(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년10월10일
(11) 등록번호 10-2586173
(24) 등록일자 2023년09월27일

(51) 국제특허분류(Int. Cl.)
G06N 3/08 (2023.01) *G06N 3/04* (2023.01)
(52) CPC특허분류
G06N 3/08 (2023.01)
G06N 3/04 (2023.01)
(21) 출원번호 10-2017-0143240
(22) 출원일자 2017년10월31일
심사청구일자 2020년10월27일
(65) 공개번호 10-2019-0048347
(43) 공개일자 2019년05월09일
(56) 선행기술조사문헌
US20170103313 A1*
US20190114499 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
김경훈
경기도 수원시 영통구 동탄원천로881번길 35, 503동 1503호(매탄동, 주공그린빌)
박영환
경기도 용인시 수지구 수지로 166, 107동 306호(풍덕천동, 정자뜰마을 태영 데시앙1차 아파트)
(뒷면에 계속)
(74) 대리인
정홍식, 김태헌

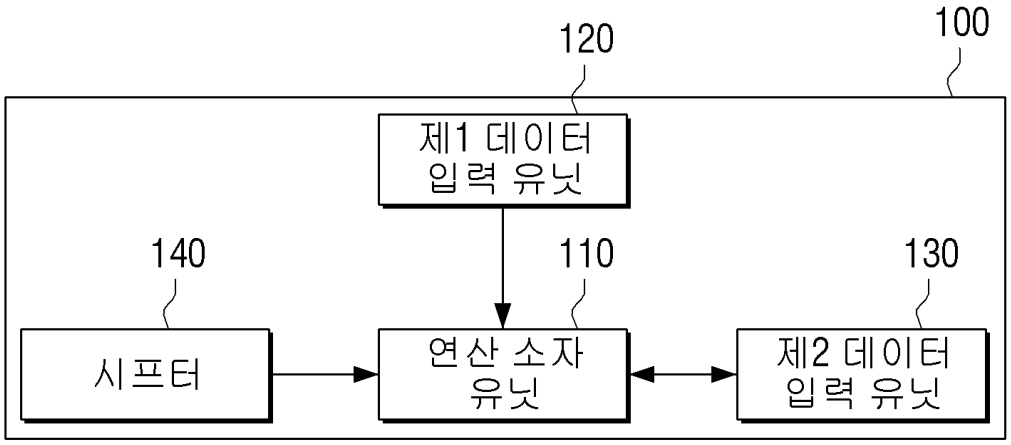
전체 청구항 수 : 총 20 항 심사관 : 송근배

(54) 발명의 명칭 **프로세서 및 그 제어 방법**

(57) 요 약

딥러닝(Deep Learning)을 수행하는 프로세서가 개시된다. 본 딥러닝을 수행하는 프로세서는 매트릭스 형태로 배열된 복수의 연산 소자(Processing Element)를 포함하는 연산 소자 유닛, 연산 소자 유닛의 제1 로우(row)에 포함된 복수의 제1 연산 소자 각각에 연결된 제1 데이터 입력 유닛, 연산 소자 유닛의 복수의 연산 소자 각각에 연결된 제2 데이터 입력 유닛 및 연산 소자 유닛의 복수의 로우 각각에 대응되는 복수의 레지스터 유닛을 포함하며, 복수의 레지스터 유닛 각각은 대응되는 연산 소자 유닛의 로우에 포함된 복수의 연산 소자와 연결된 시프터(Shifter)를 포함하며, 프로세서는 시프터를 통해 입력된 연산 명령어에 기초하여 제1 데이터 입력 유닛을 통해 입력된 데이터 및 제2 데이터 입력 유닛을 통해 입력된 데이터를 연산하도록 연산 소자 유닛을 제어한다.

대 표 도 - 도2a



(72) 발명자

서동관

경기도 용인시 기흥구 흥덕중앙로105번길 41, 1105동 1202호(영덕동, 흥덕마을 11단지 경남아너스빌)

프라사드 케샤바

경기도 수원시 영통구 효원로 363, 119동 2301호(매탄동, 매탄 위브 하늘채)

김석진

서울특별시 마포구 토정로31길 23, 102동 1501호(용강동, 래미안 마포 리버웰)

조한수

경기도 화성시 영통로50번길 14, 202동 204호(반월동, 반달마을두산위브아파트)

김현중

경기도 수원시 영통구 도청로 65, 5407동 2604호(이의동, 자연앤 힐스테이트)

명 세 서

청구범위

청구항 1

딥러닝(Deep Learning)을 수행하는 프로세서에 있어서,

매트릭스 형태로 배열된 복수의 연산 소자(Processing Element)를 포함하는 연산 소자 유닛;

상기 연산 소자 유닛의 제1 로우(row)에 포함된 복수의 제1 연산 소자 각각에 연결된 제1 데이터 입력 유닛;

상기 연산 소자 유닛의 상기 복수의 연산 소자 각각에 연결된 제2 데이터 입력 유닛; 및

상기 연산 소자 유닛의 복수의 로우 각각에 대응되는 복수의 레지스터 유닛을 포함하며, 상기 복수의 레지스터 유닛 각각은 대응되는 상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자와 연결된 시프터(Shifter);를 포함하며,

상기 프로세서는,

상기 시프터를 통해 입력된 연산 명령어에 기초하여 상기 제1 데이터 입력 유닛을 통해 입력된 데이터 및 상기 제2 데이터 입력 유닛을 통해 입력된 데이터를 연산하도록 상기 연산 소자 유닛을 제어하며,

상기 제2 데이터 입력 유닛은,

싸이클 별로 상기 연산 소자 유닛의 복수의 로우 중 하나의 로우에 포함된 연산 소자들로 대상 데이터의 일부를 입력하고, 상기 싸이클이 변경되면 입력 로우를 변경하는, 프로세서.

청구항 2

제1항에 있어서,

상기 프로세서는,

제1 싸이클에서, 상기 복수의 제1 연산 소자 각각에 필터에 포함된 복수의 엘리먼트 중 제1 엘리먼트를 입력하도록 상기 제1 데이터 입력 유닛을 제어하고,

상기 복수의 제1 연산 소자 각각에 상기 대상 데이터 중 제1 데이터를 입력하도록 상기 제2 데이터 입력 유닛을 제어하며,

상기 시프터를 통해 입력된 제1 연산 명령어에 기초하여 상기 제1 엘리먼트 및 상기 제1 데이터를 연산하도록 상기 연산 소자 유닛을 제어하는, 프로세서.

청구항 3

제2항에 있어서,

상기 프로세서는,

상기 제1 싸이클 다음의 제2 싸이클에서,

상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하도록 상기 연산 소자 유닛을 제어하고,

상기 복수의 제1 연산 소자 각각에 상기 필터에 포함된 상기 복수의 엘리먼트 중 제2 엘리먼트를 입력하도록 상기 제1 데이터 입력 유닛을 제어하며,

상기 시프터를 통해 입력된 제2 연산 명령어에 기초하여 상기 제2 엘리먼트 및 상기 제1 데이터를 연산하도록 상기 연산 소자 유닛을 제어하는, 프로세서.

청구항 4

제2항에 있어서,

상기 프로세서는,

상기 제1 싸이클 다음의 제2 싸이클에서,

상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하도록 상기 연산 소자 유닛을 제어하고,

상기 복수의 제2 연산 소자 각각에 상기 대상 데이터 중 제2 데이터를 입력하도록 상기 제2 데이터 입력 유닛을 제어하며,

상기 시프터를 통해 입력된 상기 제1 연산 명령어에 기초하여 상기 제1 로우에서 시프트된 상기 제1 엘리먼트 및 상기 제2 데이터를 연산하도록 상기 연산 소자 유닛을 제어하는, 프로세서.

청구항 5

제4항에 있어서,

상기 프로세서는,

상기 제1 싸이클에서, 상기 제1 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하고, 상기 제1 연산 명령어를 상기 제1 로우에 대응되는 제1 레지스터 유닛으로부터 상기 제2 로우에 대응되는 제2 레지스터 유닛으로 시프트하도록 상기 시프터를 제어하며,

상기 제2 싸이클에서, 상기 시프트된 제1 연산 명령어를 상기 복수의 제2 연산 소자 각각에 입력하고, 제2 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하며, 상기 제1 연산 명령어를 상기 제2 레지스터 유닛으로부터 상기 연산 소자 유닛의 제3 로우에 대응되는 제3 레지스터 유닛으로 시프트하고, 상기 제2 연산 명령어를 상기 제1 레지스터 유닛으로부터 상기 제2 레지스터 유닛으로 시프트하도록 상기 시프터를 제어하는, 프로세서.

청구항 6

제2항에 있어서,

상기 프로세서는,

상기 제1 엘리먼트 및 상기 제1 데이터를 연산한 연산 데이터를 인접한 연산 소자로 전달하고, 상기 전달된 연산 데이터를 상기 연산 소자에서 연산된 데이터와 어큐뮬레이션(accumulation)하도록 상기 연산 소자 유닛을 제어하는, 프로세서.

청구항 7

제1항에 있어서,

상기 제2 데이터 입력 유닛은,

상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자의 개수와 동일한 개수의 데이터 패스(path)를 통해 외부 메모리와 연결되는, 프로세서.

청구항 8

제1항에 있어서,

연산 명령어 레지스터 유닛; 및

상기 연산 명령어 레지스터 유닛 및 상기 시프터 사이에 연결된 멀티플렉서;를 더 포함하며,

상기 프로세서는,

상기 연산 명령어 레지스터 유닛에 저장된 복수의 연산 명령어 세트 중 하나를 상기 시프터로 제공하도록 상기 멀티플렉서를 제어하며,

상기 복수의 연산 명령어 세트 각각은, 상기 연산 소자 유닛의 복수의 로우 각각에 입력될 복수의 연산 명령어를 포함하는, 프로세서.

청구항 9

제8항에 있어서,

상기 프로세서는,

상기 시프터로 입력된 연산 명령어 세트를 상기 복수의 레지스터 유닛에 저장하고, 상기 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하며, 상기 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어를 입력하고, 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어 각각을 상기 연산 소자 유닛의 대응되는 로우 각각으로 입력하도록 상기 시프터를 제어하는, 프로세서.

청구항 10

제9항에 있어서,

상기 프로세서는,

상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어를 상기 연산 명령어 레지스터 유닛으로 제공하도록 상기 시프터를 제어하며,

상기 연산 명령어 레지스터 유닛에 저장된 상기 복수의 연산 명령어 세트 중 상기 시프터로 입력된 연산 명령어 세트를 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어로 업데이트하도록 상기 연산 명령어 레지스터 유닛을 제어하는, 프로세서.

청구항 11

딥러닝(Deep Learning)을 수행하는 프로세서의 제어 방법에 있어서,

상기 프로세서는,

매트릭스 형태로 배열된 복수의 연산 소자(Processing Element)를 포함하는 연산 소자 유닛, 상기 연산 소자 유닛의 제1 로우(row)에 포함된 복수의 제1 연산 소자 각각에 연결된 제1 데이터 입력 유닛, 상기 연산 소자 유닛의 상기 복수의 연산 소자 각각에 연결된 제2 데이터 입력 유닛 및 상기 연산 소자 유닛의 복수의 로우 각각에 대응되는 복수의 레지스터 유닛을 포함하며, 상기 복수의 레지스터 유닛 각각은 대응되는 상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자와 연결된 시프터(Shifter)를 포함하며,

상기 시프터를 통해 연산 명령어를 상기 연산 소자 유닛으로 입력하고, 상기 제1 데이터 입력 유닛을 통해 상기 연산 소자 유닛으로 데이터를 입력하며, 상기 제2 데이터 입력 유닛을 통해 상기 연산 소자 유닛으로 데이터를 입력하는 단계; 및

상기 연산 소자 유닛이 상기 연산 명령어에 기초하여 상기 제1 데이터 입력 유닛을 통해 입력된 데이터 및 상기 제2 데이터 입력 유닛을 통해 입력된 데이터를 연산하는 단계;를 포함하며,

상기 제2 데이터 입력 유닛은,

싸이클 별로 상기 연산 소자 유닛의 복수의 로우 중 하나의 로우에 포함된 연산 소자들로 대상 데이터의 일부를 입력하고, 상기 싸이클이 변경되면 입력 로우를 변경하는, 제어 방법.

청구항 12

제11항에 있어서,

상기 연산하는 단계는,

제1 싸이클에서, 상기 제1 데이터 입력 유닛을 통해 상기 복수의 제1 연산 소자 각각에 필터에 포함된 복수의 엘리먼트 중 제1 엘리먼트를 입력하는 단계;

상기 제2 데이터 입력 유닛을 통해 상기 복수의 제1 연산 소자 각각에 상기 대상 데이터 중 제1 데이터를 입력하는 단계; 및

상기 연산 소자 유닛이 상기 시프터를 통해 입력된 제1 연산 명령어에 기초하여 상기 제1 엘리먼트 및 상기 제1

데이터를 연산하는 단계;를 포함하는, 제어 방법.

청구항 13

제12항에 있어서,

상기 연산하는 단계는,

상기 제1 싸이클 다음의 제2 싸이클에서,

상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하는 단계;

상기 제1 데이터 입력 유닛을 통해 상기 복수의 제1 연산 소자 각각에 상기 필터에 포함된 상기 복수의 엘리먼트 중 제2 엘리먼트를 입력하는 단계; 및

상기 연산 소자 유닛이 상기 시프터를 통해 입력된 제2 연산 명령어에 기초하여 상기 제2 엘리먼트 및 상기 제1 데이터를 연산하는 단계;를 더 포함하는, 제어 방법.

청구항 14

제12항에 있어서,

상기 연산하는 단계는,

상기 제1 싸이클 다음의 제2 싸이클에서,

상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하는 단계;

상기 제2 데이터 입력 유닛을 통해 상기 복수의 제2 연산 소자 각각에 상기 대상 데이터 중 제2 데이터를 입력하는 단계; 및

상기 연산 소자 유닛이 상기 시프터를 통해 입력된 상기 제1 연산 명령어에 기초하여 상기 제1 로우에서 시프트된 상기 제1 엘리먼트 및 상기 제2 데이터를 연산하는 단계;를 더 포함하는, 제어 방법.

청구항 15

제14항에 있어서,

상기 연산하는 단계는,

상기 제1 싸이클에서, 상기 제1 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하고, 상기 제1 연산 명령어를 상기 제1 로우에 대응되는 제1 레지스터 유닛으로부터 상기 제2 로우에 대응되는 제2 레지스터 유닛으로 시프트하는 단계; 및

상기 제2 싸이클에서, 상기 시프트된 제1 연산 명령어를 상기 복수의 제2 연산 소자 각각에 입력하고, 제2 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하며, 상기 제1 연산 명령어를 상기 제2 레지스터 유닛으로부터 상기 연산 소자 유닛의 제3 로우에 대응되는 제3 레지스터 유닛으로 시프트하고, 상기 제2 연산 명령어를 상기 제1 레지스터 유닛으로부터 상기 제2 레지스터 유닛으로 시프트하는 단계;를 더 포함하는, 제어 방법.

