



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0002371
(43) 공개일자 2025년01월07일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2018.01)
(52) CPC특허분류
G06F 9/30036 (2023.08)
G06F 9/30032 (2013.01)
(21) 출원번호 10-2024-7037014
(22) 출원일자(국제) 2023년03월13일
심사청구일자 없음
(85) 번역문제출일자 2024년11월06일
(86) 국제출원번호 PCT/GB2023/050584
(87) 국제공개번호 WO 2023/199014
국제공개일자 2023년10월19일
(30) 우선권주장
2205491.0 2022년04월13일 영국(GB)

(71) 출원인
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리턴 폴번로드 110
(72) 발명자
그라세트, 아르노 필립 클로드
프랑스 06410 소피아 안티폴리스, 애비뉴 드 루마닐 비오 738 씨/오 에이알엠 프랑스 에스에이에스 밀라노비치, 엘레나
프랑스 06410 소피아 안티폴리스, 애비뉴 드 루마닐 비오 738 씨/오 에이알엠 프랑스 에스에이에스
(74) 대리인
이화익

전체 청구항 수 : 총 15 항

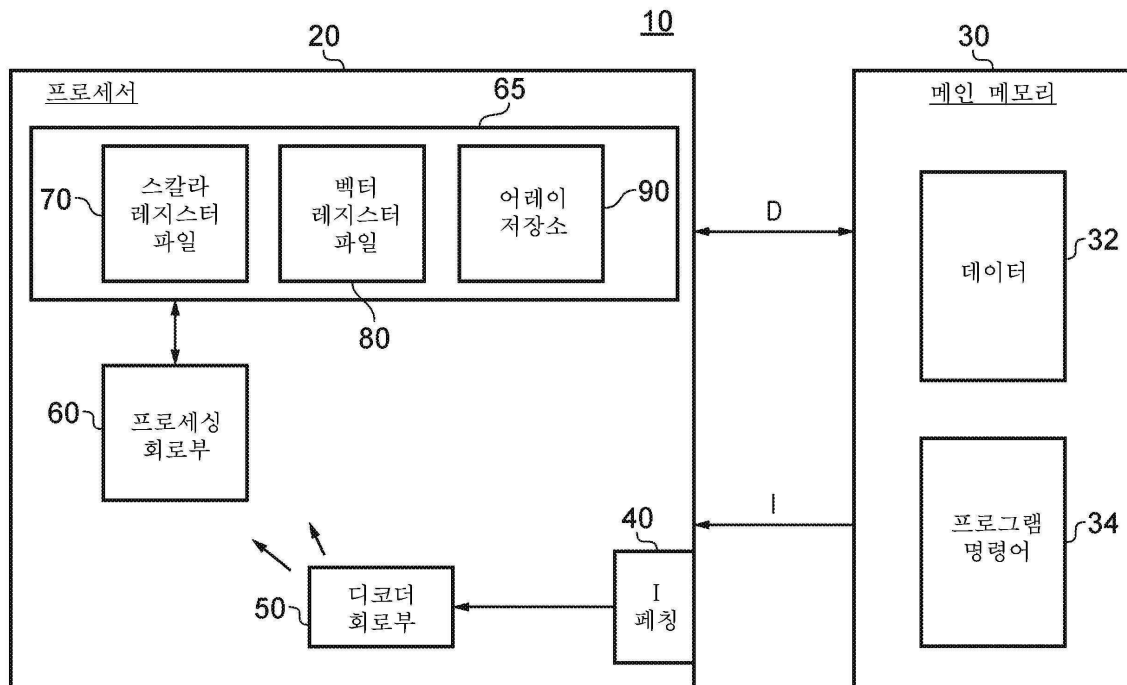
(54) 발명의 명칭 어레이 저장소에 저장되는 데이터 요소를 핸들링하기 위한 기술

(57) 요약

동작을 수행하기 위한 프로세싱 회로부, 명령어에 의해 명시되는 동작을 수행하기 위해 프로세싱 회로부를 제어하도록 명령어를 디코딩하기 위한 명령어 디코더 회로부, 및 데이터 요소를 저장하기 위한 저장 요소(storage element)를 포함하는 어레이 저장소를 포함하는 장치가 제공된다. 어레이 저장소는 동작을 수행할 때 프로세싱

(뒷면에 계속)

대표도 - 도1



회로부에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원이다. 명령어 디코더 회로부는, 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어(move and zero instruction)에 응답하여, 하나 이상의 식별된 벡터의 데이터 요소를 어레이 저장소로부터 목적지 저장소로 이동시키고, 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하기 위해 프로세싱 회로부를 제어하도록 배열된다.

(52) CPC특허분류

G06F 9/30043 (2013.01)

G06F 9/30105 (2013.01)

명세서

청구범위

청구항 1

장치로서,

동작을 수행하기 위한 프로세싱 회로부;

상기 프로세싱 회로부를 제어하여 명령어에 의해 명시되는 상기 동작을 수행하도록 상기 명령어를 디코딩하기 위한 명령어 디코더 회로부; 및

데이터 요소를 저장하기 위한 저장 요소(storage element)를 포함하는 어레이 저장소를 포함하고, 상기 어레이 저장소는 상기 동작을 수행할 때 상기 프로세싱 회로부에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원이고;

상기 명령어 디코더 회로부는, 상기 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어(move and zero instruction)에 응답하여, 상기 하나 이상의 식별된 벡터의 데이터 요소를 상기 어레이 저장소로부터 목적지 저장소로 이동시키고, 상기 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 상기 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하기 위해 상기 프로세싱 회로부를 제어하도록 배열되는, 장치.

청구항 2

제1항에 있어서,

상기 프로세싱 회로부는 누산 동작(accumulate operation)의 복수의 반복을 수행하고, 상기 누산 동작을 수행할 때 생성되는 누산 결과를 유지하기 위해 데이터 요소의 상기 주어진 2차원 어레이를 사용하도록 배열되고, 상기 누산 동작의 주어진 반복 이후에, 데이터 요소의 상기 주어진 2차원 어레이 내의 데이터 요소의 적어도 하나의 주어진 벡터가 최종 누산 결과를 저장하도록 배열되는 한편, 데이터 요소의 상기 주어진 2차원 어레이 내의 데이터 요소의 잔여 벡터는 중간 누산 결과를 저장하도록 배열되며;

상기 이동 및 제로화 명령어는 데이터 요소의 상기 적어도 하나의 주어진 벡터를 식별하도록 배열되고, 상기 프로세싱 회로부로 하여금 상기 적어도 하나의 주어진 벡터의 최종 누산 결과를 상기 어레이 저장소로부터 상기 목적지 저장소로 이동시키게 하고, 상기 적어도 하나의 주어진 벡터의 최종 누산 결과를 저장하는 데 사용된 상기 어레이 저장소의 저장 요소를 소거하여 후속 누산 동작에서 사용하기 위해 그러한 저장 요소를 확보(free)하게 하도록 상기 누산 동작의 주어진 반복 후에 실행되는, 장치.

청구항 3

제2항에 있어서,

누산 동작의 상기 복수의 반복은 입력 데이터 요소의 어레이에 대한 유한 임펄스 응답(finite impulse response; FIR) 필터링 동작을 구현하는 데 사용되는 프로세싱 및 누산 동작이고, 상기 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이는 상기 FIR 필터링 동작의 수행 동안 생성되는 출력 데이터 요소의 어레이를 유지하는 데 사용되며;

상기 프로세싱 회로부는 입력 데이터 요소의 단일 벡터를 프로세싱하고 출력 데이터 요소의 상기 어레이의 다수의 벡터 내에 누산하기 위한 출력 데이터 요소를 생성하도록 상기 누산 동작의 각각의 반복 동안 배열되는, 장치.

청구항 4

제1항 내지 제3항 중 어느 한 항에 있어서, 데이터 요소의 상기 주어진 2차원 어레이는 데이터 요소의 정사각형 2차원 어레이이고, 데이터 요소의 상기 정사각형 2차원 어레이를 형성하는 상기 복수의 벡터는 제1 어레이 방향

으로 배열되는 제1 복수의 벡터 및 상기 제1 어레이 방향에 직교하는 제2 어레이 방향으로 배열되는 제2 복수의 벡터를 포함하고, 상기 이동 및 제로화 명령어의 각각의 인스턴스는 모두 상기 제1 어레이 방향으로 연장되거나 모두 상기 제2 어레이 방향으로 연장되는 데이터 요소의 하나 이상의 벡터를 식별하도록 배열되는, 장치.

청구항 5

제4항에 있어서,

상기 프로세싱 회로부는 데이터 요소의 상기 정사각형 2차원 어레이에 대해 프로세싱 동작을 수행하도록 배열되고, 이를 수행하는 동안 상기 프로세싱 회로부는 상기 제1 어레이 방향 및 상기 제2 어레이 방향 둘 모두에서 데이터 요소의 벡터에 액세스하도록 인에이블(enable)되는, 장치.

청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서,

상기 어레이 저장소는 제1 어레이 방향으로 연장되는 복수의 어레이 벡터 레지스터를 포함하고, 상기 프로세싱 회로부는 하나 이상의 누산 동작(accumulation operation)을 수행하도록 배열되고, 여기서 각각의 누산 동작은 상기 어레이 저장소의 다수의 어레이 벡터 레지스터의 그룹 내에 누산하기 위한 출력 데이터를 생성하도록 배열되고;

데이터 요소의 상기 주어진 2차원 어레이는 다수의 어레이 벡터 레지스터의 상기 그룹 내에 저장되는 상기 데이터 요소를 포함하고;

상기 이동 및 제로화 명령어에 의해 식별되는 상기 하나 이상의 식별되는 벡터에 최종 결과 데이터가 존재하게 되는 것을 상기 하나 이상의 누산 동작의 수행이 초래했으면, 상기 이동 및 제로화 명령어는, 상기 프로세싱 회로부로 하여금 상기 하나 이상의 식별된 벡터의 데이터 요소를 상기 어레이 저장소로부터 상기 목적지 저장소로 이동시키게 하고 상기 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 다수의 어레이 벡터 레지스터의 상기 그룹 내의 각각의 어레이 벡터 레지스터를 로직 제로 값으로 설정하게 하도록 실행되는, 장치.

청구항 7

제6항에 있어서, 상기 프로세싱 회로부가 상기 하나 이상의 누산 동작의 수행을 완료한 경우, 최종 결과 데이터는 다수의 어레이 벡터 레지스터의 상기 그룹 내의 각각의 어레이 벡터 레지스터에 존재하고, 상기 이동 및 제로화 명령어의 실행은 상기 최종 결과 데이터로 하여금 다수의 어레이 벡터 레지스터의 상기 그룹으로부터 상기 목적지 저장소로 이동되게 하고, 다수의 어레이 벡터 레지스터의 상기 그룹 내의 각각의 어레이 벡터 레지스터에 대해 제로로 소거(clear)되게 하는, 장치.

청구항 8

제1항 내지 제7항 중 어느 한 항에 있어서,

복수의 벡터 레지스터를 포함하는 벡터 레지스터 파일을 추가로 포함하고;

상기 이동 및 제로화 명령어는 상기 벡터 레지스터 파일 내의 하나 이상의 벡터 레지스터를 상기 목적지 저장소로 표시하도록 배열되는, 장치.

청구항 9

제1항 내지 제7항 중 어느 한 항에 있어서, 상기 이동 및 제로화 명령어는, 상기 하나 이상의 식별된 벡터의 데이터 요소가 저장될 메모리 내의 하나 이상의 위치를 상기 목적지 저장소로 표시하도록 배열되는, 장치.

청구항 10

제1항 내지 제9항 중 어느 한 항에 있어서, 상기 이동 및 제로화 명령어는 상기 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 데 사용되는 벡터 식별 필드를 포함하는, 장치.

청구항 11

제4항에 종속되는 경우 제10항에 있어서, 상기 벡터 식별 필드는 상기 정사각형 2차원 어레이를 식별하기 위한 제1 서브필드, 및 상기 하나 이상의 벡터를 식별하는 데 사용되는 하나 이상의 라인 식별자 및 어레이 방향 표시를 제공하는 제2 서브필드를 포함하는, 장치.

청구항 12

제1항 내지 제11항 중 어느 한 항에 있어서, 상기 이동 및 제로화 명령어는, 상기 하나 이상의 식별된 벡터의 어느 데이터 요소가 상기 어레이 저장소로부터 상기 목적지 저장소로 이동되어야 하는지를 식별하는 데 사용되는 술어 정보(predicate information)를 식별하고, 이들의 연관된 저장 요소가 상기 로직 제로 값으로 설정되기 위한 술어 필드를 포함하는, 장치.

청구항 13

제12항에 있어서, 상기 이동 및 제로화 명령어는 상기 하나 이상의 식별된 벡터 내의 각각의 데이터 요소의 크기를 식별하기 위한 크기 필드를 추가로 포함하는, 장치.

청구항 14

장치의 어레이 저장소 내의 데이터 요소를 핸들링(handling)하는 방법으로서,

동작을 수행하기 위해 프로세싱 회로부를 사용하는 단계;

상기 프로세싱 회로부를 제어하여 명령어에 의해 명시되는 상기 동작을 수행하도록 상기 명령어를 디코딩하기 위해 명령어 디코더 회로부를 사용하는 단계;

상기 어레이 저장소 내에, 데이터 요소를 저장하기 위한 저장 요소를 제공하는 단계 - 상기 어레이 저장소는 상기 동작을 수행할 때 상기 프로세싱 회로부에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원임 -; 및

상기 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어에 응답하여, 상기 하나 이상의 식별된 벡터의 데이터 요소를 상기 어레이 저장소로부터 목적지 저장소로 이동시키고, 상기 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 상기 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하기 위해 상기 프로세싱 회로부를 제어하도록 상기 명령어 디코더 회로부를 사용하는 단계를 포함하는, 방법.

청구항 15

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

동작을 수행하기 위한 프로세싱 프로그램 로직;

상기 프로세싱 프로그램 로직을 제어하여 명령어에 의해 명시되는 상기 동작을 수행하도록 상기 명령어를 디코딩하기 위한 명령어 디코딩 프로그램 로직(instruction decode program logic); 및

데이터 요소를 저장하기 위한 저장 요소를 포함하는 어레이 저장소를 에뮬레이팅하기 위한 어레이 저장소 에뮬레이팅 프로그램 로직을 포함하고, 상기 어레이 저장소는 상기 동작을 수행할 때 상기 프로세싱 프로그램 로직에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원이고;

상기 명령어 디코딩 프로그램 로직은, 상기 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어에 응답하여, 상기 하나 이상의 식별된 벡터의 데이터 요소를 상기 어레이 저장소로부터 목적지 저장소로 이동시키고, 상기 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 상기 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하기 위해 상기 프로세싱 프로그램 로직을 제어하도록 배열되는, 컴퓨터 프로그램.

발명의 설명

배경 기술

[0001] 본 기술은 데이터 프로세싱 분야에 관한 것이며, 보다 상세하게는 어레이 저장소에 저장되는 데이터 요소의 핸들링(handling)에 관한 것이다.

[0002] 일부 현대의 데이터 프로세싱 시스템은 데이터 프로세싱 동작을 수행할 때 데이터 프로세싱 시스템의 프로세싱 회로부에 의해 액세스될 수 있는 데이터 요소의 하나 이상의 2차원 어레이를 저장하기 위한 어레이 저장소를 제공할 수 있다. 이는 다수의 상이한 유형의 동작, 예를 들어, 누산 기능(accumulate function)을 포함하는 동작을 수행하기 위한 효율적인 메커니즘을 제공할 수 있으며, 여기서 누산 출력은 데이터 요소의 2차원 어레이 내에서 유지될 수 있다.

[0003] 그러나, 이러한 어레이 저장소의 사용으로부터 실현될 수 있는 효율 이득을 최상으로 사용하기 위해, 후속 동작과 관련하여 사용하기 위해 어레이 저장소의 자원을 확보(free up)하는 방식으로 어레이 저장소 밖으로 데이터 요소를 전송하기 위한 효율적인 메커니즘을 제공하는 것이 유익할 것이다.

발명의 내용

[0004] 하나의 예시적인 배열에서, 동작을 수행하기 위한 프로세싱 회로부; 프로세싱 회로부를 제어하여 명령어에 의해 명시되는 동작을 수행하도록 명령어를 디코딩하기 위한 명령어 디코더 회로부; 및 데이터 요소를 저장하기 위한 저장 요소(storage element)를 포함하는 어레이 저장소를 포함하는 장치가 제공되며, 어레이 저장소는 동작을 수행할 때 프로세싱 회로부에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원이고; 명령어 디코더 회로부는, 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어(move and zero instruction)에 응답하여, 하나 이상의 식별된 벡터의 데이터 요소를 어레이 저장소로부터 목적지 저장소로 이동시키고, 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하기 위해 프로세싱 회로부를 제어하도록 배열된다.

[0005] 추가의 예시적인 배열에서, 장치의 어레이 저장소 내의 데이터 요소를 핸들링하는 방법이 제공되며, 이 방법은 동작을 수행하기 위해 프로세싱 회로부를 사용하는 단계; 프로세싱 회로부를 제어하여 명령어에 의해 명시되는 동작을 수행하도록 명령어를 디코딩하기 위해 명령어 디코더 회로부를 사용하는 단계; 어레이 저장소 내에, 데이터 요소를 저장하기 위한 저장 요소를 제공하는 단계 - 어레이 저장소는 동작을 수행할 때 프로세싱 회로부에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원임 -; 및 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어에 응답하여, 하나 이상의 식별된 벡터의 데이터 요소를 어레이 저장소로부터 목적지 저장소로 이동시키고, 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하기 위해 프로세싱 회로부를 제어하도록 명령어 디코더 회로부를 사용하는 단계를 포함한다.

[0006] 또 다른 추가적인 예시적인 배열에서, 명령어 실행 환경을 제공하도록 호스트 데이터 프로세싱 장치를 제어하기 위한 컴퓨터 프로그램이 제공되며, 컴퓨터 프로그램은 동작을 수행하기 위한 프로세싱 프로그램 로직; 프로세싱 프로그램 로직을 제어하여 명령어에 의해 명시되는 동작을 수행하도록 명령어를 디코딩하기 위한 명령어 디코딩 프로그램 로직(instruction decode program logic); 및 데이터 요소를 저장하기 위한 저장 요소를 포함하는 어레이 저장소를 에뮬레이팅하기 위한 어레이 저장소 에뮬레이팅 프로그램 로직을 포함하고, 어레이 저장소는 동작을 수행할 때 프로세싱 프로그램 로직에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원이고; 명령어 디코딩 프로그램 로직은, 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어에 응답하여, 하나 이상의 식별된 벡터의 데이터 요소를 어레이 저장소로부터 목적지 저장소로 이동시키고, 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하기 위해 프로세싱 프로그램 로직을 제어하도록 배열된다.

도면의 간단한 설명

[0007] 본 기술은 첨부 도면에 예시된 바와 같은 그의 예를 참조하여, 단지 예시로서, 추가로 설명될 것이다.

