상의 특정 명령어의 실행에 응답하여 단지 임의의 유효한 능력 표시 필드가 유효한 능력이 관련 능력 크기 블록에 저장된다는 것을 나타내도록 설정될 수 있게 하도록 배열될 수 있다. 이러한 방식으로 유효한 능력 표시 필드의 설정을 제한함으로써, 이것은 예를 들어 벡터 레지스터 내의 범용 데이터의 능력 크기 블록이 능력으로 취급되어야 함을 나타내려는 시도를 금지함으로써 보안을 향상시킬 수 있다. 따라서, 비능력 동작을 통해 또는 유효하지 않게 하는 방식으로 능력을 변경함으로써 유효한 능력을 생성하지 않는, 벡터에 대해 수행되는 동작들은 관련된 유효한 능력 표시 필드가 소거되게 하여 유효한 능력이 그 안에 저장되어 있지 않음을 나타내도록 배열될 수 있다. 따라서, 예로서, 데이터의 능력 크기 블록에 대한 부분적 기입 또는 비능력의 기입은 관련된 유효한 능력 표시 필드를 소거할 것이다. 능력 표시 필드는 또한 다양한 비명령어 동작, 예를 들어 예외 처리와 관련된 벡터 레지스터 상태의 스택킹(stack) 및 소거, 또는 일부 구현들에서는 재설정 동작에 의해 소거될 수 있다.

[0022] 앞서 언급한 바와 같이, 전술한 주어진 벡터 메모리 액세스 명령어를 실행할 때 필요한 능력을 제공하는 데 사용되는 벡터 레지스터들의 수는 메모리 액세스 동작의 대상이 되는 데이터 요소를 포함하는 벡터 레지스터들의 수보다 크다. 하나의 예시적인 구현에서, 적어도 하나의 능력 벡터 표시 필드로부터 결정된 복수의 벡터 레지스터를 형성하는 벡터 레지스터들의 수는 2의 거듭제곱이다. 특히, 능력을 저장하는 데 필요한 벡터 레지스터들의 수는 데이터 요소와 능력 간의 크기 차이에 따라 달라지며, 하나의 예시적인 구현에서는 그러한 차이가 2의 거듭제곱에 따라 달라질 수 있다. 여기서 유의해야 할 점은, 능력의 크기를 고려할 때, 능력이 유효한 능력 임을 나타내기 위해 사용되는 어떠한 관련 플래그(예컨대, 앞서 언급한 유효한 능력 표시 필드)도 능력 자체의 일부로 간주되지 않는다는 점이다.

[0023] 앞서 언급한 바와 같이, 원하는 경우, 다수의 능력 벡터 표시 필드를 사용하여 주어진 벡터 메모리 액세스 명령어를 실행할 때 필요한 능력을 저장하는 다양한 벡터 레지스터들을 지정할 수 있다. 이러한 접근법은 다양한 벡터 레지스터가 서로에 대해 임의로 위치되고 명령어 인코딩에서 지정될 수 있게 한다. 그러나, 하나의 예시적인 구현에서, 적어도 하나의 능력 벡터 표시 필드는 하나의 벡터 레지스터를 식별하도록 배열된 단일 능력 벡터 표시 필드이고, 명령어 디코더는 결정된 관계에 기초하여 복수의 벡터 레지스터 중 나머지 벡터 레지스터들을 결정하도록 배열된다. 이러한 접근법은 명령어 인코딩 관점에서 유리할 수 있는데, 이는 전형적으로 명령어 인코딩 공간이 상당히 제한적이고, 필요한 능력을 저장할 벡터 레지스터 각각을 식별하기 위해 다수의 능력 벡터 표시 필드를 제공하는 것이 실용적이지 않을 수 있기 때문이다.

[0024] 식별된 하나의 벡터 레지스터 및 결정된 관계에 기초하여 나머지 벡터 레지스터가 결정되는 방식은 구현에 따라 다양한 형태를 취할 수 있다. 예를 들어, 결정된 관계는 벡터 레지스터들이 서로 순차적이거나, 벡터 레지스터들이 짝수/홀수 쌍이거나, 다양한 벡터 레지스터들 사이에 알려진 오프셋이 존재한다는 것을 지정할 수 있다.

대안적으로, 임의의 다른 적절한 표시된 관계가 사용될 수 있다.

[0025] 하나의 특정한 예시적인 구현에서, 필요한 능력을 저장하는 복수의 벡터 레지스터 내의 벡터 레지스터들의 수는 2^N 이고, 단일 능력 벡터 표시 필드는 하나의 벡터 레지스터를 식별하는 제1 벡터 레지스터 번호를 나타내며, 제1 벡터 레지스터 번호는 논리 0 값에서 그의 N개의 최하위 비트를 갖도록 제약된다. 이어서, 명령어 디코더는 제1 벡터 레지스터 번호를 재사용하고, N개의 최하위 비트 중 적어도 하나를 논리 1 값으로 선택적으로 설정함으로써 나머지 벡터 레지스터 각각에 대한 벡터 레지스터 번호를 생성하도록 배열된다. 이는 주어진 벡터 메모리 액세스 명령어를 실행할 때 필요한 능력을 제공할 다양한 벡터 레지스터를 계산하기 위한 특히 간단하고 효율적인 메커니즘을 제공할 수 있다.

[0026] 일부 구현들에서, 능력을 보유하는 데 필요한 벡터 레지스터들의 수는 예를 들어, 주어진 벡터 메모리 액세스 명령어가 특정한 고정된 크기의 데이터 요소와 함께 사용하기 위해서만 지원되고, 능력도 고정된 크기를 갖기 때문에 고정될 것이다. 그러나, 더 일반적인 경우에, 벡터 레지스터들의 수는 주어진 벡터 메모리 액세스 명령어가 실행될 데이터 요소의 크기 및 능력의 크기에 대한 지식에 기초하여 명령어 디코더에 의해 런타임에 추론될 수 있다.

[0027] 단일 능력 벡터 표시 필드가 제1 벡터 레지스터 번호를 나타내도록 배열될 수 있는 다수의 방법이 있다. 단일 능력 벡터 표시 필드는 하나의 예시적인 구현에서 제1 벡터 레지스터 번호를 직접 식별할 수 있지만, 다른 구현에서는 그러한 제1 벡터 레지스터 번호가 결정될 수 있도록 충분한 정보를 지정할 수 있다. 예를 들어, 위의 경우, 제1 벡터 레지스터 번호는 논리 0 값에서 그의 N개의 최하위 비트를 갖도록 제약되는 경우, 이러한 최하위 N개의 비트는 단일 능력 벡터 표시 필드 내에서 식별될 필요가 없으며, 대신 논리 0 값으로 하드와이어링될 수 있다.

[0028] 다양한 데이터 요소와 관련된 능력이 능력을 제공하는 데 사용되는 벡터 레지스터 내에 레이아웃되는 방식은 구현에 따라 달라질 수 있다. 그러나, 하나의 예시적인 구현에서, 적어도 하나의 벡터 레지스터 내의 인접 위치들과 관련된 데이터 요소들의 임의의 주어진 쌍에 대해, 관련 능력들은 상기 복수의 벡터 레지스터 중 상이한 벡터 레지스터들에 저장된다. 이러한 배열은 주어진 벡터 메모리 액세스 명령어를 실행할 때 효율적인 구현을 가능하게 할 수 있는 것으로 밝혀졌다.

[0029] 임의의 특정 데이터 요소에 대한 관련 능력의 다수의 벡터 레지스터 내의 위치가 결정되는 방식은 구현에 따라 달라질 수 있다. 그러나, 하나의 예시적인 구현에서, 데이터 벡터 표시 필드로부터 결정된 적어도 하나의 벡터 레지스터는 단일 벡터 레지스터를 포함하고, 각각의 데이터 요소는 단일 벡터 레지스터의 대응하는 데이터 레인과 관련된다. 또한, 각각의 능력은 상기 복수의 벡터 레지스터 내의 벡터 레지스터들 중 하나 내의 능력 레인 내에 위치된다. 여기서 유의해야 할 점은 데이터 요소와 능력의 크기가 다르기 때문에 데이터 레인의 폭은 전형적으로 능력 레인의 폭과 다를 것이라는 점이다. 이러한 배열에 의해, 이어서 주어진 데이터 요소에 대해, 관련 능력을 포함하는 복수의 벡터 레지스터 내의 벡터 레지스터는 대응하는 데이터 레인의 레인 번호의 주어진 수의 최하위 비트들에 따라 결정될 수 있고, 관련 능력을 포함하는 능력 레인은 대응하는 데이터 레인의 레인 번호의 나머지 비트들에 따라 결정될 수 있다. 따라서, 이는 각각의 데이터 요소에 대한 관련 능력의 위치를 결정하기 위한 특히 효율적인 메커니즘을 제공한다.

[0030] 하나의 특정한 예시적인 배열에서, 복수의 능력을 포함하는 벡터 레지스터들의 수는 논리적으로 0 내지 P-1의 값들을 갖는 시퀀스로 간주되는 P이고, 임의의 주어진 벡터 레지스터 내의 능력 레인들의 수는 0 내지 M-1의 값들을 갖는 M이다. 또한, 주어진 데이터 요소와 관련된 데이터 레인은 데이터 레인 X이고, 이는 0 내지 X-1의 값을 갖는다. 이러한 용어를 사용하면, 하나의 예시적인 구현에서, 복수의 벡터 레지스터 내의 관련 능력의 위치는 X를 P로 나누어 몫과 나머지를 생성함으로써 결정될 수 있으며, 몫은 관련 능력을 포함하는 능력 레인을 식별하고, 나머지는 관련 능력을 포함하는 복수의 벡터 레지스터 내의 벡터 레지스터를 식별한다. 따라서, 이러한 구현에서는 주어진 데이터 요소에 대한 관련 능력의 위치를 정하는 데 필요한 벡터 레지스터 및 능력 레인 모두가 쉽고 효율적으로 결정될 수 있다.

[0031] 위의 예에서 복수의 능력을 포함하는 복수의 벡터 레지스터는 논리적으로 0 내지 P-1의 값을 갖는 시퀀스로 간주되지만, 이는 이러한 벡터 레지스터들과 관련된 논리 벡터 번호들이 연속적인 논리 벡터 번호들일 필요가 있음을 의미하지 않으며, 실제로 벡터 레지스터들이 벡터 레지스터 세트 내에서 서로에 대해 물리적으로 순차적으로 위치해야 한다는 것을 의미하지도 않는다는 점에 유의해야 한다.

[0032] 하나의 예시적인 구현에서, 벡터 레지스터 세트는 논리적으로 복수의 섹션으로 분할될 수 있으며, 각각의 섹션

은 벡터 레지스터 세트 내의 벡터 레지스터들 각각으로부터의 대응하는 부분을 포함하고, 복수의 능력은 각각의 데이터 요소에 대하여 관련 능력이 해당 데이터 요소와 동일한 섹션 내에 저장되도록 복수의 벡터 레지스터 내에 위치할 수 있다. 이러한 접근법은 주어진 벡터 메모리 액세스 명령어의 실행이 다수의 "박동(beat)"으로 분할되게 할 수 있으며, 각각의 박동 동안에는 주어진 벡터 메모리 액세스 명령어를 실행하기 위해 벡터 레지스터 세트의 하나의 섹션만이 액세스된다. 벡터 메모리 액세스 명령어가 다수의 박동으로 분할될 수 있게 함으로써, 벡터 메모리 액세스 명령어의 실행이 하나 이상의 다른 명령어의 실행과 중첩되게 할 수 있으며, 이는 매우 효율적인 구현으로 이어질 수 있다. 특히, 임의의 특정 박동 동안, 해당 박동 동안 메모리 액세스 동작을 수행하는 데 필요한 데이터 요소와 능력은 모두 벡터 레지스터 세트의 단일 섹션으로부터 획득될 수 있기 때문에, 이는 중첩된 명령어의 실행 동안 임의의 다른 섹션이 액세스에 이용 가능하게 남겨둔다.

[0033] 하나의 예시적인 구현에서, 처리 회로는 하나 이상의 박동에 걸쳐, 주어진 섹션 내의 데이터 요소들에 대한 메모리 액세스 동작들을 수행한 후에, 하나 이상의 박동에 걸쳐, 다음 섹션 내의 데이터 요소들에 대한 메모리 액세스 동작들을 수행하도록 배열될 수 있다. 하나의 예시적인 구현에서, 주어진 벡터 메모리 액세스 명령어를 실행하는 데 사용되는 다수의 박동 중의 각각의 박동은 상이한 섹션에 액세스할 수 있지만, 이는 필수 사항이 아니며, 일부 구현들에서는 이러한 박동들 중 2개 이상이 동일한 섹션에 액세스하는 경우가 있을 수 있다.

[0034] 위에서 언급한 주어진 벡터 메모리 액세스 명령어를 실행할 때 필요한 능력들이 메모리로부터 로드된 후 앞서 논의한 배열들로 다수의 벡터 레지스터들 내에 구성될 수 있는 여러 가지 방법들이 있으며, 실제로 벡터 레지스터들 내의 이러한 능력들이 적절한 시기에 메모리에 다시 저장될 수 있는 여러 가지 방법들이 있다. 그러나, 하나의 예시적인 구현에서, 명령어 디코더는 복수의 벡터 능력 메모리 전송 명령어를 디코딩하도록 배열되며, 이들은 함께 명령어 디코더로 하여금 처리 회로를 제어하여 메모리와 복수의 벡터 레지스터 사이에서 복수의 능력을 전송하고, 전송 동안 복수의 능력을 재배열하여 메모리에 복수의 능력이 순차적으로 저장되고, 복수의 벡터 레지스터에서 복수의 능력이 디인터리빙되어 메모리에 순차적으로 저장되는 상기 복수 내의 능력들의 임의의 주어진 쌍이 상기 복수의 벡터 레지스터 중 상이한 벡터 레지스터에 저장되게 한다.

[0035] 전술한 단계들을 수행하기 위해 사용되는 복수의 벡터 능력 메모리 전송 명령어는 서로 직접 뒤따를 필요가 없으며, 따라서 순차적으로 하나씩 실행될 필요가 없다는 점에 유의해야 한다. 대신, 각각 필요한 작업의 일부를 수행하는 여러 개의 별개의 명령어가 있을 수 있으며, 모든 명령어들이 실행되었다면 이들이 메모리와 벡터 레지스터 간에 이동(일례에서는 복사)될 때 능력들의 필요한 재배열이 수행되었을 것이다. 복수의 벡터 능력 메모리 전송 명령어는 메모리로부터 다수의 벡터 레지스터로 능력을 로드하는 데 사용되는 로드 명령어들이거나, 다수의 벡터 레지스터로부터 능력을 메모리에 다시 저장하는 데 사용되는 저장 명령어들일 수 있다.

[0036] 하나의 예시적인 구현에서, 각각의 벡터 능력 메모리 전송 명령어는 각각의 다른 벡터 능력 메모리 전송 명령어에 대한 상이한 능력들을 식별하도록 배열되고, 각각의 벡터 능력 메모리 전송 명령어는 처리 회로가 액세스 패턴에 의해 지정된 재배열을 수행하면서 식별된 능력을 전송하게 하는 그러한 액세스 패턴을 식별하도록 배열된다. 따라서, 이러한 배열에서, 각각의 개별 벡터 능력 메모리 전송 명령어의 실행은 해당 명령어에 의해 전송되는 능력들에 대해 필요한 재배열이 수행되게 할 것이고, 이어서 다른 벡터 능력 메모리 전송 명령어는 다른 능력들을 전송하고 그러한 능력들에 대한 필요한 재배열을 수행하는 데 사용된다.