청구항 16

제12항에 있어서,

상기 연산하는 단계는,

상기 제1 엘리먼트 및 상기 제1 데이터를 연산한 연산 데이터를 인접한 연산 소자로 전달하는 단계; 및

상기 전달된 연산 데이터를 상기 연산 소자에서 연산된 데이터와 어큐뮬레이션(accumulation)하는 단계;를 더 포함하는, 제어 방법.

청구항 17

제11항에 있어서,

상기 제2 데이터 입력 유닛은,

상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자의 개수와 동일한 개수의 데이터 패스(path)를 통해 외부 메모리와 연결되는, 제어 방법.

청구항 18

제11항에 있어서,

상기 프로세서는,

연산 명령어 레지스터 유닛; 및

상기 연산 명령어 레지스터 유닛 및 상기 시프터 사이에 연결된 멀티플렉서;를 더 포함하며,

상기 제어 방법은,

상기 멀티플렉서에 의해 상기 연산 명령어 레지스터 유닛에 저장된 복수의 연산 명령어 세트 중 하나를 상기 시프터로 제공하는 단계;를 더 포함하며,

상기 복수의 연산 명령어 세트 각각은, 상기 연산 소자 유닛의 복수의 로우 각각에 입력될 복수의 연산 명령어를 포함하는, 제어 방법.

청구항 19

제18항에 있어서,

상기 시프터로 입력된 연산 명령어 세트를 상기 복수의 레지스터 유닛에 저장하는 단계;

상기 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하는 단계; 및

상기 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어를 입력하는 단계;를 더 포함하며,

상기 연산 명령어를 상기 연산 소자 유닛으로 입력하는 단계는,

상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어 각각을 상기 연산 소자 유닛의 대응되는 로우 각각으로 입력하는, 제어 방법.

청구항 20

제19항에 있어서,

상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어를 상기 연산 명령어 레지스터 유닛으로 제공하는 단계; 및

상기 연산 명령어 레지스터 유닛에 저장된 상기 복수의 연산 명령어 세트 중 상기 시프터로 입력된 연산 명령어 세트를 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어로 업데이트하는 단계;를 더 포함하는, 제어 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 프로세서 및 그 제어 방법에 대한 것으로, 더욱 상세하게는 딥러닝(Deep Learning)을 수행하는 프로세서 및 그 제어 방법에 대한 것이다.

배 경 기 술

[0002] 머신 러닝(Machine Learning)은 인공 지능의 한 분야로서, 컴퓨터에 데이터를 입력하여 학습하게 함으로써 새로운 지식을 생성하는 기술을 의미한다. 특히, 머신 러닝 기술의 하나인 인공 신경망 분야에서 두드러진 발전이

이루어졌으며, 그 결과로서 딥러닝(Deep Learning)이 탄생하였다.

[0003] 딥러닝은 인공 신경망에 기반을 둔 머신 러닝 기술의 한 종류로, 인공 신경망이 다층 구조로 설계되어 깊어지더라도 학습을 위한 데이터들을 비지도 학습(Unsupervised Learning) 전처리함으로써 학습 효율을 향상시킬 수 있다. 특히, 딥러닝은 인터넷에 의한 빅데이터 및 이를 처리하기 위한 컴퓨팅 능력의 향상으로 최근 비약적인 발전을 보이고 있다.

[0004] 또한, 빅데이터의 처리를 위한 컴퓨팅 능력의 향상을 위해, 다수의 프로세서를 이용하여 다수의 신경 네트워크를 구성한 심층신경네트워크(Deep Neural Network, DNN)를 구현할 수도 있다. 이때, DNN의 구현에 있어 각 프로세서의 확장성(scalability)은 매우 중요한 요소일 수 있다. 프로세서의 확장성에 대하여는 도 1a 내지 도 1d를 통해 설명한다.

[0005] 도 1a는 종래 기술에 따른 프로세서의 구조를 간략하게 나타낸 도면이다. 프로세서는 복수의 연산 소자(Processing Element)를 포함하며, 인접한 연산 소자 간에는 데이터의 공유가 가능하다.

[0006] 연산 소자 각각은 기본적으로 곱셈기(multiplier) 및 산술 논리 연산 장치(Arithmetic Logic Unit, ALU)를 포함하며, ALU는 적어도 하나 이상의 가산기(adder)를 포함할 수 있다. 연산 소자는 곱셈기 및 ALU를 이용하여 사칙 연산을 수행할 수 있다.

[0007] 외부 메모리는 복수의 메모리 뱅크로 구분될 수 있으며, 복수의 연산 소자 각각으로 데이터를 제공할 수 있다.

[0008] 도 1b는 종래의 프로세서 구조를 이용한 컨볼루션(convolution) 연산을 설명하기 위한 도면이다.

[0009] 컨볼루션 연산은 커널(kernel) 데이터를 이용하여 이미지(image) 데이터를 변환하는 연산으로, 먼저 이미지의 좌측 상단의 9개 픽셀에 대하여 커널 데이터를 이용한 컨볼루션 연산은 하기와 같다.

[0010] a1 + b2 + c3 + d4 + e5 + f6 + g7 + h8 + i9

[0011] 이상의 연산을 통해 Accumulation의 픽셀 데이터 하나가 산출될 수 있다. 커널 데이터를 픽셀 단위로 이동해가며 이상과 같은 연산을 이미지 데이터 전체에 수행하는 경우, Accumulation이 완성된다(Accumulation의 외곽 픽셀은 별도의 계산을 통해 도출되며, 이에 대한 구체적인 설명은 생략한다).

[0012] 프로세서는 컨볼루션 연산을 수행하기 위해 복수의 싸이클 동안 연산을 수행한다. 먼저, 제1 싸이클에서 이미지 데이터의 일부는 복수의 연산 소자로 입력될 수 있다. 예를 들어, 복수의 연산 소자가 16개이므로, 이미지 데이터의 1부터 9 및 이에 인접한 7개의 픽셀 데이터가 복수의 연산 소자로 입력될 수 있다. 또한, 커널 데이터 중 a가 복수의 연산 소자로 입력될 수 있다. 그리고, 연산 명령어(Instruction)가 복수의 연산 소자로 입력될 수 있다. 복수의 연산 소자는 입력된 연산 명령어에 기초하여 a와 각각의 픽셀 데이터를 연산할 수 있다. 연산 결과를 인접한 연산 소자로 이동된다. 도 1b에서는 커널 데이터 중 a 및 연산 명령어가 일부의 연산 소자로 입력되는 것으로 도시되었으나, 이는 도면이 복잡해지는 것을 방지하기 위한 것이며, 실제로 커널 데이터 중 a 및 연산 명령어는 모든 연산 소자로 입력될 수 있다.

[0013] 제2 싸이클에서는 이미지 데이터는 그대로 이용되고, 커널 데이터 중 b 및 연산 명령어가 복수의 연산 소자로 입력될 수 있다. 복수의 연산 소자는 입력된 연산 명령어에 기초하여 b와 각각의 픽셀 데이터를 연산하고, 제1 싸이클의 연산 결과와 어큐뮬레이션(accumulation)할 수 있다.

[0014] 제2 싸이클과 유사한 동작이 커널 데이터의 개수인 제9 싸이클까지 반복되어 Accumulation의 픽셀 데이터 일부가 산출될 수 있다.

[0015] 이상의 연산에서 각 싸이클 동안의 데이터 이동량을 고려해보면, 제1 싸이클에서는 외부 메모리에 저장된 이미지 데이터로부터 16개의 픽셀 데이터가 복수의 연산 소자로 이동되나, 제2 싸이클 내지 제9 싸이클까지는 외부 메모리로부터 프로세서로의 이미지 데이터 이동이 없다. 즉, 데이터 이동량은 도 1c와 같이 나타낼 수 있다. 즉, 외부 메모리로부터 프로세서로의 데이터 패스(path)는 최소한 복수의 연산 소자에 동시에 데이터를 입력할 수 있는 정도로 형성될 필요가 있다.

[0016] 만약, 다수의 프로세서를 이용하여 DNN을 구현하는 경우라면, 도 1d와 같이 나타낼 수 있고, 프로세서와 외부 메모리 간의 데이터 패스는 기하급수적으로 증가할 수도 있다. 여기서, 2D Array 각각은 프로세서를 나타낸다. 즉, 종래의 프로세서를 이용하는 경우 단순 병렬 연결을 통해서는 DNN 구현이 어려울 수 있다.

[0017] 그에 따라, 확장성을 고려한 프로세서가 개발될 필요성이 대두되었다.

발명의 내용

해결하려는 과제

[0018] 본 발명은 상술한 필요성에 따른 것으로, 본 발명의 목적은 외부 메모리로부터의 피크 데이터 이동량을 감소시키기 위한 프로세서 및 그 제어 방법을 제공함에 있다.

과제의 해결 수단

[0019] 이상과 같은 목적을 달성하기 위한 본 발명의 일 실시 예에 따르면, 딥러닝(Deep Learning)을 수행하는 프로세서는 매트릭스 형태로 배열된 복수의 연산 소자(Processing Element)를 포함하는 연산 소자 유닛, 상기 연산 소자 유닛의 제1 로우(row)에 포함된 복수의 제1 연산 소자 각각에 연결된 제1 데이터 입력 유닛, 상기 연산 소자 유닛의 상기 복수의 연산 소자 각각에 연결된 제2 데이터 입력 유닛 및 상기 연산 소자 유닛의 복수의 로우 각각에 대응되는 복수의 레지스터 유닛을 포함하며, 상기 복수의 레지스터 유닛 각각은 대응되는 상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자와 연결된 시프터(Shifter)를 포함하며, 상기 프로세서는 상기 시프터를 통해 입력된 연산 명령어에 기초하여 상기 제1 데이터 입력 유닛을 통해 입력된 데이터 및 상기 제2 데이터 입력 유닛을 통해 입력된 데이터를 연산하도록 상기 연산 소자 유닛을 제어한다.

[0020] 또한, 상기 프로세서는 제1 싸이클에서, 상기 복수의 제1 연산 소자 각각에 필터에 포함된 복수의 엘리먼트 중 제1 엘리먼트를 입력하도록 상기 제1 데이터 입력 유닛을 제어하고, 상기 복수의 제1 연산 소자 각각에 대상 데이터 중 제1 데이터를 입력하도록 상기 제2 데이터 입력 유닛을 제어하며, 상기 시프터를 통해 입력된 제1 연산 명령어에 기초하여 상기 제1 엘리먼트 및 상기 제1 데이터를 연산하도록 상기 연산 소자 유닛을 제어할 수 있다.

[0021] 그리고, 상기 프로세서는 상기 제1 싸이클 다음의 제2 싸이클에서, 상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하도록 상기 연산 소자 유닛을 제어하고, 상기 복수의 제1 연산 소자 각각에 상기 필터에 포함된 상기 복수의 엘리먼트 중 제2 엘리먼트를 입력하도록 상기 제1 데이터 입력 유닛을 제어하며, 상기 시프터를 통해 입력된 제2 연산 명령어에 기초하여 상기 제2 엘리먼트 및 상기 제1 데이터를 연산하도록 상기 연산 소자 유닛을 제어할 수 있다.

[0022] 또한, 상기 프로세서는 상기 제1 싸이클 다음의 제2 싸이클에서, 상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하도록 상기 연산 소자 유닛을 제어하고, 상기 복수의 제2 연산 소자 각각에 상기 대상 데이터 중 제2 데이터를 입력하도록 상기 제2 데이터 입력 유닛을 제어하며, 상기 시프터를 통해 입력된 상기 제1 연산 명령어에 기초하여 상기 제1 로우에서 시프트된 상기 제1 엘리먼트 및 상기 제2 데이터를 연산하도록 상기 연산 소자 유닛을 제어할 수 있다.

[0023] 그리고, 상기 프로세서는 상기 제1 싸이클에서, 상기 제1 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하고, 상기 제1 연산 명령어를 상기 제1 로우에 대응되는 제1 레지스터 유닛으로부터 상기 제2 로우에 대응되는 제2 레지스터 유닛으로 시프트하도록 상기 시프터를 제어하며, 상기 제2 싸이클에서, 상기 시프트된 제1 연산 명령어를 상기 복수의 제2 연산 소자 각각에 입력하고, 제2 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하며, 상기 제1 연산 명령어를 상기 제2 레지스터 유닛으로부터 상기 연산 소자 유닛의 제3 로우에 대응되는 제3 레지스터 유닛으로 시프트하고, 상기 제2 연산 명령어를 상기 제1 레지스터 유닛으로부터 상기 제2 레지스터 유닛으로 시프트하도록 상기 시프터를 제어할 수 있다.

[0024] 또한, 상기 프로세서는 상기 제1 엘리먼트 및 상기 제1 데이터를 연산한 연산 데이터를 인접한 연산 소자로 전달하고, 상기 전달된 연산 데이터를 상기 연산 소자에서 연산된 데이터와 어큐뮬레이션(accumulation)하도록 상기 연산 소자 유닛을 제어할 수 있다.

[0025] 그리고, 상기 제2 데이터 입력 유닛은 상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자의 개수와 동일한 개수의 데이터 패스(path)를 통해 외부 메모리와 연결될 수 있다.

[0026] 또한, 연산 명령어 레지스터 유닛 및 상기 연산 명령어 레지스터 유닛 및 상기 시프터 사이에 연결된 멀티플렉서를 더 포함하며, 상기 프로세서는 상기 연산 명령어 레지스터 유닛에 저장된 복수의 연산 명령어 세트 중 하나를 상기 시프터로 제공하도록 상기 멀티플렉서를 제어하며, 상기 복수의 연산 명령어 세트 각각은 상기 연산 소자 유닛의 복수의 로우 각각에 입력될 복수의 연산 명령어를 포함할 수 있다.

[0027] 그리고, 상기 프로세서는 상기 시프터로 입력된 연산 명령어 세트를 상기 복수의 레지스터 유닛에 저장하고, 상

기 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하며, 상기 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어를 입력하고, 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어 각각을 상기 연산 소자 유닛의 대응되는 로우 각각으로 입력하도록 상기 시프터를 제어할 수 있다.

[0028] 또한, 상기 프로세서는 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어를 상기 연산 명령어 레지스터 유닛으로 제공하도록 상기 시프터를 제어하며, 상기 연산 명령어 레지스터 유닛에 저장된 상기 복수의 연산 명령어 세트 중 상기 시프터로 입력된 연산 명령어 세트를 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어로 업데이트하도록 상기 연산 명령어 레지스터 유닛을 제어할 수 있다.