도 1은 하나의 예시적인 구현에 따른 장치의 블록도이다;

도 2는 벡터 피연산자를 저장하기 위한 벡터 레지스터 및 데이터 요소의 2D 어레이를 저장하기 위한 어레이 레지스터를 포함하는, 장치 내에 제공될 수 있는 아키텍처 레지스터의 예를 도시하며, 어레이 레지스터의 물리적 구현의 예를 포함한다;

도 3a 및 도 3b는 데이터 요소의 주어진 2D 어레이가 비-정사각형 또는 정사각형일 수 있는 예를 예시한다;

도 4a 및 도 4b는 일 예시적인 구현에 따라 어레이 저장소 내의 정사각형 2D 어레이에 대해 액세스가 어떻게 수행될 수 있는지를 개략적으로 예시한다;

도 5a는 하나의 예시적인 구현에 따른 이동 및 제로화 명령어 내에서 제공될 수 있는 필드를 개략적으로 예시하고, 도 5b는 이동 및 제로화 명령어의 벡터 식별 필드를 구현하기 위해 하나의 특정 예시적인 구현에서 사용될 수 있는 서브필드를 개략적으로 예시한다;

도 6은 하나의 예시적인 구현에 따른, 이동 및 제로화 명령어가 어떻게 핸들링될 수 있는지를 예시하는 흐름도이다;

도 7은 어레이 저장소 내에 제공되는 데이터 요소에 대해 동작할 수 있는 하나의 예시적인 명령어 시퀀스를 예시하며, 여기서 명령어 시퀀스는 본원에 설명된 이동 및 제로화 명령어의 다수의 인스턴스를 포함한다;

도 8은 수행될 수 있는 유한 임펄스 응답(finite impulse response; FIR) 필터링 동작을 개략적으로 예시한다;

도 9a 내지 도 9d는 하나의 예시적인 구현에 따라 2D 이미지 필터링 동작을 수행할 때 어레이 저장소가 어떻게 사용될 수 있는지를 예시한다;

도 10은 어레이 저장소 내에 제공되는 데이터 요소에 대해 동작할 수 있는 대안적인 예시적인 명령어 시퀀스를 예시하며, 여기서 명령어 시퀀스는 본원에 설명된 이동 및 제로화 명령어의 다수의 인스턴스를 포함한다;

도 11은 하나의 예시적인 구현에 따라, 벡터 제로화 명령어(zero vectors instruction)가 어떻게 핸들링될 수 있는지를 예시하는 흐름도이다;

도 12는 일 예시적인 구현에 따른 벡터 제로화 명령어 내에서 제공될 수 있는 필드를 개략적으로 예시한다;

도 13은 대안적인 예시적인 구현에 따라 벡터 제로화 명령어가 어떻게 핸들링될 수 있는지를 예시하는 흐름도이다; 그리고

도 14는 사용될 수 있는 시뮬레이터 구현을 예시한다.

발명을 실시하기 위한 구체적인 내용

[0008] 일 예시적인 배열에서, 동작을 수행하기 위한 프로세싱 회로부, 및 명령어에 의해 명시되는 동작을 수행하도록 프로세싱 회로부를 제어하기 위해 명령어를 디코딩하기 위한 명령어 디코더 회로부를 갖는 장치가 제공된다. 데이터 요소를 저장하기 위한 저장 요소를 포함하는 어레이 저장소가 또한 제공된다. 어레이 저장소는 동작을 수행할 때 프로세싱 회로부에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원이다.

[0009] 앞서 언급된 바와 같이, 어레이 저장소의 사용은 특정 유형의 동작, 예를 들어, 누산 동작을 수행하기 위한 매우 효율적인 메커니즘을 제공할 수 있다. 누산 동작은 오로지 누산 기능을 수행할 수 있지만, 대안적으로 누산 기능 외에 추가적인 프로세싱을 통합할 수 있다(순전히 예로서 누산 동작은 형태 $A = A + B * C$ 의 곱셈 누산 기능을 수행할 수 있음). 실현될 수 있는 잠재적인 효율성 이점을 최대한 활용하기 위해, 데이터 요소가 어레이 저장소를 사용하여 수행되는 계산의 대상이 더 이상 아닐 때, 데이터 요소를 어레이 저장소 밖으로 이동시키기 위한 효율적인 메커니즘, 및 또한, 저장 요소가 후속 계산과 연관되어 사용하기 위해 이용 가능하도록 어레이 저장소의 연관된 저장 요소를 확보(free up)하기 위한 효율적인 메커니즘을 갖는 것이 필요할 수 있다.

[0010] 일 예시적인 구현에 따르면, 그러한 프로세스의 효율에서 유의한 개선을 가능하게 할 수 있는 이동 및 제로화 명령어가 제공된다. 특히, 하나의 예시적인 구현에서, 명령어 디코더 회로부는, 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어에 응답하여, 하나 이상의 식별된 벡터의 데이터 요소를 어레이 저장소로부터 목적지 저장소로 이동시키고, 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 어레이 저장소의 저장 요소를 로직 제로 값으로 설정

하기 위해 프로세싱 회로부를 제어하도록 배열될 수 있다.

- [0011] 따라서, 상기 기술에 따르면, 단일 명령어는, 실행될 때, 주어진 2차원 어레이 내의 데이터 요소의 하나 이상의 식별된 벡터 내의 데이터 요소 둘 모두가 어레이 저장소 밖으로 이동되게 하고, 또한, 그러한 데이터 요소를 저장하고 있던 어레이 저장소의 연관된 저장 요소가 로직 제로 값으로 소거되게 하고, 따라서 후속 계산에 사용하기 위해 이들을 준비하게 한다는 것이 명시될 수 있다.
- [0012] 이는 성능을 유의하게 개선할 수 있다. 특히, 데이터 요소를 어레이 저장소 밖으로 이동시키는 동작(act) 및 연관된 저장 요소를 재사용하기 위해 준비하는 동작은 그 자체로는 유용한 계산을 수행하지 않으며, 따라서 어레이 저장소의 사용과 연관된 오버헤드로서 여겨질 수 있다. 단일 명령어로 데이터 요소 둘 모두가 이동되게 하고, 연관된 저장 요소가 로직 제로 값으로 소거되도록 허용함으로써, 이는 해당 오버헤드를 유의하게 감소시킬 수 있다.
- [0013] 특히, 이전에 공지된 기술은 요구되는 데이터 요소를 어레이 저장소로부터 명시된 목적지 저장소로 이동시키기 위해 실행될 적어도 하나의 이동 명령어를 요구했을 것이고, 그 후 하나 이상의 추가적인 이동 명령어가 로직 제로 값의 하나 이상의 벡터를 하나 이상의 소스 벡터 레지스터로부터 어레이 저장소의 관련 저장 요소로 이동시키기 위해 요구되었을 것이다. 따라서, 이것은 차례로 수행되어야 하는 종속 명령어의 시퀀스를 생성한다. 예를 들어, 제1 이동 명령어가 데이터 요소의 하나의 벡터를 어레이 저장소 밖으로 이동시키기 위해 사용되고, 이어서 제2 이동 명령어가 로직 제로 값의 벡터를 명시된 소스 벡터 레지스터로부터 어레이 저장소의 관련 저장 요소로 이동시키기 위해 사용되는 단순한 경우를 단지 고려하면, 잇따라 수행될 필요가 있는 두 개의 종속적인(dependent) 이동 명령어가 분명히 존재하며, 이러한 명령어 종속성은 새로운 이동 및 제로화 명령어의 사용을 통해 제거된다.
- [0014] 또한, 조합된 이동 및 제로화(zeroing) 동작을 수행하는 것과 연관된 하드웨어 비용은 일부 구현에서, 데이터 요소의 벡터를 어레이 저장소 밖으로 이동시키기 위해 단지 표준적 이동 동작(standard move operation)을 수행하는 것과 연관된 하드웨어 비용과 동일할 수 있고, 따라서, 관련 저장 요소의 제로화는 추가적인 하드웨어 오버헤드가 없이 효과적으로 획득될 수 있다는 것이 발견되었다. 또한, 조합된 이동 및 제로화 동작의 성능(실행 속도)은 하나의 예시적인 구현에서 단지 단일 이동 동작을 수행하는 것과 동일한 것으로 밝혀졌다.
- [0015] 더욱이, 본 기술의 사용은 평소라면 그러한 로직 제로 값을 어레이 저장소로 이동시키는 데 사용되는 이동 명령어에 대한 소스 피연산자로서 사용되는 데 요구될, 벡터 레지스터 내에 로직 제로 값을 저장할 필요성을 회피하고, 따라서 벡터 레지스터 파일 내에서 하나 이상의 벡터 레지스터를 확보한다.
- [0016] 또한, 제로화 기능성을 수행하기 위해 별도의 이동 명령어가 필요 없으므로, 성능이 개선됨을 알 수 있다.
- [0017] 이러한 접근법은 어레이 저장소의 많은 예시적인 사용 사례에서 매우 유익한 것으로 밝혀졌다. 예를 들어, 어레이 저장소는 종종 누산 동작의 다수의 반복을 수행할 때 생성되는 결과를 누산하는 데 사용되고, 최종 누산 결과가 생성될 때, 이들은 전형적으로 예를 들어, 장치 내에 제공되는 하나 이상의 벡터 레지스터로 이동됨으로써 어레이 저장소 밖으로 이동된다. 어레이 저장소가 누산 동작의 수행에 사용되고 있을 때, 그러한 저장 요소가 로직 제로 값으로 처음 설정되는 경우에 새로운 일련의 누산 동작에 대해 그러한 최종 누산 결과를 저장하고 있던 저장 요소를 재사용하는 것만이 가능하며, 본원에 설명된 바와 같은 이동 및 제로화 명령어의 사용은 이것이 효율적으로 달성될 수 있게 할 수 있다.
- [0018] 따라서, 일 예시적인 구현에서, 프로세싱 회로부는 누산 동작의 복수의 반복을 수행하고, 누산 동작을 수행할 때 생성되는 누산 결과를 유지하기 위해 데이터 요소의 주어진 2차원 어레이를 사용하도록 배열될 수 있으며, 누산 동작의 주어진 반복 이후에, 데이터 요소의 주어진 2차원 어레이 내의 데이터 요소의 적어도 하나의 주어진 벡터가 최종 누산 결과를 저장하도록 배열되는 한편, 데이터 요소의 주어진 2차원 어레이 내의 데이터 요소의 잔여 벡터는 중간 누산 결과를 저장하도록 배열된다. 이러한 구현에서, 이동 및 제로화 명령어는 데이터 요소의 적어도 하나의 주어진 벡터를 식별하도록 배열될 수 있고, 누산 동작의 주어진 반복 후에 실행되어, 프로세싱 회로부로 하여금, 적어도 하나의 주어진 벡터의 최종 누산 결과를 어레이 저장소로부터 목적지 저장소로 이동시키게 하고, 후속 누산 동작에서 사용하기 위해 그러한 저장 요소를 확보하기 위해 적어도 하나의 주어진 벡터의 최종 누산 결과를 저장하는 데 사용된 어레이 저장소의 저장 요소를 소거하게 할 수 있다.
- [0019] 상기에 지칭된 누산 동작은 단지 (예를 들어, 형태 $A = A+B$ 의) 누산 기능을 수행할 수 있지만, 더 전형적으로 누산 기능에 더하여 일부 추가적인 프로세싱 동작을 수반할 수도 있다는 것에 유의해야 한다. 따라서, 누산 동작은 프로세싱 동작 결과 값을 생성하기 위해 수행되는 프로세싱 동작을 포함할 수 있고, 그 후 프로세싱 동작

결과 값은 어레이 저장소의 연관된 저장 요소 내의 기존 데이터 요소 값과 누산되어 어레이 저장소의 연관된 저장 요소 내에 저장될 새로운 데이터 요소 값을 생성한다. 순전히 예로서, 상기에 지칭된 누산 동작은 (예를 들어, 형태 $A = A + B * C$ 의) 곱셈 누산 동작일 수 있다.

[0020] 위에서 언급된 누산 기능성을 사용할 수 있고, 어레이 저장소의 사용이 효율적인 구현 기술(implementation technique)을 제공할 수 있는 다양한 유형의 데이터 프로세싱 동작이 존재한다. 하나의 특정 예시적인 사용 사례에서, 누산 동작의 복수의 반복은 입력 데이터 요소의 어레이에 대한 유한 임펄스 응답(finite impulse response; FIR) 필터링 동작을 구현하는 데 사용되는 프로세싱 및 누산 동작이고, 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이는 FIR 필터링 동작의 수행 동안 생성되는 출력 데이터 요소의 어레이를 유지하는 데 사용될 수 있다. 프로세싱 회로부는 누산 동작의 각각의 반복 동안 입력 데이터 요소의 단일 벡터를 프로세싱하고 출력 데이터 요소의 어레이의 다수의 벡터 내에 누산하기 위한 출력 데이터 요소를 생성하도록 배열될 수 있다.

[0021] 입력 데이터 요소 및 출력 데이터 요소 사이의 대응은 구현에 따라 변할 수 있다. 예를 들어, 입력 데이터 요소의 하나의 벡터는 출력 데이터 요소의 다수의 벡터와 연관될 수 있다. 또한, 출력 데이터 요소의 그러한 다수의 벡터는 어레이 저장소 내의 수평 방향 및 수직 방향 중 어느 하나 또는 양쪽으로 배열될 수 있다(벡터가 수평 방향 및 수직 방향 둘 모두에서 액세스될 수 있는 구현을 지원하기 위해, 데이터 요소의 2차원 어레이는 전형적으로 데이터 요소의 2차원 정사각형 어레이일 것임). 또한, 입력 데이터 요소와 출력 데이터 요소의 크기는 상이할 수 있다.

[0022] 전술된 바와 같은 이러한 기술은 데이터 요소의 정사각형 어레이를 사용하여, 슬라이딩 윈도우 기술에 의해 구현되는 FIR 필터링을 계산하기 위해 외적 접근법(outer product approach)을 이용할 수 있다. 이러한 기술은 전형적으로 출력 데이터 요소의 정사각형 어레이의 벡터 중 일부가 출력 데이터 요소의 다른 벡터 이전에 최종화(finalise)되는 결과를 초래하고, 따라서 전술된 이동 및 제로화 명령어의 사용은 출력 데이터 요소의 그러한 최종화된 벡터가 어레이 저장소 밖으로 이동될 수 있게 하면서, 연관된 저장 요소는 출력 데이터 요소의 다른 벡터와 연관하여 사용하기 위해 확보될 수 있다.

[0023] 입력 데이터 요소의 어레이는 다양한 형태를 취할 수 있지만, 하나의 예시적인 구현에서 픽셀 값의 어레이를 나타낼 수 있다. 그러나, 본원에 설명된 기술은 예를 들어, 이미지 데이터를 나타내지 않을 수 있는 데이터 요소의 다른 어레이에 동등하게 적용 가능하다.

[0024] 일부 예시적인 구현에서, 입력 데이터 요소의 행을 프로세싱함으로써 생성되는 출력 데이터 요소의 다수의 벡터는 출력 데이터 요소의 다수의 "행"인 것으로 지칭될 수 있다. 그러나, 앞서 언급된 바와 같이, 출력 데이터 요소의 데이터 요소의 주어진 정사각형 2D 어레이(이러한 정사각형 2D 어레이는 본원에서 정사각형 서브어레이(sub-array)로도 지칭될 수 있음) 내에 누산되는 출력 데이터 요소의 행이 어레이 저장소 내의 임의의 원하는 배향(orientation)에 저장될 수 있다는 것에 유의해야 한다. 예를 들어, 행은 정사각형 서브어레이 내의 수평 벡터 또는 정사각형 서브어레이 내의 수직 벡터로서 저장될 수 있고, 따라서 용어 "행"은 어레이 저장소 내의 데이터 요소의 임의의 특정 배향을 암시하는 것으로 본원에서 취해져서는 안 된다.

[0025] 하나의 예시적인 배열에서, 데이터 요소의 주어진 2차원 어레이는 데이터 요소의 정사각형 2차원 어레이이고, 데이터 요소의 정사각형 2차원 어레이를 형성하는 복수의 벡터는 제1 어레이 방향으로 배열되는 제1 복수의 벡터 및 제1 어레이 방향에 직교하는 제2 어레이 방향으로 배열되는 제2 복수의 벡터를 포함하고, 이동 및 제로화 명령어의 각각의 인스턴스는 모두 제1 어레이 방향으로 연장되거나 모두 제2 어레이 방향으로 연장되는 데이터 요소의 하나 이상의 벡터를 식별하도록 배열된다. 따라서, 이는 어레이 저장소 밖으로 이동될 데이터 요소의 다양한 벡터가 어떻게 식별되는지에 있어서 큰 유연성의 정도를 제공한다.

[0026] 어레이 저장소 내에 저장되는 데이터 요소의 하나 이상의 2차원 어레이는 구현에 따라 다양한 형태를 취할 수 있다. 하나의 특정 예시적인 구현에서, 프로세싱 회로부는 데이터 요소의 정사각형 2차원 어레이에 대해 프로세싱 동작을 수행하도록 배열되고, 수행하는 동안 프로세싱 회로부는 제1 어레이 방향 및 제2 어레이 방향 둘 모두에서 데이터 요소의 벡터에 액세스하도록 인에이블된다.

[0027] 일 예시적인 구현에서, 어레이 저장소는 제1 어레이 방향으로 연장되는 복수의 어레이 벡터 레지스터를 포함하도록 구성될 수 있다. 프로세싱 회로부는 하나 이상의 누산 동작(accumulation operation)을 수행하도록 배열될 수 있으며, 여기서 각각의 누산 동작은 어레이 저장소의 다수의 어레이 벡터 레지스터의 그룹 내에 누산하기 위한 출력 데이터를 생성하도록 배열된다. 따라서, 이러한 구현에서, 어레이 저장소는 단일 방향 내에서 연장

되는 다수의 별개로 어드레싱가능한 어레이 벡터 레지스터를 포함하는 것으로 간주되고, 데이터 요소의 앞서 언급된 주어진 2차원 어레이는 위에서 지칭된 다수의 어레이 벡터 레지스터의 그룹 내에 저장되는 데이터 요소를 포함하는 것으로 간주될 수 있다.

[0028] 이러한 예시적인 구현에서, 이동 및 제로화 명령어에 의해 식별되는 하나 이상의 식별되는 벡터에 최종 결과 데이터가 존재하게 되는 것을 하나 이상의 누산 동작의 수행이 초래했으면, 이동 및 제로화 명령어는, 프로세싱 회로부로 하여금 하나 이상의 식별된 벡터의 데이터 요소를 어레이 저장소로부터 목적지 저장소로 이동시키게 하고 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용된 다수의 어레이 벡터 레지스터의 그룹 내의 각각의 어레이 벡터 레지스터를 로직 제로 값으로 설정하게 하도록 실행될 수 있다.

[0029] 하나의 특정 예시적인 구현에서, 프로세싱 회로부가 하나 이상의 누산 동작의 수행을 완료한 경우, 최종 결과 데이터는 다수의 어레이 벡터 레지스터의 그룹 내의 각각의 어레이 벡터 레지스터에 존재한다. 이어서 이동 및 제로화 명령어의 실행은 최종 결과 데이터가 다수의 어레이 벡터 레지스터의 그룹으로부터 목적지 저장소로 이동되게 하고, 다수의 어레이 벡터 레지스터의 그룹 내의 각각의 어레이 벡터 레지스터에 대해 제로로 소거되게 할 수 있다. 그 후, 이것은 프로세싱 회로부가 임의의 원하는 후속 프로세싱 동작을 위해 어레이 벡터 레지스터의 그룹으로부터 하나 이상의 어레이 벡터 레지스터를 재사용할 수 있게 한다(따라서, 예를 들어, 그러한 어레이 벡터 레지스터 중 임의의 것 또는 전부를 사용하는 후속 누산 명령어의 실행은 어레이 벡터 레지스터의 내용물(content)이 제로로 소거되는 것으로 인해 비-누산 변형(variant)이 수행되게 할 것임).

[0030] 이동 및 제로화 명령어에 대해 명시되는 목적지 저장소는 다양한 형식을 취할 수 있다. 일 예시적인 구현에서, 장치는 복수의 벡터 레지스터를 포함하는 벡터 레지스터 파일을 추가로 제공할 수 있고, 이동 및 제로화 명령어는 벡터 레지스터 파일 내의 하나 이상의 벡터 레지스터를 목적지 저장소로서 표시하도록 배열될 수 있다. 하나 이상의 벡터 레지스터를 식별하기 위해 이동 및 제로화 명령어가 배열될 수 있는 다양한 방식이 있다. 예를 들어, 데이터 요소의 단일 벡터가 어레이 저장소 밖으로 이동되어야 하는 단일 벡터 레지스터 경우에서, 이동 및 제로화 명령어는 해당 단일 벡터 레지스터를 결정하는 데 사용되는 식별자를 제공할 수 있다. 데이터 요소의 다수의 벡터가 어레이 저장소 밖으로 이동되어야 하는 다수의 벡터 레지스터 경우에 대해, 그러한 다수의 벡터 레지스터는 벡터 레지스터 각각에 대해 별개의 식별자 정보를 사용하여 명시적으로 식별될 수 있거나, 대안적으로 하나의 벡터 레지스터는 명령어에 의해 식별될 수 있으며, 다수의 벡터 레지스터 내의 다른 벡터 레지스터는 암시적이다. 예를 들어, 다수의 벡터 레지스터는 명시적으로 식별된 벡터 레지스터로 시작하는 인접한 벡터 레지스터의 시퀀스일 수 있거나, 다수의 벡터 레지스터는 각각 일정한 스트라이드 값(stride value)에 의해 분리될 수 있다.