[0037] 이러한 구현을 통해, 다양한 상이한 명령어들이 모두 동일한 최대 데이터 양, 즉 임의의 특정 시스템에서 이용 가능한 유한한 메모리 대역폭과 관련하여 선택되는 최대 데이터 양을 전송하도록 배열하는 것이 가능하다. 이러한 접근법은 임의의 개별 명령어가 중단되는 것을 방지할 수 있으며, 따라서 이러한 접근법을 구현하기 위해 시퀀싱 상태 머신이 필요하지 않다. 이러한 접근법은 또한 이러한 능력 전송 프로세스가 진행되는 동안 다른 명령어들이 스케줄링되는 것을 가능하게 한다. 또한, 위에서 논의한 방식으로 상이한 능력들에 대해 동작하도록 각각의 명령어를 배열함으로써, 각각의 박동에 대해 임의의 개별 명령어가 벡터 레지스터들의 동일한 섹션 내에서만 동작하도록 배열될 수 있다. 앞서 논의한 바와 같이, 주어진 섹션 내에서만 동작하는 것은 상이한 섹션들에서 동작하는 명령어들의 중첩을 허용한다.

[0038] 하나의 예시적인 구현에서, 메모리는 다수의 메모리 뱅크로 형성되고, 각각의 벡터 능력 메모리 전송 명령어에 대해, 액세스 패턴은 해당 벡터 능력 메모리 전송 명령어가 처리 회로에 의해 실행될 때 메모리 뱅크들 중 2개 이상이 액세스되게 하도록 정의된다. 뱅크형 메모리는 하드웨어가 메모리로의/메모리로부터의 병렬 전송을 구현하는 것을 더 쉽게 하며, 따라서 이를 가능하게 하는 액세스 패턴을 지정하는 것이 유리하다.

[0039] 위에서 언급한 벡터 능력 메모리 전송 명령어 외에도 벡터 로드 및 저장 명령어를 사용하여 필요에 따라 그리고 필요할 때 메모리에서 벡터 레지스터로 데이터 요소를 로드하거나 벡터 레지스터에서 그러한 데이터 요소를 메

모리에 다시 저장할 수 있다.

- [0040] 데이터 요소를 보유하는 데 사용되는 벡터 레지스터들의 수 및 관련 능력을 보유하는 데 사용되는 벡터 레지스터들의 수는 구현에 따라 달라질 수 있지만, 하나의 특정한 예시적인 구현에서, 주어진 벡터 메모리 액세스 명령어의 데이터 벡터 표시 필드로부터 결정된 적어도 하나의 벡터 레지스터는 단일 벡터 레지스터를 포함하고, 능력은 데이터 요소 크기의 두 배이고(앞서 언급한 바와 같이, 능력이 유효한 능력임을 나타내기 위해 사용되는 임의의 플래그는 능력의 크기를 고려할 때 능력의 일부로 간주되지 않음), 적어도 하나의 능력 벡터 표시 필드로부터 결정된 복수의 벡터 레지스터는 두 개의 벡터 레지스터를 포함한다. 이러한 배열은 능력에서 도출된 메모리 어드레스를 사용하여 벡터 수집 및 분산 동작을 수행하기 위한 특히 유용한 구현을 제공하는 것으로 밝혀졌다.
- [0041] 하나의 예시적인 구현에서, 주어진 벡터 메모리 액세스 명령어는 어드레스 오프셋을 나타내는 즉시 값을 추가로 포함할 수 있고, 처리 회로는 복수의 데이터 요소 내의 각각의 주어진 데이터 요소에 대해, 어드레스 오프셋을 관련 능력에 의해 제공되는 어드레스 표시와 결합함으로써 주어진 데이터 요소의 메모리 어드레스를 결정하도록 배열될 수 있다. 이는 다양한 능력에서 제공되는 어드레스 표시로부터 메모리 어드레스를 계산하기 위한 효율적인 구현을 제공할 수 있다.
- [0042] 하나의 예시적인 구현에서, 주어진 벡터 메모리 액세스 명령어는 어드레스 오프셋을 나타내는 즉시 값을 추가로 포함할 수 있고, 각각의 주어진 데이터 요소에 대해, 처리 회로는 어드레스 오프셋에 따라 어드레스 표시를 조정함으로써 복수의 벡터 레지스터에서 관련 능력의 어드레스 표시를 업데이트하도록 배열될 수 있다. 따라서, 예로서, 특정 능력의 어드레스 표시가 제1 벡터 메모리 액세스 명령어의 실행 중에 사용되면, 벡터 레지스터에 저장된 능력 내에 표시된 바와 같은 해당 어드레스 표시는 후속 벡터 메모리 액세스 명령어와 관련하여 사용할 준비가 되도록 전술한 방식으로 업데이트될 수 있다.
- [0043] 일부 경우들에서, 전술한 조정 프로세스들 모두가 수행될 수 있으며, 이에 따라 어드레스 오프셋이 액세스할 메모리 어드레스를 식별하기 위해 능력에 의해 제공되는 어드레스 표시와 결합(예를 들어, 그에 추가)되고, 동일한 업데이트된 어드레스가 업데이트된 어드레스 표시로서 능력 레지스터에 다시 기입된다. 전형적으로, 두 조정 프로세스 모두에 대해 동일한 즉시 값이 사용될 것이지만, 원하는 경우 각각의 조정 프로세스에 대해 상이한 즉시 값이 사용될 수 있다.
- [0044] 이제, 특정한 예시적인 구현들이 도면들을 참조하여 논의될 것이다.
- [0045] 도 1은 벡터 명령어의 처리를 지원하는 데이터 처리 장치(2)의 예를 개략적으로 예시한다. 이것은 설명의 용이성을 위해 단순화된 도면이며, 실제로 장치는 간결성을 위해 도 1에 도시되지 않은 많은 요소를 가질 수 있다는 것을 이해할 것이다. 장치(2)는 명령어 디코더(6)에 의해 디코딩된 명령어들에 응답하여 데이터 처리를 수행하기 위한 처리 회로(4)를 포함한다. 프로그램 명령어는 메모리 시스템(8)으로부터 인출되고, 명령어 디코더에 의해 디코딩되어 처리 회로(4)가 아키텍처에 의해 정의된 방식으로 명령어를 처리하도록 제어하는 제어 신호를 생성한다. 예를 들어, 디코더(6)는 디코딩된 명령어의 연산 코드 및 명령어의 임의의 추가 제어 필드를 해석하여 처리 회로(4)로 하여금 산술 연산, 로드/저장 동작 또는 논리 동작과 같은 동작을 수행하기 위해 적절한 하드웨어 유닛을 활성화하게 하는 제어 신호를 생성할 수 있다. 장치는 스칼라 레지스터들(10)의 세트와 벡터 레지스터들(12)의 세트를 갖는다. 장치는 또한 예를 들어 처리 회로의 동작을 구성하는 데 사용되는 제어 정보를 저장하기 위한 다른 레지스터(도시되지 않음)를 가질 수 있다. 산술 또는 논리 명령어에 응답하여, 처리 회로는 전형적으로 레지스터(10, 12)로부터 소스 피연산자를 판독하고, 명령어의 결과를 레지스터(10, 12)에 다시 기입한다. 로드/저장 명령어에 응답하여, 데이터 값이 처리 회로(4) 내의 로드/저장 유닛(18)을 통해 레지스터(10, 12)와 메모리 시스템(8) 사이에서 전송된다. 메모리 시스템(8)은 메인 메모리뿐만 아니라 하나 이상의 캐시 레벨을 포함할 수 있다.
- [0046] 스칼라 레지스터들(10)의 세트는 단일 데이터 요소를 포함하는 스칼라 값을 저장하기 위한 다수의 스칼라 레지스터를 포함한다. 명령어 디코더(6) 및 처리 회로(4)에 의해 지원되는 일부 명령어들은 스칼라 레지스터들(10)로부터 판독된 스칼라 피연산자들을 처리하여 스칼라 레지스터에 다시 기입되는 스칼라 결과를 생성하는 스칼라 명령어들일 수 있다.
- [0047] 벡터 레지스터들(12)의 세트는 다수의 벡터 레지스터를 포함하며, 각각은 다수의 요소를 포함하는 벡터 값을 저장하도록 배열된다. 벡터 명령어에 응답하여, 명령어 디코더(6)는 처리 회로(4)를 제어하여, 벡터 레지스터들(12) 중 하나로부터 판독된 벡터 피연산자의 각각의 요소에 대해 벡터 처리의 다수의 레인을 수행하여, 스칼라

레지스터(10)에 기입될 스칼라 결과 또는 벡터 레지스터(12)에 기입될 추가적인 벡터 결과를 생성할 수 있다. 일부 벡터 명령어는 하나 이상의 스칼라 피연산자로부터 벡터 결과를 생성할 수 있거나, 스칼라 레지스터 파일 내의 스칼라 피연산자에 대해 추가적인 스칼라 연산을 수행할 수 있을 뿐만 아니라, 벡터 레지스터 파일(12)로부터 판독된 벡터 피연산자에 대한 벡터 처리 레인을 수행할 수 있다. 따라서, 일부 명령어는 하나 이상의 소스 레지스터 및 명령어의 목적지 레지스터 중 적어도 하나가 벡터 레지스터(12)이고 하나 이상의 소스 레지스터 및 목적지 레지스터 중 다른 하나가 스칼라 레지스터(10)인 혼합 스칼라-벡터 명령어일 수 있다.

[0048] 벡터 명령어는 또한 벡터 로드/저장 명령어를 포함할 수 있으며, 이는 벡터 레지스터(12)와 메모리 시스템(8) 내의 위치 사이에서 데이터 값이 전송되게 한다. 로드/저장 명령어는 메모리 내의 위치들이 어드레스들의 연속 범위에 대응하는 연속 로드/저장 명령어들을 포함할 수 있거나, 다수의 불연속 어드레스를 지정하고, 처리 회로(4)를 제어하여 이들 어드레스 각각으로부터 벡터 레지스터의 각각의 요소로 데이터를 로드하거나 벡터 레지스터의 각각의 요소로부터 불연속 어드레스들로 데이터를 저장하는 수집/분산 유형 벡터 로드/저장 명령어들을 포함할 수 있다.

[0049] 처리 회로(4)는 상이한 데이터 요소 크기들의 범위를 갖는 벡터들의 처리를 지원할 수 있다. 예를 들어, 128 비트 벡터 레지스터(12)는 16개의 8 비트 데이터 요소, 8개의 16 비트 데이터 요소, 4개의 32 비트 데이터 요소 또는 2개의 64 비트 데이터 요소로 분할될 수 있다. 제어 레지스터는 사용되는 현재 데이터 요소 크기를 지정하는 데 사용될 수 있거나, 대안적으로 이는 실행될 주어진 벡터 명령어의 파라미터일 수 있다.

[0050] 처리 회로(4)는 상이한 명령어 클래스들을 처리하기 위한 다수의 별개의 하드웨어 블록을 포함할 수 있다. 예를 들어, 메모리 시스템(8)과 상호 작용하는 로드/저장 명령어는 전용 로드/저장 유닛(18)에 의해 처리될 수 있는 반면, 산술 또는 논리 명령어는 산술 논리 유닛(ALU)에 의해 처리될 수 있다. ALU 자체는 곱셈을 수신하는 연산을 수행하기 위한 곱셈-누산 유닛(MAC) 및 다른 종류의 ALU 연산을 처리하기 위한 추가 유닛으로 추가로 분할될 수 있다. 부동 소수점 명령어를 처리하기 위해 부동 소수점 유닛도 제공될 수 있다. 어떠한 벡터 처리도 수반하지 않는 순수 스칼라 명령어는 또한 벡터 명령어와 비교하여 별개의 하드웨어 블록에 의해 처리되거나 동일한 하드웨어 블록을 재사용할 수 있다.

[0051] 앞서 논의한 바와 같이, 지원될 수 있는 벡터 로드/저장 명령어의 한 가지 유형은 벡터 수집/분산 명령어이다. 이러한 벡터 명령어는 메모리 내의 다수의 불연속 어드레스를 표시하고, 처리 회로(4)를 제어하여 (벡터 수집 명령어의 경우) 이들 불연속 어드레스로부터 벡터 레지스터의 각각의 요소로 데이터를 로드하거나 (벡터 분산 명령어의 경우) 벡터 레지스터의 각각의 요소로부터 불연속 어드레스에 데이터를 저장할 수 있다. 본 명세서에 설명된 기술들에 따르면, 다양한 메모리 어드레스를 식별하기 위해 표준 어드레스 표시들의 벡터를 사용하는 대신, 다양한 메모리 어드레스를 결정하기 위해 사용될 능력들의 벡터들을 지정할 수 있는 새로운 형태의 벡터 수집/분산 명령어가 제공된다. 이는 벡터 수집/분산 동작을 구현하는 데 사용되는 개별 메모리 액세스 동작의 수를 더 세밀하게 제어할 수 있는데, 그 이유는 각각의 개별 메모리 액세스 동작과 관련하여 사용하기 위해 별개의 능력을 정의할 수 있기 때문이다. 어드레스 표시를 제공하는 것 외에도 각각의 능력은 전형적으로 해당 능력을 사용할 때 수행할 수 있는 동작을 제한하는 데 사용되는 제약 정보를 포함할 것이다. 예를 들어, 제약 정보는 능력에 의해 제공되는 어드레스 표시를 사용할 때 처리 회로에 의해 액세스 가능한 메모리 어드레스들의 확장 불가능한 범위를 식별할 수 있으며, 관련 허가를 식별하는 하나 이상의 허가 플래그(예를 들어, 판독 액세스가 허용되는지 여부, 기입 액세스가 허용되는지 여부, 지정된 특권 또는 보안 레벨에서 액세스가 허용되는지 여부, 능력을 사용하여 인출 및 실행될 명령어의 메모리 어드레스를 생성할 수 있는지 여부 등)를 제공할 수도 있다.

[0052] 이 새로운 형태의 벡터 수집/분산 명령어를 실행할 때, 메모리와 벡터 레지스터 사이에서 이동될 각각의 데이터 요소(이동 방향은 벡터 수집 동작 또는 벡터 분산 동작이 수행되고 있는지에 따라 달라짐)는 관련 능력을 가질 것이며, 처리 회로(4) 내의 능력 액세스 검사 회로(16)는 그러한 주어진 데이터 요소에 액세스하는 데 사용되는 메모리 액세스 동작이 관련 능력에 의해 지정된 제약 정보와 관련하여 허용되는지 여부를 결정하기 위해 각각의 데이터 요소에 대한 능력 검사를 수행하는 데 사용될 수 있다. 따라서, 이는 능력 내의 임의의 범위 제약 정보가 주어지면 메모리 어드레스가 액세스 가능한지 여부 및 능력 내의 제약 정보가 주어지면 액세스 유형이 허용되는지 여부를 모두 검사하는 것을 포함할 수 있다. 이러한 벡터 수집/분산 명령어를 실행할 때 필요한 복수의 능력이 일련의 벡터 레지스터 내에 어떻게 배열되는지에 대한 더 상세한 내용은 다수의 나머지 도면들을 참조하여 더 상세히 논의될 것이다.

[0053] 도 1에 도시된 바와 같이, 박동 제어 회로(20)는 원하는 경우 명령어 디코더(6) 및 처리 회로(4)의 동작을 제어

하기 위해 제공될 수 있다. 특히, 일부 예시적인 구현에서, 벡터 명령어의 실행은 "박동"으로 지칭되는 부분들로 분할될 수 있으며, 각각의 박동은 미리 결정된 크기의 벡터의 일부의 처리에 대응한다. 도 10 및 도 11을 참조하여 나중에 더 상세히 논의될 바와 같이, 이는 벡터 명령어들의 중첩 실행을 허용할 수 있으며, 이에 따라 성능을 개선할 수 있다.