[0029] 한편, 본 발명의 일 실시 예에 따르면, 딥러닝(Deep Learning)을 수행하는 프로세서의 제어 방법에 있어서, 상기 프로세서는 매트릭스 형태로 배열된 복수의 연산 소자(Processing Element)를 포함하는 연산 소자 유닛, 상기 연산 소자 유닛의 제1 로우(row)에 포함된 복수의 제1 연산 소자 각각에 연결된 제1 데이터 입력 유닛, 상기 연산 소자 유닛의 상기 복수의 연산 소자 각각에 연결된 제2 데이터 입력 유닛 및 상기 연산 소자 유닛의 복수의 로우 각각에 대응되는 복수의 레지스터 유닛을 포함하며, 상기 복수의 레지스터 유닛 각각은 대응되는 상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자와 연결된 시프터(Shifter)를 포함하며, 상기 시프터를 통해 연산 명령어를 상기 연산 소자 유닛으로 입력하고, 상기 제1 데이터 입력 유닛을 통해 상기 연산 소자 유닛으로 데이터를 입력하며, 상기 제2 데이터 입력 유닛을 통해 상기 연산 소자 유닛으로 데이터를 입력하는 단계 및 상기 연산 소자 유닛이 상기 연산 명령어에 기초하여 상기 제1 데이터 입력 유닛을 통해 입력된 데이터 및 상기 제2 데이터 입력 유닛을 통해 입력된 데이터를 연산하는 단계를 포함한다.

[0030] 또한, 상기 연산하는 단계는 제1 싸이클에서, 상기 제1 데이터 입력 유닛을 통해 상기 복수의 제1 연산 소자 각각에 필터에 포함된 복수의 엘리먼트 중 제1 엘리먼트를 입력하는 단계, 상기 제2 데이터 입력 유닛을 통해 상기 복수의 제1 연산 소자 각각에 대상 데이터 중 제1 데이터를 입력하는 단계 및 상기 연산 소자 유닛이 상기 시프터를 통해 입력된 제1 연산 명령어에 기초하여 상기 제1 엘리먼트 및 상기 제1 데이터를 연산하는 단계를 포함할 수 있다.

[0031] 그리고, 상기 연산하는 단계는 상기 제1 싸이클 다음의 제2 싸이클에서, 상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하는 단계, 상기 제1 데이터 입력 유닛을 통해 상기 복수의 제1 연산 소자 각각에 상기 필터에 포함된 상기 복수의 엘리먼트 중 제2 엘리먼트를 입력하는 단계 및 상기 연산 소자 유닛이 상기 시프터를 통해 입력된 제2 연산 명령어에 기초하여 상기 제2 엘리먼트 및 상기 제1 데이터를 연산하는 단계를 더 포함할 수 있다.

[0032] 또한, 상기 연산하는 단계는 상기 제1 싸이클 다음의 제2 싸이클에서, 상기 제1 엘리먼트를 상기 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하는 단계, 상기 제2 데이터 입력 유닛을 통해 상기 복수의 제2 연산 소자 각각에 상기 대상 데이터 중 제2 데이터를 입력하는 단계 및 상기 연산 소자 유닛이 상기 시프터를 통해 입력된 상기 제1 연산 명령어에 기초하여 상기 제1 로우에서 시프트된 상기 제1 엘리먼트 및 상기 제2 데이터를 연산하는 단계를 더 포함할 수 있다.

[0033] 그리고, 상기 연산하는 단계는 상기 제1 싸이클에서, 상기 제1 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하고, 상기 제1 연산 명령어를 상기 제1 로우에 대응되는 제1 레지스터 유닛으로부터 상기 제2 로우에 대응되는 제2 레지스터 유닛으로 시프트하는 단계 및 상기 제2 싸이클에서, 상기 시프트된 제1 연산 명령어를 상기 복수의 제2 연산 소자 각각에 입력하고, 제2 연산 명령어를 상기 복수의 제1 연산 소자 각각에 입력하며, 상기 제1 연산 명령어를 상기 제2 레지스터 유닛으로부터 상기 연산 소자 유닛의 제3 로우에 대응되는 제3 레지스터 유닛으로 시프트하고, 상기 제2 연산 명령어를 상기 제1 레지스터 유닛으로부터 상기 제2 레지스터 유닛으로 시프트하는 단계를 더 포함할 수 있다.

[0034] 또한, 상기 연산하는 단계는 상기 제1 엘리먼트 및 상기 제1 데이터를 연산한 연산 데이터를 인접한 연산 소자로 전달하는 단계 및 상기 전달된 연산 데이터를 상기 연산 소자에서 연산된 데이터와 어큐뮬레이션(accumulation)하는 단계를 더 포함할 수 있다.

[0035] 그리고, 상기 제2 데이터 입력 유닛은 상기 연산 소자 유닛의 로우에 포함된 복수의 연산 소자의 개수와 동일한 개수의 데이터 패스(path)를 통해 외부 메모리와 연결될 수 있다.

[0036] 또한, 상기 프로세서는 연산 명령어 레지스터 유닛 및 상기 연산 명령어 레지스터 유닛 및 상기 시프터 사이에

연결된 멀티플렉서를 더 포함하며, 상기 제어 방법은 상기 멀티플렉서에 의해 상기 연산 명령어 레지스터 유닛에 저장된 복수의 연산 명령어 세트 중 하나를 상기 시프터로 제공하는 단계를 더 포함하며, 상기 복수의 연산 명령어 세트 각각은 상기 연산 소자 유닛의 복수의 로우 각각에 입력될 복수의 연산 명령어를 포함할 수 있다.

[0037] 그리고, 상기 시프터로 입력된 연산 명령어 세트를 상기 복수의 레지스터 유닛에 저장하는 단계, 상기 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하는 단계 및 상기 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어를 입력하는 단계를 더 포함하며, 상기 연산 명령어를 상기 연산 소자 유닛으로 입력하는 단계는 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어 각각을 상기 연산 소자 유닛의 대응되는 로우 각각으로 입력할 수 있다.

[0038] 또한, 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어를 상기 연산 명령어 레지스터 유닛으로 제공하는 단계 및 상기 연산 명령어 레지스터 유닛에 저장된 상기 복수의 연산 명령어 세트 중 상기 시프터로 입력된 연산 명령어 세트를 상기 추가 연산 명령어 및 상기 시프트된 연산 명령어 세트에 포함된 명령어로 업데이트하는 단계를 더 포함할 수 있다.

발명의 효과

[0039] 이상과 같은 본 발명의 다양한 실시 예에 따르면, 프로세서가 외부 메모리로부터의 피크 데이터 이동량을 감소시키는 구조로 설계되고, 설계된 구조에 대응되도록 연산을 수행함에 따라 프로세서의 확장성(scalability)을 향상시킬 수 있다.

도면의 간단한 설명

[0040] 도 1a 내지 도 1d는 종래 기술의 문제점을 설명하기 위한 도면들이다.

도 2a는 본 발명의 일 실시 예에 따른 딥러닝(Deep Learning)을 수행하는 프로세서의 구성을 나타내는 블럭도이다.

도 2b는 본 발명의 일 실시 예에 따른 딥러닝을 수행하는 프로세서에서 연산 소자 유닛의 세부 구성을 나타내는 블럭도이다.

도 3a 내지 도 3e는 본 발명의 일 실시 예에 따른 컨볼루션 연산을 수행하는 방법을 설명하기 위한 도면들이다.

도 4는 본 발명의 일 실시 예에 따른 외부 메모리로부터 프로세서로의 데이터 이동량을 나타내는 도면이다.

도 5는 본 발명의 일 실시 예에 따른 딥러닝을 수행하는 프로세서의 세부 구성을 나타내는 블럭도이다.

도 6a 내지 도 6d는 본 발명의 일 실시 예에 따른 풀링(pooling) 동작을 설명하기 위한 도면들이다.

도 7은 본 발명의 일 실시 예에 따른 프로세서의 제어 방법을 설명하기 위한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0041] 이하에서, 첨부된 도면을 이용하여 본 발명의 다양한 실시 예들에 대하여 구체적으로 설명한다.

[0042] 도 2a는 본 발명의 일 실시 예에 따른 딥러닝(Deep Learning)을 수행하는 프로세서(100)의 구성을 나타내는 블럭도이다.

[0043] 도 2a에 도시된 바와 같이, 프로세서(100)는 연산 소자 유닛(110), 제1 데이터 입력 유닛(120), 제2 데이터 입력 유닛(130) 및 시프터(Shifter, 140)를 포함한다.

[0044] 연산 소자 유닛(110)은 매트릭스 형태로 배열된 복수의 연산 소자(Processing Element)를 포함할 수 있다. 인접한 연산 소자 간에는 데이터의 일방향 시프트 또는 양방향 시프트가 가능하다.

[0045] 연산 소자 각각은 기본적으로 곱셈기(multiplier) 및 산술 논리 연산 장치(Arithmetic Logic Unit, ALU)를 포함하며, ALU는 적어도 하나 이상의 가산기(adder)를 포함할 수 있다. 연산 소자는 곱셈기 및 ALU를 이용하여 사칙 연산을 수행할 수 있다. 다만, 이에 한정되는 것은 아니며, 사칙 연산 및 시프트 등과 같은 기능을 수행할 수 있다면 얼마든지 다른 구조로 형성될 수도 있다.

[0046] 연산 소자 각각은 데이터를 저장하기 위한 레지스터를 포함할 수 있다. 예를 들어, 연산 소자 각각은 특정 싸이클에서의 연산 결과를 저장하기 위한 레지스터를 포함할 수 있다. 또는, 연산 소자 각각은 특정 싸이클에서의

연산 결과를 인접한 연산 소자로 시프트하고, 인접한 연산 소자로부터 시프트된 연산 결과를 저장하기 위한 레지스터를 포함할 수도 있다.

[0047] 제1 데이터 입력 유닛(120)은 연산 소자 유닛(110)의 제1 로우(row)에 포함된 복수의 제1 연산 소자 각각에 연결될 수 있다. 여기서, 연산 소자 유닛(110)의 제1 로우는 연산 소자 유닛의 두 개의 최외곽 로우 중 하나일 수 있다.

[0048] 제1 데이터 입력 유닛(120)은 커널 데이터를 입력받아 연산 소자 유닛(110)의 제1 로우에 포함된 복수의 제1 연산 소자 각각에 입력할 수 있다. 예를 들어, 제1 데이터 입력 유닛(120)은 a, b, c, d의 필터 데이터를 입력받아 제1 싸이클에서 a를 복수의 제1 연산 소자 각각에 입력할 수 있다. 그리고, 제1 데이터 입력 유닛(120)은 제2 싸이클에서 b를 복수의 제1 연산 소자 각각에 입력하고, 싸이클이 변경되면 c, d를 순차적으로 복수의 제1 연산 소자 각각에 입력할 수 있다. 한편, 필터 데이터는 컨볼루션 연산에 사용되는 입력 데이터 중 하나로, 커널 데이터와 그 쓰임이 동일하므로 이하에서는 커널 데이터와 혼용해서 사용한다.

[0049] 제1 데이터 입력 유닛(120)은 커널 데이터 전체를 저장하는 복수의 레지스터 유닛으로 구현될 수 있다. 여기서, 레지스터 유닛 하나는 커널 데이터에 포함된 복수의 엘리먼트 중 하나를 저장할 수 있다. 예를 들어, 레지스터 유닛 하나는 커널 데이터 a, b, c, d 중 하나만을 저장할 수 있다.

[0050] 그리고, 제1 데이터 입력 유닛(120)은 커널 데이터 전체 중 싸이클 별로 필요한 엘리먼트만을 복수의 제1 연산 소자 각각에 입력하기 위한 멀티플렉서(multiplexer)를 더 포함할 수 있다. 예를 들어, 멀티플렉서는 복수의 레지스터 유닛으로부터 a, b, c, d의 커널 데이터를 입력받고, 이중 하나만을 연산 소자 유닛(110)으로 제공할 수 있다.

[0051] 또는, 제1 데이터 입력 유닛(120)은 커널 데이터 전체 중 싸이클 별로 필요한 엘리먼트만을 저장하는 레지스터 유닛을 포함할 수도 있다. 이 경우, 제1 데이터 입력 유닛(120)은 레지스터 유닛에 저장된 데이터를 복수의 제1 연산 소자 각각에 입력할 수 있다.

[0052] 또는, 제1 데이터 입력 유닛(120)은 외부 메모리로부터 직접 커널 데이터 전체를 입력받고, 이중 하나만을 복수의 제1 연산 소자 각각에 제공하는 멀티플렉서만을 포함할 수도 있다.

[0053] 이상에서 설명한 제1 데이터 입력 유닛(120)의 내부 구조는 일 실시 예에 불과하고, 얼마든지 다른 형태로 구현될 수도 있다.

[0054] 한편, 제1 데이터 입력 유닛(120)이 연산 소자 유닛(110)의 제1 로우에 포함된 복수의 제1 연산 소자 각각에만 연결된 결과, 데이터 패스를 종래와 비교하여 감소시킬 수 있다. 예를 들어, 프로세서(100)가 5×5 형태의 복수의 연산 소자를 포함하는 경우, 도 1b의 종래 기술에 의하면 커널 데이터 중 엘리먼트 하나가 전체 25개의 복수의 연산 소자로 입력되기 위해 25개의 데이터 패스가 필요하였다. 이에 대해, 본원의 제1 데이터 입력 유닛(120)은 복수의 제1 연산 소자 각각에만 연결된 결과, 5개의 데이터 패스만이 필요하다. 그에 따라, 하드웨어 설계가 용이해지고, 집적도를 향상시킬 수 있으며, 프로세서(100)의 확장성을 향상시킬 수 있다.

[0055] 제2 데이터 입력 유닛(130)은 연산 소자 유닛(110)의 복수의 연산 소자 각각에 연결될 수 있다. 다만, 제2 데이터 입력 유닛(130)은 대상 데이터를 입력받아, 각 싸이클 별로 연산 소자 유닛(110)의 로우 단위로 대상 데이터의 일부를 입력할 수 있다.

[0056] 그에 따라, 제2 데이터 입력 유닛(130)은 연산 소자 유닛(110)의 로우에 포함된 복수의 연산 소자의 개수와 동일한 개수의 데이터 패스를 통해 외부 메모리와 연결될 수 있다. 다만, 이에 한정된 것은 아니며, 제2 데이터 입력 유닛(130)과 외부 메모리 간 형성되어 있는 데이터 패스의 개수는 연산 소자 유닛(110)의 로우에 포함된 복수의 연산 소자의 개수 이상이면 무방하다.

[0057] 제2 데이터 입력 유닛(130)은 대상 데이터 중 싸이클 별로 연산 소자 유닛(110)에 입력될 일부만을 저장하는 레지스터 유닛을 포함할 수 있다. 예를 들어, 제2 데이터 입력 유닛(130)은 연산 소자 유닛(110)의 로우에 포함된 복수의 연산 소자에 입력될 데이터를 저장할 수 있다. 제2 데이터 입력 유닛(130)은 싸이클이 변경되면 저장된 데이터를 연산 소자 유닛(110)으로 입력하고, 외부 메모리부터 새로운 데이터를 입력받을 수 있다. 이때, 프로세서(100)는 싸이클 별로 필요한 데이터를 외부 메모리로부터 독출하여 제2 데이터 입력 유닛(130)으로 입력되도록 외부 메모리를 제어할 수 있다.