[0031] 그러나, 대안적인 구현에서, 원하는 경우, 이동 및 제로화 명령어는 하나 이상의 식별된 벡터의 데이터 요소가 저장될 메모리 내의 하나 이상의 위치를 목적지 저장소로서 표시하도록 배열될 수 있다. 이러한 경우에, 이동 및 제로화 명령어는 대신에 저장 및 제로화 명령어로 지칭될 수 있다.

[0032] 데이터 요소의 하나 이상의 벡터가 이동될 메모리 위치를 식별하기 위해 이동 및 제로화 명령어가 배열될 수 있는 다양한 방식이 있다. 예를 들어, 데이터 요소의 단일 벡터만이 메모리로 이동될 단일 벡터 경우를 고려하여, 이동 및 제로화 명령어는 메모리 내의 위치를 식별하도록 배열될 수 있으며, 그 다음, 데이터 요소의 벡터는 해당 위치에 의해 식별되는 연속적인 메모리 어드레스(이 경우, 위치는 예를 들어, 제1 데이터 요소의 메모리 어드레스일 수 있음)에 기록된다. 다수의 벡터가 이동되어야 하는 경우, 메모리 내의 다수의 이산 위치는 명령어에 의해 식별될 수 있으며, 데이터 요소의 벡터 각각은 특정된 위치 중 하나에 의해 식별되는 메모리 어드레스의 시퀀스로 이동된다. 대안적으로, 하나의 위치는 명령어에 의해 명시될 수 있고, 다른 위치는 암시적일 수 있다(예를 들어, 식별된 위치로부터 고정된 스트라이드/오프셋에서 메모리 어드레스를 식별하는 그러한 위치).

[0033] 단일 벡터의 개별 요소가 액세스되는 어레이 방향 및 어레이 내의 해당 액세스되는 벡터 내에 유지되는 데이터 요소의 특성에 따라 별개의 메모리 위치에 저장될 필요가 있을 수 있다. 그러나, 단일 벡터 내의 개별 데이터 요소가 별개의 메모리 위치에 저장될 필요가 있는 상황에서, 이는 데이터 요소의 벡터가 적절한 과정에서 메모리로 전송되기 전에 벡터 레지스터로 먼저 이동되는 경우가 전형적으로 될 것이다.

[0034] 이동될 데이터 요소의 하나 이상의 벡터가 이동 및 제로화 명령어에 의해 식별될 수 있는 다수의 방식이 존재한다. 일 예시적인 구현에서, 이동 및 제로화 명령어는 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 데 사용되는 벡터 식별 필드를 포함할 수 있다. 예를 들어, 데이터 요소의 단일 벡터가 이동될 때, 그 단일 벡터를 식별하기에 충분한 식별자가 제공될 수 있다. 데이터

요소의 다수의 벡터가 이동되어야하는 경우, 벡터 식별 필드는 그러한 다수의 벡터 각각을 명시적으로 식별하기 위한 충분한 정보를 제공하는 데 사용될 수 있거나, 대안적으로 데이터 요소의 하나의 벡터가 명시적으로 식별될 수 있으며, 그런 다음, 다른 벡터는 암시적, 예를 들어 인접한 벡터, 또는 규칙적으로 이격된 벡터(종종 스트라이드 액세스(stride access)로서 지칭됨)이다. 이 후자의 경우, 이동될 벡터의 수를 식별하기 위해 벡터 식별 필드에 의해 숫자가 제공될 수 있다.

[0035] 액세스가 어레이 방향 중 어느 하나에 있을 수 있는 앞서 언급된 접근법에서, 벡터 식별 필드는 또한 액세스되고 있는 어레이 방향을 식별하기에 충분한 정보를 제공하는 데 사용될 수 있다. 예를 들어, 일 구현에서, 벡터 식별 필드는 정사각형 2차원 어레이를 식별하기 위한 제1 서브필드, 및 하나 이상의 벡터를 식별하는 데 사용되는 하나 이상의 라인 식별자 및 어레이 방향 표시를 제공하는 제2 서브필드를 포함할 수 있다.

[0036] 일 예시적인 구현에서, 이동 및 제로화 명령어는 하나 이상의 식별된 벡터의 어느 데이터 요소가 어레이 저장소로부터 목적지 저장소로 이동되어야 하는지를 식별하는 데 사용되는 술어 정보(predicate information)를 식별하고, 이들의 연관된 저장 요소가 로직 제로 값으로 설정되도록 하는 술어 필드를 포함할 수 있다. 이는 기능이 특정 벡터 내의 특정 데이터 요소로 제한될 수 있게 함으로써, 추가적인 유연성을 제공할 수 있다.

[0037] 예측이 사용되는 일부 구현에서, 데이터 요소 크기는 변화될 수 있고, 그러한 경우에서 이동 및 제로화 명령어는 하나 이상의 식별된 벡터 내의 각각의 데이터 요소의 크기를 식별하기 위한 크기 필드를 포함할 수 있다. 이러한 추가 정보를 제공하기 위한 명령어를 가능하게 함으로써, 하나 이상의 식별된 벡터 내의 전체 데이터 요소의 서브세트에 대해 이동 및 제로화 동작(move and zero operation)이 수행될 수 있게 하면서, 시스템 내에서 프로세싱되고 있는 다양한 상이한 데이터 요소 크기에 대해 명령어가 사용될 수 있게 하는 것이 가능하다.

[0038] 본원에서 설명된 또 다른 기술에 따라, 어레이 저장소 내의 데이터 요소의 벡터를 제로화하기 위해 그리고 이러한 어레이 저장소를 사용하여 누산 동작을 수행할 때 성능 개선을 제공하기 위해 또한 사용될 수 있는 추가적인 새로운 형태의 명령어가 제공된다. 이 기술에 따르면, 동작을 수행하기 위한 프로세싱 회로부, 명령어에 의해 명시되는 동작을 수행하기 위해 프로세싱 회로부를 제어하도록 명령어를 디코딩하기 위한 명령어 디코더 회로부, 및 데이터 요소를 저장하기 위한 저장 요소를 포함하는 어레이 저장소를 갖는 장치가 제공된다. 앞서 설명된 기술과 같이, 어레이 저장소는 동작을 수행할 때 프로세싱 회로부에 액세스 가능한 데이터 요소의 적어도 하나의 2차원 어레이를 저장하도록 배열되고, 데이터 요소의 각각의 2차원 어레이는 데이터 요소의 복수의 벡터를 포함하고, 각각의 벡터는 1차원이다. 이 추가적인 기술에 따르면, 명령어 디코더 회로부는, 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 다수의 벡터를 식별하는 벡터 제로화 명령어를 디코딩하는 것에 응답하여, 데이터 요소의 식별된 다수의 벡터에 대해 동작하도록 배열되는 후속 누산 명령어를 또한 디코딩하고, 어레이 저장소 내의 식별된 다수의 벡터에 저장하기 위한 결과 데이터 요소를 생성하기 위해 누산 명령어에 의해 명시되는 누산 동작의 비-누산 변형을 수행하게끔 프로세싱 회로부를 제어하도록 배열된다.

[0039] 상기 접근법의 사용에 의해, 벡터 제로화 명령어는, 명령어 디코더 회로부에 의한 디코딩 시에, 벡터 제로화 명령어에 의해 명시되는 바와 같은 데이터 요소의 동일한 다수의 벡터를 명시하는 후속 누산 명령어와 융합(fuse)/병합되어, 사실상 해당 누산 명령어의 비-누산 변형을 생성할 수 있다. 명령어 인코딩 공간이 전형적으로 상당히 제약되기 때문에, 이러한 접근법은 매우 유익할 수 있고, 어레이 저장소 내의 다수의 벡터에 관하여 누산 동작을 수행하기 위한 다수의 상이한 누산 명령어를 제공하는 것이 바람직할 수 있지만, 그러한 비-누산 변형을 제공함으로써 소모될 것인 명령어 인코딩 공간의 양으로 인해 그러한 명령어의 비-누산 변형을 또한 제공하는 것을 추구하는 것은 매우 바람직하지 않을 수 있다는 것이 밝혀졌다. 본 기술을 사용함으로써, 명령어의 비-누산 변형을 제공할 필요가 없고, 대신에, 프로세싱 회로부로 하여금 비-누산 변형을 수행하게 하기 위해 벡터 제로화 명령어를 후속 누산 명령어와 조합함으로써 시뮬레이션될 수 있다.

[0040] 일 예시적인 구현에서, 어레이 저장소는 제1 어레이 방향으로 연장되는 복수의 어레이 벡터 레지스터를 포함할 수 있고, 어레이 저장소 내의 식별되는 다수의 벡터는 어레이 저장소의 다수의 어레이 벡터 레지스터의 그룹에 의해 제공된다. 이러한 구성에서, 데이터 요소의 주어진 2차원 어레이는 다수의 어레이 벡터 레지스터의 그룹 내에 저장되는 데이터 요소를 포함할 수 있다. 또한, 후속 누산 명령어는 데이터 요소의 식별되는 다수의 벡터(즉, 벡터 제로화 명령어에 의해 명시되는 바와 같은 다수의 어레이 벡터의 동일한 그룹)에 대해 수행될 누산 동작을 포함하는 프로세싱 동작을 명시할 수 있고, 벡터 제로화 명령어는 프로세싱 동작의 비누산 변형의 프로세싱 회로부에 의한 수행을 가능하게 하기 위해 후속 누산 명령어와 조합되어 사용될 수 있다.

[0041] 일 예시적인 구현에서, 벡터 제로화 명령어는 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데

이터 요소의 다수의 벡터를 식별하는 데 사용되는 벡터 식별 필드를 포함할 수 있다.

[0042] 앞서 논의된 이동 및 제로화 명령어와 같이, 벡터 제로화 명령어는 원하는 경우, 다수의 식별된 벡터 내의 어떤 저장 요소가 로직 제로 값으로 설정되어야 하는지를 식별하는 데 사용되는 술어 정보를 식별하기 위한 술어 필드를 포함할 수 있다. 그러한 접근법은 사실상 일부 데이터 요소가 후속 누산 명령어의 비-누산 변형의 대상이 되는 반면, 다른 데이터 요소는 진정한(true) 누산 변형의 대상이 되는 것을 허용할 수 있다. 또한, 원하는 경우, 벡터 제로화 명령어는 다수의 식별되는 벡터 내의 각각의 데이터 요소의 크기를 식별하기 위한 크기 필드를 포함할 수 있다.

[0043] 이제, 특정한 예시적인 구현이 도면을 참조하여 논의될 것이다.

[0044] 도 1은 데이터 값(32) 및 프로그램 명령어(34)를 저장하는 메모리(30)에 커플링된 프로세서(20)를 포함하는 데이터 프로세싱 시스템(10)을 개략적으로 예시한다. 프로세서(20)는 메모리(30)로부터 프로그램 명령어(34)를 페칭하고 페칭된 프로그램 명령어를 명령어 디코더 회로부(50)에 공급하기 위한 명령어 페칭 유닛(40)을 포함한다. 디코더 회로부(50)는 페칭된 프로그램 명령어를 디코딩하고, 디코딩된 벡터 명령어에 의해 명시된 바와 같이 레지스터 저장소(65)의 저장 요소 내에 유지되는 데이터 값에 대해 프로세싱 동작을 수행하기 위해 프로세싱 회로부(60)를 제어하도록 제어 신호를 생성한다. 도 1에 도시된 바와 같이, 레지스터 저장소(65)는 다수의 상이한 블록으로 형성될 수 있다. 예를 들어, 명령어에 의해 명시될 수 있는 복수의 스칼라 레지스터를 포함하는 스칼라 레지스터 파일(70)이 제공될 수 있고, 유사하게 명령어에 의해 명시될 수 있는 복수의 벡터 레지스터를 포함하는 벡터 레지스터 파일(80)이 제공될 수 있다.

[0045] 또한, 도 1에 도시된 바와 같이, 프로세서(20)는 어레이 저장소(90)에 액세스할 수 있다. 도 1에 도시된 예에서, 어레이 저장소(90)는 프로세서(20)의 일부로서 제공되지만, 이는 필수는 아니다. 다양한 예에서, 어레이 저장소는 하기 중 임의의 하나 이상으로서 구현될 수 있다: 아키텍처적으로-어드레스가능(architecturally-addressable) 레지스터; 비-아키텍처적으로-어드레스가능 레지스터; 스크래치패드 메모리; 및 캐시.

[0046] 프로세싱 회로부(60)는 하나의 예시적인 구현에서 프로세싱 회로부와 스칼라 프로세싱 회로부 둘 모두를 포함할 수 있다. 스칼라 프로세싱과 벡터 프로세싱 사이의 일반적인 구별은 다음과 같다. 벡터 프로세싱은 데이터 벡터 내의 각각의 포지션에서 복수의 데이터 요소를 갖는 데이터 벡터의 데이터 요소에 단일 벡터 프로세싱 명령어를 적용하는 것을 수반할 수 있다. 본 기술에 따르면, 프로세싱 회로부는 또한 어레이 저장소(90) 내에 저장되는 데이터 요소의 2차원 어레이(이는 서브어레이로도 지칭될 수 있음) 내의 복수의 벡터에 대해 동작을 수행하기 위한 벡터 프로세싱을 수행할 수 있다. 스칼라 프로세싱은 데이터 벡터에 대해서보다는 단일 데이터 요소에 대해 효과적으로 동작한다. 벡터 프로세싱은 프로세싱될 데이터의 많은 상이한 인스턴스에 대해 프로세싱 동작이 수행되는 경우에 유용할 수 있다. 벡터 프로세싱 배열에서, 단일 명령어가 (데이터 벡터의) 다수의 데이터 요소에 동시에 적용될 수 있다. 이는 스칼라 프로세싱에 비해 데이터 프로세싱의 효율 및 처리량을 개선할 수 있다.

[0047] 프로세서(20)는 어레이 저장소(90)에 저장되는 데이터 요소의 2차원 어레이를 프로세싱하도록 배열될 수 있다. 2차원 어레이는 적어도 일부 예에서, 다수의 방향에서 데이터 요소의 1차원 벡터로서 액세스될 수 있다. 일 예시적인 구현에서, 어레이 저장소(90)는 데이터 요소의 하나 이상의 2차원 어레이를 저장하도록 배열될 수 있고, 데이터 요소의 각각의 2차원 어레이는 메모리 내의 데이터 요소의 더 크거나 훨씬 더 높은 차원의 어레이의 정사각형 어레이 부분을 형성할 수 있다.

[0048] 도 2는 하나의 예시적인 구현에서 제공될 수 있는 프로세서(20)의 아키텍처 레지스터(65)의 예를 도시한다. (명령어 세트 아키텍처(instruction set architecture; ISA)에서 정의된 바와 같은) 아키텍처 레지스터는 프로세싱 회로부(60) 내의 스칼라 프로세싱 회로부에 의해 수행되는 프로세싱 동작을 위한 범용 레지스터로서 작용하는 스칼라 정수 레지스터(100)의 세트를 포함할 수 있다. 예를 들어, 제공되는 특정 수의 범용 레지스터(100), 예를 들어, 이러한 예에서는 31개의 레지스터(X0 내지 X30)가 있을 수 있다(스칼라 레지스터 필드의 32번째 인코딩은 하드웨어로 제공되는 레지스터에 대응하지 않을 수 있는데, 이는, 그것이 기본적으로, 예를 들어, 0의 값을 나타내는 것으로 고려될 수 있거나, 또는 범용 레지스터가 아닌 전용 유형의 레지스터를 나타내는 데 사용될 수 있기 때문임). 동일한 물리적 저장소에 맵핑되는 상이한 크기의 스칼라 레지스터에 액세스하는 것이 가능할 수 있다. 예를 들어, 레지스터 라벨(X0 내지 X30)은 64 비트 레지스터를 지칭할 수 있지만, 동일한 레지스터가 또한 32 비트 레지스터로서 액세스될 수 있는데(예를 들어, 하드웨어로 제공되는 각각의 64 비트 레지스터의 하위 32 비트를 사용하여 액세스됨), 이 경우, 레지스터 라벨(W0 내지 W30)은 어셈블러 코드에서 동일한 레지스터를 참조하는 데 사용될 수 있다.