[0054] 도 2는 태그 비트를 개별 데이터 블록들과 관련하여 사용하여 그러한 데이터 블록들이 능력을 나타내는지 또는 일반 데이터를 나타내는지를 식별할 수 있는 방법을 개략적으로 예시한다. 특히, 메모리 어드레스 공간(110)은, 전형적으로 지정된 크기를 가질, 일련의 데이터 블록(115)을 저장할 것이다. 순전히 예시를 위해, 이 예에서 각각의 데이터 블록이 64 비트를 포함하는 것으로 가정되지만, 다른 예시적인 구현에서 상이한 크기의 데이터 블록, 예를 들어 능력이 128 비트의 정보에 의해 정의될 때 128 비트 데이터 블록이 사용될 수 있다. 각각의 데이터 블록(115)과 관련하여, 태그 필드(120)가 제공되는데, 이는 일례에서, 관련 데이터 블록이 능력을 나타낸다는 것을 식별하도록 설정되고, 관련 데이터 블록이 일반 데이터를 나타내고, 따라서 능력으로 취급될 수 없다는 것을 표시하도록 소거되는, 태그 비트로 지칭되는 단일 비트 필드이다. 설정 또는 소거 상태와 관련된 실제 값은 예시적인 구현에 따라 달라질 수 있지만, 순전히 예시로서, 하나의 예시적인 구현에서, 태그 비트가 1의 값을 갖는 경우, 그것은 관련 데이터 블록이 능력이라는 것을 표시하고, 그것이 0의 값을 갖는 경우, 그것은 관련 데이터 블록이 일반 데이터를 포함한다는 것을 표시한다는 것을 이해할 것이다. 하나의 예시적인 구현에서, 태그 비트들은 통상의 메모리 어드레스 공간의 일부를 형성하지 않을 수 있고, 대신에 예를 들어 별개의 태그 메모리에 "대역 외(out-of-band)"로 저장될 수 있다.

[0055] 능력이 처리 회로에 의해 액세스 가능한 레지스터(100)에 로드될 때, 태그 비트는 능력 정보와 함께 이동한다. 따라서, 능력이 레지스터(100)에 로드될 때, 어드레스 표시(102)(본 명세서에서 포인터로도 지칭될 수 있음) 및 제약 정보(예컨대, 앞서 언급한 범위 정보 및 허가 정보)를 제공하는 메타데이터(104)가 레지스터에 로드될 것이다. 또한, 그러한 레지스터와 관련하여, 또는 그것 내의 특정 비트 필드로서, 내용이 유효한 능력을 나타낸다는 것을 식별하도록 태그 비트(106)가 설정될 것이다. 유사하게, 유효한 능력이 다시 메모리에 저장될 때, 능력이 저장되는 데이터 블록과 관련하여 관련 태그 비트(120)가 설정될 것이다. 이러한 접근법에 의해, 능력과 일반 데이터 간을 구별하고, 따라서 일반 데이터가 능력으로서 사용될 수 없도록 보장하는 것이 가능하다.

[0056] 장치는 능력을 저장하기 위한 전용 능력 레지스터(도 1에 도시되지 않음)를 구비할 수 있으며, 따라서 도 2의 레지스터(100)는 전용 능력 레지스터일 수 있다. 그러나, 전술한 새로운 형태의 벡터 수집/분산 명령어를 실행하기 위해서는, 벡터 레지스터들(12)의 세트 내의 다수의 벡터 레지스터들 내에 필요한 능력들을 배치하는 것이 바람직하다. 벡터 레지스터 내에 저장된 유효한 능력과 범용 데이터를 구별할 수 있도록, 벡터 레지스터 세트는 관련된 유효한 능력 표시 저장소의 제공에 의해 보완되며, 이것이 구현될 수 있는 두 가지 상이한 방법이 도 3a 및 3b에 개략적으로 도시되어 있다. 도 3a에 도시된 예에서, 벡터 레지스터들(130)의 세트는 복수의 벡터 레지스터(135)를 포함하며, 각각의 벡터 레지스터는 다수의 능력 크기 블록(137)을 제공하기에 충분한 크기를 갖는다. 순전히 예로서, 능력의 길이가 64 비트일 때, 각각의 능력 크기 블록(137)은 64 비트일 수 있고, 각각의 벡터 레지스터의 길이는 $2^N \times 64$ 비트일 수 있으며, 여기서 N은 0 이상의 정수이다.

[0057] 도 3a의 특정 예에서는, 각각의 벡터 레지스터의 길이가 128 비트이고, 따라서 각각의 벡터 레지스터는 2개의 능력 크기 블록(137)을 갖는다고 가정한다. 유효한 능력 표시 저장소(140)는 벡터 레지스터들의 세트와 관련하여 제공되며, 유효한 능력 표시 저장소(140)는 각각의 벡터 레지스터(135)에 대한 엔트리(145)를 갖는다. 각각의 엔트리(145)는 관련 벡터 레지스터(135) 내의 각각의 능력 크기 블록(137)에 대한 유효한 능력 표시 필드를 제공한다. 유효한 능력 표시 필드는 다양한 형태를 취할 수 있지만, 하나의 예시적인 구현은 단일 비트 필드일 수 있으며, 따라서 일례에서는 앞서 설명한 태그 비트의 형태를 취할 수 있다. 이러한 경우에, 각각의 엔트리(145)는 관련 벡터 레지스터(135) 내의 각각의 능력 크기 블록(137)에 대한 태그 비트를 제공하여, 해당 능력 크기 블록이 유효한 능력을 저장하고 있는지 여부를 식별한다는 것을 이해할 것이다.

[0058] 도 3a의 예에서는, 유효한 능력 표시 저장소(140)가 벡터 레지스터들(130)의 세트에 대한 별개의 구조로 간주되지만, 대안적인 구현에서는, 필요한 태그 비트를 수용하도록 벡터 레지스터들의 크기를 증가시킴으로써 벡터 레지스터들의 세트 내에 유효한 능력 표시 저장소가 효과적으로 통합될 수 있다. 도 3b에 도시된 바와 같은 배열에서, 벡터 레지스터들(150)의 세트는 다수의 능력 크기 블록들(160, 164)을 포함하며, 이들 각각은 관련 태그 비트를 저장하기 위한 관련된 유효한 능력 표시 필드(162, 166)를 갖는다. 이 배열에서는 능력의 크기가 변경되는 것으로 간주되지 않으므로 앞서 언급한 예에서 각각의 능력의 길이는 여전히 64 비트라는 점에 유의해야 한다. 그러나, 벡터 레지스터는 관련 태그 비트를 위한 공간을 제공하기 위해 확장된다. 따라서, 도 3b의 예

를 고려할 때, 다시 두 개의 능력이 각각의 벡터 레지스터 내에 저장될 수 있는 것으로 간주되고, 각각의 능력의 길이가 64 비트라고 가정하면, 능력을 저장할 수 있는 임의의 벡터 레지스터는 길이가 130 비트이도록 배열될 수 있으므로, 두 개의 능력 및 그들의 관련 태그 비트가 모두 저장되게 할 수 있다. 이 예에서 태그 비트들은 벡터 레지스터들(155)의 일부이지만, 태그 비트들에 대한 액세스는 앞서 설명한 바와 같이 여전히 엄격하게 제어될 수 있으므로, 이들은 범용 처리 명령어에 의해 직접 액세스 가능하지 않으며, 비능력 명령어를 사용하여 벡터 레지스터 내의 값을 변경하면 태그가 소거된다.

[0059] 도 3a 및 도 3b의 예들에서는 모든 벡터 레지스터들이 능력들을 저장할 수 있다고 가정하지만, 대안적인 구현에서는 세트 내의 벡터 레지스터들의 서브세트가 능력들을 저장하기 위해 예약될 수 있으며, 그러한 경우에는 개별 저장소인지(도 3a의 예에 따름) 또는 벡터 레지스터 구조 자체에 통합되는지(도 3b의 예에 따름)에 관계없이, 관련된 유효한 능력 표시 저장소와 함께 제공되어야 하는 것은 단지 그러한 벡터 레지스터들의 서브세트라는 점에 유의해야 한다.

[0060] 도 4a 및 도 4b는 하나의 예시적인 구현에 따른, 벡터 레지스터의 각각의 능력 크기 블록과 관련하여 유지되는 태그 비트가 관리될 수 있는 방법을 예시하는 흐름도이다. 도 4a는 기입되는 벡터 레지스터 내의 능력 크기 블록에 대해 유지되는 관련 태그 비트와 관련하여 수행할 동작을 결정하기 위해 수행되는 몇 가지 단계를 예시한다. 특히, 단계 170에서 기입 동작이 벡터 레지스터에 대해 수행되고 있다고 결정되면, 도 4a의 프로세스의 나머지는 기입되고 있는 벡터 레지스터 내의 각각의 능력 크기 블록과 관련하여 수행된다.

[0061] 단계 172에서, 벡터 레지스터의 주어진 능력 크기 부분과 관련하여 기입되는 데이터가 전체 능력 블록 크기를 갖는지 여부가 결정된다. 그렇지 않다면, 태그 비트가 이전에 설정된 경우 소거되고, 그에 따라 프로세스는 태그 비트가 소거되는 단계 174로 진행한다. 이러한 접근법은 능력의 불법적인 수정을 방지한다. 예를 들어, 벡터 레지스터 내에 저장된 유효한 능력의 특정 비트 수를 수정하려고 시도하는 경우, 위의 프로세스는 태그 비트를 소거하여 벡터 레지스터에 현재 저장된 수정된 버전이 능력으로 사용되는 것을 방지할 것이다.

[0062] 그러나, 전체 능력 크기 블록의 정보가 벡터 레지스터의 주어진 능력 크기 부분에 기입되고 있다고 가정하면, 유효한 능력이 기입되고 있는지 여부가 단계 176에서 결정된다. 그렇지 않다면, 다시 프로세스는 태그 비트가 소거되는 단계 174로 진행한다. 그러나, 유효한 능력이 기입되고 있는 경우, 프로세스는 태그 비트가 설정되는 단계 178로 진행한다.

[0063] 벡터 레지스터 내의 능력 크기 블록과 관련된 태그 비트가 소거될 수 있는 것은 벡터 레지스터에 기입하는 명령어의 실행 동안만이 아니라는 점에 유의해야 한다. 특히, 도 4b에 의해 표시된 바와 같이, 벡터 레지스터의 능력 크기 블록에 저장된 능력이 더 이상 유효하지 않게 하기 위해 임의의 단계들이 취해졌는지의 여부가 단계 180에서 결정될 수 있다. 이러한 조건이 검출되지 않으면, 단계 185에 의해 표시된 바와 같이, 관련 태그 비트에 대한 업데이트가 이루어지지 않지만, 그러한 조건이 검출될 때마다 관련 태그 비트는 단계 190에서 소거된다.

[0064] 도 5a는 하나의 예시적인 구현에 따른, 벡터 메모리 액세스 명령어(200)(본 명세서에서 벡터 수집 또는 벡터 분산 명령어로도 지칭됨) 내에 제공될 수 있는 필드들을 개략적으로 예시한다. 연산 코드 필드(205)는 벡터 메모리 액세스 명령어의 형태를 식별하기 위해 사용되며, 따라서 이 경우에는 수집 변형 또는 분산 변형이 지정되고 있는지 여부를 식별하고, 명령어가 액세스될 메모리 어드레스를 결정하기 위해 능력을 사용하는 앞서 설명한 유형임을 식별하는 데 사용될 수 있다.

[0065] 데이터 벡터 표시 필드(210)는 명령어의 실행을 통해 벡터 레지스터 세트와 메모리 사이에서 이동될 데이터 요소들과 관련된 적어도 하나의 벡터 레지스터를 식별하는 데 사용된다. 하나의 예시적인 구현에서, 단일 벡터 레지스터는 데이터 벡터 표시 필드(210)에 의해 식별된다. 이러한 식별된 벡터 레지스터는 벡터 분산 동작을 수행할 때 소스 벡터 레지스터로서 작용하거나, 벡터 수집 동작을 수행할 때 목적지 벡터 레지스터로서 작용할 것임을 이해할 것이다.

[0066] 벡터 분산 또는 벡터 수집 동작의 대상인 각각의 데이터 요소의 메모리 어드레스를 결정하기 위해 필요한 능력을 저장하는 복수의 벡터 레지스터를 식별하기 위해 사용되는 내용을 갖는 적어도 하나의 능력 벡터 표시 필드(215)도 제공될 수 있다. 하나의 구현에서는 다수의 능력 벡터 표시 필드, 예를 들어 필요한 능력을 포함하는 벡터 레지스터들 각각에 대한 하나의 필드가 제공될 수 있지만, 다른 예시적인 구현에서는 단일 능력 벡터 표시 필드가 능력을 저장하는 벡터 레지스터들 중 하나를 결정하기에 충분한 정보를 제공하는 데 사용되며, 나머지 벡터 레지스터들은 소정의 미리 결정된 관계에 기초하여 결정된다. 이 후자 접근법은 명령어 인코딩 관점에서

유리할 수 있다. 미리 결정된 관계는 다양한 형태를 취할 수 있다. 예를 들어, 벡터 레지스터들은 서로 순차적일 수 있거나, 짝수/홀수 쌍을 형성할 수 있거나, 다양한 벡터 레지스터들 사이에 알려진 오프셋이 존재할 수 있다.

[0067] 도 5a에 도시된 바와 같이, 명령어(200)는 또한 추가 정보를 캡처하기 위한 하나 이상의 선택적 필드(220)를 포함할 수 있다. 예를 들어, 다양한 방식으로 사용될 수 있는 어드레스 오프셋을 나타내는 즉시 값이 지정될 수 있다. 예를 들어, 그러한 어드레스 오프셋은 액세스될 메모리 어드레스를 식별하기 위해 각각의 능력 내의 어드레스 표시와 결합될 수 있다(예를 들어, 그에 추가될 수 있음). 다른 예로서, 어드레스 오프셋은 (다시 예를 들어, 어드레스 오프셋을 기존 어드레스 표시와 결합하여) 각각의 능력 내의 어드레스 표시를 업데이트하는 데 사용될 수 있으며, 따라서 벡터 레지스터들 내의 업데이트된 능력은 후속 벡터 메모리 액세스 명령어와 관련하여 사용될 준비가 된다. 실제로, 하나의 예시적인 구현에서, 전술한 어드레스 표시 조정 프로세스들 모두가 수행될 수 있으며, 동일한 즉시 값이 조정 프로세스들 모두에 대해 전형적으로 사용될 것이다.

[0068] 하나 이상의 필드(220) 내에 제공될 수 있는 선택적 정보의 다른 예로서, 명령어의 실행 동안에 액세스될 데이터 요소의 데이터 요소 크기, 및/또는 능력 크기를 지정하기 위해 정보가 제공될 수 있다. 일부 구현에서는 이러한 정보가 불필요할 수 있는데, 그 이유는 능력 크기가 고정될 수 있고, 또한 본 명세서에 설명된 유형의 벡터 메모리 액세스 명령어가 특정 크기의 데이터 요소에 대해서만 수행되도록 허용되는 경우가 있을 수 있으며, 따라서 그 예시적인 경우에서는 데이터 요소 크기와 능력 크기 모두가 명령어에 의해 별개로 지정될 필요 없이 알려지기 때문이다.

[0069] 도 5a에는 각각의 필드를 형성하는 다양한 비트들이 연속적인 것으로 도시되지만, 이는 순전히 예시를 위한 것이며, 명령어 내의 정확히 어떤 비트가 어떤 필드와 관련되는지는 구현에 따라 달라진다는 점에 유의해야 한다. 순전히 예로서, 벡터 레지스터 식별자 필드의 폭이 4 비트인 경우, 3 비트가 함께 그룹화될 수 있지만, 제4 비트는 명령어 인코딩 내의 어딘가 다른 곳에 제공된다.

[0070] 도 5b는 도 5a에 도시된 것과 같은 벡터 메모리 액세스 명령어를 실행할 때 수행되는 단계들을 예시하는 흐름도이다. 단계 230에서, 벡터 메모리 액세스 명령어가 실행될 것인지의 여부가 결정되고, 그렇다면, 프로세스는 데이터 요소들과 관련된 벡터 레지스터가 데이터 벡터 표시 필드 내의 정보로부터 결정되는 단계 235로 진행된다.

[0071] 단계 240에서, 필요한 능력을 포함하는 다수의 벡터 레지스터가 또한 적어도 하나의 능력 벡터 표시 필드 내의 정보를 사용하여 결정된다. 앞서 논의한 바와 같이, 다수의 능력 벡터 표시 필드가 제공될 수 있으며, 각각은 예를 들어 벡터 레지스터들 중 하나를 식별하거나, 대안적으로 단일 능력 벡터 표시 필드가 제공되어 벡터 레지스터들 중 하나의 결정을 가능하게 할 수 있고, 나머지 벡터 레지스터들은 알려진 관계와 관련하여 결정될 수 있다.