[0058] 그리고, 제2 데이터 입력 유닛(130)은 레지스터 유닛에 저장된 대상 데이터 중 일부를 연산 소자 유닛(110)의

복수의 로우 중 하나로 입력하기 위한 디멀티플렉서(demultiplexer)를 더 포함할 수 있다.

[0059] 또는, 제2 데이터 입력 유닛(130)은 외부 메모리로부터 대상 데이터의 일부를 싸이클 별로 입력받고, 입력된 데이터를 연산 소자 유닛(110)의 복수의 로우 중 하나로 입력하기 위한 디멀티플렉서(demultiplexer)만을 포함할 수도 있다.

[0060] 이상에서 설명한 제2 데이터 입력 유닛(130)의 내부 구조는 일 실시 예에 불과하고, 얼마든지 다른 형태로 구현될 수도 있다.

[0061] 한편, 제2 데이터 입력 유닛(130)은 대상 데이터 중 싸이클 별로 연산 소자 유닛(110)에 입력될 일부만을 외부 메모리로부터 입력받을 수 있다. 즉, 제2 데이터 입력 유닛(130)은 대상 데이터를 한번에 연산 소자 유닛(110)으로 입력하는 것이 아니라 싸이클 별로 일부를 입력하기 때문에, 외부 메모리와 연결되는 데이터 패스를 종래와 비교하여 감소시킬 수 있다. 예를 들어, 프로세서(100)가 5×5 형태의 복수의 연산 소자를 포함하는 경우, 도 1b의 종래 기술에 의하면 제2 데이터 입력 유닛(130)은 최소 25개의 데이터 패스를 통해 외부 메모리로부터 대상 데이터를 입력받아야 한다. 이에 대해, 본원의 제2 데이터 입력 유닛(130)은 최소 5개의 데이터 패스를 통해 외부 메모리로부터 대상 데이터의 일부만을 싸이클 별로 입력받으면 된다. 그에 따라, 하드웨어 설계가 용이해지고, 집적도를 향상시킬 수 있으며, 프로세서(100)의 확장성을 향상시킬 수 있다.

[0062] 시프터(140)는 연산 소자 유닛(110)의 복수의 로우 각각에 대응되는 복수의 레지스터 유닛을 포함하며, 복수의 레지스터 유닛 각각은 대응되는 연산 소자 유닛(110)의 로우에 포함된 복수의 연산 소자와 연결될 수 있다. 여기서, 레지스터 유닛 하나는 연산 명령어 하나를 저장할 수 있다.

[0063] 프로세서(100)는 시프터(140)를 통해 입력된 연산 명령어에 기초하여 제1 데이터 입력 유닛(120)을 통해 입력된 데이터 및 제2 데이터 입력 유닛(130)을 통해 입력된 데이터를 연산하도록 연산 소자 유닛(110)을 제어할 수 있다.

[0064] 예를 들어, 프로세서(100)는 내부에 클럭을 생성하는 클럭 생성 유닛(미도시)을 더 포함할 수 있다. 클럭 생성 유닛은 프로세서(100) 내부의 각 유닛으로 생성된 클럭을 전송하며, 프로세서(100) 내부의 각 유닛은 입력된 클럭에 기초하여 데이터를 처리할 수 있다. 가령, 제1 데이터 입력 유닛(120) 및 제2 데이터 입력 유닛(130)은 클럭의 상승 엣지에 반응하여 저장하고 있던 데이터를 연산 소자 유닛(110)으로 입력하고, 외부 메모리로부터 데이터를 입력받을 수 있다. 연산 소자 유닛(110)은 클럭의 상승 엣지에 반응하여 입력된 데이터를 연산할 수 있다. 시프터(140)는 클럭의 상승 엣지에 반응하여 연산 명령어를 연산 소자 유닛(110)으로 입력하고, 연산 명령어를 인접한 레지스터 유닛으로 시프트할 수 있다. 또는, 프로세서(100) 내부의 각 유닛은 클럭의 하강 엣지에 반응할 수도 있고, 상승 엣지 및 하강 엣지 둘 다에 반응할 수도 있다. 이상의 프로세서(100)의 내부의 동기화 동작에 대하여는 종래 기술이므로 구체적인 설명은 생략한다.

[0065] 또는, 프로세서(100)는 비동기화 방식으로 구현될 수 있다. 예를 들어, 프로세서(100)는 내부에 별도의 제어부를 포함하며, 제어부는 프로세서(100) 내부의 각 유닛이 동작해야할 시점에 각 유닛으로 직접 제어 명령을 전송할 수도 있다.

[0066] 이하에서는 설명의 편의를 위하여, 프로세서(100)의 동기화 여부와 무관하게 프로세서(100)가 각 유닛을 제어하는 것으로 설명한다.

[0067] 도 2b는 본 발명의 일 실시 예에 따른 딥러닝을 수행하는 프로세서(100)에서 연산 소자 유닛(110)의 세부 구성을 나타내는 블럭도이다. 도 2b에서 연산 소자 유닛(110)은 25개의 연산 소자를 포함하는 것으로 도시되었으나, 이는 일 실시 예에 불과하고 얼마든지 다른 개수의 연산 소자로 구성될 수도 있다.

[0068] 연산 소자 유닛(110)은 5개의 로우로 구분될 수 있다. 즉, 연산 소자 유닛(110)은 제1 로우에 포함된 복수의 제1 연산 소자(110-1), 제2 로우에 포함된 복수의 제2 연산 소자(110-2), 제3 로우에 포함된 복수의 제3 연산 소자(110-3), 제4 로우에 포함된 복수의 제4 연산 소자(110-4) 및 제5 로우에 포함된 복수의 제5 연산 소자(110-5)를 포함할 수 있다.

[0069] 동일한 로우 내의 인접한 연산 소자 간에는 데이터의 양방향 시프트가 가능하다. 그리고, 동일한 로우가 아닌 인접한 연산 소자 간에는 데이터의 일방향 시프트만이 가능하다. 즉, 동일한 로우가 아닌 인접한 연산 소자 간에는 제1 로우에서 제5 로우를 향하는 방향으로만 데이터의 시프트가 가능하다. 여기서, 인접한 연산 소자는 특정 연산 소자를 기준으로 상하좌우에 배치된 연산 소자를 의미하며, 대각선에 배치된 연산 소자를 의미하는 것은 아니다.

[0070] 제1 데이터 입력 유닛(120)은 복수의 제1 연산 소자(110-1) 각각에 연결될 수 있다. 즉, 제1 데이터 입력 유닛(120)은 제1 로우에 포함된 5개의 연산 소자 각각에 연결되며, 커널 데이터에 포함된 복수의 엘리먼트 중 하나의 엘리먼트를 5개의 연산 소자 모두에 입력할 수 있다. 즉, 한 싸이클 동안 제1 데이터 입력 유닛(120)은 복수의 제1 연산 소자(110-1)에 동일한 데이터를 입력할 수 있다.

[0071] 다만, 이는 일 실시 예에 불과하고, 한 싸이클 동안 제1 데이터 입력 유닛(120)은 복수의 제1 연산 소자(110-1) 각각에 서로 다른 데이터를 입력할 수도 있다. 즉, 이상의 예는 컨볼루션 연산의 경우를 설명하기 위한 경우로, 프로세서(100)가 다른 연산을 수행하는 경우에는 얼마든지 다르게 동작할 수도 있다. 또한, 컨볼루션 연산의 경우라도 연산 순서를 변경하여 한 싸이클 동안 제1 데이터 입력 유닛(120)은 복수의 제1 연산 소자(110-1) 각각에 서로 다른 데이터를 입력할 수도 있다.

[0072] 제2 데이터 입력 유닛(130)은 연산 소자 유닛(110)의 복수의 연산 소자 각각에 연결될 수 있다. 그리고, 제1 싸이클에서 제2 데이터 입력 유닛(130)은 복수의 제1 연산 소자(110-1) 각각으로 대상 데이터의 일부를 입력할 수 있다. 이때, 복수의 제1 연산 소자(110-1) 각각으로 입력되는 데이터는 서로 다를 수 있다. 그리고, 제2 싸이클에서 제2 데이터 입력 유닛(130)은 복수의 제2 연산 소자(110-2) 각각으로 대상 데이터의 다른 일부를 입력할 수 있다.

[0073] 시프터(140)에 포함된 복수의 레지스터 유닛은 각각 연산 소자 유닛(110)의 복수의 로우 각각에 연결될 수 있다. 예를 들어, 제1 레지스터 유닛은 복수의 제1 연산 소자(110-1)에 연결되며, 제2 레지스터 유닛은 복수의 제2 연산 소자(110-2)에 연결되며, 제3 레지스터 유닛은 복수의 제3 연산 소자(110-3)에 연결되며, 제4 레지스터 유닛은 복수의 제4 연산 소자(110-4)에 연결되며, 제5 레지스터 유닛은 복수의 제5 연산 소자(110-5)에 연결될 수 있다. 각 레지스터 유닛은 연결된 연산 소자로 동일한 연산 명령어를 입력할 수 있다. 예를 들어, 제1 레지스터 유닛은 복수의 제1 연산 소자(110-1)로 하나의 연산 명령어를 입력할 수 있다.

[0074] 한편, 도 1b의 종래 기술에 의하면 연산 명령어가 모든 연산 소자로 입력될 필요가 있으며, 그에 따라 연산 명령어가 저장된 레지스터 유닛은 모든 연산 소자와 연결되어야 한다. 이에 대해, 본원의 레지스터 유닛 각각은 대응되는 로우의 복수의 연산 소자가 아닌 다른 연산 소자에 연결되지 않는다. 그에 따라, 하드웨어 설계가 용이해지고, 집적도를 향상시킬 수 있으며, 프로세서(100)의 확장성을 향상시킬 수 있다.

[0075] 이상에서는 프로세서(100)의 하드웨어적인 구조 및 그에 따른 확장성의 향상에 대하여 설명하였다. 이하에서는 본 프로세서(100)를 이용한 컨볼루션 연산 등을 수행하는 방법을 설명한다.

[0076] 도 3a 내지 도 3e는 본 발명의 일 실시 예에 따른 컨볼루션 연산을 수행하는 방법을 설명하기 위한 도면들이다. 필터 데이터(커널 데이터) 및 대상 데이터(이미지 데이터)는 도 3a와 같은 경우를 예로 들어 설명하나, 이는 일 실시 예에 불과하다.

[0077] 도 3b는 본 발명의 일 실시 예에 따른 제1 싸이클에서 프로세서(100)의 동작을 나타낸다.

[0078] 프로세서(100)는 제1 싸이클에서, 연산 소자 유닛(110)의 제1 로우에 포함된 복수의 제1 연산 소자(110-1) 각각에 필터 데이터에 포함된 복수의 엘리먼트 중 제1 엘리먼트(a)를 입력하도록 제1 데이터 입력 유닛(130)을 제어하고, 복수의 제1 연산 소자(110-1) 각각에 대상 데이터 중 제1 데이터(1, 2, 3, 4, 5)를 입력하도록 제2 데이터 입력 유닛(130)을 제어하며, 시프터(140)를 통해 입력된 제1 연산 명령어(Mul)에 기초하여 제1 엘리먼트 및 제1 데이터를 연산하도록 연산 소자 유닛(110)을 제어할 수 있다.

[0079] 프로세서(100)는 제1 싸이클에서, 제1 연산 명령어를 복수의 제1 연산 소자(110-1) 각각에 입력하고, 제1 연산 명령어를 제1 로우에 대응되는 제1 레지스터 유닛으로부터 제2 로우에 대응되는 제2 레지스터 유닛으로 시프트하도록 시프터(140)를 제어할 수 있다.

[0080] 연산 결과는 도 3c에 도시된 바와 같다. 제1 싸이클에서 복수의 제1 연산 소자(110-1) 각각은 제1 엘리먼트, 제1 데이터 및 연산 결과를 저장할 수 있다.

[0081] 프로세서(100)는 제1 싸이클 다음의 제2 싸이클에서, 제2 연산 명령어(Shift+MAC)를 복수의 제1 연산 소자(110-1) 각각에 입력하며, 제2 연산 명령어를 제1 레지스터 유닛으로부터 제2 레지스터 유닛으로 시프트하도록 시프터(140)를 제어할 수 있다.

[0082] Shift 명령어에 따라 복수의 제1 연산 소자(110-1) 각각은 제1 싸이클에서 연산한 연산 결과를 인접한 연산 소자로 시프트할 수 있다. 특히, Shift 명령어는 우측에 있는 인접한 연산 소자로 연산 결과에 따른 연산 데이터를 시프트할 것을 지시하는 명령어일 수 있다. 즉, 프로세서(100)는 제1 엘리먼트 및 제1 데이터를 연산한 연산

데이터를 인접한 연산 소자로 전달하도록 연산 소자 유닛(110)을 제어할 수 있다.

[0083] 프로세서(100)는 제2 싸이클에서, 제1 엘리먼트를 연산 소자 유닛(110)의 제2 로우에 포함된 복수의 제2 연산 소자(110-2) 각각으로 시프트하도록 연산 소자 유닛(110)을 제어하고, 복수의 제1 연산 소자(110-1) 각각에 필터 데이터에 포함된 복수의 엘리먼트 중 제2 엘리먼트(b)를 입력하도록 제1 데이터 입력 유닛(120)을 제어하며, 시프터(140)를 통해 입력된 제2 연산 명령어(Shift+MAC)에 기초하여 제2 엘리먼트 및 제1 데이터를 연산하도록 연산 소자 유닛(110)을 제어할 수 있다. 여기서, 복수의 제1 연산 소자(110-1)는 싸이클이 변경되도 제1 데이터를 저장할 수 있다.

[0084] 즉, MAC 명령어에 따라 복수의 제1 연산 소자(110-1) 각각은 제2 싸이클에서 제1 데이터 입력 유닛(120)으로부터 입력된 제2 엘리먼트(b) 및 제1 싸이클에서 제2 데이터 입력 유닛(130)으로부터 입력된 제1 데이터(1, 2, 3, 4, 5)를 연산하고, 인접한 연산 소자로부터 전달된 연산 데이터를 복수의 제1 연산 소자(110-1) 각각에서 연산된 데이터와 어큐뮬레이션(accumulation)할 수 있다.

[0085] 프로세서(100)는 제2 싸이클에서, 제1 싸이클에서 시프트된 제1 연산 명령어(mul)를 복수의 제2 연산 소자(110-2) 각각에 입력하고, 제1 연산 명령어를 제2 레지스터 유닛으로부터 연산 소자 유닛(110)의 제3 로우에 대응되는 제3 레지스터 유닛으로 시프트하도록 시프터(140)를 제어할 수 있다.