- [0049] 또한, 디코더(50)에 의해 지원되는 ISA에서 프로그램 명령어에 의한 선택을 위해 이용가능한 아키텍처 레지스터는 특정 수의 벡터 레지스터(105)(이러한 예에서 Z0 내지 Z31로 라벨링됨)를 포함할 수 있다. 물론, 도 2에 도시된 수의 스칼라/벡터 레지스터를 제공하는 것이 필수는 아니며, 다른 예는 프로그램 명령어에 의해 명시가능한 상이한 수의 레지스터를 제공할 수 있다. 각각의 벡터 레지스터는 가변적인 수의 데이터 요소를 포함하는 벡터 피연산자를 저장할 수 있으며, 여기서 각각의 데이터 요소는 독립적인 데이터 값을 표현할 수 있다. 벡터 프로세싱(SIMD) 명령어에 응답하여, 프로세싱 회로부는 레지스터에 저장되는 벡터 피연산자에 대해 벡터 프로세싱을 수행하여 결과를 생성할 수 있다. 예를 들어, 벡터 프로세싱은, 대응하는 동작이 하나 이상의 피연산자 벡터 내의 각각의 레인의 요소에 대해 수행되어 결과 벡터의 요소에 대해 대응하는 결과를 생성하는, 레인별(lane-by-lane) 동작을 포함할 수 있다. 벡터 또는 SIMD 프로세싱을 수행할 때, 각각의 벡터 레지스터는 특정 벡터 길이(vector length; VL)를 가질 수 있으며, 여기서 벡터 길이는 주어진 벡터 레지스터에서의 비트의 수를 지칭한다. 벡터 프로세싱 모드에서 사용되는 벡터 길이(VL)는 주어진 하드웨어 구현에 대해 고정될 수 있거나 가변적일 수 있다. 프로세서(20)에 의해 지원되는 ISA는 가변적인 벡터 길이를 지원할 수 있고, 그에 따라 상이한 프로세서 구현이 상이한 크기의 벡터 레지스터를 구현하는 것을 선택할 수 있지만, ISA는 벡터 길이에 독립적(agnostic)일 수 있고, 그에 따라 코드가 그 프로그램을 실행하는 주어진 CPU 상에서 구현되는 특정 벡터 길이에 관계없이 올바르게 기능할 수 있도록 명령어가 설계된다.
- [0050] 벡터 레지스터(Z0 내지 Z31)는 또한, 어레이 저장소(90) 내에 저장되는 데이터 요소의 2차원 어레이에 대해 프로세싱 회로부(60)에 의해 수행되는 프로세싱 동작 및 누산 동작에 입력을 제공하는 벡터 피연산자를 저장하기 위한 피연산자 레지스터로서 기능할 수 있다. 벡터 레지스터가 그러한 동작에 입력을 제공하기 위해 사용될 때, 벡터 레지스터는 벡터 동작에 사용되는 벡터 길이(VL)와 동일할 수 있거나 상이한 벡터 길이일 수 있는 벡터 길이(MVL)를 가질 수 있다.
- [0051] 도 2에 도시된 바와 같이, 아키텍처 레지스터는 전술된 어레이 저장소(90)를 형성하는 특정 수 N_A 의 어레이 레지스터(110), ZA0 내지 ZA(N_A-1)을 또한 포함한다. 각각의 어레이 레지스터는 데이터 요소의 단일 2D 어레이, 예를 들어, 프로세싱 및 누산 동작의 결과를 저장하기 위한 한 세트의 레지스터 저장소로서 보여질 수 있다. 그러나, 프로세싱 및 누산 동작이 어레이 레지스터를 사용할 수 있는 유일한 동작은 아닐 수 있다. 어레이 레지스터는 또한, 메모리 내에서의 어레이 구조의 행/열 방향의 전위(transposition)를 수행하는 동안 정사각형 어레이를 저장하기 위해 사용될 수 있다. 프로그램 명령어가 어레이 레지스터(110) 중 하나를 참조할 때, 그것은 어레이 식별자(ZAi)를 사용하여 단일 엔티티로서 참조되지만, 일부 유형의 명령어(예를 들어, 데이터 전송 명령어)는 또한 어레이의 일부(예를 들어, 요소의 하나의 수평/수직 그룹)를 선택하는 인덱스 값을 정의함으로써 어레이의 하위 부분을 선택할 수 있다.
- [0052] 실제로, 어레이 레지스터에 대응하는 레지스터 저장소의 물리적 구현(physical implementation)은 도 2에 또한 도시된 바와 같이, 특정 수 N_R 의 어레이 벡터 레지스터(ZAR0 내지 ZAR(N_R-1))를 포함할 수 있다. 어레이 레지스터 저장소(110)를 형성하는 어레이 벡터 레지스터(ZAR)는 SIMD 프로세싱에 사용되는 벡터 레지스터(Z0 내지 Z31) 및 어레이 프로세싱에 대한 벡터 입력과는 별개의 세트의 레지스터일 수 있다. 어레이 벡터 레지스터(ZAR) 각각은 벡터 길이(MVL)를 가질 수 있으며, 이에 따라, 각각의 어레이 벡터 레지스터(ZAR)는 가변적인 수의 데이터 요소로 논리적으로 파티셔닝될 수 있는 길이(MVL)의 1D 벡터를 저장할 수 있다. 예를 들어, MVL이 512 비트인 경우, 이것은, 예를 들어, 64개의 8 비트 요소, 32개의 16 비트 요소, 16개의 32 비트 요소, 8개의 64 비트 요소 또는 4개의 128 비트 요소의 세트일 수 있다. 주어진 구현에서 이러한 옵션 모두가 지원될 필요는 없을 것이 이해될 것이다. 가변적인 요소 크기를 지원함으로써, 이것은 상이한 정밀도의 데이터 구조를 수반하는 계산을 핸들링하기 위한 유연성을 제공한다. 데이터의 2D 어레이를 표현하기 위해, 한 그룹의 어레이 벡터 레지스터(ZAR0 내지 ZAR(N_R-1))는 어레이 레지스터 식별자(ZA0 내지 ZA(N_A-1)) 중 주어진 하나가 할당되는 단일 엔티티로서 논리적으로 고려될 수 있으며, 그에 따라, 2D 어레이는 어레이의 하나의 차원에 대응하는 단일 벡터 레지스터 내에서 연장되는 요소 및 다수의 벡터 레지스터에 걸쳐 스트라이핑되는(striped) 어레이의 다른 차원 내의 요소로 형성된다.
- [0053] 어레이 레지스터(ZA)가, 수평 방향의 요소의 수가 수직 방향의 요소의 수와 같은, 데이터의 정사각형 어레이를 저장하도록 어레이 레지스터(ZA)를 배열하는 것이 필수는 아니지만 유용할 수 있다. 이것은, 수평 방향으로 또는 수직 방향으로 어레이 레지스터(110)를 판독/기입하기 위한 지원을 제공함으로써, 메모리 내의 어레이 구조의 행/열 차원이 어레이 레지스터(110)와 메모리 사이에서 어레이 구조를 전송할 시에 스위칭될 수 있는 어레이의 즉각적인(on-the-fly) 전위를 지원하는 것을 도울 수 있다. 2D 어레이 레지스터로부터의 데이터를 수평 방

향 또는 수직 방향으로 기입/판독하기 위한 지원을 제공함으로써, 이것은, 메모리로부터 하나의 방향으로(예를 들어, 행별로) 로딩된 데이터가, 메모리와 벡터 레지스터 사이에서 데이터를 전송하기 위해 다수의 수집/분산 로드/저장 또는 치환 동작을 사용하여 가능할 것보다 더 빠르게, 메모리로 반대 방향으로(예를 들어, 열별로) 다시 기록될 수 있게 할 수 있다.

[0054] 위에서 언급된 바와 같이, 데이터 요소의 주어진 2D 어레이는 일부 예시적인 구현에서 정사각형 어레이를 형성할 수 있지만, 이는 필수가 아니다. 따라서, 도 3a에 도시된 바와 같이, 하나의 예시적인 구현에서, 데이터 요소(115)의 주어진 2D 어레이는 비-정사각형 어레이를 형성할 수 있다. 대안적으로, 도 3b에 도시된 바와 같이, 데이터 요소(120)의 주어진 2D 어레이는 정사각형 어레이를 형성할 수 있다. 도면 3a 및 도 3b 각각에서, 개별 정사각형 박스는 데이터 요소를 나타내고, 일부 구현에서 데이터 요소 크기는 변할 수 있다. 어느 예에서든, 데이터 요소의 2차원 어레이는 다양한 방식으로 명시될 수 있지만, 하나의 예시적인 구현에서, 도 3a 및 3b에 도시된 바와 같이, 데이터 요소의 주어진 2D 어레이는 어레이 벡터 레지스터(ZAR)의 시퀀스에 의해 명시될 수 있다.

[0055] 위에서 논의된 바와 같이, 프로세싱 회로부(60)는, 디코더 회로부(50)에 의해 디코딩되는 명령어의 제어 하에, 스칼라 레지스터(70), 벡터 레지스터(80), 및/또는 어레이 저장소(90)에 액세스하도록 배열된다. 이러한 후자의 배열의 추가 상세는 이제, 특히 어레이 저장소 내의 정사각형 2D 어레이에 대한 액세스를 고려하면서, 어레이 저장소가 어떻게 액세스될 수 있는지의 하나의 예시적인 예를 단순히 제공하는 도 4a를 참조하여 설명될 것이다.

[0056] 예시된 예에서, 어레이 저장소(90) 내의 정사각형 2D 어레이는 $n \times n$ 개의 저장 요소/위치(200)의 어레이(205)로서 배열되며, 여기서 n 은 1보다 큰 정수이다. 본 예에서, n 은 16이며, 이는 저장 위치(200)에 대한 액세스의 입도가 수평 또는 수직 어레이 방향 중 어느 하나에서 전체 저장소의 1/16임을 암시한다.

[0057] 프로세싱 회로부의 관점에서, $n \times n$ 개의 위치의 어레이는 제1 방향(예를 들어, 도시된 바와 같은 수평 방향)으로 n 개의 선형(1차원) 벡터로서 및 제2 어레이 방향(예를 들어, 도시된 바와 같은 수직 방향)으로 n 개의 선형 벡터로서 액세스 가능하다. 따라서, $n \times n$ 개의 저장 위치는, 프로세싱 회로부(60)의 관점에서, 각각이 n 개의 데이터 요소인 $2n$ 개의 선형 벡터로서 배열되거나 적어도 액세스 가능하다.

[0058] 저장 위치(200)의 어레이는, 적어도 프로세싱 회로부(60) 및 선택적으로 디코더 회로부(50)와 통신하는 제어 회로부(250)의 제어 하에서, 액세스 회로부(210, 220), 열 선택 회로부(230) 및 행 선택 회로부(240)에 의해 액세스 가능하다.

[0059] 도 4b를 참조하면, 제1 방향(도시된 바와 같은 수평 또는 "H" 방향)으로의 n 개의 선형 벡터는, "A1"로 지정된 예시적인 정사각형 2D 어레이의 경우(이하에서 논의되는 바와 같이, 어레이 저장소(90) 내에 제공되는 하나 초과개의 그러한 2D 어레이, 예를 들어 A0, A1, A2 등이 있을 수 있음에 유의함), 각각 16개의 데이터 요소 0...F(16진수 표기법)이고, 이 예에서 A1H0...A1H15로서 참조될 수 있다. 도 4b의 어레이 저장소(90) A1의 256개의 엔트리(16 x 16개의 엔트리)에 저장된 동일한 기본 데이터(underlying data)는, 대신에, 제2 방향(도시된 바와 같은 수직 또는 "V" 방향)에서 A1V0...A1V15로서 참조될 수 있다. 예를 들어, 데이터 요소(260)는 A1H0의 항목 F로 참조되지만 A1V15의 항목 0으로 참조되는 것에 유의한다. "H" 및 "V"의 사용은 어레이 저장소(90)를 구성하는 데이터 요소의 저장소와 관련된 어떠한 공간적 또는 물리적 레이아웃 요건도 암시하지 않으며, 어레이 저장소 내의 2D 어레이가 임의의 예시 애플리케이션에서 행 또는 열 데이터를 저장하는지 여부와의 관련성도 없다.

[0060] 앞서 논의된 바와 같이, 어레이 저장소(90)의 사용은 특정 유형의 동작, 예를 들어, 이러한 누산 동작의 복수의 반복이 어레이 저장소(90) 내의 데이터 요소의 주어진 2차원 어레이에 대해 수행될 수 있는 누산 동작과 관련하여 성능을 유의하게 개선할 수 있으며, 데이터 요소의 2차원 어레이는 그러한 누산 동작을 수행할 때 결과를 누산하는 데 사용된다. 그러나, 일단 이러한 누산 동작이 완료되면, 데이터 요소의 생성된 벡터를 어레이 저장소 밖으로 이동시키고, 이들이 후속 누산 동작에 사용될 수 있도록 어레이 저장소 내에서 연관된 저장 요소를 준비하기 위한 효율적인 메커니즘이 바람직할 것이다.

[0061] 앞서 논의된 바와 같이, 하나의 예시적인 구현에서, 이것은 어레이 저장소(90) 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 하나 이상의 벡터를 식별하는 이동 및 제로화 명령어의 사용을 통해 달성된다. 그러한 이동 및 제로화 명령어가 디코딩될 때, 프로세싱 회로부(60)는 그 후, 하나 이상의 식별된 벡터의 데이터 요소를 어레이 저장소로부터 목적지 저장소(이는 예를 들어, 벡터 레지스터 파일(80) 내의 하나 이상의 벡터

레지스터일 수 있음)로 이동시키고, 또한 하나 이상의 식별된 벡터의 데이터 요소를 저장하는 데 사용되고 있었던 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하도록 제어된다.

[0062] 도 5a는 하나의 예시적인 구현에 따른 이동 및 제로화 명령어 내에서 제공될 수 있는 필드를 개략적으로 예시한다. opcode 필드(305)는 이동 및 제로화 명령어로서 명령어를 식별하는 데 사용된다. 일부 예시적인 구현에서, 제공되는 이동 및 제로화 명령어의 상이한 변형이 존재할 수 있고, 따라서 이동 및 제로화 명령어를 식별할 수 있는 하나 초과와 상이한 opcode가 존재할 수 있다. 하나의 특정 예로서, 이동 및 제로화 명령어가 어레이 저장소 내의 식별되는 벡터를 벡터 레지스터 파일(80) 내의 타겟 벡터 레지스터로 이동시키는 데 사용될 때 이동 및 제로화 명령어에 대해 정의되는 하나의 변형이 있을 수 있고, 어레이 저장소 내의 하나 이상의 벡터가 메모리로 이동될 때 상이한 변형이 제공될 수 있다(이 후자의 경우, 명령어는 예를 들어 저장 및 제로화 명령어로 지칭될 수 있음).

[0063] 벡터 식별 필드(310)는 또한 이동 동작의 대상이 될 어레이 저장소 내의 하나 이상의 벡터를 식별하기 위해 제공된다. 일부 경우에서, 오직 단일 벡터만이 식별될 수 있지만, 다른 경우에는 다수의 벡터가 이 필드에 의해 식별될 수 있다. 후자의 경우에, 그러한 다수의 벡터는 모두 하나의 예시적인 구현에서 독립적으로 식별될 수 있지만, 또 다른 예시적인 구현에서, 다수의 벡터는, 예를 들어, 제1 벡터의 표시 및 이동될 벡터의 수의 표시로부터 추론될 수 있다.

[0064] 도 5a에 도시된 바와 같이, 벡터(들)가 이동되어야 하는 목적지 저장소를 식별하기 위해, 목적지 저장소 식별 필드(315)가 또한 이동 및 제로화 명령어 내에 제공된다. 일 예시적인 구현에서, 이 필드는 벡터 레지스터 파일(80)에서 하나 이상의 벡터 레지스터를 식별하는 데 사용되고, 다수의 이러한 벡터 레지스터가 식별되는 예에서, 이들은 다수의 벡터가 벡터 식별 필드에 의해 식별되는 것과 유사한 방식으로 식별될 수 있다(예를 들어, 제1 벡터 레지스터가 식별될 수 있으며, 다른 벡터 레지스터는 그 후 목적지 저장소를 형성하기 위해 요구되는 벡터 레지스터의 수에 대한 지식에 기초하여 암시적임). 대안적인 구현에서, 어레이 저장소 내의 하나 이상의 벡터가 메모리로 이동될 경우, 목적지 저장소 식별 필드(315)에 제공되는 정보는 하나 이상의 벡터의 데이터 요소가 저장되어야 하는 메모리 내의 위치를 식별하도록 배열될 수 있다. 이는 예를 들어, 메모리 내의 필요한 위치를 식별하기 위해 그 내용물이 사용되는 하나 이상의 레지스터를 식별하는 것을 수반할 수 있다.

[0065] 원하는 경우, 하나 이상의 선택적인 추가 필드(320)가 명령어(300) 내에 제공될 수 있다. 예를 들어, 술어 필드는 하나 이상의 식별되는 벡터 내의 어떤 데이터 요소가 이동 및 제로화 동작의 대상이 될지 제어하는 데 사용되는 술어 정보를 식별하는 데 사용될 수 있다. 이는 특정 데이터 요소에 대해서는 동작이 적용되게 할 수 있지만 다른 요소에 대해서는 적용될 수 없도록함으로써 유연성을 제공한다. 또 다른 예로서, 데이터 요소 크기 표시가 명령어 내에 제공될 수 있고, 그에 의해 데이터 요소 크기가 고정되지 않은 벡터에 대해 명령어가 적용되게 할 수 있다.

[0066] 일 예시적인 구현에서, 이동 및 제로화 명령어는 제1 어레이 방향으로 연장되는 어레이 벡터 레지스터에 대해 동작하도록 배열될 수 있고, 이러한 구현에서 수평 방향 및 수직 방향 모두가 벡터 식별 필드(310) 내에서 인코딩될 수 있게 할 필요는 없을 것이다. 그러나, 어레이 저장소(90) 내의 2D 어레이가 수평 또는 수직 방향 중 어느 하나로 액세스될 수 있는 예에서는(하나의 이러한 구현에서, 2D 어레이는 정사각형 어레이임), 그러면 벡터 식별 필드는 도 5b에 도시된 형태를 취할 수 있다. 특히, 이 벡터 식별 필드(310')는 액세스될 어레이 저장소(90) 내의 주어진 정사각형 2D 어레이를 식별하기 위해 사용되는 제1 서브필드(312), 및 2개의 부분(313 및 314)으로 형성된 제2 서브필드를 포함할 수 있다. 제1 부분(313)은 정사각형 2D 어레이 내의 데이터 요소의 하나 이상의 라인을 식별하기 위한 하나 이상의 라인 식별자를 제공하고, 제2 부분(314)은 어레이 방향 표시를 제공하고, 이에 따라 라인 식별자에 의해 식별되는 데이터 요소의 라인이 수평 방향 또는 수직 방향으로 연장되는지에 대한 결정을 가능하게 한다. 제1 부분(313) 및 제2 부분(314)의 조합은 정사각형 2D 어레이 내의 하나 이상의 벡터가 식별될 수 있게 한다는 것을 이해할 것이다.

[0067] 도 6은 이동 및 제로화 명령어의 동작을 예시하는 흐름도이다. 단계(350)에서, 이동 및 제로화 명령어가 디코더 회로부(50)에 의해 조우되었다고 결정될 때, 단계(355)에서, 어레이 저장소 내의 데이터 요소의 하나 이상의 벡터가 이동 및 제로화 명령어의 벡터 식별 필드(310)에 제공되는 정보로부터 식별된다. 또한, 단계(360)에서, 사용될 목적지 저장소는 이동 및 제로화 명령어의 목적지 저장소 식별 필드(315)로부터 식별된다. 앞서 언급된 바와 같이, 이 단계는 전형적으로 벡터 레지스터 파일(80) 내의 하나 이상의 벡터 레지스터로 하여금 어레이 저장소 밖으로 이동되는 데이터 요소에 대한 목적지로서 식별되게 할 수 있지만, 대안적으로 일부 구현에서, 이는 식별되는 목적지 저장소가 메모리 내의 하나 이상의 위치의 형태를 취하는 경우일 수 있다. 일 예시적인 구현

에서, 단계(355 및 360)는 디코더 회로부(50)에 의해 수행될 수 있지만, 대안적인 구현에서, 프로세싱 회로부(60)는 디코더 회로부(50)에 의해 제공되는 정보에 기초하여 그러한 결정 단계를 수행할 수 있다.

[0068] 단계(365)에서, 프로세싱 회로부(60)는 데이터 요소의 각각의 식별된 벡터를 목적지 저장소로 이동시키는 데 사용되고, 이어서 어레이 저장소의 관련 저장 요소, 즉, 이제 목적지 저장소로 이동된 데이터 요소를 저장하는 데 사용되는 그러한 저장 요소를 제로로 설정하는 데 사용된다.

[0069] 도 7은 하나의 예시적인 구현에서 어레이 저장소(90)에 대해 동작할 수 있는 예시적인 명령어 시퀀스를 예시한다. 도 7의 예에 도시된 바와 같이, 어레이 저장소 내에 제공되는 주어진 2D 어레이 내에서 프로세싱 및 누산 동작을 수행하기 위해 일련의 데이터 프로세싱 명령어(이 예에서 3개의 명령어)가 수행될 수 있다. 이러한 다수의 명령어의 수행 동안, 결과는 주어진 2D 어레이 내에 누산된다.

[0070] 이 예에서, 일단 제3 데이터 프로세싱 명령어가 완료되면, 주어진 2D 어레이 내의 제1 수직 벡터는 최종 누산 결과를 저장하는 반면, 이 스테이지에서 주어진 2D 어레이 내의 다른 수직 벡터는 중간 누산 결과만을 저장한다고 가정된다. 제1 수직 벡터가 최종 누산 결과를 저장한다는 것을 고려하면, 후속 프로세싱 및 누산 동작에서 사용하기 위해 제1 수직 벡터의 저장 요소를 확보하기 위해 그러한 결과를 어레이 저장소 밖으로 이동시키는 것이 유용할 것이다.

[0071] 도 7에 도시된 바와 같이, 이것은 이동 및 제로화 명령어를 실행하고, 수직 벡터 1을 식별하고, 그 수직 벡터의 내용물이 이동되어야 하는 목적지 벡터 레지스터를 정의함으로써 달성되며, 이 예에서 벡터 레지스터는 레지스터 Z_i 로서 지칭된다. 이러한 명령어의 실행은 수직 벡터 1에서의 최종 누산 결과가, 식별되는 벡터 레지스터로 이동되게 하고, 주어진 2D 어레이 내의 연관된 저장 요소(즉, 수직 벡터 1을 구현하는 저장 요소)가 로직 제로 값으로 소거되게 한다. 이 단일 명령어의 실행 결과로서, 최종 누산 결과가 어레이 저장소 밖으로 이동되었을 뿐만 아니라 기본 저장 요소도 준비되고, 그에 따라 후속 프로세싱 및 누산 동작에서 즉시 재사용 가능하다. 특히, 기본 저장 요소의 내용물을 0으로 소거함으로써, 기본 저장 요소는 후속되는 명령어에 의해 산출되는 새로운 누산 결과를 위한 목적지로 즉시 명시되기 시작할 수 있다.