[0072] 단계 245에서, 벡터 메모리 액세스 명령어와 관련되는 각각의 주어진 데이터 요소에 대해, 관련 능력에 의해 제공되는 어드레스 표시에 기초하여 그러한 주어진 데이터 요소에 대한 메모리 어드레스가 결정된다. 또한, 관련 능력의 제약 정보에 기초하여, 그러한 주어진 데이터 요소에 액세스하는 데 사용될 메모리 액세스 동작이 허용되는지 여부가 결정된다. 이는 메모리 어드레스가 관련 능력의 범위 제약 정보에 의해 지정된 허용 범위 내에 있는지 여부뿐만 아니라, 관련 능력의 메타데이터에 의해 지정된 임의의 다른 제약이 충족되는지 여부(예를 들어, 벡터 분산 동작이 수행되고 있고, 따라서 주어진 데이터 요소에 대해 수행되는 개별 메모리 액세스 동작이 기입 동작인 경우, 관련 능력을 사용하여 기입 액세스가 허용되는지 여부)를 결정하는 것을 포함할 수 있다.

[0073] 단계 250에서, 메모리 액세스 동작이 허용되는 것으로 결정된 각각의 데이터 요소에 대해 메모리 액세스 동작의 수행이 가능해질 수 있다. 하나의 예시적인 구현에서는 메모리 액세스 동작들이 이러한 메모리 액세스 동작들이 허용되는 각각의 데이터 요소에 대해 수행될 수 있지만, 다른 구현들에서는 다른 메모리 액세스 동작들이 허용되지 않는 경우들에서 하나 이상의 허용된 메모리 액세스 동작의 수행을 억제하도록 결정될 수 있다. 앞서 언급한 바와 같이, 이러한 상황에서 정확히 어떤 허용 가능한 액세스가 억제되는지는 구현, 및 관련 액세스가 허용되지 않는 데이터 요소의 데이터 요소 벡터 내의 위치에 의존할 수 있다.

[0074] 도 6a는 단일 능력 벡터 필드 표시가 제공되는 구현에서 필요한 능력을 보유하는 다수의 벡터 레지스터를 결정하는 데 사용될 수 있는 기술을 예시하는 흐름도이다. 단계 300에서, 필요한 능력을 보유하는 하나의 벡터 레지스터가 그러한 단일 능력 벡터 표시 필드 내의 정보로부터 결정된다. 그 다음, 단계 310에서, 필요한 능력을 보유하는 각각의 다른 벡터 레지스터가 단계 300에서 식별된 벡터 레지스터 및 알려진 결정된 관계로부터 결정

된다. 그러한 결정된 관계는 암시적일 수 있거나, 대안적으로 능력 벡터 표시 필드 내에서 또는 실제로 명령어의 다른 필드 내에서 지정될 수 있다.

[0075] 도 6b는 필요한 능력을 보유하는 다양한 벡터 레지스터를 계산하는 데 사용될 수 있는 특정한 예시적인 구현을 예시한다. 단계 320에서, 필요한 능력을 포함하는 벡터 레지스터들의 수가 결정되며, 이 예시적인 구현에서는 2^N 개의 이러한 벡터 레지스터가 존재한다. 일부 구현들에서, 능력을 보유하는 데 필요한 벡터 레지스터들의 수는 예를 들어, 주어진 벡터 메모리 액세스 명령어가 특정한 고정된 크기의 데이터 요소와 함께 사용하기 위해서만 지원되고, 능력도 고정된 크기를 갖기 때문에 고정될 것이다. 그러나, 대안적으로 벡터 레지스터들의 수는 명령어 디코더에 의해 런타임에 예를 들어 명령어에 의해 지정된 데이터 요소 크기 및 능력 크기 정보에 기초하여 결정될 수 있다.

[0076] 단계 330에서, 제1 벡터 레지스터 번호가 능력 벡터 표시 필드에 제공된 정보로부터 결정되지만, 이 구현에서 해당 벡터 레지스터 번호의 최하위 N 비트는 논리 0 값으로 제약된다. 이러한 구현에서는, 능력 벡터 표시 필드가 이러한 비트들을 지정할 필요가 없다는 것을 이해할 것인데, 그 이유는 이들은 0으로 하드와이어링될 수 있기 때문이다.

[0077] 단계 340에서, 필요한 능력을 포함하는 다수의 벡터 레지스터에 대한 각각의 다른 벡터 레지스터 번호가 결정된 제1 벡터 레지스터 번호의 N개의 최하위 비트의 조합에 의해 결정된다. 이는 필요한 능력을 포함하는 다수의 벡터 레지스터를 지정하기 위한 특히 간단하고 효율적인 메커니즘을 제공한다.

[0078] 도 7은 벡터 레지스터들(350)의 세트가 어떻게 다수의 논리적 섹션들(360, 365)로 형성되는 것으로 간주될 수 있는지를 예시한다. 각각의 벡터 레지스터(355)는 각각의 섹션 내의 부분(357, 359)을 갖는다. 도 7에는 두 개의 섹션이 도시되어 있지만, 다른 구현에서는 3개 이상의 섹션이 제공될 수 있다. 일부 구현에서는 벡터 레지스터의 부분(357, 359)마다 단일 능력만이 제공되는 반면, 다른 구현에서는 레지스터의 각각의 부분이 여러 능력을 보유할 만큼 충분히 클 수 있다. 이러한 접근법은 주어진 벡터 메모리 액세스 명령어를 포함한 벡터 명령어의 실행이 다수의 "박동"으로 분할되게 할 수 있으며, 각각의 박동 동안에는 벡터 명령어를 실행하기 위해 벡터 레지스터 세트의 하나의 섹션만이 액세스된다. 벡터 명령어가 다수의 박동으로 분할될 수 있게 함으로써, 벡터 명령어의 실행이 하나 이상의 다른 벡터 명령어의 실행과 중첩되게 할 수 있으며, 이는 매우 효율적인 구현으로 이어질 수 있다. 예를 들어, 주어진 벡터 메모리 액세스 명령어는 벡터 산술 명령어와 중첩될 수 있다. 특히, 하나의 예시적인 구현에서, 임의의 특정 박동 동안 메모리 액세스 동작을 수행하는 데 필요한 데이터 요소와 능력은 모두 벡터 레지스터 세트의 단일 섹션으로부터 획득될 수 있으며, 이는 이어서 중첩된 명령어의 실행 동안 임의의 다른 섹션이 액세스에 이용 가능하게 남겨둔다. 박동 기반 구현에 대한 더 상세한 내용은 나중에 도 10 및 11을 참조하여 더 상세히 논의될 것이다.

[0079] 도 8a 내지 도 8c는 본 명세서에 설명된 수집 또는 분산 동작을 수행할 때 사용될 수 있는 데이터 요소들 및 관련 능력들의 상이한 특정한 예시적인 배열을 예시한다. 도 8a에서 표시된 바와 같이, "CX"라는 용어는 대응하는 데이터 값 "DX"에 대한 메모리 어드레스를 결정하는 데 사용되는 능력을 식별한다. 도 8a에 도시된 벡터 레지스터(400)는 벡터 메모리 액세스 명령어의 실행 동안에 액세스되는 데이터 요소들과 관련된 벡터 레지스터이다. 이 예시적인 구현에서는 벡터 레지스터(400)의 폭이 128 비트이고, 각각의 데이터 요소의 폭이 32 비트이며, 결과적으로 4개의 데이터 요소가 벡터 레지스터(400)와 관련된다고 가정한다. 각각의 데이터 요소는 벡터 레지스터(400)의 대응하는 데이터 레인과 관련되는 것으로 간주될 수 있으며, 따라서 도 8a에 도시된 바와 같이 데이터 레인은 0 내지 3의 값을 가질 수 있다.

[0080] 도 8a 내지 도 8c에 도시된 예들에서, 능력들의 폭은 64 비트이고, 따라서 도 8a의 예에서 각각의 벡터 레지스터(405, 410)는 두 개의 능력을 저장할 수 있다(도 8a 내지 도 8c에서의 예시의 목적상, 앞서 논의한 태그 값들을 보유하기 위해 제공되는 임의의 추가 비트들은 생략됨). 특정 벡터 레지스터 내의 각각의 능력은 관련 능력 레인을 점유하는 것으로 간주될 수 있으므로, 도 8a의 예에는 레인 0 및 1이라고 하는 두 개의 능력 레인이 있다. 도 8a에 도시된 바와 같이, 능력 C0은 제1 능력 레지스터 $Q_N(405)$ 내에서 능력 레인 0을 점유하고, 능력 C1은 제2 능력 레지스터 $Q_{N+1}(410)$ 내에서 능력 레인 0을 점유하고, 능력 C2는 제1 능력 레지스터 $Q_N(405)$ 내에서 능력 레인 1을 점유하고, 능력 C3은 제2 능력 레지스터 $Q_{N+1}(410)$ 내에서 능력 레인 1을 점유한다. 따라서, 이러한 배열에 의해, 벡터 레지스터(400) 내의 인접 위치들과 관련된 데이터 요소들의 임의의 주어진 쌍에 대해, 관련 능력은 복수의 벡터 레지스터(405, 410) 중의 상이한 벡터 레지스터들에 저장된다는 것이 인식될 수 있다.

- [0081] 이러한 배열은 매우 유리한 것으로 밝혀졌는데, 이는 그것이 데이터 요소들의 특정 시퀀스와 관련하여 요구되는 능력들이 모두 벡터 레지스터들의 동일한 부분(357, 359) 내에서 발견될 수 있음을 의미하기 때문이다. 특히, 도 8a에 도시된 예에서, 데이터 요소 D0 및 D1 및 이들 데이터 요소의 메모리 어드레스를 식별하는 데 필요한 능력 C0 및 C1은 모두 관련 벡터 레지스터의 하부 절반에서 발견될 수 있으며, 마찬가지로 데이터 요소 D2 및 D3 및 이들 데이터 요소의 메모리 어드레스를 식별하는 데 필요한 능력 C2 및 C3은 모두 관련 벡터 레지스터의 상부 절반에서 발견될 수 있다. 이는 예를 들어 앞서 언급한 벡터 메모리 액세스 명령어의 박동별 실행을 지원할 수 있다.
- [0082] 도 8a에서 데이터 값은 32 비트이지만, 이는 필수 사항이 아니며, 도 8b는 데이터 요소의 폭이 16 비트인 대안적인 예를 도시한다. 따라서, 128 비트 폭의 벡터 레지스터(415)는 8개의 데이터 요소와 관련될 수 있으며, 관련 능력을 보유하기 위해 4개의 벡터 레지스터 Q_N 내지 Q_{N+3} (420, 425, 430, 435)이 필요하다. 다시, 능력은 도 8a의 방식과 유사한 방식으로 레이아웃되며, 처음 4개의 능력은 벡터 레지스터들(420, 425, 430, 435)의 하부 절반 내에 저장되고, 마지막 4개의 능력은 이들 벡터 레지스터의 상부 절반 내에 저장된다.
- [0083] 벡터 레지스터를 128 비트 레지스터로 간주할 필요도 없으며, 도 8c의 예에서 각각의 레지스터의 폭은 256 비트이다. 이 특정 예에서 데이터 요소의 폭은 32 비트이고, 능력의 폭은 다른 예와 동일하게 유지되는데, 즉 64 비트이다. 따라서, 이 예에서, 벡터 레지스터(440)와 관련된 8개의 데이터 요소가 있고, 2개의 벡터 레지스터(445, 450)가 8개의 능력을 저장하는 데 사용되며, 각각의 레지스터 내에 4개의 능력이 배치됨을 알 수 있다. 능력들은 능력 레인 0, 능력 레인 1, 능력 레인 2 및 능력 레인 3에 오름차순으로 저장되도록 배열되므로, 도 8a 및 8b의 다른 두 예를 참조하여 앞서 논의한 일반 패턴을 따른다.
- [0084] 벡터 메모리 액세스 명령어의 앞서 설명한 박동별 실행을 수행할 때, 하나의 예시적인 구현에서 벡터 레지스터의 각각의 섹션은 하나 이상의 능력을 저장하도록 배열될 수 있다. 따라서, 도 8a 또는 도 8b의 예들을 고려하면, 벡터 레지스터들은 두 개의 섹션으로 형성되는 것으로 간주될 수 있으며, 이는 필요한 액세스 동작들의 절반이 제1 박동에서 처리되고 나머지 절반이 제2 박동에서 처리될 수 있게 한다. 유사하게, 도 8c를 고려하면, 벡터 레지스터 세트는 2개 또는 4개의 섹션으로 형성되는 것으로 간주될 수 있으며, 이는 필요한 액세스 동작들이 각각 2개 또는 4개의 박동에 걸쳐 수행될 수 있게 한다. 그러나, 벡터 레지스터의 각각의 논리 섹션이 적어도 하나의 능력을 수용할 만큼 충분히 넓어야 하는 것이 반드시 필요하지는 않다는 점에 유의해야 한다. 예를 들어, 일부 구현에서는 64 비트 능력과 함께 능력 크기보다 작은 섹션 크기, 예컨대 32 비트 섹션 크기를 갖는 것이 가능할 수 있다.
- [0085] 도 9는 도 8a 내지 도 8c에 개략적으로 예시된 바와 같은 능력들의 레이아웃을 사용할 때 각각의 데이터 요소에 대한 관련 능력이 어떻게 결정될 수 있는지를 예시하는 흐름도이다. 단계 450에서, 파라미터 M은 능력 레인들의 수와 동일하게 설정되고, 파라미터 P는 능력을 보유하는 벡터 레지스터들의 수와 동일하게 설정된다. 단계 455에서, 벡터 레지스터들은 값 0 내지 P-1의 시퀀스에 의해 식별되는 것으로 간주되고, 능력 레인들은 값 0 내지 M-1의 시퀀스에 의해 식별되는 것으로 간주된다. 단계 460에서, 파라미터 X는 0으로 설정되고, 이어서 단계 465에서, 레인 X 내의 데이터 요소에 대해 계산 X/P가 수행된다.
- [0086] 단계 470에서, 위의 계산으로부터 생성된 몫 및 나머지는 관련 능력을 포함하는 능력 레인 및 벡터 레지스터를 각각 식별하는 데 사용된다. 단계 475에서, 데이터 레인 X가 마지막 데이터 레인인지 여부가 결정되고, 그렇지 않은 경우, 단계 465로 돌아가기 전에 X의 값이 단계 480에서 증가된다. 단계 475에서 데이터 레인 X가 마지막 데이터 레인인 것으로 결정되면, 프로세스는 단계 485에서 종료된다.
- [0087] 디지털 신호 처리(DSP)와 같은 일부 응용에서는 거의 동일한 수의 ALU 및 로드/저장 명령어가 있을 수 있으므로, MAC과 같은 일부 대형 블록은 상당한 양의 시간 동안 유향 상태로 남겨질 수 있다. 이러한 비효율성은 벡터 아키텍처에서 악화될 수 있는데, 이는 더 높은 성능을 얻기 위해 실행 자원이 벡터 레인들의 수에 따라 스케일링되기 때문이다. 더 작은 프로세서(예를 들어, 단일 발행, 순차 코어)에서는 완전히 스케일링된 벡터 파이프라인의 영역 오버헤드가 엄청나게 클 수 있다. 이용 가능한 실행 자원을 더 잘 사용하면서 영역 영향을 최소화하는 한 가지 접근법은 도 10에 도시된 바와 같이 명령어들의 실행을 중첩시키는 것이다. 이 예에서, 세 개의 벡터 명령어는 로드 명령어 VLDR, 곱셈 명령어 VMUL 및 시프트 명령어 VSHR을 포함하며, 이러한 모든 명령어들은 이들 간에 데이터 종속성이 있더라도 동시에 실행될 수 있다. 이는 VMUL의 요소 1이 Q1 레지스터 전체가 아니라 Q1의 요소 1에만 의존하므로 VLDR의 실행이 완료되기 전에 VMUL의 실행이 시작될 수 있기 때문이다. 명령어들이 중첩되는 것을 허용함으로써, 곱셈기와 같은 고가의 블록을 더 많은 시간 동안 활성 상태로 유지할 수 있다.