[0086] 프로세서(100)는 제2 싸이클에서, 복수의 제2 연산 소자(110-2) 각각에 대상 데이터 중 제2 데이터(6, 7, 8, 9, 10)를 입력하도록 제2 데이터 입력 유닛(130)을 제어하며, 시프터(140)를 통해 입력된 제1 연산 명령어에 기초하여 제1 로우에서 시프트된 제1 엘리먼트(a) 및 제2 데이터를 연산하도록 연산 소자 유닛(110)을 제어할 수 있다.

[0087] 연산 결과는 도 3e에 도시된 바와 같다. 제2 싸이클에서 복수의 제1 연산 소자(110-1) 각각은 제2 엘리먼트(b), 제1 데이터(1, 2, 3, 4, 5) 및 연산 결과를 저장하고, 복수의 제2 연산 소자(110-2) 각각은 제1 엘리먼트(a), 제2 데이터(6, 7, 8, 9, 10) 및 연산 결과를 저장할 수 있다.

[0088] 제2 싸이클에서는 제1 싸이클에서보다 두 배 많은 연산 소자가 연산을 수행할 수 있다. 다만, 연산 소자 유닛(110)으로 입력되는 대상 데이터의 양은 싸이클이 변경되도 동일할 수 있으며, 그에 따라 외부 메모리로부터의 프로세서(100)로 입력되는 데이터의 양을 일정하게 유지할 수 있다. 즉, 프로세서(100)는 대상 데이터를 한꺼번에 수신하지 않고 일부를 순차적으로 수신하여, 프로세서(100)와 외부 메모리 사이의 데이터 패스가 축소되어도 데이터 전송에 문제가 발생하지 않는다.

[0089] 프로세서(100)는 제2 싸이클 다음의 제3 싸이클에서, 제1 엘리먼트(a)를 연산 소자 유닛(110)의 제3 로우에 포함된 복수의 제3 연산 소자(110-3) 각각으로 시프트하고 제2 엘리먼트(b)를 복수의 제2 연산 소자(110-2) 각각으로 시프트하도록 연산 소자 유닛(110)을 제어하며, 복수의 제1 연산 소자(110-1) 각각에 필터 데이터에 포함된 복수의 엘리먼트 중 제4 엘리먼트(d)를 입력하도록 제1 데이터 입력 유닛(120)을 제어할 수 있다.

[0090] 여기서, 프로세서(100)가 제4 엘리먼트(d)를 입력하도록 제1 데이터 입력 유닛(120)을 제어하는 것은 필터 데이터의 형태가 2d 형태이기 때문이다. 구체적으로, 제3 엘리먼트(c)를 입력하는 경우는 제4 엘리먼트(d)를 입력하는 경우보다 시프트가 한 번 더 필요할 수 있기 때문에, 프로세서(100)는 효율적인 연산을 위해 제4 엘리먼트(d)를 제3 엘리먼트(c) 보다 먼저 입력할 수 있다.

[0091] 그리고, 프로세서(100)는 제3 싸이클에서, 제2 싸이클에서 시프트된 제1 연산 명령어(mul)를 복수의 제3 연산 소자(110-3) 각각에 입력하고, 제1 연산 명령어를 제3 레지스터 유닛으로부터 연산 소자 유닛(110)의 제4 로우에 대응되는 제4 레지스터 유닛으로 시프트하도록 시프터(140)를 제어할 수 있다. 또한, 프로세서(100)는 제3 싸이클에서, 제2 싸이클에서 시프트된 제2 연산 명령어(Shift+MAC)를 복수의 제2 연산 소자(110-2) 각각에 입력하며, 제2 연산 명령어를 제2 레지스터 유닛으로부터 제3 레지스터 유닛으로 시프트하도록 시프터(140)를 제어할 수 있다.

[0092] 그리고, 프로세서(100)는 제3 싸이클에서, 복수의 제3 연산 소자(110-3) 각각에 대상 데이터 중 제3 데이터(11, 12, 13, 14, 15)를 입력하도록 제2 데이터 입력 유닛(130)을 제어하며, 시프터(140)를 통해 입력된 제1 연산 명령어에 기초하여 제2 로우에서 시프트된 제1 엘리먼트(a) 및 제3 데이터를 연산하고, 시프터(140)를 통해 입력된 제2 연산 명령어에 기초하여 제1 로우에서 시프트된 제1 엘리먼트(b) 및 제2 데이터를 연산하도록 연산 소자 유닛(110)을 제어할 수 있다.

[0093] 그리고, 제2 싸이클에서 연산된 복수의 제1 연산 소자(110-1)의 연산 결과는 복수의 제2 연산 소자(110-2)로 시

프트될 수 있다. 이때, 시프트된 연산 결과와 제3 싸이클에서 연산된 복수의 제2 연산 소자(110-2)의 연산 결과는 복수의 제2 연산 소자(110-2) 내에서 서로 분리되어 저장될 수 있다. 이후, 제3 싸이클에서 연산된 복수의 제2 연산 소자(110-2)의 연산 결과는 제3 싸이클의 다음의 제4 싸이클에서 제3 연산 소자(110-3)로 시프트될 수 있다.

[0094] 프로세서(100)는 제4 싸이클에서, 제1 엘리먼트(a), 제2 엘리먼트(b) 및 제4 엘리먼트(d)를 이전 싸이클과 유사하게 시프트하도록 연산 소자 유닛(110)을 제어하며, 복수의 제1 연산 소자(110-1) 각각에 필터 데이터에 포함된 복수의 엘리먼트 중 제3 엘리먼트(c)를 입력하도록 제1 데이터 입력 유닛(120)을 제어할 수 있다.

[0095] 또한, 프로세서(100)는 시프터(140) 내의 연산 명령어들을 대응되는 로우의 복수의 연산 소자로 입력하고, 이전 싸이클과 유사하게 시프트하도록 시프터(140)를 제어할 수 있다.

[0096] 이하에서는 설명의 편의를 위하여, 복수의 제2 연산 소자(110-2)에 대하여만 설명한다. 복수의 제2 연산 소자(110-2)에는 제2 싸이클에서 연산된 복수의 제1 연산 소자(110-1)의 연산 결과 및 제3 싸이클에서 연산된 복수의 제2 연산 소자(110-2)의 연산 결과가 저장되어 있으며, 프로세서(100)는 제3 싸이클에서 연산된 복수의 제2 연산 소자(110-2)의 연산 결과를 복수의 제3 연산 소자(110-3)로 시프트하도록 연산 소자 유닛(110)을 제어할 수 있다.

[0097] 프로세서(100)는 복수의 제1 연산 소자(110-1)로부터 시프트된 제4 엘리먼트(d) 및 제2 데이터(6, 7, 8, 9, 10)를 연산하고, 연산된 결과를 제2 싸이클에서 연산된 복수의 제1 연산 소자(110-1)의 연산 결과와 어큐뮬레이션할 수 있다.

[0098] 대상 데이터의 좌측 상단과 필터 데이터로부터 연산되는 첫 번째 데이터를 예로 들면, 이상의 동작을 통해 a1 + b2 + d7가 연산 결과로서 생성될 수 있다. a1 + b2 + d7의 연산 결과는 복수의 제2 연산 소자(110-2) 중 좌측에서 두 번째 연산 소자에 의해 생성될 수 있다.

[0099] 제5 싸이클은 제2 싸이클과 유사하나, Shift 명령어에 따른 시프트의 방향이 우측이 아니라 좌측일 수 있다. 대상 데이터의 좌측 상단과 필터 데이터로부터 연산되는 첫 번째 데이터를 예로 들면, 이상의 동작을 통해 a1 + b2 + d7 + c6이 연산 결과로서 생성될 수 있다. a1 + b2 + d7 + c6의 연산 결과는 복수의 제2 연산 소자(110-2) 중 좌측에서 첫 번째 연산 소자에 의해 생성될 수 있다.

[0100] 이상과 같은 과정을 통해 프로세서(100)의 컨볼루션 연산이 가능하다.

[0101] 도 4는 본 발명의 일 실시 예에 따른 외부 메모리로부터 프로세서(100)로의 데이터 이동량을 나타내는 도면이다.

[0102] 도 4에 도시된 바와 같이, 실선 부분은 본원의 프로세서(100)에 의한 데이터 이동량을 나타내고, 점선은 도 1c의 종래 프로세서에 의한 데이터 이동량을 나타낸다.

[0103] 즉, 프로세서(100)는 연산을 순차적으로 진행하며, 그에 따라 외부 메모리로부터 프로세서(100)로의 데이터 이동량 역시 일정하게 유지할 수 있다. 이때, 본원의 프로세서(100)를 이용한 데이터 이동량은 종래 프로세서를 이용한 순간적인 데이터 이동량보다 작을 수 있으며, 그에 따라 본원의 프로세서(100)는 외부 메모리와의 데이터 패스가 종래 프로세서보다 작더라도 연산에 지장이 없을 수 있다.

[0104] 도 5는 본 발명의 일 실시 예에 따른 딥러닝을 수행하는 프로세서(100)의 세부 구성을 나타내는 블럭도이다. 도 5에 따르면, 프로세서(100)는 연산 소자 유닛(110), 제1 데이터 입력 유닛(120), 제2 데이터 입력 유닛(130), 시프터(140), 연산 명령어 레지스터 유닛(150) 및 멀티플렉서(160)를 포함한다. 도 5에 도시된 구성요소들 중 도 2a에 도시된 구성요소와 중복되는 부분에 대해서는 자세한 설명을 생략하도록 한다.

[0105] 연산 명령어 레지스터 유닛(150)은 복수의 연산 명령어 세트를 저장할 수 있다. 여기서, 복수의 연산 명령어 세트 각각은 연산 소자 유닛(110)의 복수의 로우 각각에 입력될 복수의 연산 명령어를 포함할 수 있다. 즉, 복수의 연산 명령어 세트 각각은 연산 소자 유닛(110)의 로우의 개수만큼의 연산 명령어를 포함할 수 있다.

[0106] 프로세서(100)는 멀티플렉서(160)로 복수의 연산 명령어 세트를 제공하고, 시프터(140)로부터 연산 명령어 세트를 입력받아 저장하도록 연산 명령어 레지스터 유닛(150)을 제어할 수 있다.

[0107] 멀티플렉서(160)는 연산 명령어 레지스터 유닛(150) 및 시프터(140) 사이에 연결될 수 있다.

[0108] 프로세서(100)는 연산 명령어 레지스터 유닛(150)에 저장된 복수의 연산 명령어 세트 중 하나를 시프터(140)로

제공하도록 멀티플렉서(160)를 제어할 수 있다.

[0109] 도 6a 내지 도 6d는 본 발명의 일 실시 예에 따른 풀링(pooling) 동작을 설명하기 위한 도면들이다. 풀링 동작은 도 5의 프로세서(100) 구조를 통해 동작할 수 있다. 풀링 동작은 두 개의 연산 명령어 세트의 교번적인 입력 및 연산 명령어 세트의 시프트가 중요하며, 도 6a 내지 도 6d에서는 커널 데이터 및 대상 데이터의 입력을 위한 제1 데이터 입력 유닛(120) 및 제2 데이터 입력 유닛(130)을 생략하였다.

[0110] 먼저, 풀링 동작을 시작하기 전의 연산 명령어 레지스터 유닛(150)에는 복수의 연산 명령어 세트가 저장되어 있을 수 있다.

[0111] 예를 들어, 도 6a에 도시된 바와 같이, 풀링 동작을 시작하기 전의 연산 명령어 레지스터 유닛(150)에는 제1 연산 명령어 세트 및 제2 연산 명령어 세트가 저장되어 있을 수 있다.

[0112] 그리고, 프로세서(100)는 연산 명령어 레지스터 유닛(150)에 저장된 복수의 연산 명령어 세트 중 하나를 시프터(140)로 제공하도록 멀티플렉서(160)를 제어할 수 있다.

[0113] 예를 들어, 도 6b와 같이, 프로세서(100)는 연산 명령어 레지스터 유닛(150)에 저장된 제1 연산 명령어 세트(1, 2, 3, 4, 5) 및 제2 연산 명령어 세트(A, B, C, D, E) 중 하나를 시프터(140)로 제공하도록 멀티플렉서(160)를 제어할 수 있다.

[0114] 그리고, 프로세서(100)는 시프터(140)로 입력된 연산 명령어 세트를 복수의 레지스터 유닛에 저장하고, 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하며, 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어를 입력하고, 추가 연산 명령어 및 시프트된 연산 명령어 세트에 포함된 명령어 각각을 연산 소자 유닛(110)의 대응되는 로우 각각으로 입력하도록 시프터(140)를 제어할 수 있다.

[0115] 예를 들어, 프로세서(100)는 시프터(140)로 입력된 제2 연산 명령어 세트(A, B, C, D, E)를 복수의 레지스터 유닛에 저장하고, 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하며, 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어(K)를 입력하고, 추가 연산 명령어(K) 및 시프트된 연산 명령어 세트에 포함된 명령어 각각을 연산 소자 유닛(110)의 대응되는 로우 각각으로 입력하도록 시프터(140)를 제어할 수 있다.

[0116] 여기서, 마지막 레지스터 유닛에 저장된 연산 명령어(E)는 시프트할 레지스터 유닛이 없으므로, 프로세서(100)는 마지막 레지스터 유닛에 저장된 연산 명령어(E)를 삭제할 수 있다.

[0117] 다만, 이에 한정되는 것은 아니며, 프로세서(100)는 마지막 레지스터 유닛에 저장된 연산 명령어(E)를 추가 연산 명령어 K 대신 입력할 수도 있다.

[0118] 이후, 프로세서(100)는 추가 연산 명령어 및 시프트된 연산 명령어 세트에 포함된 명령어를 연산 명령어 레지스터 유닛(150)으로 제공하도록 시프터(140)를 제어하며, 연산 명령어 레지스터 유닛(150)에 저장된 제1 연산 명령어 세트 및 제2 연산 명령어 세트 중 시프터(140)로 입력된 연산 명령어 세트를 추가 연산 명령어 및 시프트된 연산 명령어 세트에 포함된 명령어로 업데이트하도록 연산 명령어 레지스터 유닛(150)을 제어할 수 있다.

[0119] 동시에 프로세서(100)는 연산 명령어 레지스터 유닛(150)에 저장된 복수의 연산 명령어 세트 중 다른 하나를 시프터(140)로 제공하도록 멀티플렉서(160)를 제어할 수 있다.

[0120] 즉, 도 6c에 도시된 바와 같이, 프로세서(100)는 연산 명령어 레지스터 유닛(150)의 제2 연산 명령어 세트(A, B, C, D, E)를 K, A, B, C, D로 업데이트할 수 있다. 그리고, 프로세서(100)는 연산 명령어 레지스터 유닛(150)에 저장된 제1 연산 명령어 세트(1, 2, 3, 4, 5)를 시프터(140)로 제공하도록 멀티플렉서(160)를 제어할 수 있다.