[0072] 따라서, 도 7에 도시된 바와 같이, 후속 데이터 프로세싱 명령어(4)가 실행될 때, 이는 2D 어레이에 누산될 수 있고, 원하는 경우 수직 벡터(1)를 재사용할 수 있다. 일단 해당 데이터 프로세싱 명령어가 실행되면, 수직 벡터(2)는 이제 최종 누산 결과를 유지하는 것으로 가정되고, 따라서, 주어진 2D 어레이 내의 수직 벡터(2)의 내용물을 어레이 저장소 밖으로 그리고 목적지 벡터 레지스터(이 예시적인 레지스터(Z_{i+x}))로 이동시키기 위해 추가적인 이동 및 제로화 명령어가 실행될 수 있다. 다시, 이 명령어의 실행은 식별되는 벡터의 내용물이 어레이 저장소 밖으로 이동되게 하고, 대응하는 저장 요소가 로직 제로 값으로 소거되게 하며, 따라서 후속 프로세싱 및 누산 동작에서 사용하기 위해 이러한 저장 요소를 확보한다. 따라서, 도 7에 도시된 바와 같이, 그런 다음, 데이터 프로세싱 명령어의 후속 반복이 실행될 수 있고, 원한다면 수직 열(2)을 재사용할 수 있다.

[0073] 결과를 누산하기 위해 어레이 저장소(90) 내의 주어진 2D 어레이를 사용하여 수행될 수 있는 다양한 유형의 동작이 존재하며, 여기서 해당 주어진 2D 어레이 내의 모든 벡터가 반드시 최종 누산 결과를 동시에 보유할 것은 아니다. 이러한 상황에서, 재사용을 위해 2D 어레이 내의 자원을 확보하도록, 도 7에서 예로서 개략적으로 예시된 접근법을 채택하는 것이 유용할 수 있다. 이러한 접근법을 위한 하나의 사용 사례는 슬라이딩 윈도우 접근법을 사용하여 2D 유한 임펄스 응답(FIR) 필터링을 수행하는 경우이다. 이러한 접근법은 입력 이미지(400)가 고려되는 도 8에 개략적으로 도시되어 있다. 특히, FIR 필터링 동작은 대응하는 출력 이미지(430)를 생성하기 위해 입력 이미지에 적용되며, 여기서 출력 이미지(430) 내의 각각의 픽셀(415, 420)은 대응하는 필터링 동작(407, 412)의 결과로서 생성된다. 그러한 각각의 필터링 동작에 대해, 다수의 입력 픽셀이 고려되고, 출력 픽셀에 대한 값을 생성하기 위해 그러한 다수의 입력 픽셀에 대해 필터링 계수가 적용된다.

[0074] 도 8에 도시된 예에서, 각 출력 픽셀은 입력 픽셀의 3x3 어레이를 고려하여 생성되는 것으로 가정된다. 따라서, 입력 픽셀(405)의 제1 3x3 어레이는 필터 동작(407)에 제공되고, 여기서 픽셀(415)에 대해 출력 값을 생성하기 위해 필터 계수의 대응하는 어레이를 사용하여 필터링을 수행한다. 유사하게, 입력 픽셀(410)의 제2 3x3 어레이는 출력 픽셀(420)에 대한 값을 생성하기 위해 필터 계수의 대응하는 세트를 사용하여 필터 동작(412)의 대상이 된다. 어레이(410)가 어레이(405)에 비해 우측으로 한 픽셀 포지션만큼 이동된 것을 알 수 있을 것이며, 위에 설명된 프로세스가 반복됨에 따라 3x3 픽셀의 슬라이딩 윈도우를 추출하여 각 필터 동작의 입력으로서 사용할 수 있다. 행의 끝이 도달되는 경우, 프로세스는 입력 이미지의 좌측으로 되돌아가지만, 이미지에서 한 행 아래에서 시작하여 다시 이미지의 왼쪽에서 우측으로 진행할 수 있다. 따라서, 각각의 출력 이미

지 픽셀을 계산할 때 사용될 입력 이미지 픽셀의 3x3 어레이를 추출하는 입력 이미지 위로 이동하는, 도 8에 도시된 배열에서 "수평" 방향으로 먼저 이동하는 슬라이딩 윈도우가 효과적으로 존재함을 알 것이다.

[0075] 도 9a 내지 도 9d는 이러한 2D 이미지 필터링 동작이 어레이 저장소(90) 내의 정사각형 2D 어레이를 사용하여 그리고 해당 2D 어레이에 보유되는 데이터에 대해 외적(outer product) 누산 동작을 수행함으로써 어떻게 효율적으로 수행될 수 있는지를 예시한다. 도 9a 내지 도 9d에 예시된 예에서, 슬라이딩 윈도우는 "수직" 방향, 즉 도 8의 예에 도시된 것과 직교하는 방향으로 먼저 이동할 것이라는 점에 유의한다. 따라서, 도 9a 내지 도 9d에 예시된 바와 같은 입력 이미지(440)는 그의 법선 뷰잉 방향에 대해 그 측면에 있는 것으로 간주될 수 있다.

[0076] 도 9a 내지 도 9d에 도시된 바와 같이, 입력 이미지(440)의 하나의 열(또는 이미지의 일부)은 한번에 프로세싱되고, 필터 계수의 벡터를 사용하는 필터링 동작의 대상이 된다. 필터 계수의 벡터 내의 각각의 블록(예를 들어, 도 9a의 블록(465) 참조)은 계수의 3x3 어레이로부터 3개의 필터 계수를 나타낸다. 패딩 요소(예를 들어, 도 9a의 요소(467) 참조)는 0 또는 정의되지 않은 값에 대응하고, 단지 예시되는 구현의 아티팩트(artefact)에 불과하다. 특히, 예시되는 구현에서, 사용되는 명령어는 결과당 4개의 곱셈 및 누산을 수행할 수 있지만, 도시된 예시적인 구현에서 3개의 곱셈 및 누산을 수행하는 것만이 요구된다.

[0077] 예시된 예에서, 계수의 4개의 세트(예를 들어, 3x3개의 계수의 4개의 세트의 하나의 행으로부터 취해진 도 9a의 4개의 블록(468) 참조)는 제공된 입력 벡터에 대해 4개의 출력 벡터를 계산하기 위해 입력으로서 사용된다((3+1)개의 계수의 하나의 세트가 4개의 출력 벡터를 생성하기 위해 사용되는 4개의 곱셈 동작 각각에서 사용됨). 또한, 도 9a 내지 9d에 예시된 프로세스는 명령어당 계수의 각각의 3x3 어레이로부터 3개의 계수의 하나의 열만을 사용하고, 따라서 프로세스는 4개의 벡터의 임의의 특정 세트 내에서 최종 누산 결과를 생성하기 위해 3개의 명령어를 취한다.

[0078] 프로세스가 진행 중이고 정상 상태일 때, 도 9c 및 도 9d를 참조하여 나중에 논의될 바와 같이, 프로세스는 동시에 4개의 출력 벡터의 3개의 세트(예를 들어, 도 9c에 도시된 출력 벡터의 3개의 세트(475, 485, 495) 참조)에 대해 동작하고 있다.

[0079] 도 9a에 도시된 바와 같이, 입력 이미지(440)의 제1 열(470)이 계수(460)의 벡터를 사용하여 프로세싱될 때, 이는 누산 결과가 4개의 벡터(475) 내에 저장되게 한다. 도 9b에 도시된 바와 같이, 입력 이미지(440)의 제2 열(480)이 계수(482)의 벡터를 사용하여 프로세싱될 때, 이는 4개의 벡터(475) 및 4개의 벡터(485) 둘 모두 내에 누산 결과를 채운다. 그 후, 제3 열(490)이 계수(492)의 벡터를 사용하여 도 9c에 도시된 바와 같이 프로세싱되면, 이는 누산 결과가 4개의 벡터(475), 4개의 벡터(485) 및 4개의 벡터(495) 내에 채워지게 한다. 도 8의 앞선 논의로부터, 이 지점에서 입력 이미지의 첫 3개의 열 내의 모든 픽셀이 프로세싱되었을 것이고, 따라서 4개의 수평 벡터(475)의 내용물은 출력 이미지의 열의 첫 번째 그룹에 대해 최종 누산 결과를 나타낼 것이라는 것이 이해될 것이다. 따라서, 도 9c에 도시된 바와 같이, 그러한 4개의 레지스터(475)의 내용물은, 그러한 내용물을 목적지 저장소, 예를 들어, 벡터 레지스터 파일(80) 내의 4개의 벡터 레지스터로 이동시키고, 2D 어레이(450) 내의 그러한 4개의 벡터(475)를 형성하는 저장 요소를 소거하여, 이들이 후속 누산 동작에 대해 이용가능하도록 하기 위해, 앞서 설명된 이동 및 제로화 명령어의 대상이 될 수 있다.

[0080] 따라서, 예로서, 도 9d에 도시된 바와 같이, 입력 이미지(440)의 제4 열(500)이 프로세싱되는 경우, 계수(460)의 벡터를 사용하여, 이는 누산 결과가 4개의 벡터(485), 4개의 벡터(495), 및 4개의 벡터(475) 내에 채워지게 할 수 있다(이는 그러한 벡터 내의 저장 요소가 이전의 이동 및 제로화 명령어에 의해 로직 제로 값으로 소거되었기 때문에 이제 재사용될 수 있음). 또한 도 9d에 도시된 바와 같이, 4개의 벡터(485)는 출력 이미지의 열의 두 번째 그룹을 나타내는 최종 누산 결과를 이제 저장할 것이다(이 지점에서 제2, 제3 및 제4 입력 열 각각이 프로세싱되었을 것이기 때문임). 따라서, 그러한 4개의 수평 벡터(485) 내에 저장되는 데이터 요소는 어레이로부터 벡터 레지스터 파일의 벡터 레지스터 내로 이동될 수 있고, 그 후, 기존 저장 요소는 후속 누산 동작에서 이들의 재사용을 허용하기 위해 소거될 수 있다.

[0081] 도 9a 내지 9d에서, 이동 및 제로화 명령어의 대상이 되는 임의의 벡터가 즉시 재사용되지만(이는 더 용이한 프로그래밍을 초래할 수 있기 때문임), 소거된 벡터가 즉시 재사용되어야 하는 요건은 없으며, 대신, 프로세싱은 원한다면 벡터(475, 485, 495)의 그룹 아래의 벡터를 계속 소모할 수 있고, 어레이(450)의 하단에 도달되었을 때에만 시작으로 되돌아갈 수 있다.

[0082] 일 예시적인 구현에서, 어레이 저장소(90) 내의 주어진 정사각형 2D 어레이는 수평 또는 수직 방향 중 어느 하나로 액세스될 수 있다. 그러나, 일부 구현에서, 벡터가 방향 중 하나에서만 액세스되는 어레이 저장소(90) 내

의 2D 어레이를 사용하여 수행될 수 있는 특정 프로세싱 동작이 존재한다. 따라서, 예로서, 앞서 논의된 도 2를 다시 참조하면, 어레이를 통해 제1 어레이 방향으로 연장되는 어레이 벡터 레지스터(ZAR)를 구체적으로 식별하는 일부 프로세싱 명령어가 존재할 수 있다. 이러한 명령어는 연관된 프로세싱 및 누산 동작이 수행될 다수의 ZAR 레지스터를 명시함으로써, 특정 프로세싱 및 누산 동작의 고도로 효율적인 수행을 허용할 수 있다. 그러나, 일련의 그러한 명령어가 실행되었으면, 모든 식별된 어레이 벡터 레지스터(ZAR)는 전형적으로 최종 누산 결과를 포함할 것이지만, 그러한 결과가 어레이 저장소 밖으로 이동되고 그러한 어레이 벡터 레지스터를 형성하는 저장 요소의 현재 내용물이 로직 제로 값으로 소거될 때까지 후속 프로세싱 및 누산 동작을 위해 그러한 레지스터를 재사용하는 것은 불가능할 것이다.

[0083] 도 10은 이러한 상황에서 성능을 유의하게 증가시키기 위해 앞에서 설명된 이동 및 제로화 명령어가 어떻게 사용될 수 있는지를 개략적으로 예시한다. 특히, 도 10에 도시된 바와 같이, 3개의 어레이 벡터 레지스터(ZAR2, ZAR3, 및 ZAR4)가 0으로 초기화되고, 이어서 프로세싱 및 누산 동작을 수행하기 위해 상기 유형의 일련의 데이터 프로세싱 명령어가 수행되고, 누산 결과는 상기 3개의 어레이 벡터 레지스터 내에 유지되는 것으로 가정된다. 요구되는 일련의 데이터 프로세싱 명령어가 완료되면(이 예에서, 실행되는 2개의 그러한 데이터 프로세싱 명령어가 존재하는 것으로 가정됨), 상기 3개의 어레이 벡터 레지스터 모두는 최종 누산 결과를 저장할 것이다. 따라서, 앞서 언급된 이동 및 제로화 명령어는 그러한 3개의 어레이 벡터 레지스터를, 데이터 요소가 목적지 저장소로 이동되어야 하는 벡터로서 명시하는 데 사용될 수 있고, 또한 목적지 저장소로서 사용될 저장소를 식별할 수 있으며, 이 예에서, 벡터 레지스터 파일(80) 내의 3개의 인접한 벡터 레지스터가 사용되는 것으로 가정된다. 따라서, 이동 및 제로화 명령어의 실행은 모든 누산 결과가 어레이 저장소로부터 벡터 레지스터 파일의 식별된 벡터 레지스터 내로 이동되게 하고, 또한 그러한 3개의 어레이 벡터 레지스터를 형성하는 저장 요소가 로직 제로 값으로 소거되게 할 것이다. 따라서, 도 10에 도시된 바와 같이, 프로세스는 이어서, 동일한 일련의 어레이 벡터 레지스터(ZAR2, ZAR3 및 ZAR4) 내로도 누산될 후속 데이터 프로세싱 명령어의 시퀀스를 실행하도록 즉시 진행할 수 있다. 이는 매우 효율적인 구현을 제공한다.

[0084] 본원에 설명된 또 다른 기술에 따르면, 어레이 저장소 내의 데이터 요소의 벡터를 제로화하고, 이러한 어레이 저장소를 사용하여 누산 동작을 수행할 때(하나 이상의 벡터 레지스터로부터 제로를 어레이 저장소의 원하는 벡터 내로 전달하기 위해 이동 명령어를 사용할 필요가 있을 것이고, 그러한 제로 값을 유지하기 위해 하나 이상의 벡터 레지스터를 예약할 필요가 있을 것인 구현과 비교될 때) 성능의 개선을 제공하기 위해 또한 사용될 수 있는 추가적인 새로운 형태의 명령어(본원에서 벡터 제로화 명령어로 지칭됨)가 제공된다. 이 추가적인 기술에 따르면, 명령어 디코더 회로부(50)는 이러한 벡터 제로화 명령어(이는 어레이 저장소 내의 데이터 요소의 주어진 2차원 어레이 중의 데이터 요소의 다수의 벡터를 식별하도록 배열됨)를 디코딩하는 것에 응답하여, 데이터 요소의 식별된 다수의 벡터에 대해 동작하도록 배열되는 후속 누산 명령어를 또한 디코딩하도록 배열된다. 그 후, 프로세싱 회로부는 식별된 다수의 벡터의 데이터 요소를 저장하기 위해 사용되는 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하고, 그 후, 어레이 저장소 내의 식별된 다수의 벡터에 저장하기 위한 결과 데이터 요소를 생성하기 위해 누산 명령어에 의해 명시되는 누산 동작을 수행하도록 야기된다.

[0085] 상기 접근법의 사용에 의해, 벡터 제로화 명령어는, 명령어 디코더 회로부에 의한 디코딩 시에, 벡터 제로화 명령어에 의해 명시되는 바와 같은 데이터 요소의 동일한 다수의 벡터를 명시하는 후속 누산 명령어와 병합되어, 사실상 해당 누산 명령어의 비-누산 변형을 생성할 수 있다. 이는 매우 유익할 수 있는데, 명령어 인코딩 공간이 통상적으로 높은 프리미엄(high premium)에 있으므로, 어레이 저장소 내의 데이터 요소의 다수의 벡터에 대해 동작하도록 정의될 수 있는 다양한 누산 명령어의 비-누산 변형을 명시하는 것이 불가능 할 수 있다.

[0086] 도 11은 하나의 예시적인 구현에 따른 이러한 벡터 제로화 명령어의 핸들링을 예시하는 흐름도이다. 단계(520)에서 벡터 제로화 명령어가 디코더 회로부(50)에 의해 조우되는 경우, 단계(525)에서, 벡터 제로화 명령어의 벡터 식별 필드를 참조하여, 어레이 저장소 내의 데이터 요소의 다수의 벡터가 식별된다. 예를 들어, 다수의 어레이 벡터 레지스터(ZAR)는 벡터 식별 필드에서 식별될 수 있다. 그 후, 단계(530)에서, 디코더 회로부는 디코딩될 다음 명령어가 벡터 제로화 명령어에 의해 식별되는 것과 동일한 벡터에 대해 동작하는 누산 명령어인지 여부를 결정한다.

[0087] 그렇지 않은 경우, 단계(535)에서, 프로세싱 회로부는 단계(525)에서 결정되는 식별되는 다수의 벡터의 데이터 요소를 저장하는 데 사용되는 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하도록 제어되고, 그 후 프로세싱은 단지 다음 명령어의 실행으로 계속된다.

[0088] 그러나, 단계(530)에서 다음 명령어가 벡터 제로화 명령어에 의해 식별되는 것과 동일한 벡터에 대해 동작하고

있는 누산 명령어라는 것이 결정되면, 디코더는 2개의 명령어를 효과적으로 융합하고, 단계(540)에서, 식별되는 다수의 벡터 각각에서의 저장을 위한 결과를 생성하기 위해 누산 명령어에 의해 명시되는 누산 동작의 비-누산 변형(전형적으로, 이는 프로세싱 동작 및 후속 누산 둘 모두를 수반함)을 수행하도록 프로세싱 회로부를 제어한다. 앞서 언급된 바와 같이, 이러한 접근법에 의해, 어레이 저장소 내의 다수의 벡터에 대해 동작하도록 배열되는 임의의 누산 명령어의 비누산 변형을 구체적으로 인코딩할 필요는 없으며, 이는 그러한 비누산 변형은 벡터 제로화 명령어에 뒤따르는 요구되는 누산 명령어의 사용을 통해 상기 융합 절차에 의해 효과적으로 구현될 수 있기 때문이다(이에 의해, 누산 명령어에 의해 정의되는 프로세싱 동작을 구현하지만 누산 기능은 효과적으로 무효화됨).

[0089] 도 12는 일 예시적인 구현에서 벡터 제로화 명령어 내에서 제공될 수 있는 필드를 개략적으로 예시하는 도면이다. 특히, 벡터 제로화 명령어(550)는 명령어가 실제로 벡터 제로화 명령어임을 식별하기 위해 그 비트가 사용되는 opcode 필드(555)를 포함한다. 또한, 어레이 저장소 내의 다수의 벡터를 식별하는 데 사용되는 벡터 식별 필드(560)가 제공된다. 이 필드에서의 정보는 일반적으로 도 5a의 이동 및 제로화 명령어(300)의 벡터 식별 필드(310)를 설명할 때 앞서 논의된 형태를 취할 수 있지만, 하나의 예시적인 구현에서 후속 누산 명령어는 제1 어레이 방향으로 연장되는 어레이 벡터 레지스터에 대해 동작하도록 배열되는 것이고, 따라서 전형적으로 수평 및 수직 방향 둘 모두가 벡터 식별 필드(560) 내에서 인코딩되도록 할 필요는 없을 것이다.