- [0088] 따라서, 마이크로아키텍처 구현이 벡터 명령어들의 실행을 중첩시킬 수 있게 하는 것이 바람직할 수 있다. 그러나 아키텍처가 고정된 양의 명령어 중첩이 있다고 가정하면, 마이크로아키텍처 구현이 실제로 아키텍처에 의해 가정되는 명령어 중첩의 양과 매칭되는 경우에는 높은 효율성을 제공할 수 있지만, 상이한 중첩을 사용하거나 전혀 중첩시키지 않는 상이한 마이크로아키텍처로 스케일링되는 경우에는 문제가 발생할 수 있다.
- [0089] 대신, 아키텍처는 도 11의 예에 도시된 바와 같이 상이한 중첩들의 범위를 지원할 수 있다. 벡터 명령어의 실행은 "박동"으로 지칭되는 부분들로 분할되며, 각각의 박동은 미리 결정된 크기의 벡터의 일부의 처리에 대응한다. 박동은 완전히 실행되거나 전혀 실행되지 않는 벡터 명령어의 원자 부분이며, 부분적으로 실행될 수 없다. 하나의 박동에서 처리되는 벡터 부분의 크기는 아키텍처에 의해 정의되며 벡터의 임의의 분수일 수 있다. 도 11의 예에서 박동은 벡터 폭의 1/4에 대응하는 처리로서 정의되므로 벡터 명령어당 4개의 박동이 있다. 분명히, 이것은 단지 하나의 예일 뿐이며, 다른 아키텍처는 2 또는 8과 같은 상이한 수의 박동을 사용할 수 있다. 하나의 박동에 대응하는 벡터의 부분은 처리 중인 벡터의 데이터 요소 크기와 동일한 크기이거나 그보다 크거나 작을 수 있다. 따라서, 요소 크기가 구현마다 다르거나 상이한 명령어들 간에 런타임에서 다르더라도, 박동은 벡터 처리의 특정 고정 폭이다. 하나의 박동에서 처리되는 벡터 부분이 여러 데이터 요소를 포함하는 경우, 각각의 요소가 독립적으로 처리되도록 각각의 요소들 사이의 경계에서 운반 신호들이 불능화될 수 있다. 하나의 박동에서 처리되는 벡터 부분이 요소의 일부에만 대응하고 하드웨어가 여러 박동을 병렬로 계산하기에 불충분한 경우, 처리의 하나의 박동 동안 생성된 운반 출력은 2개의 박동의 결과들이 함께 데이터 요소를 형성하도록 처리의 다음 박동에 운반 입력으로 입력될 수 있다.
- [0090] 도 11에 도시된 바와 같이, 처리 회로(4)의 상이한 마이크로아키텍처 구현들은 추상적 아키텍처 클록의 하나의 "틱(tick)"에서 상이한 수의 박동들을 실행할 수 있다. 여기서, "틱"은 아키텍처 상태 진전의 단위에 대응한다 (예를 들어, 간단한 아키텍처에서 각각의 틱은 다음 명령어를 가리키도록 프로그램 카운터를 업데이트하는 것을 포함하여, 명령어 실행과 관련된 모든 아키텍처 상태를 업데이트하는 경우에 대응할 수 있음). 이 분야의 통상의 기술자는, 파이프라이닝과 같은 공지된 마이크로아키텍처 기술은 단일 틱이 하드웨어 레벨에서 수행하기 위해 다수의 클록 사이클을 필요로 할 수 있다는 것을 의미할 수 있고, 실제로 하드웨어 레벨에서의 단일 클록 사이클은 다수의 명령어의 다수의 부분을 처리할 수 있다는 것을 이해할 것이다. 그러나, 이러한 마이크로아키텍처 기술은 소프트웨어에 대해 가시적이지 않은데, 이는 틱이 아키텍처 레벨에서 원자적이기 때문이다. 간결성을 위해, 이러한 마이크로아키텍처는 본 개시에 대한 추가 설명 동안 무시된다.
- [0091] 도 11의 하부 예에 도시된 바와 같이, 일부 구현은 하나의 틱 내에서 모든 박동들을 병렬로 처리하기 위한 충분한 하드웨어 자원을 제공함으로써 동일한 틱에서 벡터 명령어의 4개의 박동 모두를 스케줄링할 수 있다. 이는 더 높은 성능 구현에 적합할 수 있다. 이 경우, 전체 명령어를 하나의 틱에서 완료할 수 있으므로 아키텍처 레벨에서 명령어들 간의 어떠한 중첩도 필요하지 않다.
- [0092] 반면에, 더 영역 효율적인 구현은 틱당 2개의 박동만을 처리할 수 있는 더 좁은 처리 유닛을 제공할 수 있으며, 도 11의 중간 예에 도시된 바와 같이, 명령어 실행은 제1 명령어의 제3 또는 제4 박동과 병렬로 수행되는 제2 벡터 명령어의 제1 및 제2 박동과 중첩될 수 있으며, 이러한 명령어들은 처리 회로 내의 상이한 실행 유닛들에서 실행된다(예를 들어, 도 11에서, 제1 명령어는 로드/저장 유닛(18)을 사용하여 실행되는 로드 명령어이고(그리고, 예를 들어, 본 명세서에 설명된 유형의 벡터 수집 명령어일 수 있음), 제2 명령어는 처리 회로(4) 내에 제공된 MAC 유닛을 사용하여 실행되는 곱셈 누산 명령어임).
- [0093] 훨씬 더 에너지/영역 효율적인 구현은 더 좁고 한 번에 하나의 박동만을 처리할 수 있는 하드웨어 유닛을 제공할 수 있으며, 이 경우, 도 11의 상부 예에 도시된 바와 같이 명령어 실행이 두 박동만큼 중첩되고 엇갈리면서 틱당 하나의 박동이 처리될 수 있다. 하나의 예시적인 구현에서, 섹션 크기는 명령어들 사이의 엇갈림의 양에 영향을 주기 위해 사용될 수 있다(왜냐하면 특정 박동을 실행할 때 동일한 섹션으로부터 모든 데이터를 획득하는 것이 바람직하기 때문이다). 도 11에 예시된 상부 예에서, 예를 들어 박동 크기가 32 비트이지만 섹션 크기가 64 비트일 수 있으며, 따라서 이는 명령어들이 두 박동만큼 엇갈리는 이유이다.
- [0094] 도 11에 도시된 중첩은 일부 예일 뿐이며, 다른 구현도 가능하다는 것을 이해할 것이다. 예를 들어, 처리 회로(4)의 일부 구현은 동일한 틱에서 병렬로 여러 명령어를 이중 발행하는 것을 지원할 수 있으므로, 명령어들의 더 큰 처리량이 존재한다. 이 경우, 한 사이클에서 함께 시작되는 두 개 이상의 벡터 명령어는 다음 사이클에서 시작되는 두 개 이상의 벡터 명령어와 일부 박동이 중첩될 수 있다.
- [0095] 상이한 수행 포인트로 스케일링하기 위해 구현마다 중첩의 양을 변경할 뿐만 아니라, 벡터 명령어들 간의 중첩의 양도 프로그램 내의 벡터 명령어들의 실행의 상이한 인스턴스들 간에 런타임에 변경될 수 있다. 따라서, 처

리 회로(4)는 이전 명령어에 대한 주어진 명령어의 실행 타이밍을 제어하기 위해 도 1에 도시된 바와 같이 박동 제어 회로(20)를 구비할 수 있다. 이는 구현하기 더 어렵거나 명령어에 의해 이용 가능한 자원에 의존하는 특정 코너 사례에서 명령어들을 중첩시키지 않도록 선택할 수 있는 자유를 마이크로아키텍처에 제공한다. 예를 들어, 동일한 자원을 필요로 하는 주어진 유형(예를 들어, 곱셈 누산)의 연속 명령어들이 있고, 이용 가능한 모든 MAC 또는 ALU 자원들이 이미 다른 명령어에서 사용되고 있다면, 다음 명령어 실행을 시작하기에 충분한 자유 자원이 없을 수 있으므로, 중첩시키는 것이 아니라, 제1 명령어가 완료될 때까지 제2 명령어의 발행이 대기될 수 있다.

[0096] 도 12는 벡터 능력 메모리 전송 명령어 시퀀스가 메모리와 다수의 벡터 레지스터 사이에서 일련의 능력들을 이동시키면서 이러한 능력들이 도 8a 내지 8c의 이전 예들을 참조하여 예로서 예시된 형태의 배열들로 벡터 레지스터들 내에 저장되도록 보장하기 위해 필요한 재배열을 수행하는 데 사용될 수 있는 방법을 예시하는 흐름도이다.

[0097] 단계 490에서, 벡터 능력 메모리 전송 명령어 시퀀스가 디코딩되며, 각각의 이러한 명령어는 관련 액세스 패턴을 정의하고, 앞서 설명한 벡터 수집/분산 명령어의 임의의 특정 인스턴스에 의해 요구되는 능력 서브세트를 식별한다. 하나의 예시적인 구현에서, 각각의 개별 벡터 능력 메모리 전송 명령어는 시퀀스 내의 각각의 다른 벡터 능력 메모리 전송 명령어에 대한 상이한 능력 서브세트를 식별한다.

[0098] 이어서, 단계 492에서, 각각의 벡터 능력 메모리 전송 명령어의 액세스 패턴에 의해 정의된 바와 같이 디인터리빙(로드 동작이 수행되는 경우) 또는 인터리빙(저장 동작이 수행되는 경우)을 수행하는 동안, 능력들이 메모리와 식별된 벡터 레지스터 사이에서 이동된다. 결과적으로, 복수의 능력은 메모리에 순차적으로 저장되도록 배열될 수 있는 반면, 다수의 벡터 레지스터에서는 복수의 능력이 디인터리빙되어 메모리에 순차적으로 저장되는 능력들의 임의의 주어진 쌍이 상이한 벡터 레지스터들에 저장되게 한다.

[0099] 도 12에 예시된 단계들을 수행하기 위해 사용되는 복수의 벡터 능력 메모리 전송 명령어는 프로그램 순서로 서로 직접 뒤따를 필요가 없으며, 따라서 순차적으로 하나씩 실행될 필요가 없다. 시퀀스 내의 모든 벡터 능력 메모리 전송 명령어가 실행되었다면, 메모리와 벡터 레지스터 간에 이동할 때 필요한 능력 재배열이 수행되었을 것이다.

[0100] 하나의 예시적인 구현에서, 메모리는 다수의 메모리 뱅크로 형성되고, 각각의 벡터 능력 메모리 전송 명령어에 대해, 액세스 패턴은 해당 벡터 능력 메모리 전송 명령어가 실행될 때 2개 이상의 메모리 뱅크가 액세스되게 하도록 정의된다. 뱅크형 메모리는 하드웨어가 메모리로의/메모리로부터의 병렬 전송을 구현하는 것을 더 쉽게 하며, 따라서 이를 가능하게 하는 액세스 패턴을 지정하는 것이 유리하다. 이는 2개의 메모리 뱅크(496, 498)로 형성된 메모리의 예에 대해 도 13에 개략적으로 예시되어 있으며, 여기서 각각의 메모리 뱅크의 폭은 64 비트이다. 이러한 메모리 뱅크 구성을 이용하면, 메모리 액세스 논리(494)가 메모리 어드레스를 처리할 때, 어드레스의 비트 3은 어떤 뱅크에 액세스할지 결정하기 위해 고려될 수 있다. 특히, 어드레스의 비트 3(즉, 제1 어드레스 비트가 비트 0이라고 가정하면 제4 어드레스 비트)이 논리 0 값이면, 메모리 뱅크(496)가 액세스되는 반면, 어드레스의 비트 3이 논리 1 값이면, 다른 메모리 뱅크(498)가 액세스된다. 능력은 64 비트 능력이므로, 홀수 능력은 하나의 뱅크에 저장될 것인 반면, 짝수 능력은 다른 뱅크에 저장된다는 것을 이해할 것이다.

[0101] 순전히 예로서, 도 8a에 도시된 능력들의 배열을 고려하면, 메모리 내에 순차적으로 위치되는 능력 C0 내지 C3은 다음과 같이 두 개의 벡터 능력 메모리 전송 명령어를 사용하여 능력 레지스터들(405 및 410)에 로드될 수 있다:

[0102] VLDRC2_1: C0 → Qn[63:0], C3 → Q(n+1)[127:64]

[0103] VLDRC2_2: C1 → Q(n+1)[63:0], C2 → Qn[127:64]

[0104] 도 12를 참조하면, 이러한 명령어들 각각을 실행할 때, 두 뱅크(496, 498)가 모두 액세스된다는 것을 이해할 것인데, 그 이유는 능력 C0이 능력 C3에 대한 상이한 뱅크에 있을 것이고, 능력 C1이 능력 C2에 대한 상이한 뱅크에 있을 것이기 때문이다. 또한, 각각의 명령어에 의해 전송되는 2개의 능력은 벡터 레지스터들의 상이한 능력 레인들 내에 상주하므로, 하나의 예시적인 구현에서 동시에 벡터 레지스터들에 기입될 수 있다.

[0105] 도 14는 사용될 수도 있는 시뮬레이터 구현을 예시한다. 앞서 설명한 예들은 관련 기술을 지원하는 특정 처리 하드웨어를 동작시키기 위한 장치 및 방법의 측면에서 본 발명을 구현하지만, 본 명세서에 설명된 예들에 따라 컴퓨터 프로그램의 사용을 통해 구현되는 명령어 실행 환경을 제공하는 것이 또한 가능하다. 이러한 컴퓨터 프로그램은, 하드웨어 아키텍처의 소프트웨어 기반 구현을 제공하는 한, 시뮬레이터로서 종종 지칭된다. 다양한

시뮬레이터 컴퓨터 프로그램은 에뮬레이터, 가상 머신, 모델, 및 동적 바이너리 변환기를 비롯한, 바이너리 변환기를 포함한다. 전형적으로, 시뮬레이터 구현은 호스트 프로세서(515) 상에서 실행될 수 있으며, 선택적으로 호스트 운영 체제(510)를 실행하여, 시뮬레이터 프로그램(505)을 지원할 수 있다. 일부 배열들에서, 하드웨어와 제공된 명령어 실행 환경 사이에 다수의 시뮬레이션 계층이 있을 수 있고/있거나, 동일한 호스트 프로세서 상에 제공된 다수의 별개의 명령어 실행 환경이 있을 수 있다. 역사적으로, 합리적인 속도로 실행되는 시뮬레이터 구현을 제공하기 위해 강력한 프로세서가 필요하였지만, 그러한 접근법은 특정 상황에서, 예컨대 호환성 또는 재사용 이유 때문에 다른 프로세서에 기본적인(native) 코드를 실행하려는 요구가 있을 때, 정당화될 수 있다. 예를 들어, 시뮬레이터 구현은 호스트 프로세서 하드웨어에 의해 지원되지 않는 추가적인 기능을 갖는 명령어 실행 환경을 제공하거나, 상이한 하드웨어 아키텍처와 전형적으로 관련되는 명령어 실행 환경을 제공할 수 있다. 시뮬레이션의 개요는 문헌["Some Efficient Architecture Simulation Techniques", Robert Bedichek, Winter 1990, USENIX Conference, Pages 53 to 63]에 제공되어 있다.