[0121] 이후의 동작은 상술한 제2 연산 명령어 세트(A, B, C, D, E)가 입력된 경우와 동일하다.

[0122] 프로세서(100)는 시프터(140)로 입력된 제1 연산 명령어 세트(1, 2, 3, 4, 5)를 복수의 레지스터 유닛에 저장하고, 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하며, 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어(K)를 입력하고, 추가 연산 명령어(K) 및 시프트된 연산 명령어 세트에 포함된 명령어 각각을 연산 소자 유닛(110)의 대응되는 로우 각각으로 입력하도록 시프터(140)를 제어할 수 있다.

[0123] 마찬가지로, 마지막 레지스터 유닛에 저장된 연산 명령어(5)는 시프트할 레지스터 유닛이 없으므로, 프로세서(100)는 마지막 레지스터 유닛에 저장된 연산 명령어(5)를 삭제할 수 있다.

[0124] 또는, 프로세서(100)는 마지막 레지스터 유닛에 저장된 연산 명령어(5)를 추가 연산 명령어 K 대신 입력할 수도 있다.

[0125] 이후, 도 6d에 도시된 바와 같이, 프로세서(100)는 연산 명령어 레지스터 유닛(150)의 제1 연산 명령어 세트(1, 2, 3, 4, 5)를 K, 1, 2, 3, 4로 업데이트할 수 있다. 그리고, 프로세서(100)는 연산 명령어 레지스터 유닛(150)에 저장된 제2 연산 명령어 세트(K, A, B, C, D)를 시프터(140)로 제공하도록 멀티플렉서(160)를 제어할 수 있다.

[0126] 이상과 같은 동작을 반복하며 프로세서(100)는 풀링 동작을 수행할 수 있다. 이때, 연산 소자 유닛(110), 제1 데이터 입력 유닛(120) 및 제2 데이터 입력 유닛(130)의 동작은 컨볼루션의 경우와 동일할 수 있다.

[0127] 예를 들어, 제1 데이터 입력 유닛(120)은 복수의 제1 연산 소자(110-1)로 커널 데이터를 입력하며, 입력된 데이터는 싸이클이 변경됨에 따라 다음 로우에 포함된 복수의 연산 소자로 시프트될 수 있다. 그리고, 제2 데이터 입력 유닛(130)은 복수의 제1 연산 소자(110-1)부터 복수의 제5 연산 소자(110-5)까지 싸이클에 따라 순차적으로 대상 데이터의 일부를 입력할 수 있다. 연산 소자 유닛(110)은 입력된 데이터를 입력된 연산 명령어에 기초하여 연산하며, 커널 데이터를 싸이클 별로 시프트할 수 있다.

[0128] 도 7은 본 발명의 일 실시 예에 따른 프로세서의 제어 방법을 설명하기 위한 흐름도이다.

[0129] 딥러닝(Deep Learning)을 수행하는 프로세서의 제어 방법은 먼저, 시프터를 통해 연산 명령어를 연산 소자 유닛으로 입력하고, 제1 데이터 입력 유닛을 통해 연산 소자 유닛으로 데이터를 입력하며, 제2 데이터 입력 유닛을 통해 연산 소자 유닛으로 데이터를 입력한다(S710). 그리고, 연산 소자 유닛이 연산 명령어에 기초하여 제1 데이터 입력 유닛을 통해 입력된 데이터 및 제2 데이터 입력 유닛을 통해 입력된 데이터를 연산한다(S720). 여기서, 프로세서는 매트릭스 형태로 배열된 복수의 연산 소자(Processing Element)를 포함하는 연산 소자 유닛, 연산 소자 유닛의 제1 로우(row)에 포함된 복수의 제1 연산 소자 각각에 연결된 제1 데이터 입력 유닛, 연산 소자 유닛의 복수의 연산 소자 각각에 연결된 제2 데이터 입력 유닛 및 연산 소자 유닛의 복수의 로우 각각에 대응되는 복수의 레지스터 유닛을 포함하며, 복수의 레지스터 유닛 각각은 대응되는 연산 소자 유닛의 로우에 포함된 복수의 연산 소자와 연결된 시프터(Shifter)를 포함할 수 있다.

[0130] 여기서, 연산하는 단계(S720)는 제1 싸이클에서, 제1 데이터 입력 유닛을 통해 복수의 제1 연산 소자 각각에 필터에 포함된 복수의 엘리먼트 중 제1 엘리먼트를 입력하는 단계, 제2 데이터 입력 유닛을 통해 복수의 제1 연산 소자 각각에 대상 데이터 중 제1 데이터를 입력하는 단계 및 연산 소자 유닛이 시프터를 통해 입력된 제1 연산 명령어에 기초하여 제1 엘리먼트 및 제1 데이터를 연산하는 단계를 포함할 수 있다.

[0131] 또한, 연산하는 단계(S720)는 제1 싸이클 다음의 제2 싸이클에서, 제1 엘리먼트를 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하는 단계, 제1 데이터 입력 유닛을 통해 복수의 제1 연산 소자 각각에 필터에 포함된 복수의 엘리먼트 중 제2 엘리먼트를 입력하는 단계 및 연산 소자 유닛이 시프터를 통해 입력된 제2 연산 명령어에 기초하여 제2 엘리먼트 및 제1 데이터를 연산하는 단계를 더 포함할 수 있다.

[0132] 또는, 연산하는 단계(S720)는 제1 싸이클 다음의 제2 싸이클에서, 제1 엘리먼트를 연산 소자 유닛의 제2 로우에 포함된 복수의 제2 연산 소자 각각으로 시프트하는 단계, 제2 데이터 입력 유닛을 통해 복수의 제2 연산 소자 각각에 대상 데이터 중 제2 데이터를 입력하는 단계 및 연산 소자 유닛이 시프터를 통해 입력된 제1 연산 명령어에 기초하여 제1 로우에서 시프트된 제1 엘리먼트 및 제2 데이터를 연산하는 단계를 더 포함할 수 있다.

[0133] 여기서, 연산하는 단계(S720)는 제1 싸이클에서, 제1 연산 명령어를 복수의 제1 연산 소자 각각에 입력하고, 제1 연산 명령어를 제1 로우에 대응되는 제1 레지스터 유닛으로부터 제2 로우에 대응되는 제2 레지스터 유닛으로 시프트하는 단계 및 제2 싸이클에서, 시프트된 제1 연산 명령어를 복수의 제2 연산 소자 각각에 입력하고, 제2 연산 명령어를 복수의 제1 연산 소자 각각에 입력하며, 제1 연산 명령어를 제2 레지스터 유닛으로부터 연산 소자 유닛의 제3 로우에 대응되는 제3 레지스터 유닛으로 시프트하고, 제2 연산 명령어를 제1 레지스터 유닛으로부터 제2 레지스터 유닛으로 시프트하는 단계를 더 포함할 수 있다.

[0134] 한편, 연산하는 단계(S720)는 제1 엘리먼트 및 제1 데이터를 연산한 연산 데이터를 인접한 연산 소자로 전달하는 단계 및 전달된 연산 데이터를 연산 소자에서 연산된 데이터와 어큐뮬레이션(accumulation)하는 단계를 더 포함할 수 있다.

[0135] 또한, 제2 데이터 입력 유닛은 연산 소자 유닛의 로우에 포함된 복수의 연산 소자의 개수와 동일한 개수의 데이터 패스(path)를 통해 외부 메모리와 연결될 수 있다.

[0136] 그리고, 프로세서는 연산 명령어 레지스터 유닛 및 연산 명령어 레지스터 유닛 및 시프터 사이에 연결된 멀티플렉서를 더 포함하며, 제어 방법은 멀티플렉서에 의해 연산 명령어 레지스터 유닛에 저장된 복수의 연산 명령어 세트 중 하나를 시프터로 제공하는 단계를 더 포함하며, 복수의 연산 명령어 세트 각각은 연산 소자 유닛의 복수의 로우 각각에 입력될 복수의 연산 명령어를 포함할 수 있다.

[0137] 여기서, 시프터로 입력된 연산 명령어 세트를 복수의 레지스터 유닛에 저장하는 단계, 저장된 연산 명령어 세트에 포함된 명령어 각각을 기설정된 방향의 인접한 레지스터 유닛으로 시프트하는 단계 및 제1 로우에 대응되는 제1 레지스터 유닛으로 추가 연산 명령어를 입력하는 단계를 더 포함하며, 연산 소자 유닛으로 입력하는 단계(S710)는 추가 연산 명령어 및 시프트된 연산 명령어 세트에 포함된 명령어 각각을 연산 소자 유닛의 대응되는 로우 각각으로 입력할 수 있다.

[0138] 그리고, 추가 연산 명령어 및 시프트된 연산 명령어 세트에 포함된 명령어를 연산 명령어 레지스터 유닛으로 제공하는 단계 및 연산 명령어 레지스터 유닛에 저장된 복수의 연산 명령어 세트 중 시프터로 입력된 연산 명령어 세트를 추가 연산 명령어 및 시프트된 연산 명령어 세트에 포함된 명령어로 업데이트하는 단계를 더 포함할 수 있다.

[0139] 이상과 같은 본 발명의 다양한 실시 예에 따르면, 프로세서가 외부 메모리로부터의 피크 데이터 이동량을 감소시키는 구조로 설계되고, 설계된 구조에 대응되도록 연산을 수행함에 따라 프로세서의 확장성(scalability)을 향상시킬 수 있다.

[0140] 이상에서는 본 개시의 바람직한 실시 예에 대하여 도시하고 설명하였지만, 본 개시는 상술한 특정의 실시 예에 한정되지 아니하며, 청구범위에서 청구하는 본 개시의 요지를 벗어남이 없이 당해 개시에 속하는 기술분야에서 통상의 지식을 가진 자에 의해 다양한 변형실시가 가능한 것은 물론이고, 이러한 변형실시들은 본 개시의 기술적 사상이나 전망으로부터 개별적으로 이해되어져서는 안될 것이다.

부호의 설명

[0141] 100 : 프로세서 110 : 연산 소자 유닛

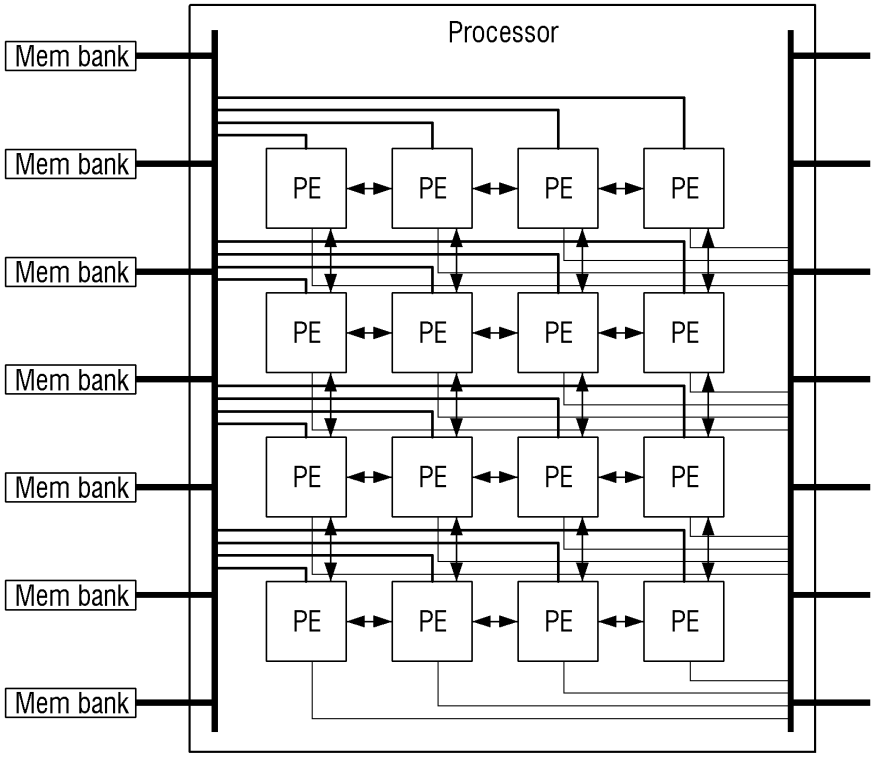
120 : 제1 데이터 입력 유닛 130 : 제2 데이터 입력 유닛

140 : 시프터

150 : 연산 명령어 레지스터 유닛 160 : 멀티플렉서

도면

도면1a


Processor
Mem bank
Mem bank
Mem bank
Mem bank
Mem bank
Mem bank
Mem bank
PE
PE
PE
PE
PE
PE
PE
PE
PE
PE
PE
PE
PE
PE
PE
PE