[0090] 원하는 경우, 이어서 박스(565)에 의해 도시된 바와 같이, 이동 및 제로화 명령어 예를 참조하여 앞서 논의된 바와 같은 술어 정보 필드 및 데이터 요소 크기 필드와 같은 특정한 선택적인 추가 필드가 제공될 수 있다.

[0091] 도 13은 또 다른 예시적인 구현에 따른 벡터 제로화 명령어의 핸들링을 예시하는 흐름도이다. 단계(570)에서 벡터 제로화 명령어가 디코더 회로부(50)에 의해 조우되는 경우, 단계(575)에서, 벡터 제로화 명령어의 벡터 식별 필드를 참조하여, 어레이 저장소 내의 데이터 요소의 다수의 벡터가 식별된다. 예를 들어, 다수의 어레이 벡터 레지스터(ZAR)는 벡터 식별 필드에서 식별될 수 있다.

[0092] 그런 다음, 단계(580)에서, 프로세싱 회로부는 단계(575)에서 결정되는 식별되는 다수의 벡터의 데이터 요소를 저장하는 데 사용되는 어레이 저장소의 저장 요소를 로직 제로 값으로 설정하도록 제어되고, 그 후 프로세싱은 단지 다음 명령어의 실행으로 계속된다.

[0093] 이러한 구현에서도, 벡터 제로화 명령어와 후속 누산 명령어를 조합하기 위한 융합이 발생하지 않는 경우에도 여전히 유의한 이점이 달성될 수 있다. 특히, 다수의 이동 명령어를 실행할 필요가 없고, 각각의 이동 명령어는 제로의 벡터를 벡터 레지스터 파일의 벡터 레지스터로부터 어레이 저장소의 식별되는 벡터로 이동시킨다. 또한, 이러한 제로화 기능성은 (제로의) 벡터 이동 기능성을 구현해야 하는 것보다 하드웨어에서 구성하는 것이 더 간단하고 더 저렴하다. 또한, 이동 명령어의 사용에 기초한 상기 구현이 사용되면 요구될 것인, 로직 제로 값을 유지하기 위해 벡터 레지스터 파일 내의 하나 이상의 벡터 레지스터를 예약해야 하는 요건이 없기 때문에 추가적인 절약이 있다.

[0094] 도 14는 사용될 수 있는 시뮬레이터 구현을 예시한다. 앞서 설명한 예는 관련 기술을 지원하는 특정 프로세싱 하드웨어를 동작시키기 위한 장치 및 방법의 측면에서 본 발명을 구현하지만, 본 명세서에 설명된 예에 따라 컴퓨터 프로그램의 사용을 통해 구현되는 명령어 실행 환경을 제공하는 것이 또한 가능하다. 이러한 컴퓨터 프로그램은, 이들이 하드웨어 아키텍처의 소프트웨어 기반 구현을 제공하는 한, 종종 시뮬레이터로 지칭된다. 다양한 시뮬레이터 컴퓨터 프로그램은 에뮬레이터, 가상 머신, 모델, 및 동적 이진 변환기를 비롯한, 이진 변환기를 포함한다. 전형적으로, 시뮬레이터 구현은 호스트 프로세서(615) 상에서 실행될 수 있으며, 선택적으로 호스트 운영 체제(610)를 실행하여, 시뮬레이터 프로그램(605)을 지원할 수 있다. 일부 배열에서, 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 재사용을 이유로 다른 프로세서에 기본적인(native) 코드를 실행하기를 바라는 경우, 정당화될 수 있다. 예를 들어, 시뮬레이터 구현은 호스트 프로세서 하드웨어에 의해 지원되지 않는 추가적인 기능성을 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.

[0095] 예가 특정 하드웨어 구조 또는 특징을 참조하여 진술된 경우, 시뮬레이션된 구현에서 동등한 기능성이 적합한 소프트웨어 구조 또는 특징에 의해 제공될 수 있다. 예를 들어, 특정 회로부는 시뮬레이션된 구현에 컴퓨터 프

로그로 로직으로서 제공될 수 있다. 마찬가지로, 레지스터 또는 캐시와 같은 메모리 하드웨어가 시뮬레이션된 구현에서 소프트웨어 데이터 구조로 제공될 수 있다. 또한, 하드웨어 장치(10) 내의 메모리(30)에 액세스하는 데 사용되는 물리적 어드레스 공간은 시뮬레이터(605)에 의한 호스트 운영 체제(610)에 의해 사용되는 가상 어드레스 공간에 맵핑되는 시뮬레이션된 어드레스 공간으로 에뮬레이팅될 수 있다. 앞서 설명된 예에서 언급된 하드웨어 요소 중 하나 이상이 호스트 하드웨어(예를 들어, 호스트 프로세서(615)) 상에 존재하는 배열에서, 일부 시뮬레이션된 구현은 적합한 경우 호스트 하드웨어를 사용할 수 있다.

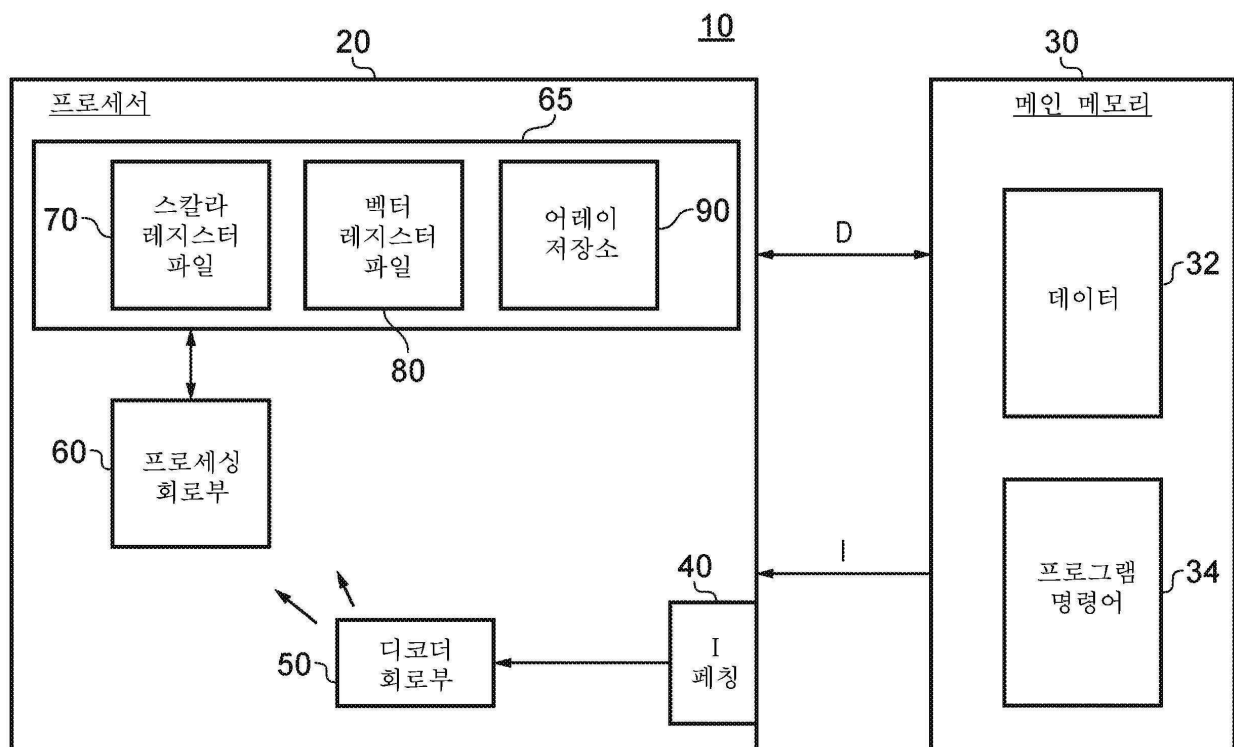
[0096] 시뮬레이터 프로그램(605)은 컴퓨터 판독 가능 저장 매체(이는 비일시적 매체일 수 있음) 상에 저장될 수 있으며, 타겟 코드(600)(이는 애플리케이션, 운영 체제, 및 하이퍼바이저(hypervisor)를 포함할 수 있음)에 가상 하드웨어 인터페이스(명령어 실행 환경)를 제공하고, 이는 시뮬레이터 프로그램(605)에 의해 모델링되는 하드웨어 아키텍처의 하드웨어 인터페이스와 동일하다. 따라서, 타겟 코드(600)의 프로그램 명령어는 시뮬레이터 프로그램(605)을 사용하는 명령어 실행 환경 내로부터 실행될 수 있고, 그에 따라 위에서 논의된 장치(10)의 하드웨어 특징을 실제로 갖지 않는 호스트 컴퓨터(615)가 그러한 특징을 에뮬레이팅할 수 있다. 시뮬레이터 프로그램은 프로세싱 회로부(60)의 거동을 에뮬레이팅하기 위한 프로세싱 프로그램 로직(620), 명령어 디코더 회로부(50)의 거동을 에뮬레이팅하기 위한 명령어 디코딩 프로그램 로직(625), 및 어레이 저장소(90)를 에뮬레이팅하기 위해 데이터 구조를 유지하기 위한 어레이 저장소 에뮬레이팅 프로그램 로직(622)을 포함할 수 있다. 따라서, 본원에서 설명된 기술은, 도 14의 예에서 시뮬레이터 프로그램(605)에 의해 소프트웨어에서 수행될 수 있다.

[0097] 본 출원에서, "...하도록 구성된"이라는 말은 장치의 요소가 정의된 동작을 수행할 수 있는 구성을 갖는다는 것을 의미하는 데 사용된다. 이런 맥락에서, "구성"은 하드웨어 또는 소프트웨어의 상호접속의 배열 또는 방식을 의미한다. 예를 들어, 장치는 정의된 동작을 제공하는 전용 하드웨어를 가질 수 있거나, 프로세서 또는 다른 프로세싱 디바이스가 기능을 수행하도록 프로그래밍될 수 있다. "하도록 구성된"은, 장치 요소가, 정의된 동작을 제공하기 위해 임의의 방식으로 변경될 필요가 있음을 암시하지는 않는다.

[0098] 예시적인 예가 첨부 도면을 참조하여 본원에서 상세히 설명되었지만, 본 발명은 그러한 정확한 예에 제한되지 않으며, 첨부된 청구범위에 의해 정의된 바와 같은 본 발명의 범위 및 사상으로부터 벗어남이 없이 다양한 변경, 추가 및 수정이 당업자에 의해 이루어질 수 있다는 것이 이해되어야 한다. 예를 들어, 본 발명의 범주로부터 벗어남이 없이 독립항의 특징과 종속항의 특징의 다양한 조합이 이루어질 수 있다.