[0106] 예들이 특정 하드웨어 구조 또는 특징을 참조하여 전술된 경우, 시뮬레이션된 구현에서 동등한 기능이 적합한 소프트웨어 구조 또는 특징에 의해 제공될 수 있다. 예를 들어, 특정 회로는 시뮬레이션된 구현에 컴퓨터 프로그램 논리로 제공될 수 있다. 마찬가지로, 레지스터 또는 캐시와 같은 메모리 하드웨어가 시뮬레이션된 구현에서 소프트웨어 데이터 구조로 제공될 수 있다. 또한, 하드웨어 장치(2) 내의 메모리(8)에 액세스하는 데 사용되는 물리적 어드레스 공간은 시뮬레이터(505)에 의한 호스트 운영 체제(510)에 의해 사용되는 가상 어드레스 공간에 매핑되는 시뮬레이션된 어드레스 공간으로 에뮬레이팅될 수 있다. 앞서 설명한 예에서 언급된 하드웨어 요소들 중 하나 이상이 호스트 하드웨어(예를 들어, 호스트 프로세서(515)) 상에 존재하는 배열들에서, 일부 시뮬레이션된 구현은 적합한 경우 호스트 하드웨어를 사용할 수 있다.

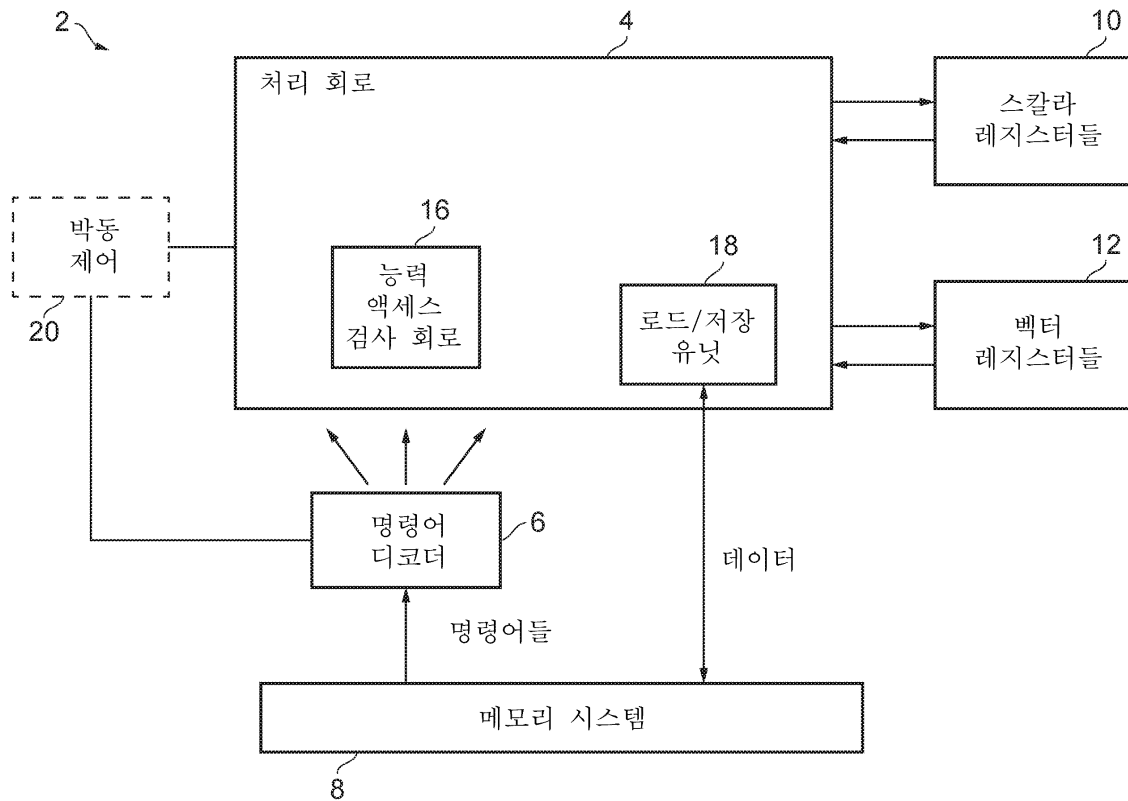
[0107] 시뮬레이터 프로그램(505)은 컴퓨터 판독 가능 저장 매체(이는 비일시적 매체일 수 있음) 상에 저장될 수 있으며, 타겟 코드(500)(이는 애플리케이션, 운영 체제, 및 하이퍼바이저를 포함할 수 있음)에 가상 하드웨어 인터페이스(명령어 실행 환경)를 제공하고, 이는 시뮬레이터 프로그램(505)에 의해 모델링되는 하드웨어 아키텍처의 하드웨어 인터페이스와 동일하다. 따라서, 타겟 코드(500)의 프로그램 명령어는 시뮬레이터 프로그램(505)을 사용하는 명령어 실행 환경 내로부터 실행될 수 있고, 따라서 위에서 논의한 장치(2)의 하드웨어 특징을 실제로 갖지 않는 호스트 컴퓨터(515)가 그러한 특징을 에뮬레이팅할 수 있다. 시뮬레이터 프로그램은 처리 회로(4)의 거동을 에뮬레이팅하기 위한 처리 프로그램 논리(520), 명령어 디코더(6)의 거동을 에뮬레이팅하기 위한 명령어 디코드 프로그램 논리(525), 및 벡터 레지스터(12)를 에뮬레이팅하기 위해 데이터 구조를 유지하기 위한 벡터 레지스터 에뮬레이팅 프로그램 논리(522)를 포함할 수 있다. 따라서, 능력을 사용하여 벡터 수집 또는 분산 동작을 수행하기 위한 본 명세서에 설명된 기술들은 도 14의 예에서 시뮬레이터 프로그램(505)에 의해 소프트웨어에서 수행될 수 있다.

[0108] 본 출원에서 "...하도록 구성된"이라는 단어는 장치의 요소가 정의된 동작을 수행할 수 있는 구성을 갖는다는 것을 의미하는 데 사용된다. 이러한 맥락에서, "구성"은 하드웨어 또는 소프트웨어의 배열 또는 상호 연결 방식을 의미한다. 예를 들어, 장치는 정의된 동작을 제공하는 전용 하드웨어를 가질 수 있거나, 프로세서 또는 다른 처리 디바이스가 기능을 수행하도록 프로그래밍될 수 있다. "구성된"은 정의된 동작을 제공하기 위해 장치 요소를 어떤 방식으로든 변경해야 한다는 것을 의미하지 않는다.

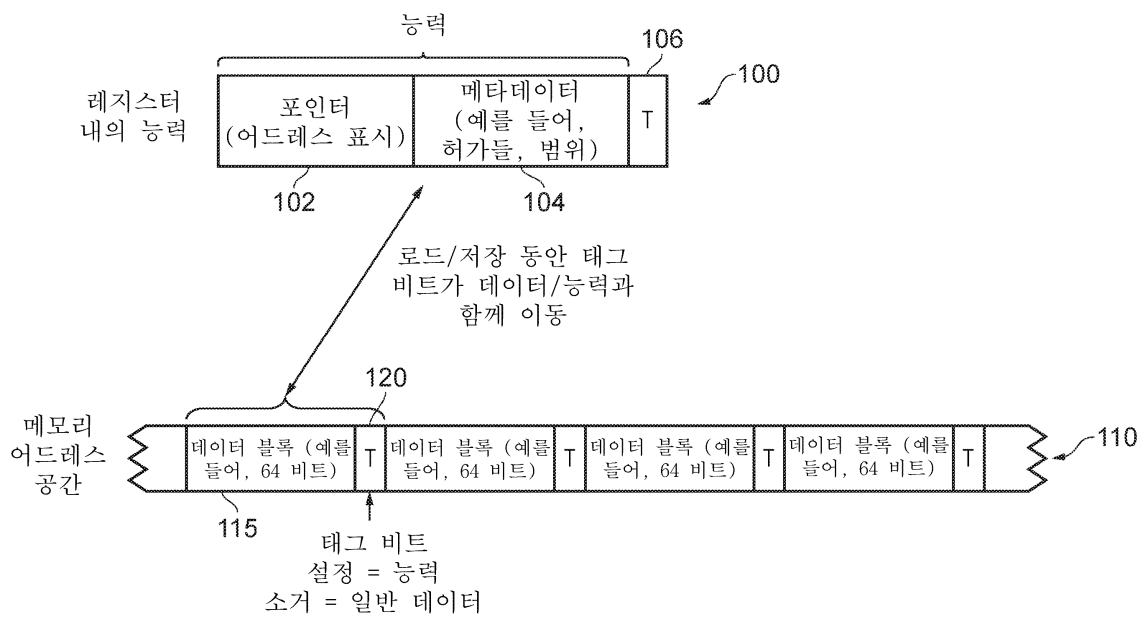
[0109] 본 발명의 예시적인 예들이 첨부 도면들을 참조하여 본 명세서에 상세히 설명되었지만, 본 발명은 그러한 정확한 예들로 제한되지 않으며, 첨부된 청구항들에 의해 정의된 바와 같은 본 발명의 범주로부터 벗어남이 없이 예들에서 다양한 변경들, 추가들 및 수정들이 이 분야의 통상의 기술자에 의해 이루어질 수 있다는 것이 이해되어야 한다. 예를 들어, 본 발명의 범주로부터 벗어남이 없이 독립 청구항들의 특징들과의 종속 청구항들의 특징들의 다양한 조합들이 이루어질 수 있다.