도면1b

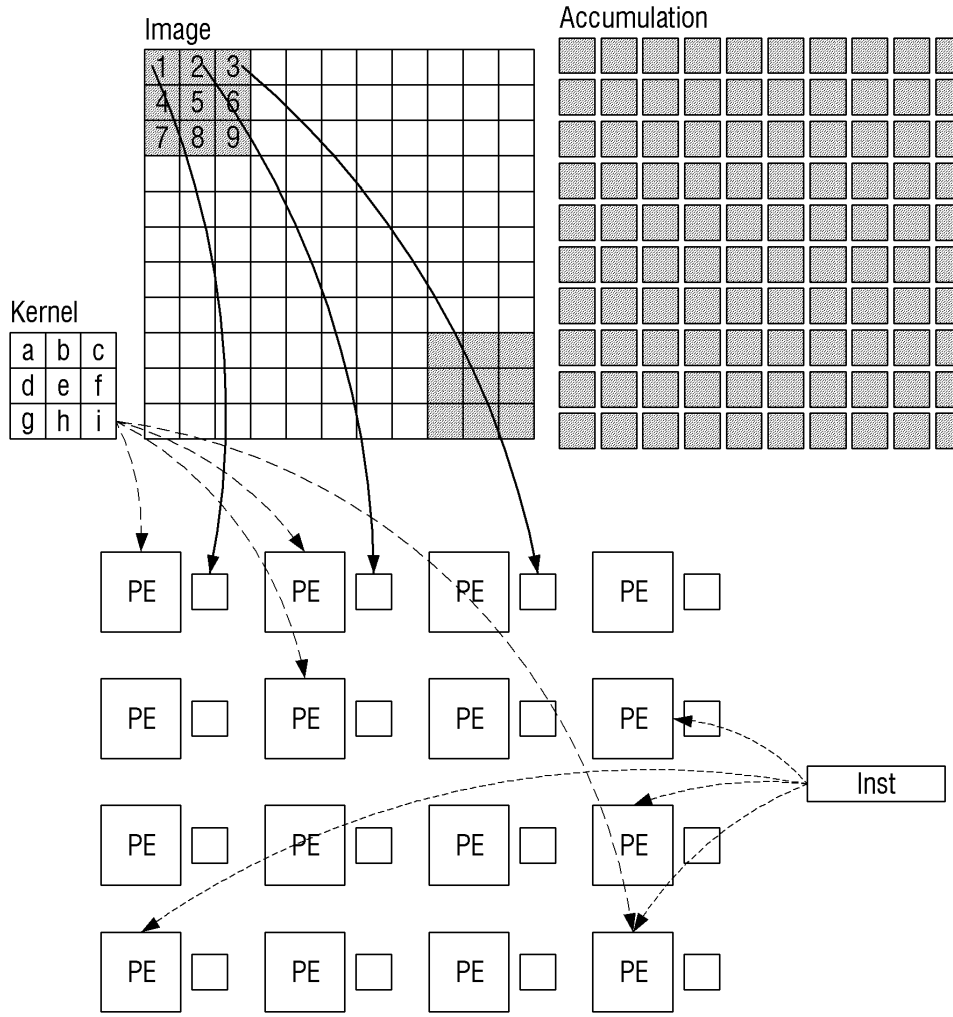
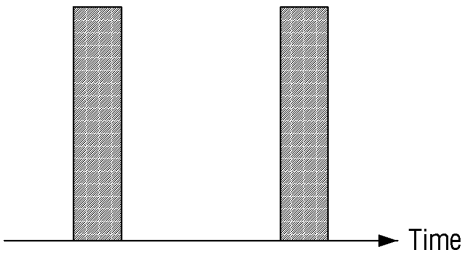
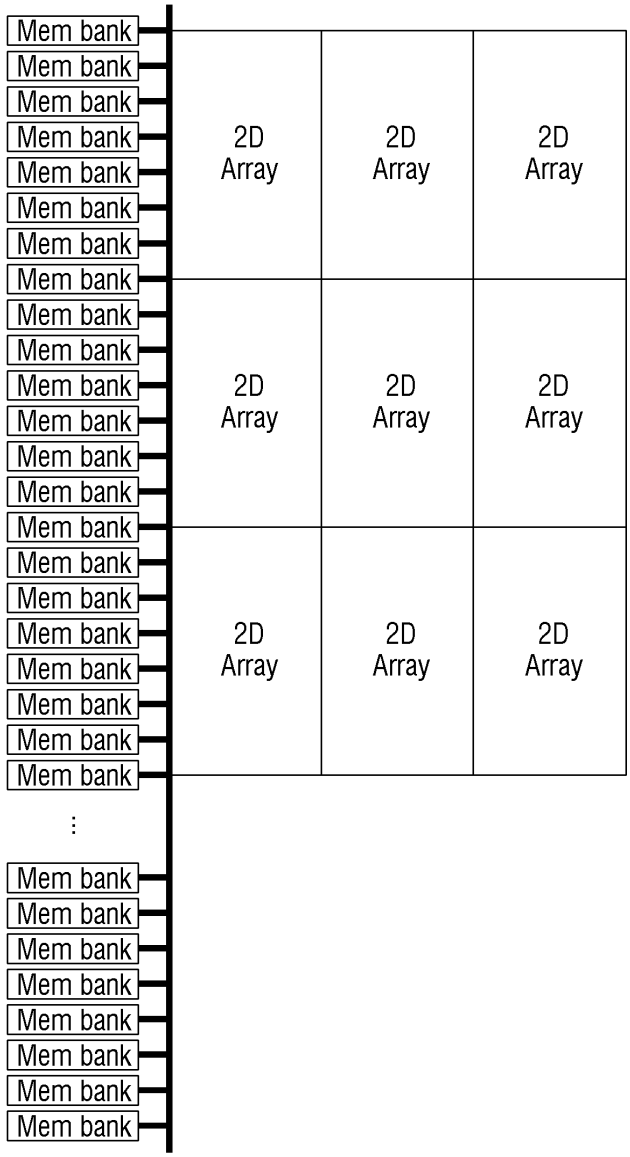


Image
1 2 3
4 5 6
7 8 9
Accumulation
Kernel
a b c
d e f
g h i
PE
Inst

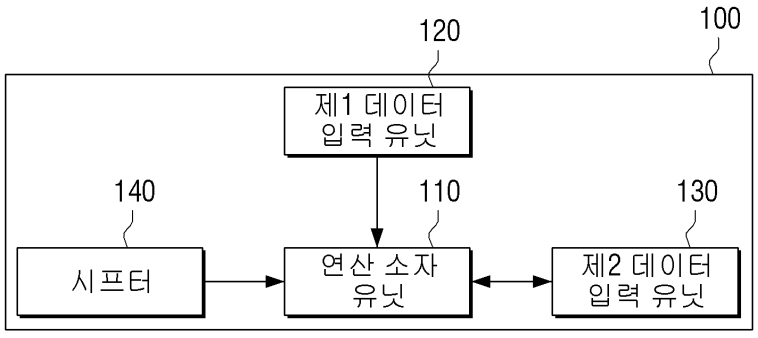
도면1c


Time

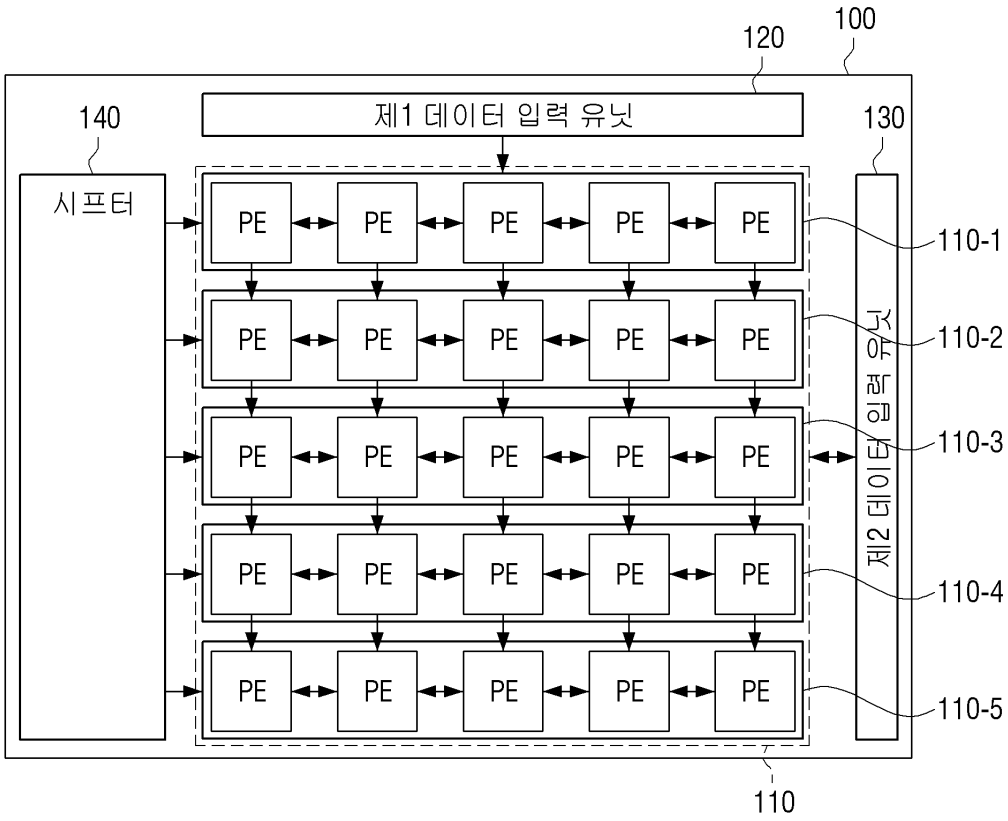
도면1d


Mem bank
2D Array

도면2a


120
100
제1 데이터 입력 유닛
140
110
130
시프터
연산 소자 유닛
제2 데이터 입력 유닛

도면2b



도면3a

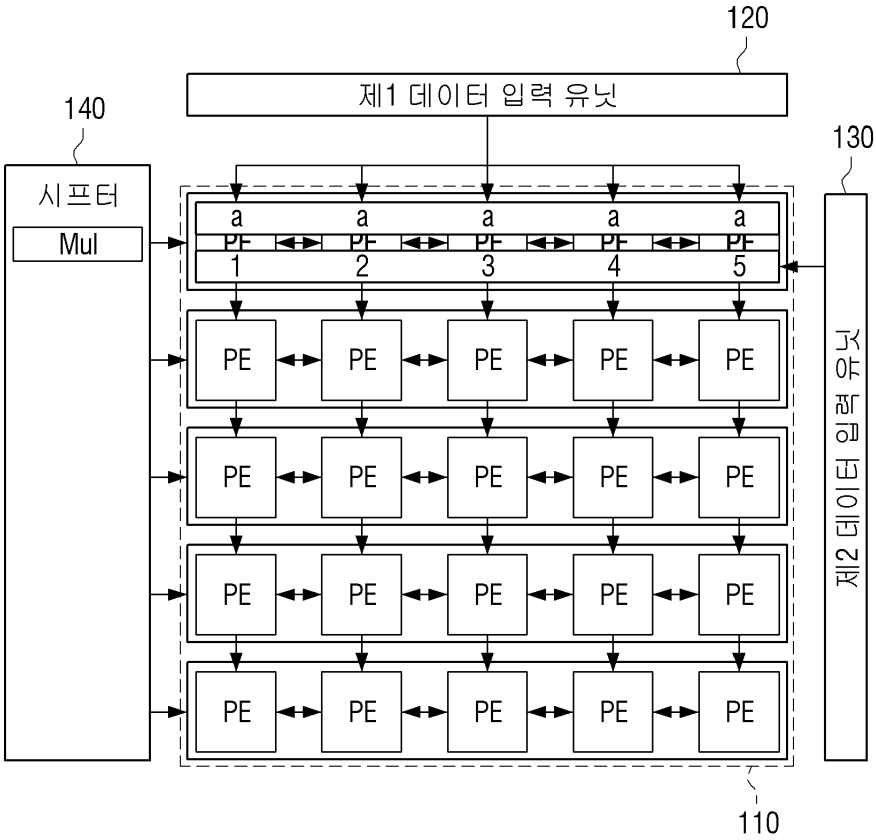
Kernel

a	b
c	d

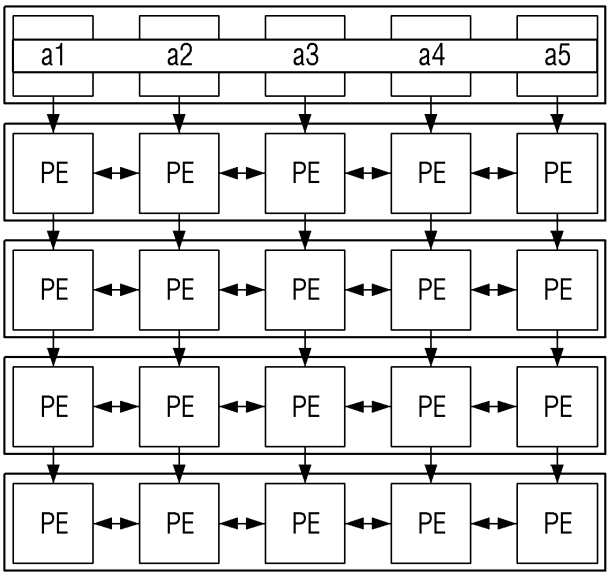
Image

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

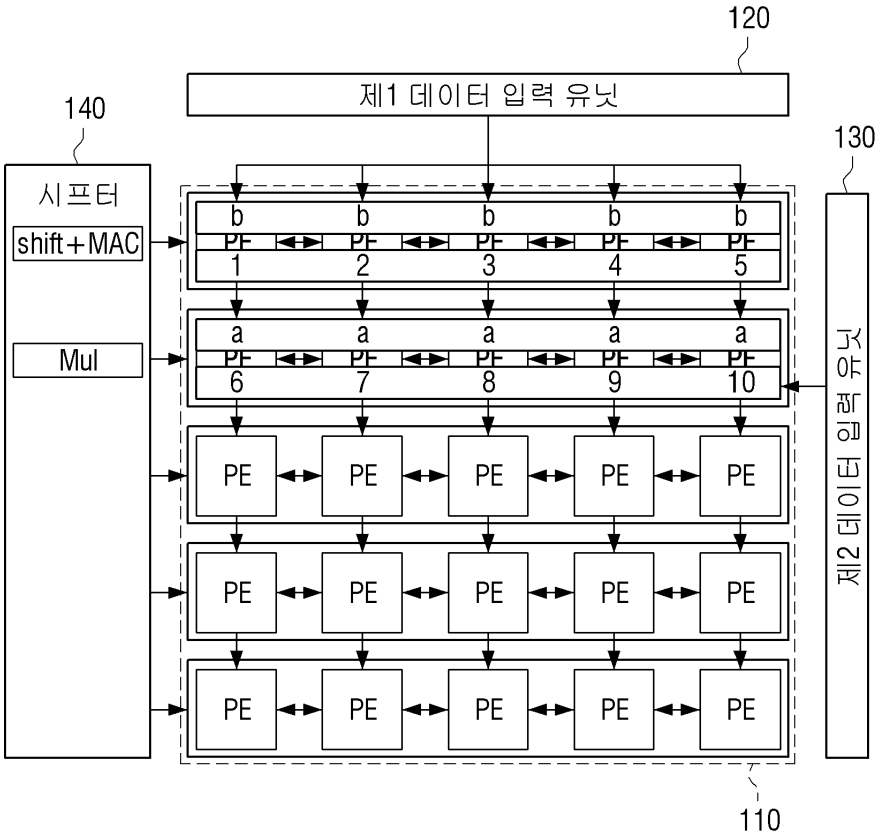
도면3b


120
제1 데이터 입력 유닛
140
시프터
Mul
130
제2 데이터 입력 유닛
a
PE
1
2
3
4
5
110

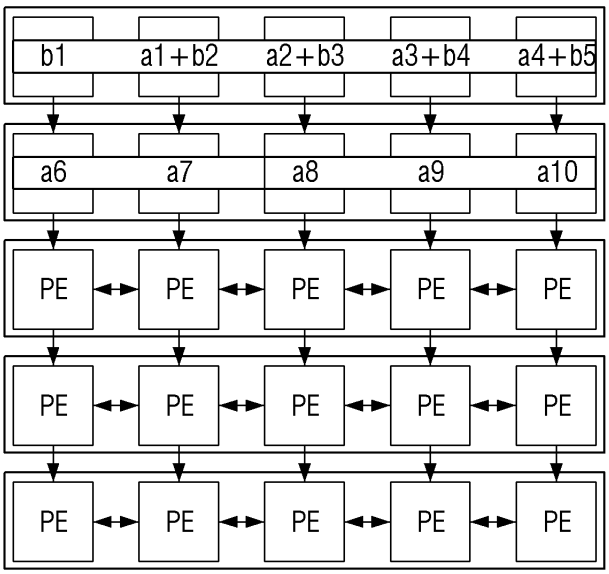
도면3c


a1
a2
a3
a4
a5
PE

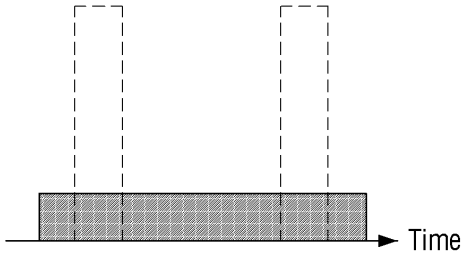
도면3d


120
제1 데이터 입력 유닛
140
시프터
shift+MAC
Mul
130
제2 데이터 입력 유닛
b b b b b
PE PE PE PE PE
1 2 3 4 5
a a a a a
PE PE PE PE PE
6 7 8 9 10
PE PE PE PE PE
PE PE PE PE PE
PE PE PE PE PE
110