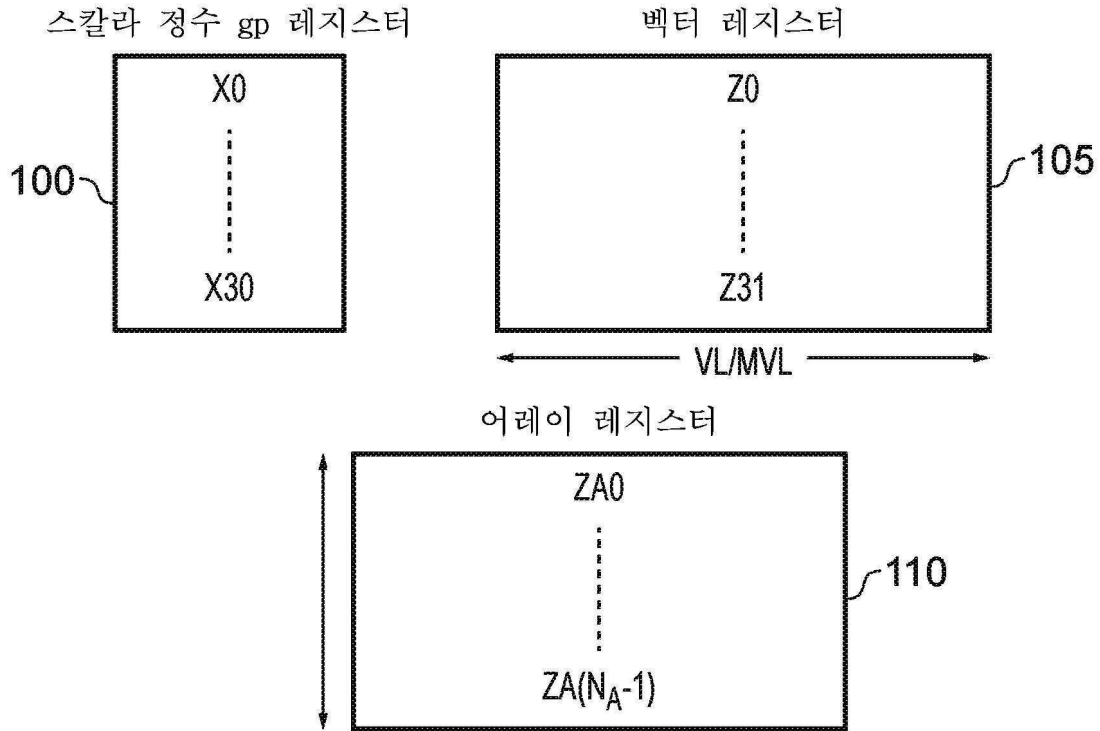
도면

도면1

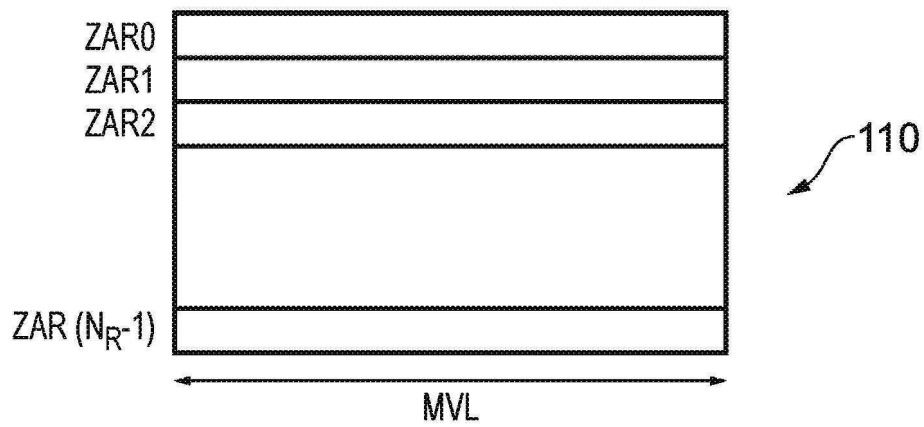


도면2

아키텍처 레지스터

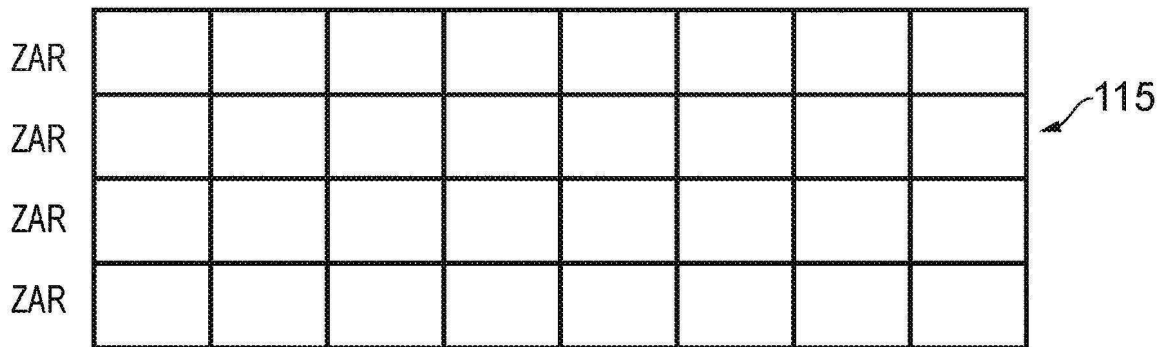


어레이 저장소의 물리적 구현



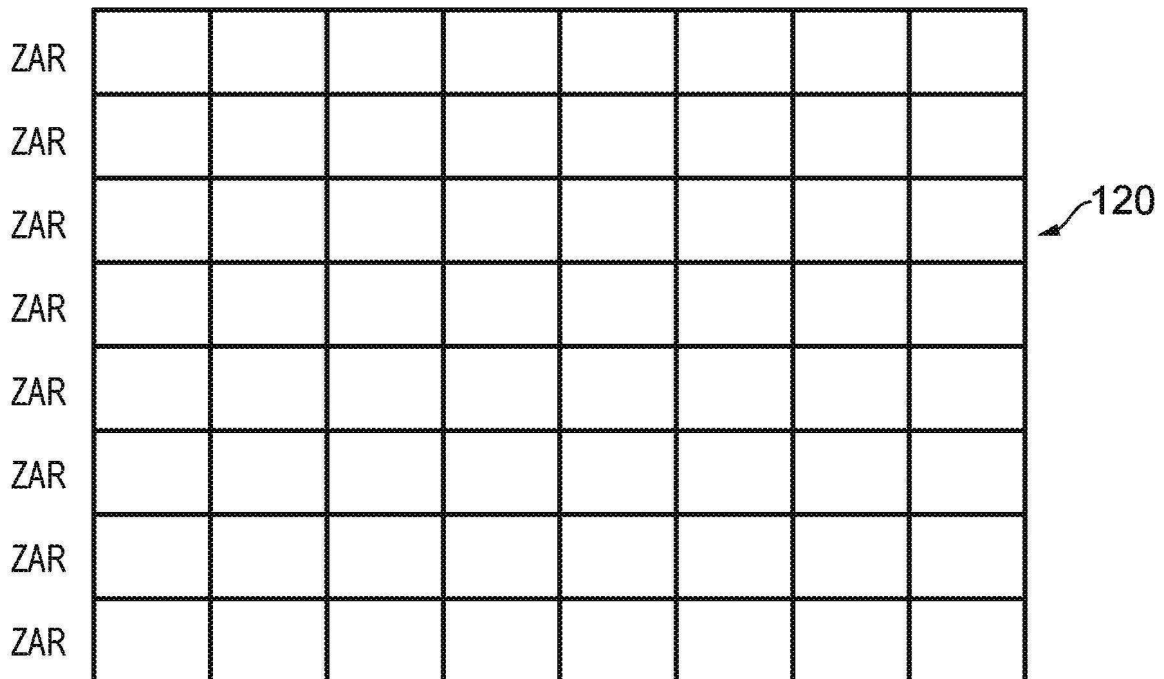
도면3a

데이터 요소의 비-정사각형 2D 어레이

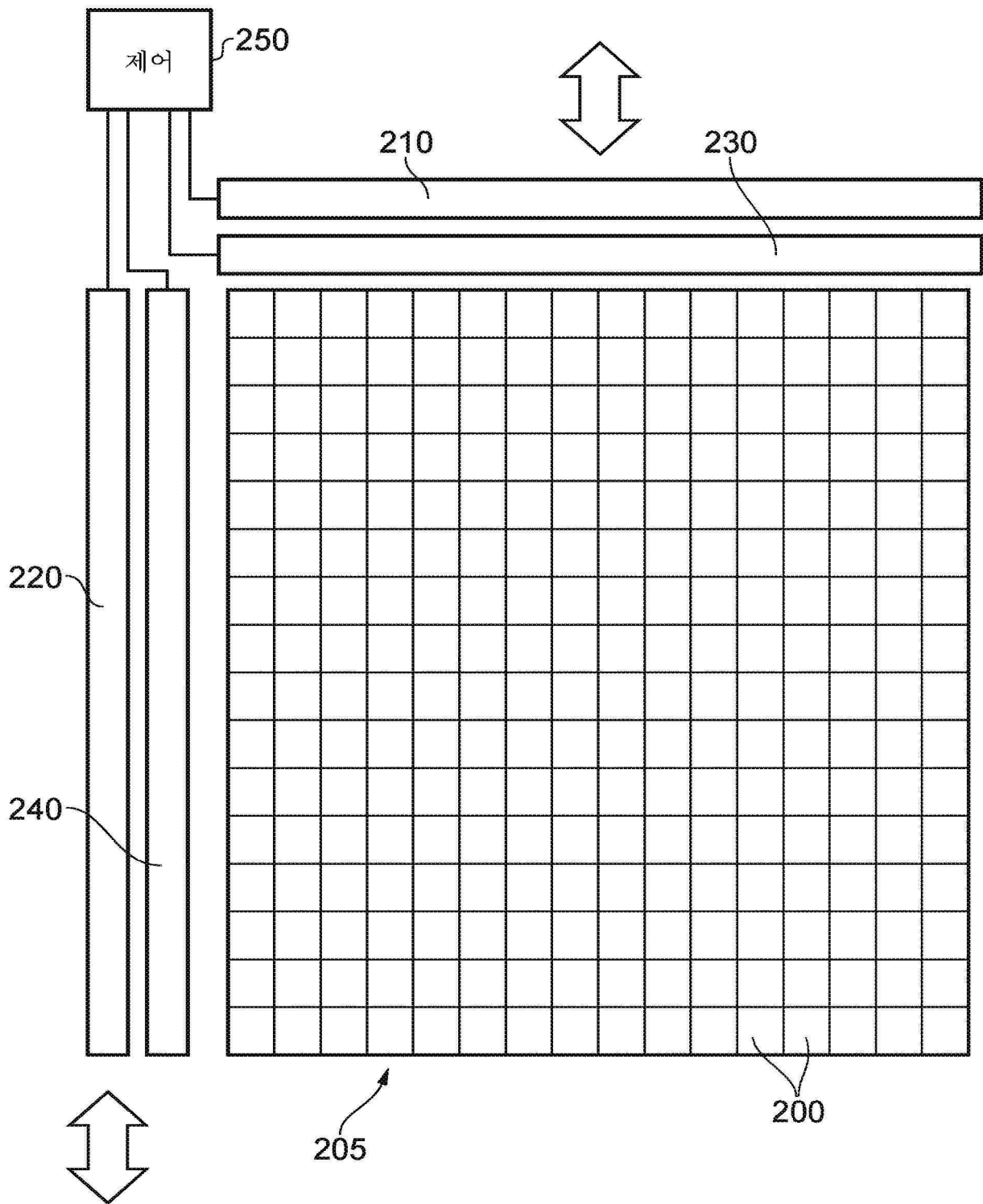


도면3b

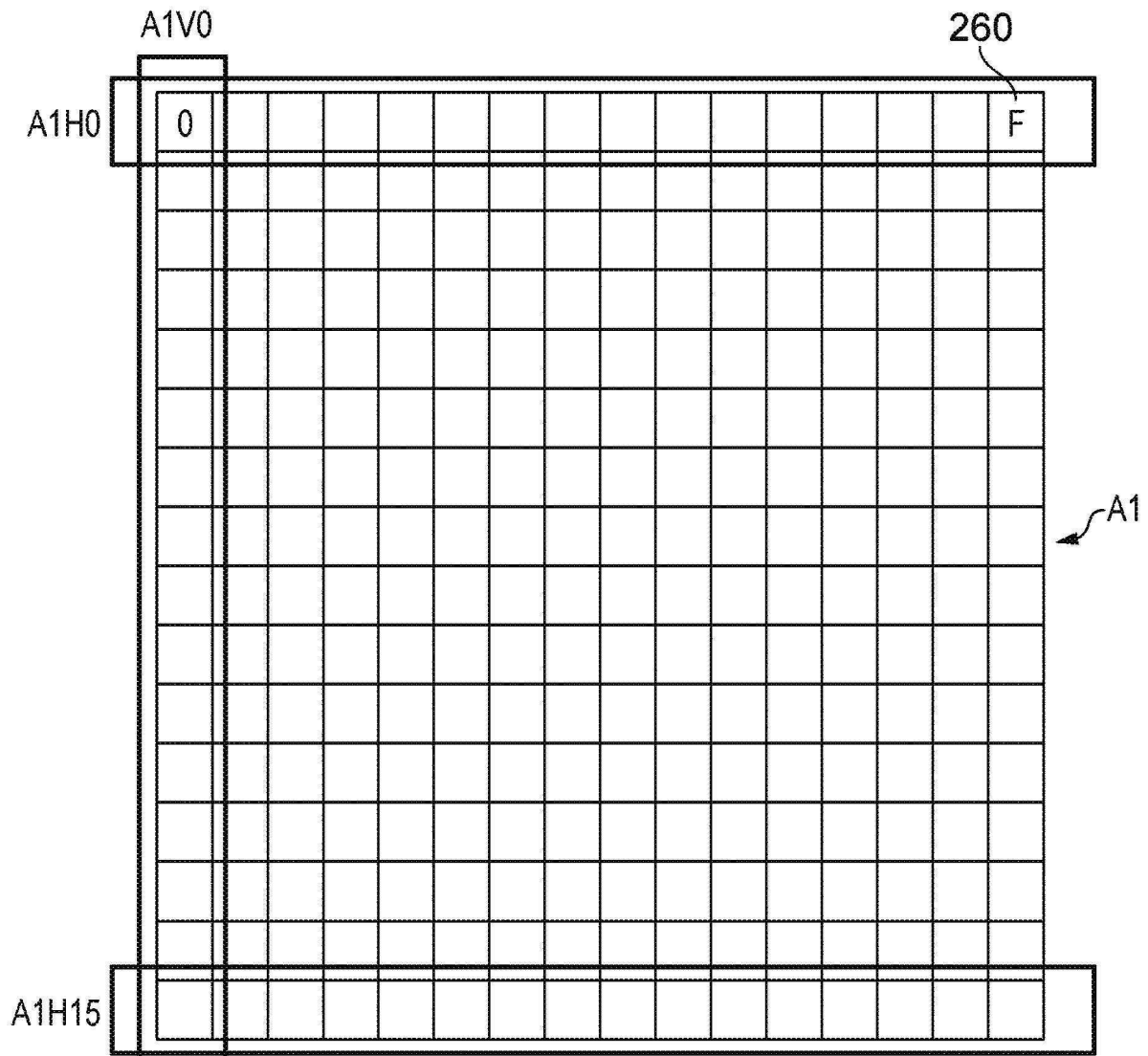
데이터 요소의 정사각형 2D 어레이



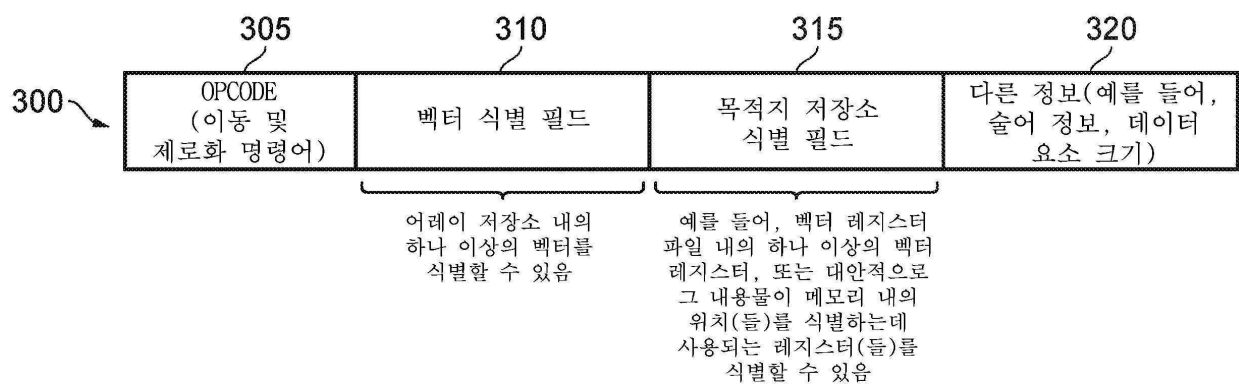
도면4a



도면4b



도면5a



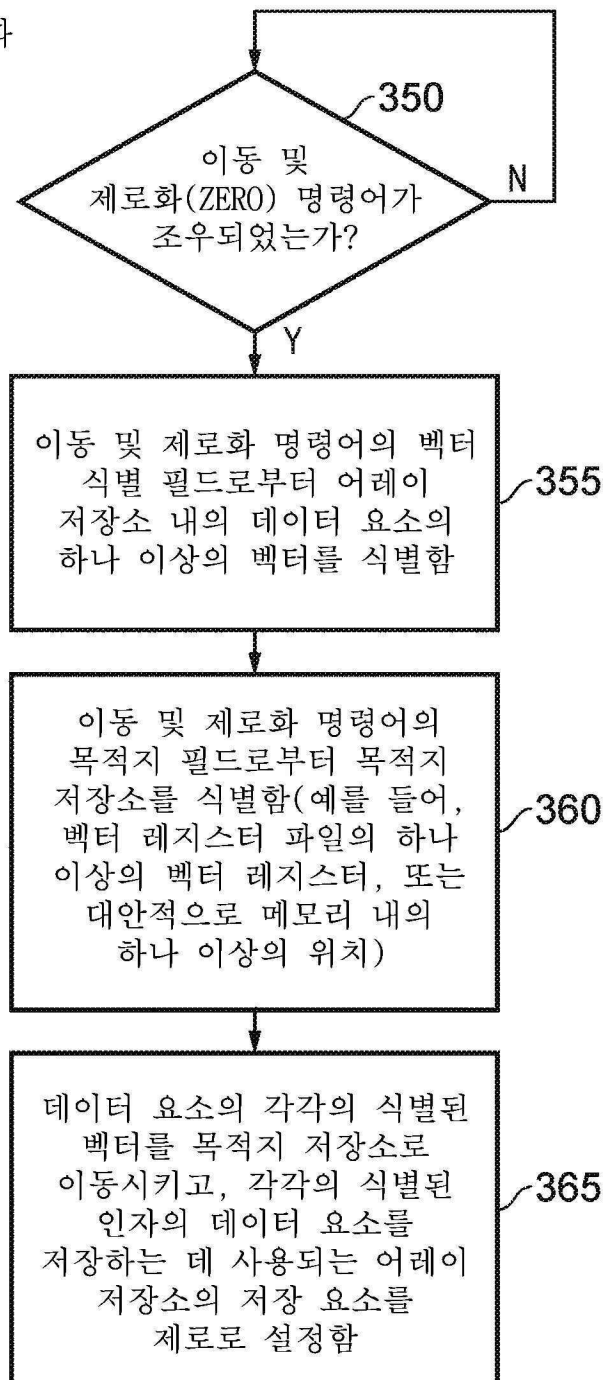
도면5b

벡터 식별 필드 예



도면6

이동 및 제로화
명령어의
핸들링



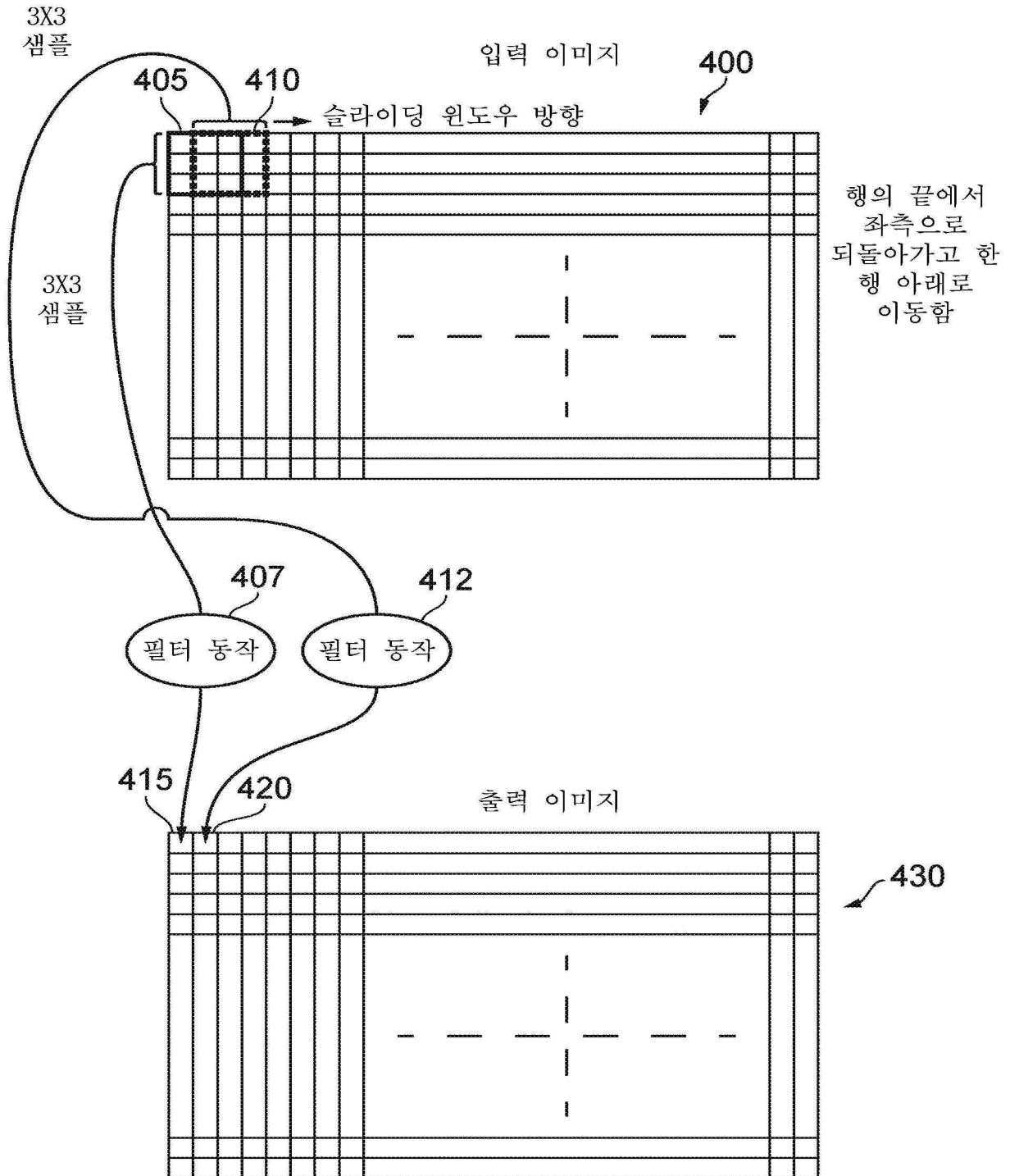
도면7

어레이 저장소에 대해 동작하는 예시적인 명령어 시퀀스

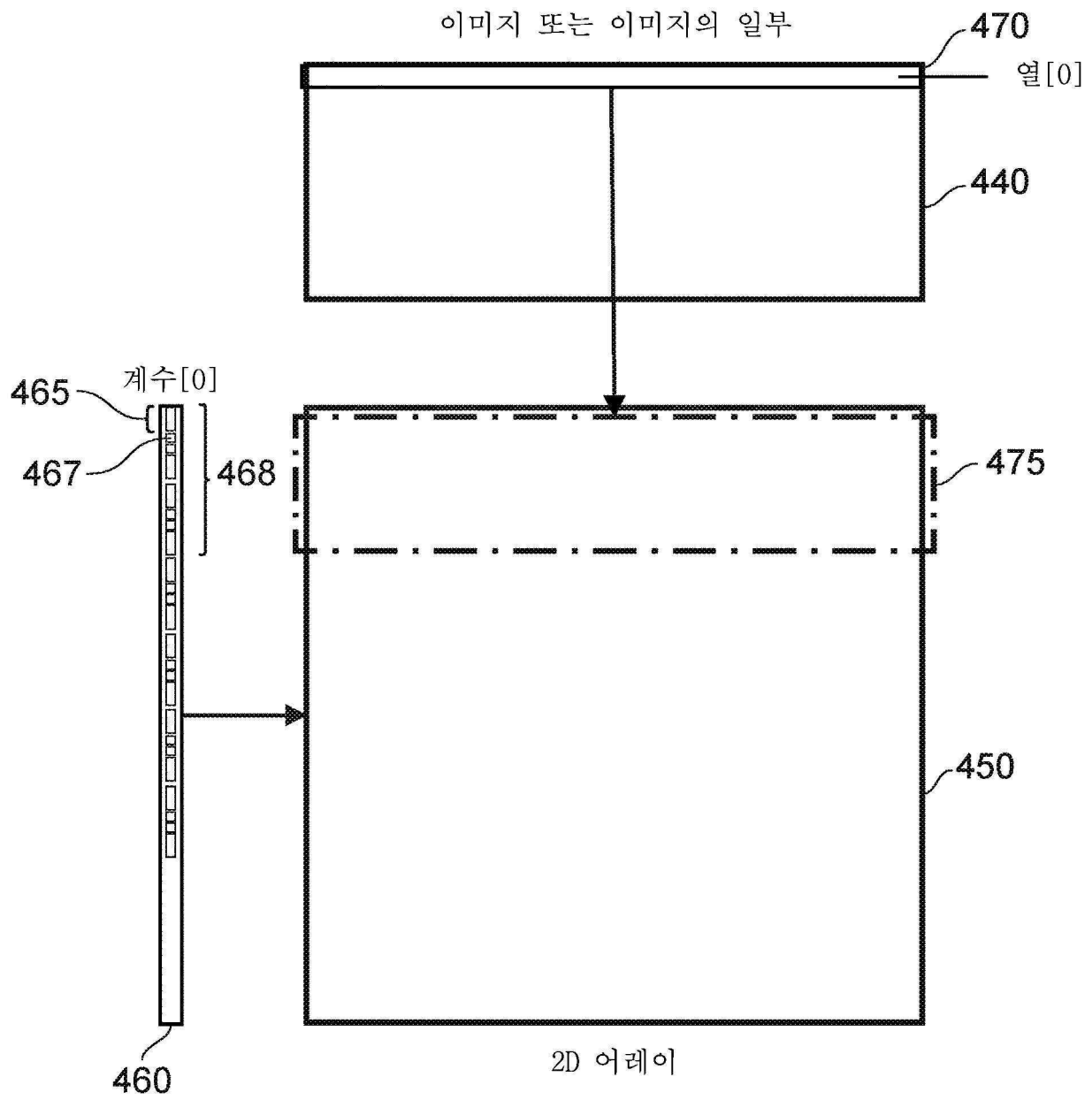
- 1) 데이터 프로세싱 명령어 1(2D 어레이 내로 누산함)
- 2) 데이터 프로세싱 명령어 2(2D 어레이 내로 누산함)
- 3) 데이터 프로세싱 명령어 3(2D 어레이 내로 누산함)
- 4) 이동 및 제로화 명령어(수직 벡터 1, 벡터 레지스터 Z_i) - 수직 벡터 1은 최종 누산 결과를 유지하고 이동 및 제로화 명령어의 실행은 그러한 최종 누산 결과를 Z_i 로 이동시키고 수직 벡터 1에 대한 저장 요소를 소거함
- 5) 데이터 프로세싱 명령어 4(2D 어레이 내로 누산함- 수직 열 1을 재사용할 수 있음)
- 6) 이동 및 제로화 명령어(수직 벡터 2, 벡터 레지스터 Z_{i+x}) - 수직 벡터 2는 최종 누산 결과를 유지하고 이동 및 제로화 명령어의 실행은 그러한 최종 누산 결과를 Z_{i+x} 로 이동시키고 수직 벡터 2에 대한 저장 요소를 소거함
- 7) 데이터 프로세싱 명령어 5(2D 어레이 내로 누산함- 수직 열 2를 재사용할 수 있음)