도면

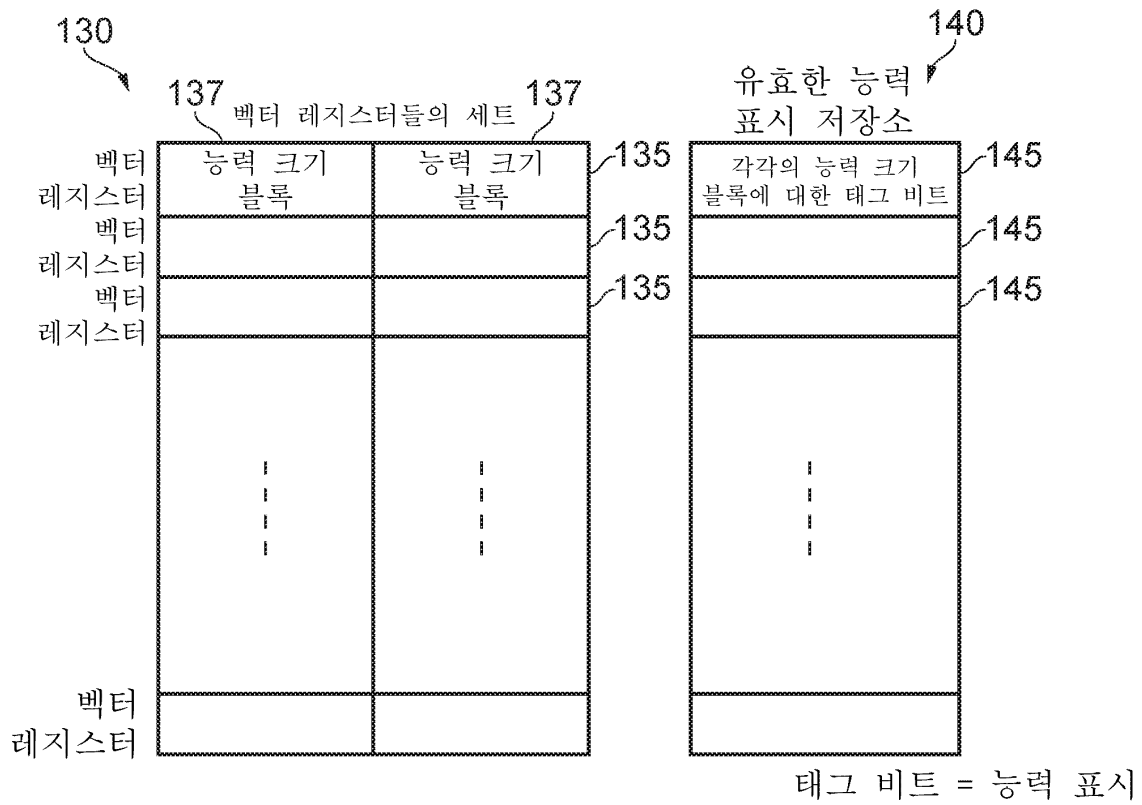
도면1



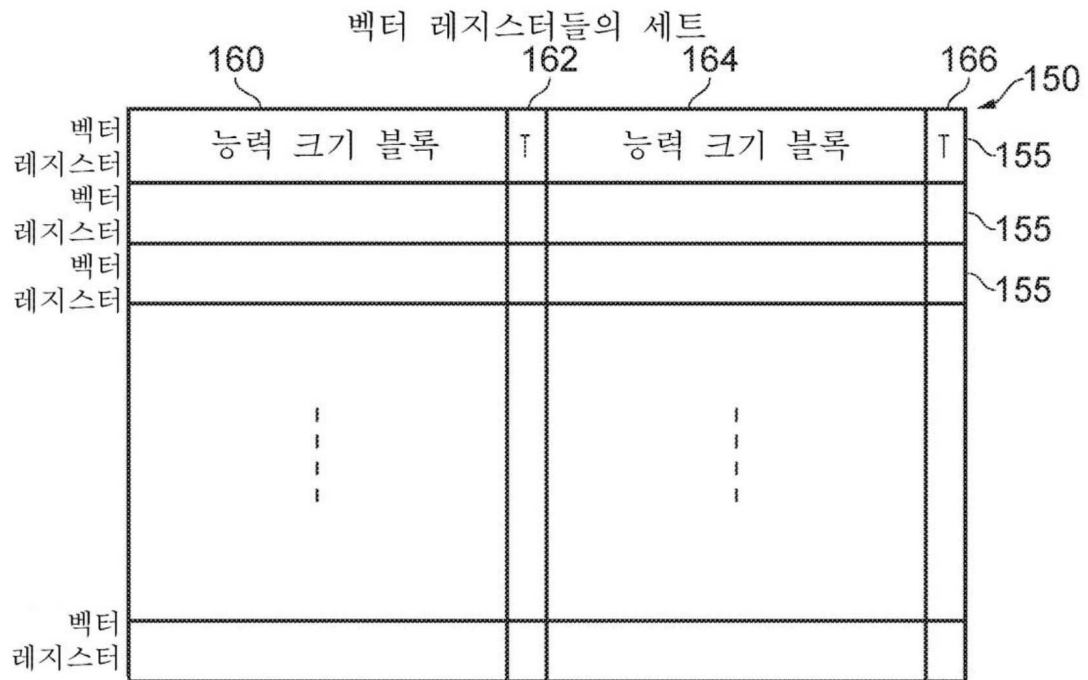
도면2



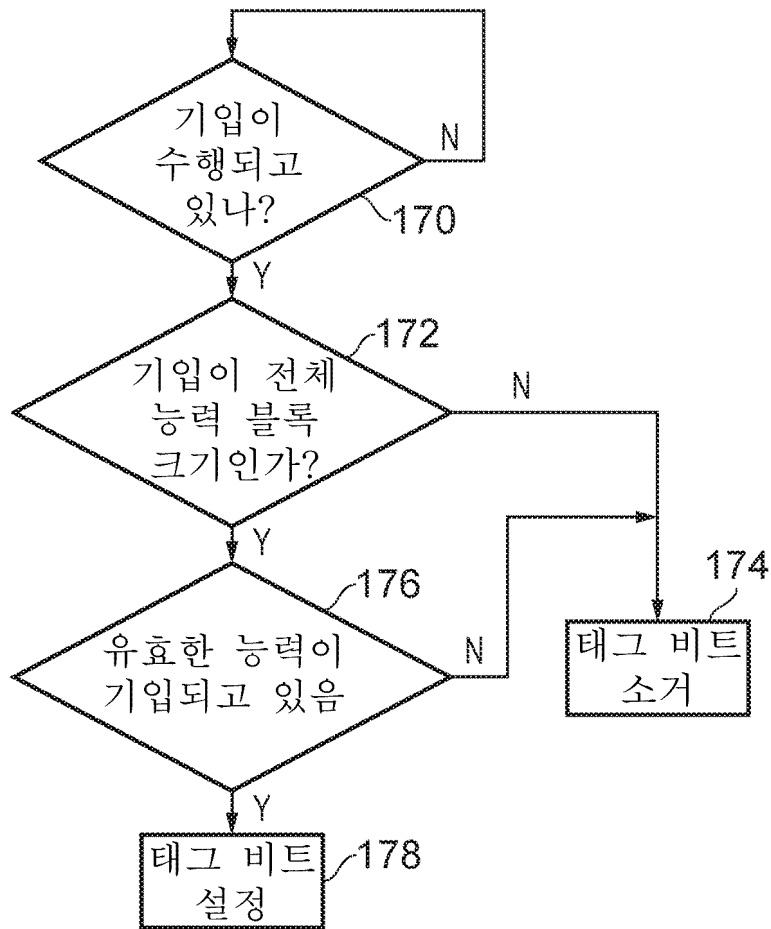
도면3a



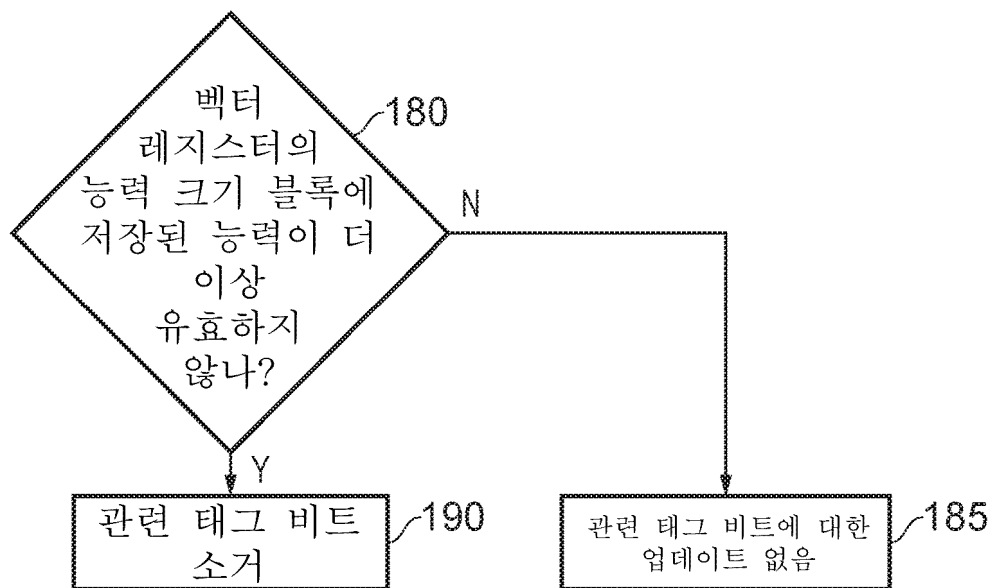
도면3b



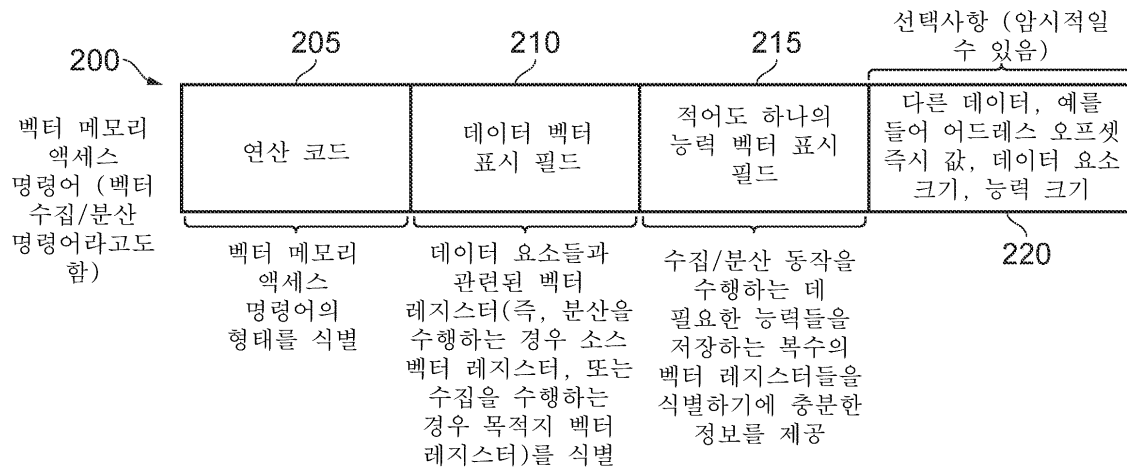
도면4a



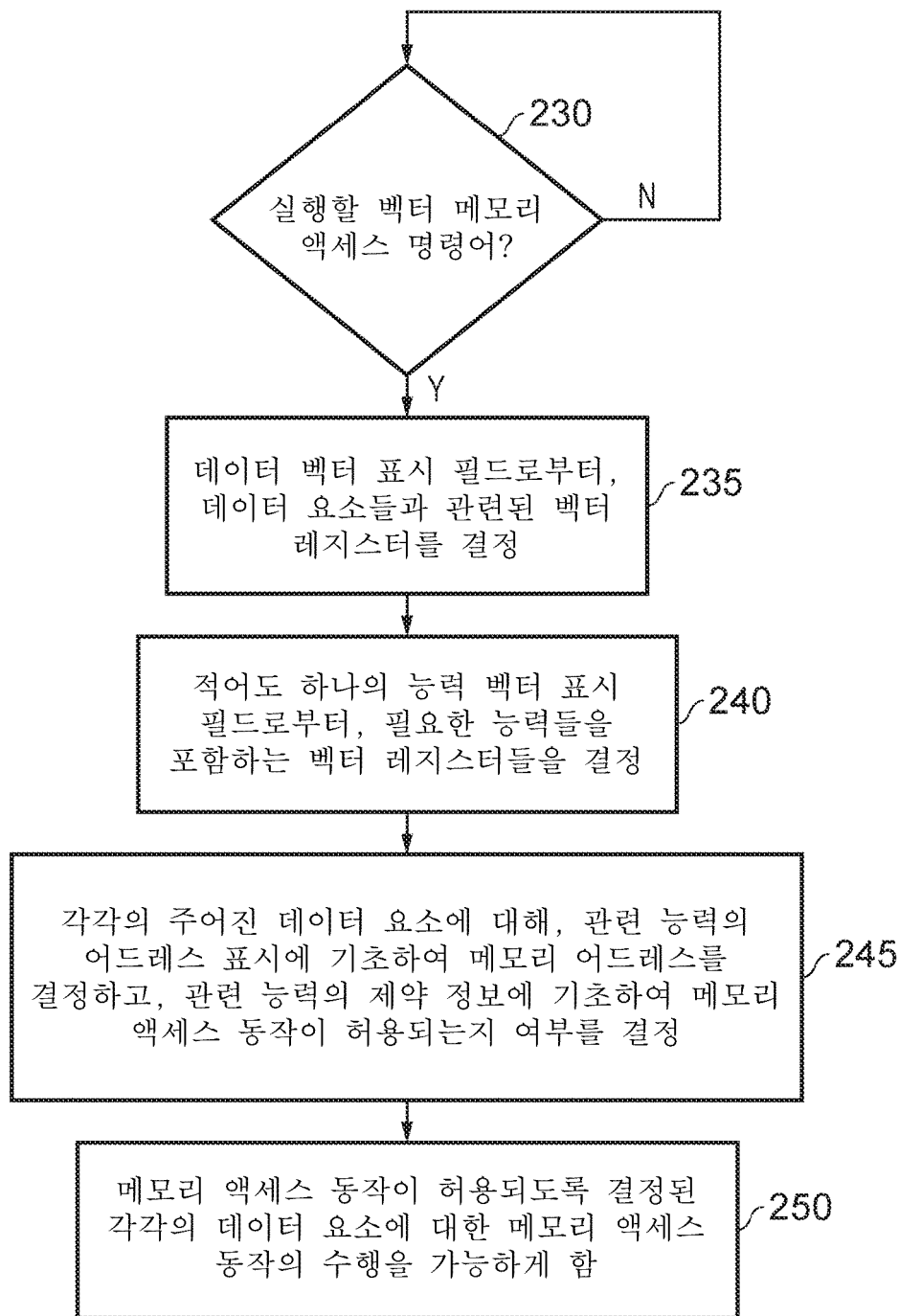
도면4b



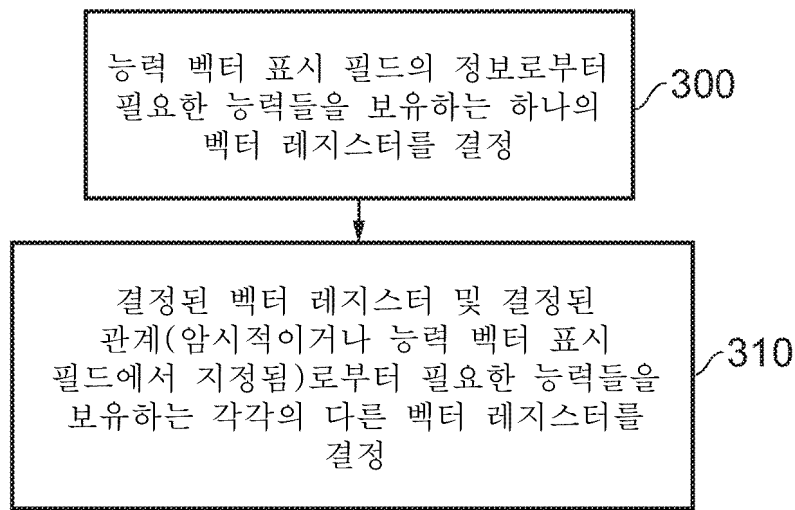
도면5a



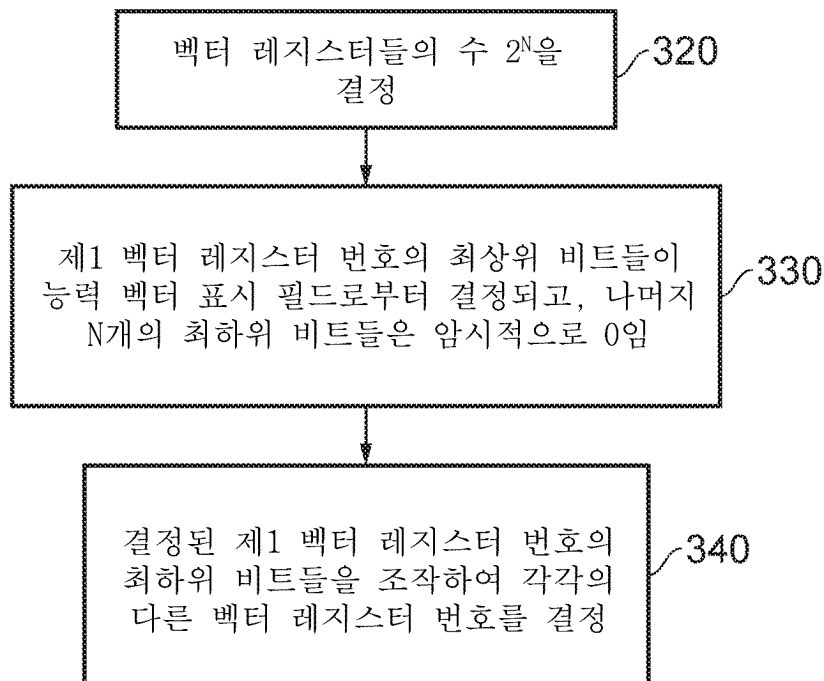
도면5b



도면6a

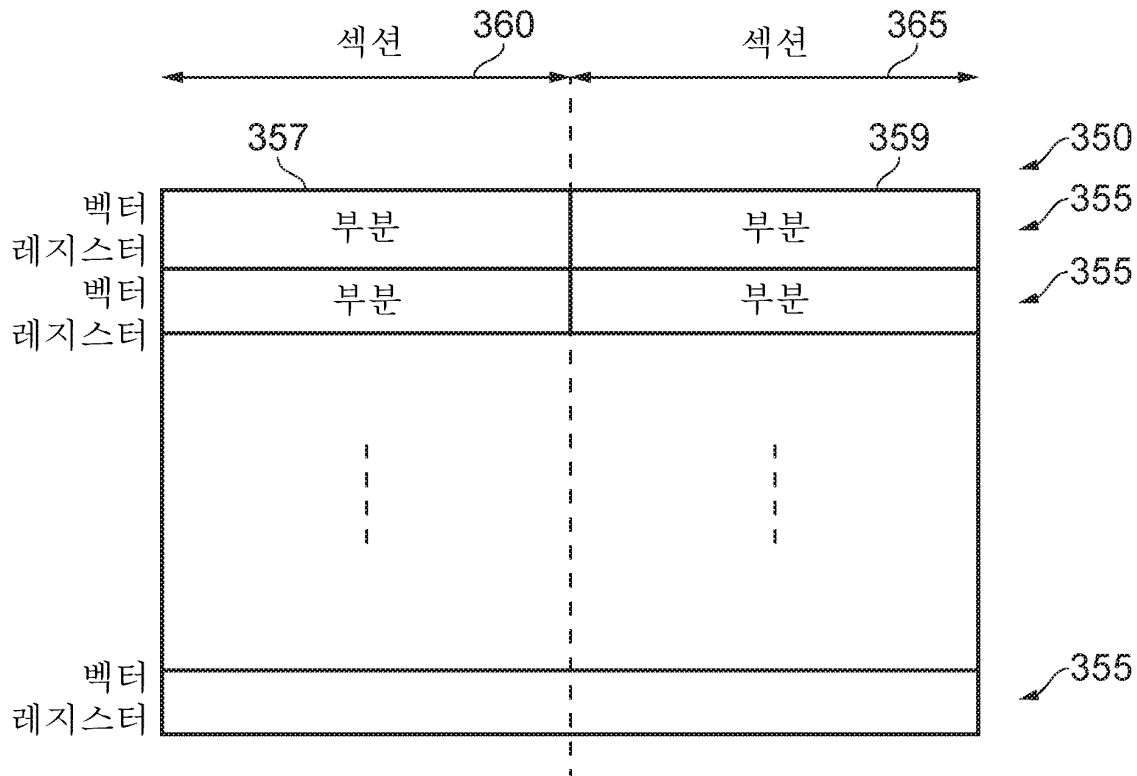


도면6b

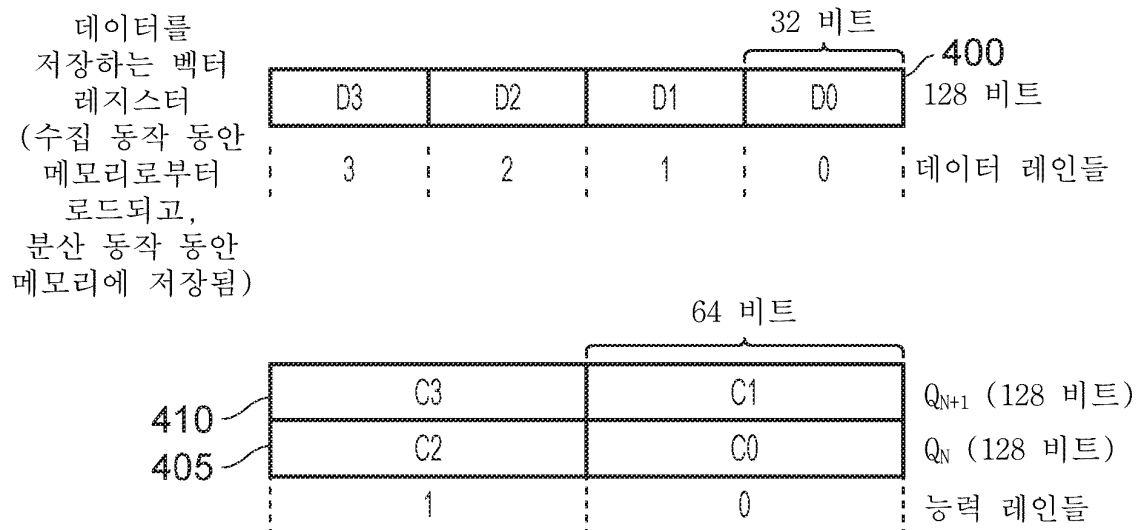


도면7

벡터 레지스터들의 세트

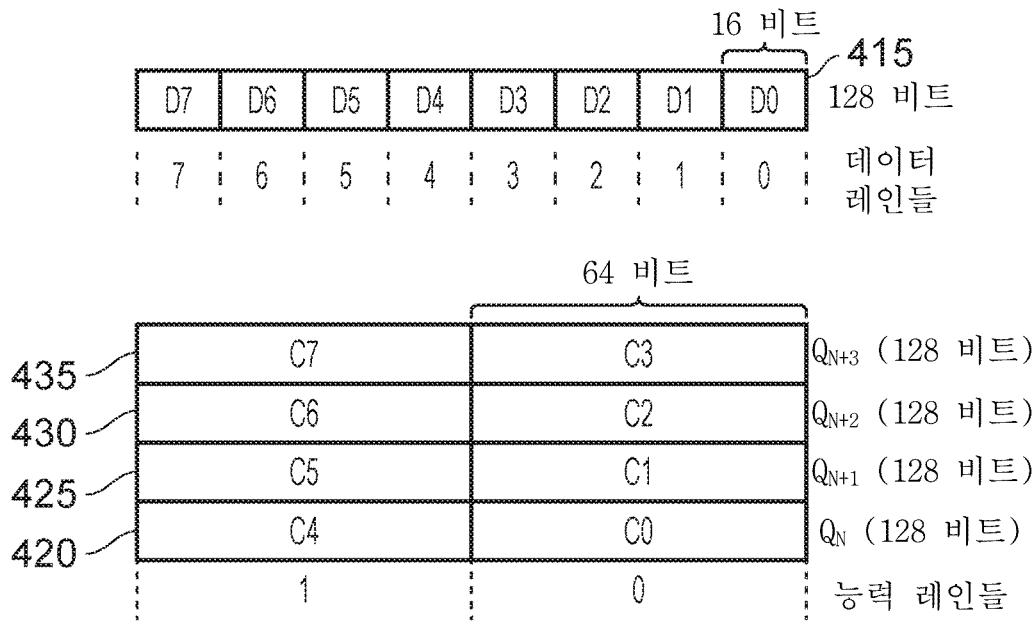


도면8a