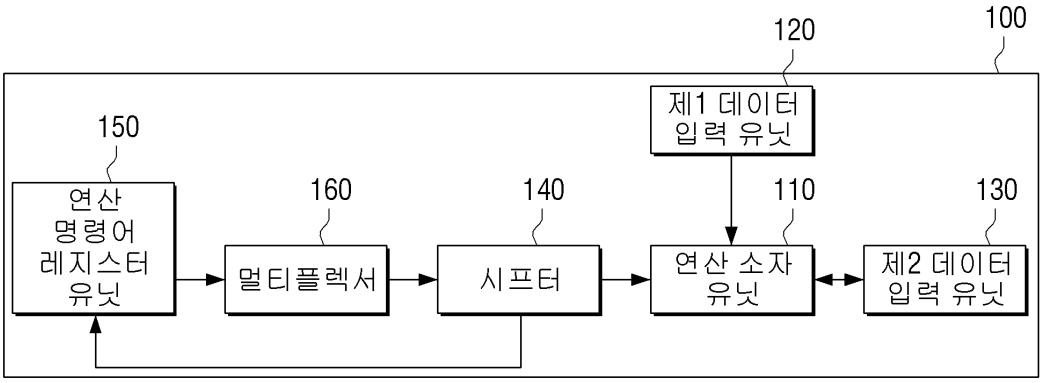
도면3e


b1 a1+b2 a2+b3 a3+b4 a4+b5
a6 a7 a8 a9 a10
PE PE PE PE PE
PE PE PE PE PE
PE PE PE PE PE

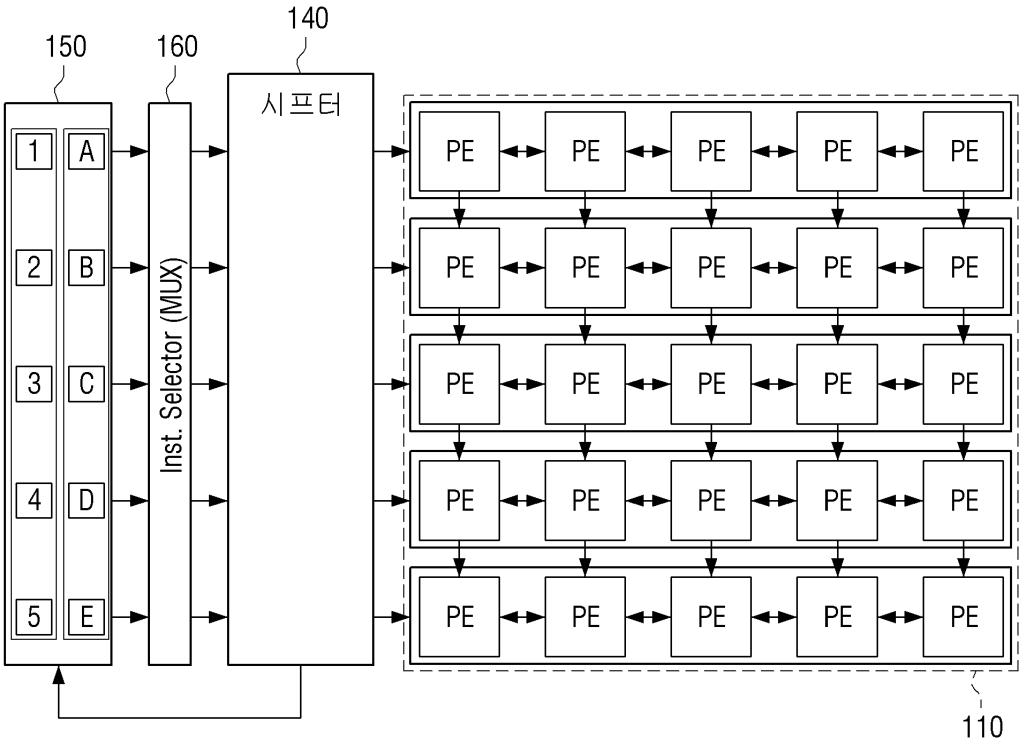
도면4


Time

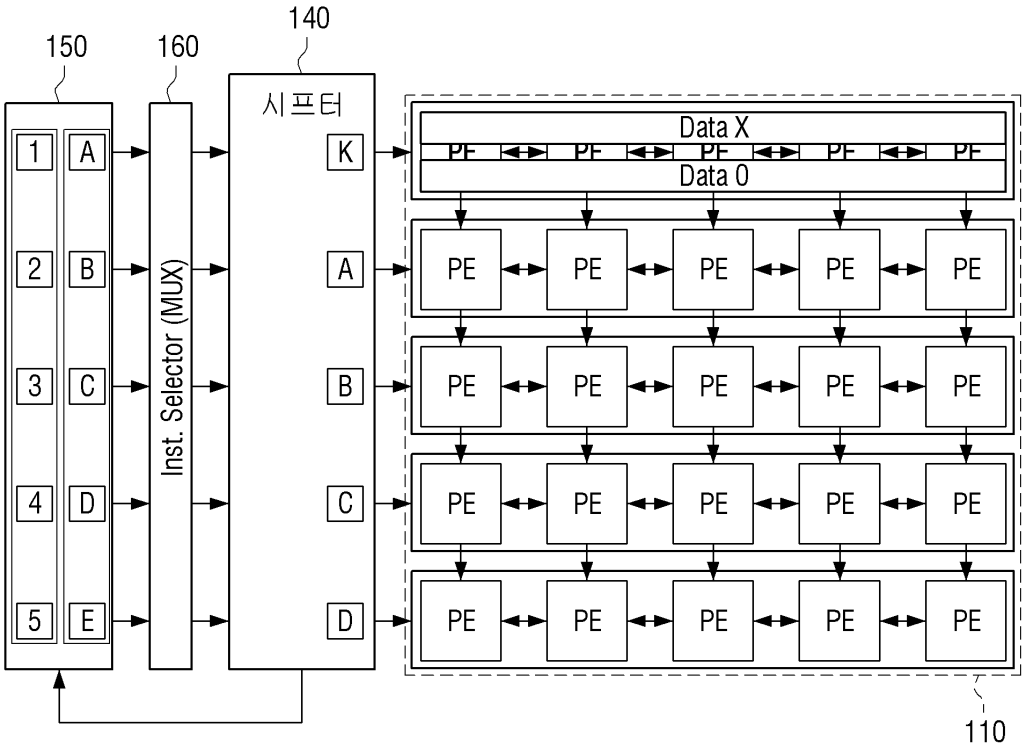
도면5


120
100
제1 데이터 입력 유닛
150
연산 명령어 레지스터 유닛
160
멀티플렉서
140
시프터
110
연산 소자 유닛
130
제2 데이터 입력 유닛

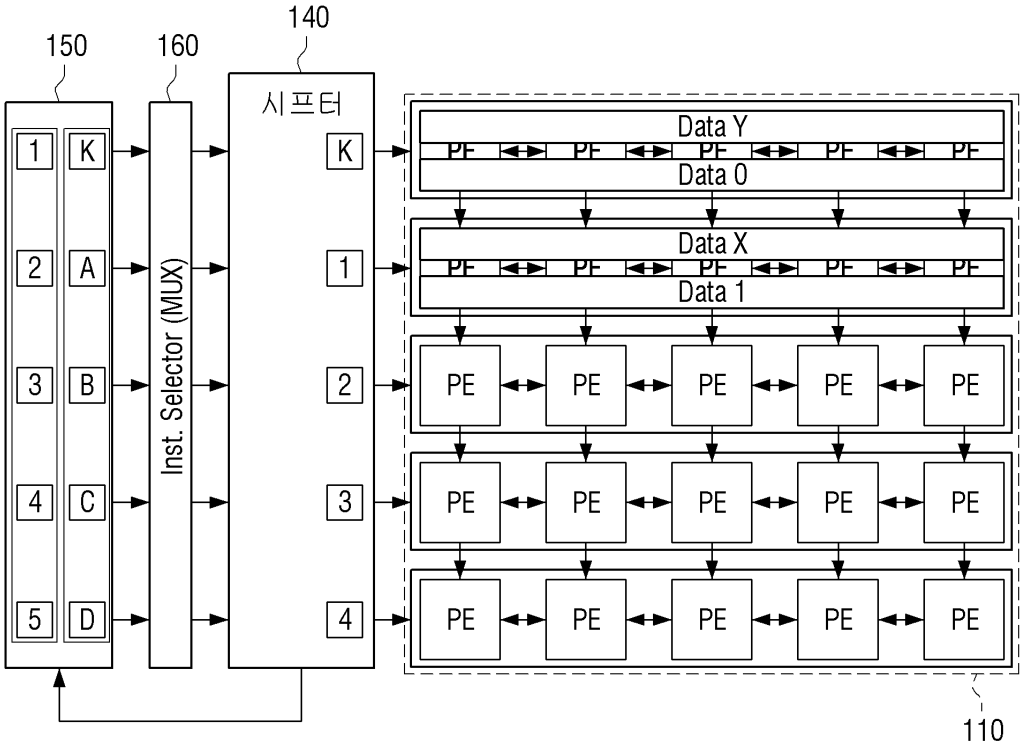
도면6a


150
160
140
시프터
1
A
2
B
3
C
4
D
5
E
Inst. Selector (MUX)
PE
110

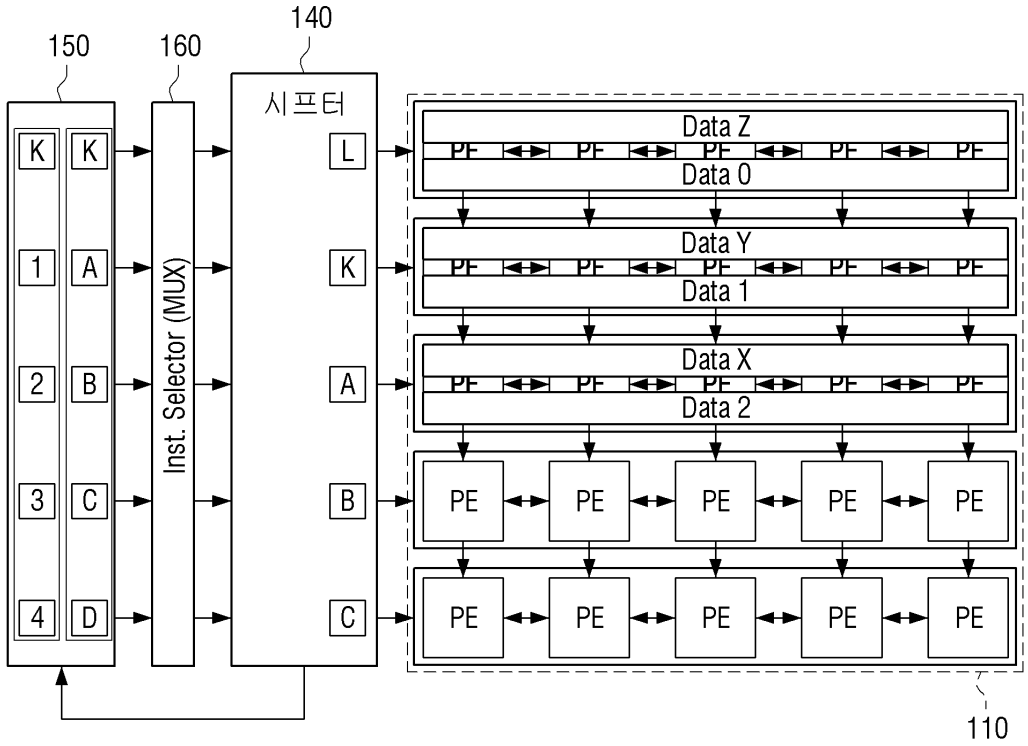
도면6b


150
160
140
시프터
Inst. Selector (MUX)
1
2
3
4
5
A
B
C
D
E
K
A
B
C
D
Data X
Data 0
PE
110

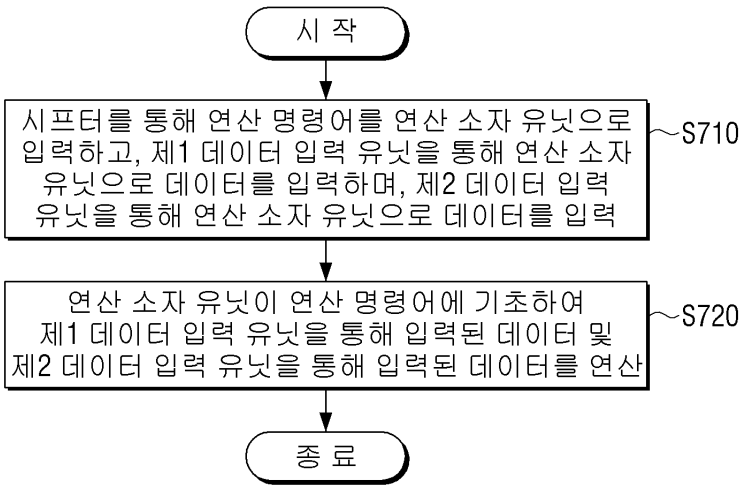
도면6c


150
160
140
시프터
Inst. Selector (MUX)
1
2
3
4
5
K
A
B
C
D
K
1
2
3
4
Data Y
Data 0
Data X
Data 1
PE
110

도면6d


150
160
140
시프터
Inst. Selector (MUX)
K
K
1
A
2
B
3
C
4
D
L
K
A
B
C
Data Z
Data 0
Data Y
Data 1
Data X
Data 2
PE
110

도면7


시 작
시프터를 통해 연산 명령어를 연산 소자 유닛으로 입력하고, 제1 데이터 입력 유닛을 통해 연산 소자 유닛으로 데이터를 입력하며, 제2 데이터 입력 유닛을 통해 연산 소자 유닛으로 데이터를 입력
S710
연산 소자 유닛이 연산 명령어에 기초하여 제1 데이터 입력 유닛을 통해 입력된 데이터 및 제2 데이터 입력 유닛을 통해 입력된 데이터를 연산
S720
종 료