도면8

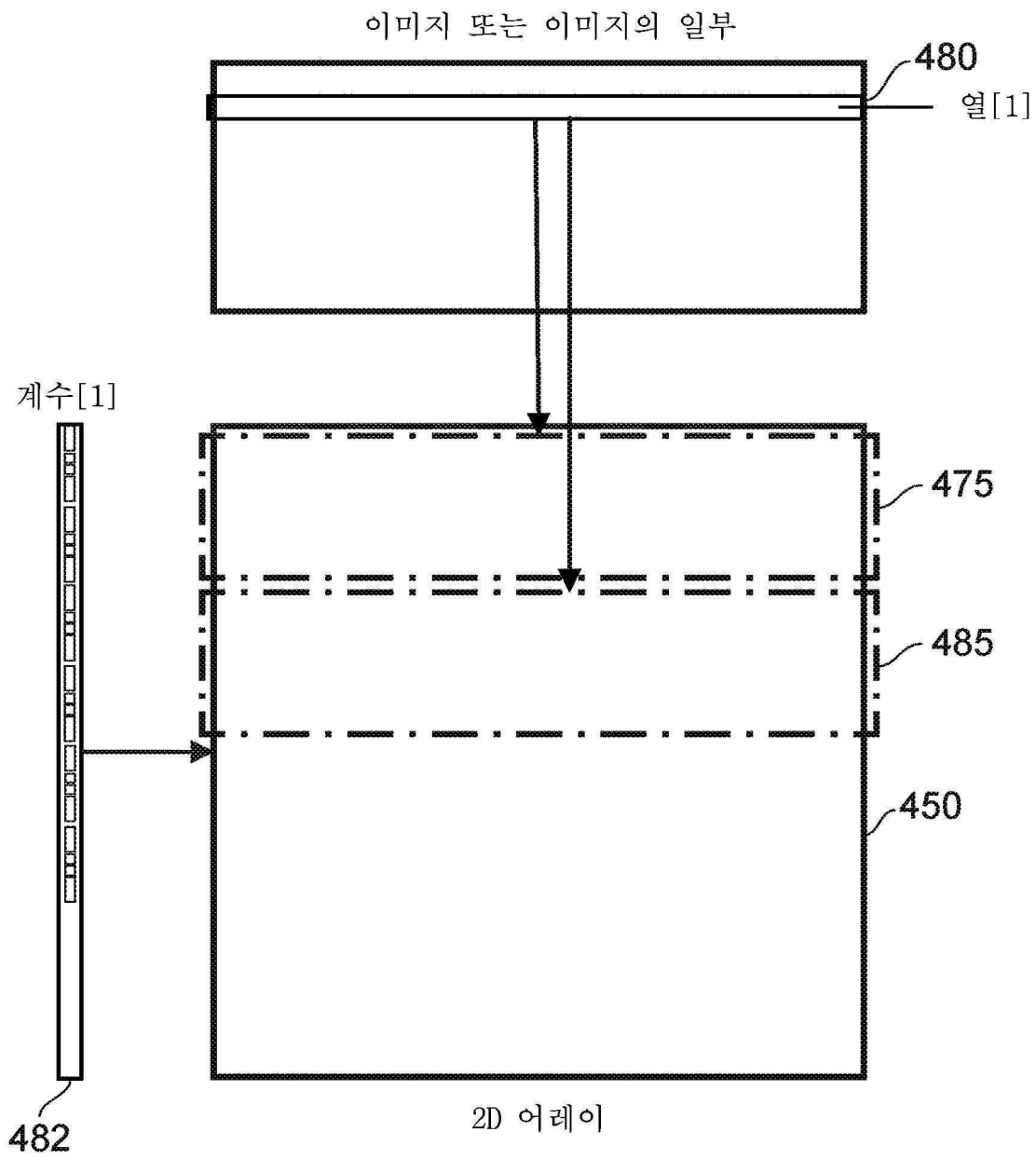
슬라이딩 윈도우 접근법을 사용한 FIR 필터링



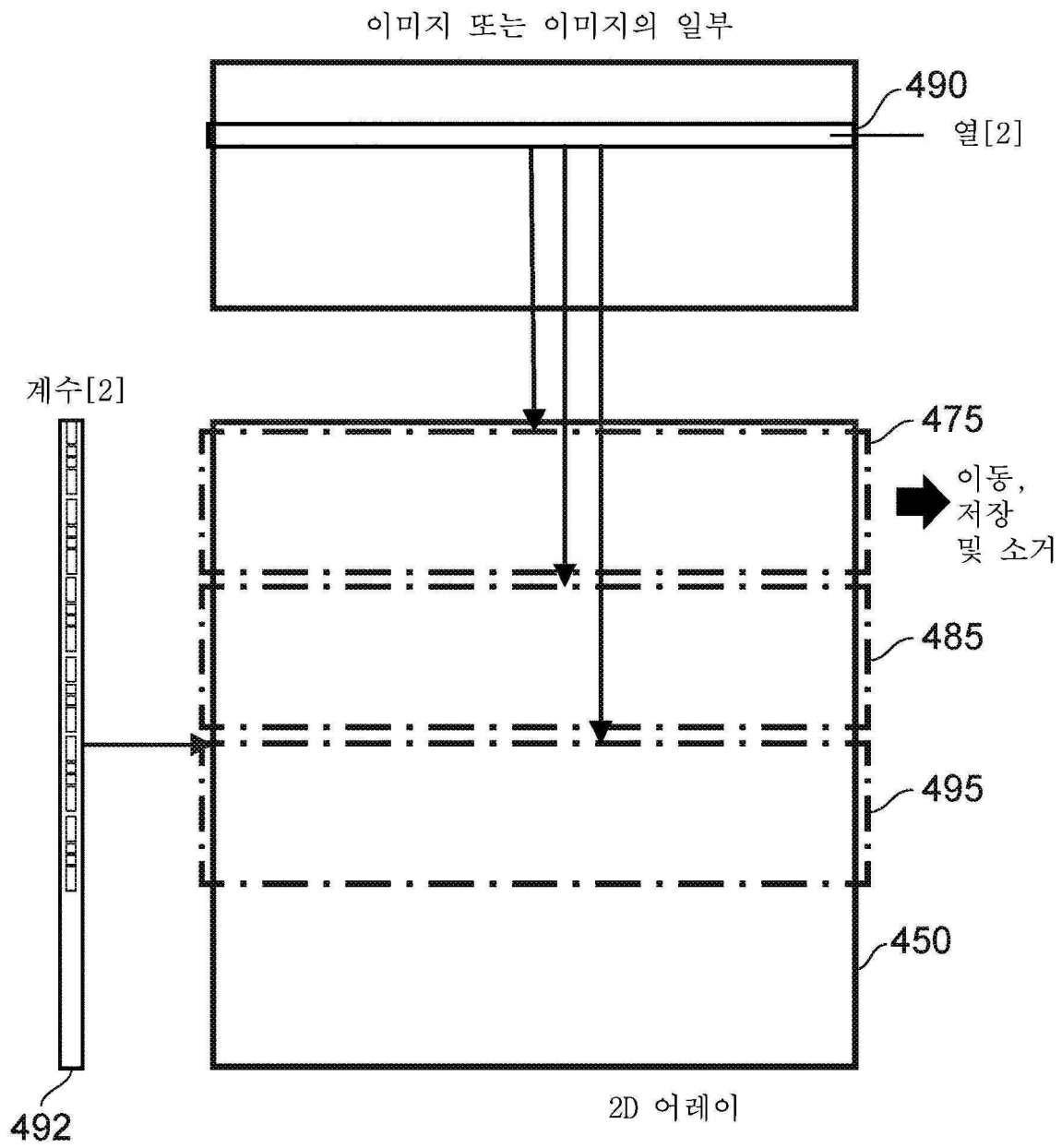
도면9a



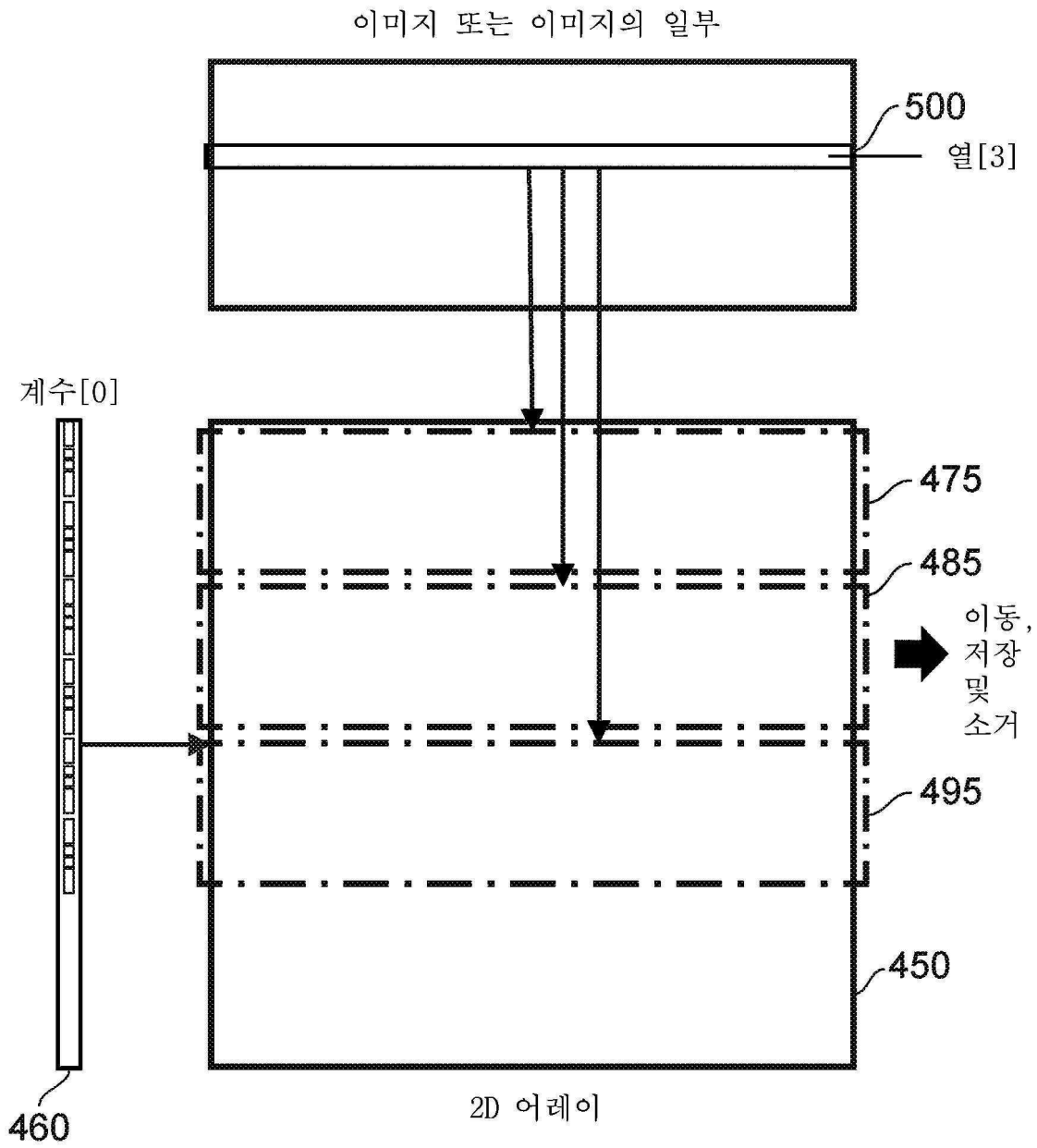
도면9b



도면9c



도면9d



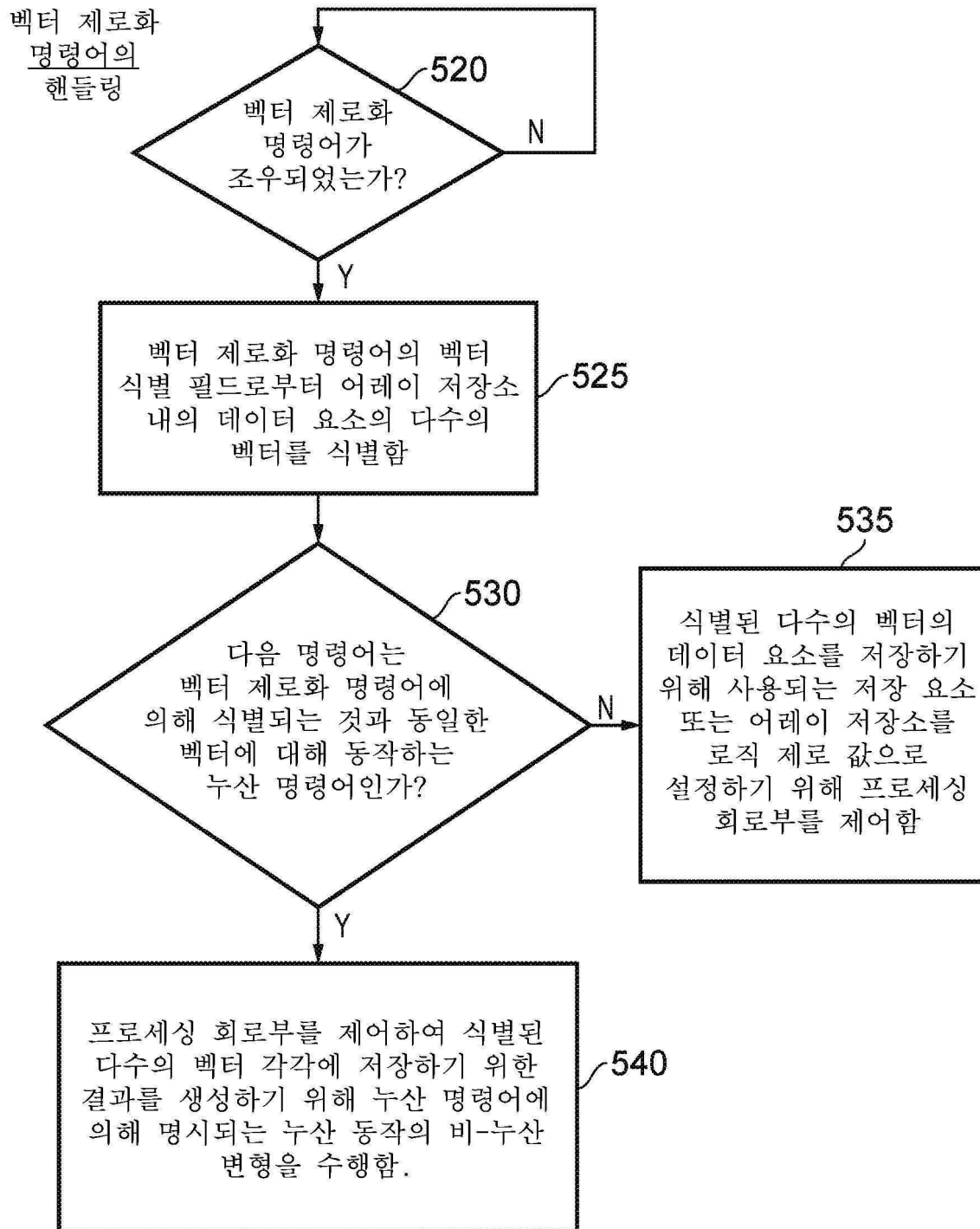
도면10

어레이 저장소에 대해 동작하는 대안적인 예시적인 명령어 시퀀스

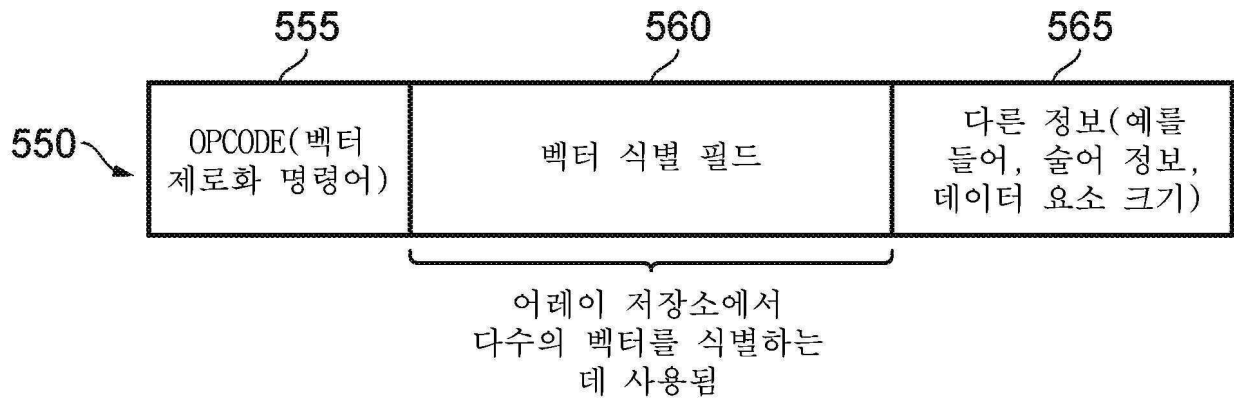
ZAR 2, ZAR 3 및 ZAR 4가 제로로 초기화되었다고 가정함

- 1) 데이터 프로세싱 명령어(ZAR 2, ZAR 3 및 ZAR 4 내로 누산함)
- 2) 데이터 프로세싱 명령어(ZAR 2, ZAR 3 및 ZAR 4 내로 누산함)
- 3) 이동 및 제로화 명령어(ZAR 2, ZAR 3, ZAR 4에서 Z_i , Z_{i+1} , Z_{i+2} 로) -
모든 누산 결과를 벡터 레지스터 파일 내의
벡터 레지스터로 이동시키고 ZAR 2,
ZAR 3 및 ZAR 4)를 소거함
- 4) 데이터 프로세싱 명령어(ZAR 2, ZAR 3 및 ZAR 4 내로 누산함)
실행은 효과적으로, 정의된 동작의 비-누산 변형이
수행되게 하고, 결과가 생성되고 ZAR 2,
ZAR 3 및 ZAR 4) 각각에 저장됨
- 5) 데이터 프로세싱 명령어(ZAR 2, ZAR 3 및 ZAR 4 내로 누산함)

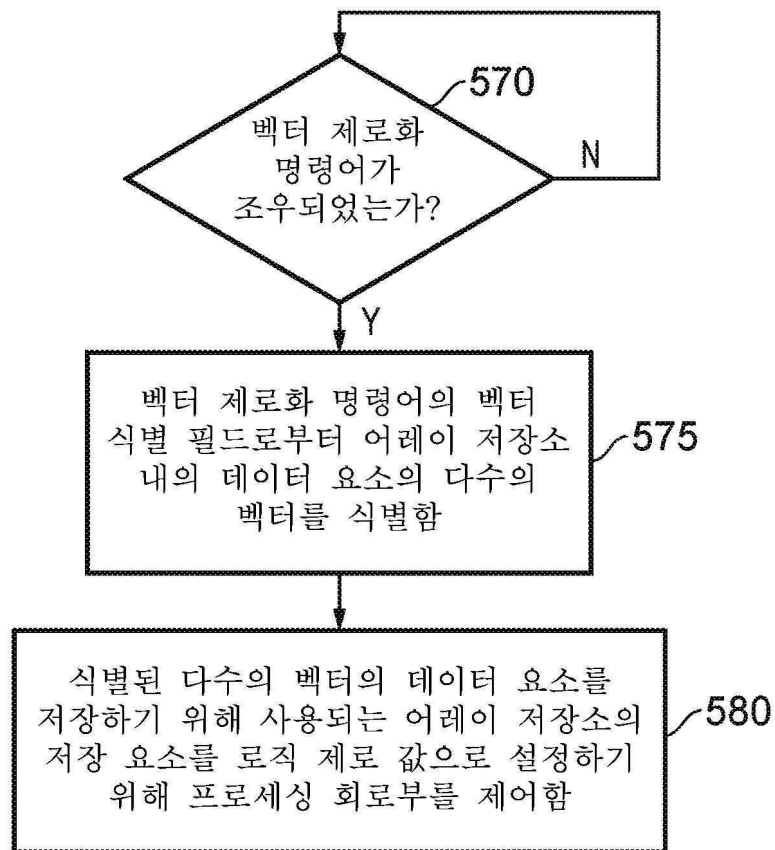
도면11



도면12



도면13



도면14

시뮬레이터 구현