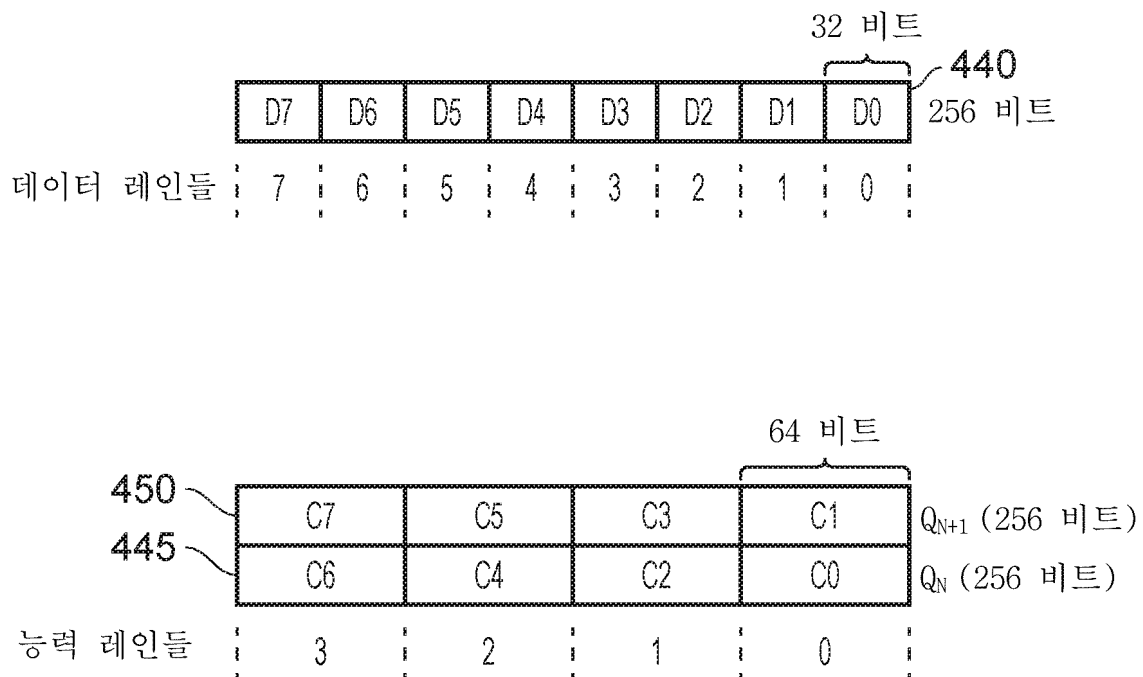


CX = 데이터 값 DX에 대한 어드레스를 결정하는 데 사용되는 능력

도면8b

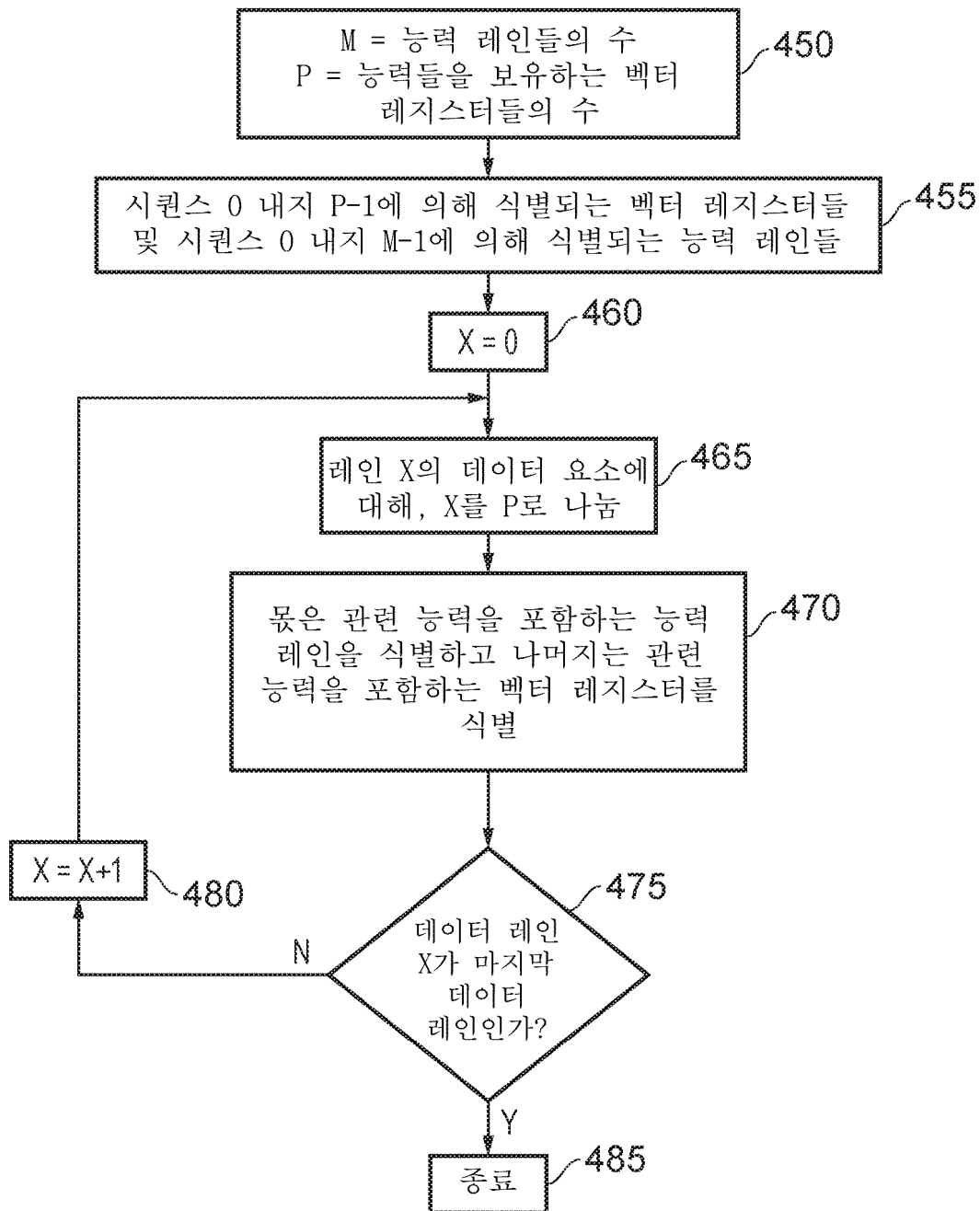


도면8c



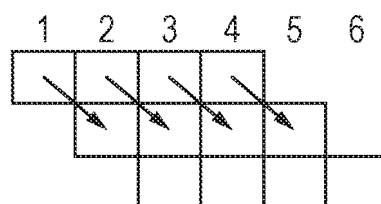
도면9

각각의 데이터 요소에
대한 관련 능력을 결정



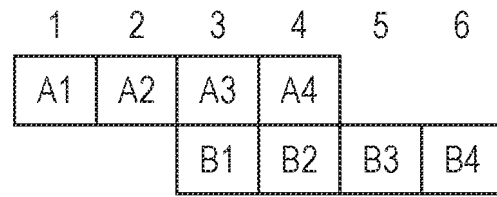
도면10

VLDR Q1, [R0], #16
VMUL Q0, Q1, Q2
VSHR Q0, Q0, #1

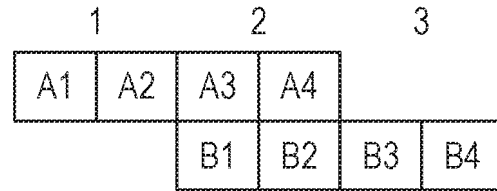


도면11

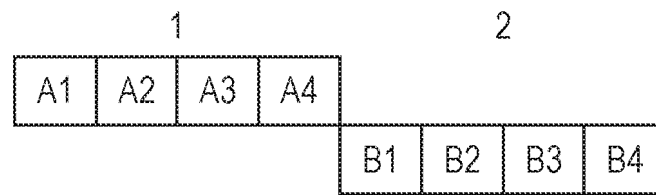
1 박동 / 틱
VLDR
VMLA



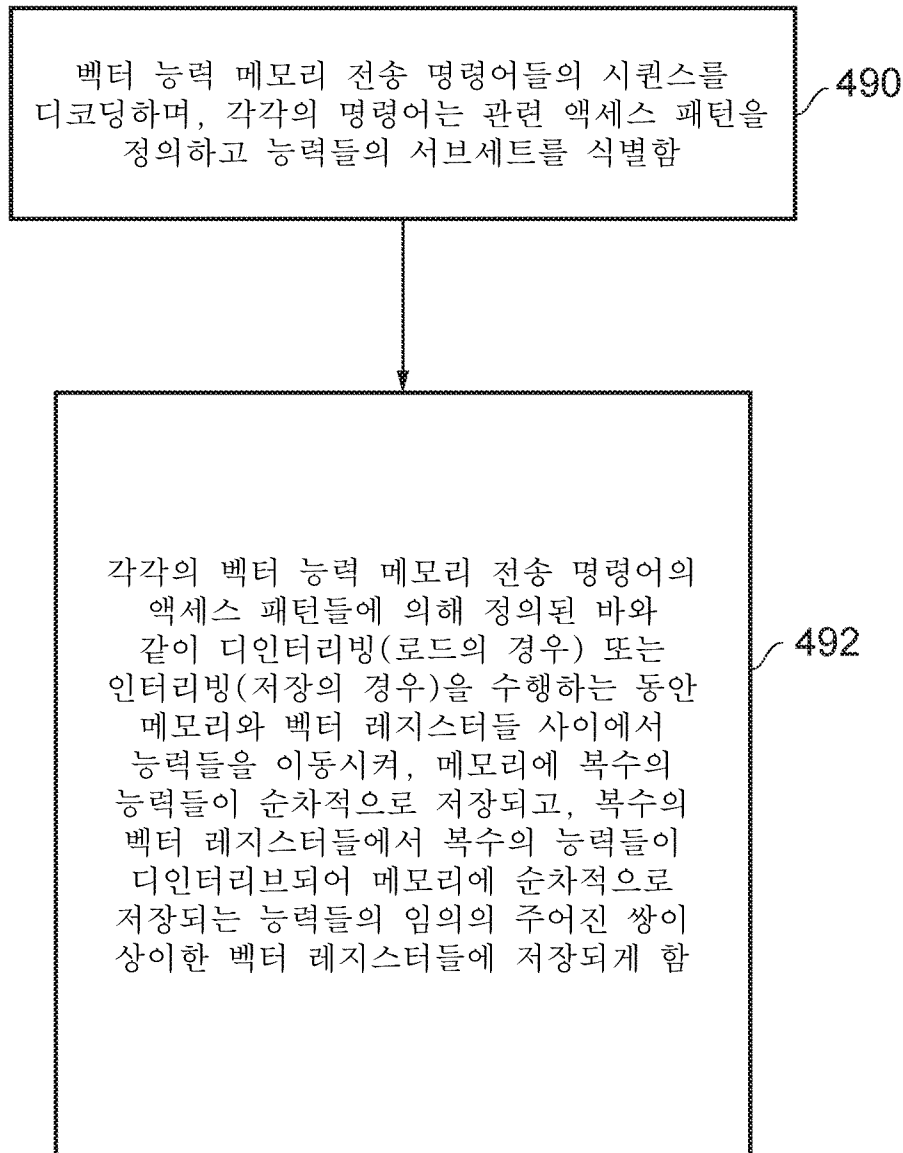
2 박동 / 틱
VLDR
VMLA



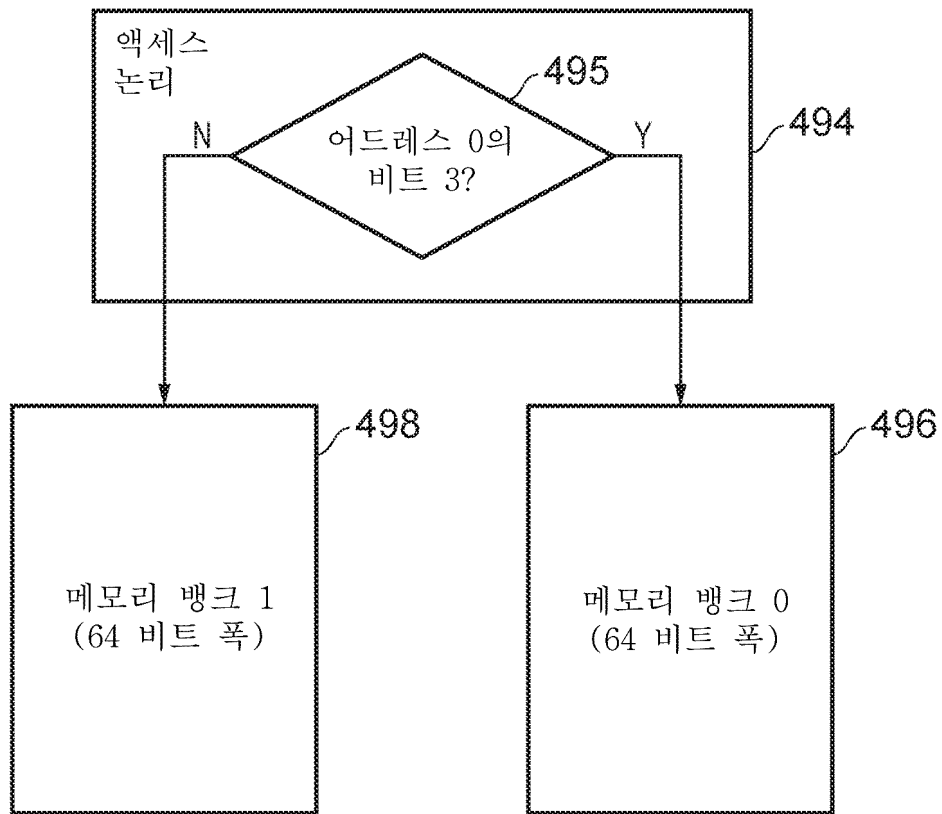
4 박동 / 틱
VLDR
VMLA



도면12



도면13



도면14

시뮬레이터 구현

