



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0107042
(43) 공개일자 2025년07월11일

(51) 국제특허분류(Int. Cl.)
G06N 3/063 (2023.01) G06N 3/0464 (2023.01)
G06N 3/0495 (2023.01)
(52) CPC특허분류
G06N 3/063 (2013.01)
G06N 3/0464 (2023.01)
(21) 출원번호 10-2024-0001739
(22) 출원일자 2024년01월04일
심사청구일자 2024년01월04일

(71) 출원인
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50
성균관대학교산학협력단
경기도 수원시 장안구 서부로 2066 (천천동, 성균관대학교내)
(72) 발명자
이종은
울산광역시 울주군 언양읍 유니스트길 50
송민준
울산광역시 울주군 언양읍 유니스트길 50
권기원
경기도 수원시 장안구 서부로 2066 (천천동)
(74) 대리인
특허법인 무한

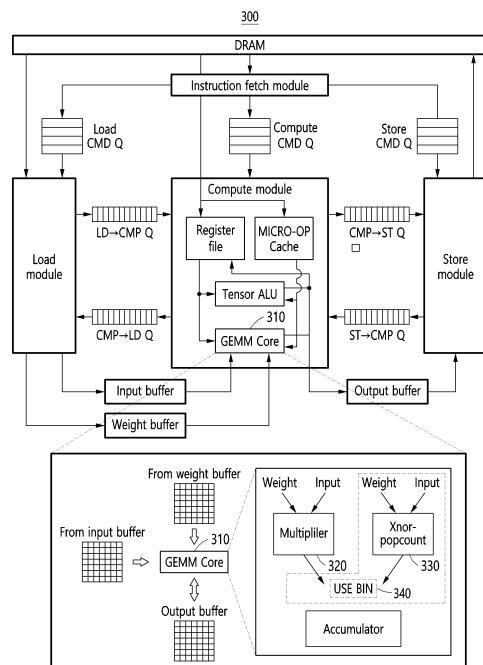
전체 청구항 수 : 총 16 항

(54) 발명의 명칭 BNN 및 QNN을 선택적으로 수행하는 장치 및 이의 동작 방법

(57) 요약

일 실시예에 따른, BNN(binarized neural networks) 연산 또는 QNN(quantized neural networks) 연산을 선택적으로(selectively) 수행하는 뉴럴 네트워크 동작 방법은, CPU(central processing unit)가 입력 데이터를 비트패킹(bit-packing)하는 단계; 상기 비트패킹된 입력 데이터를 메모리에 저장하는 단계; NPU(neural processing unit)에 (뒷면에 계속)

대표도 - 도3



unit)가 컴파일러(compiler)를 이용하여 상기 비트패킹된 입력 데이터 및 가중치 간의 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택하도록 결정하는, 선택 명령어를 생성하는 단계; 상기 NPU가, 상기 비트패킹된 입력 데이터 및 상기 가중치를 상기 메모리로부터 로딩하고, 상기 비트패킹된 입력 데이터 및 상기 가중치 간의 가중합 연산에 기초하여 BNN 연산 및 QNN 연산을 수행하는 단계; 상기 NPU가, 상기 컴파일러를 이용하여 생성한 상기 선택 명령어에 기초하여, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 하나를 선택(select)하는 단계; 및 상기 NPU가, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 선택된 연산 결과를 축적하는 단계를 포함할 수 있다.

(52) CPC특허분류

G06N 3/0495 (2023.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	2710007820
과제번호	II201336
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성(R&D)
연구과제명	인공지능대학원지원(울산과학기술원)
과제수행기관명	울산과학기술원
연구기간	2024.01.01 ~ 2024.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호	2710002839
과제번호	00258527
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	차세대지능형반도체기술개발(소자)
연구과제명	온칩 학습을 위한 전하 저장 기반의 고집적 · 저전력 · 비휘발성 3차원 수직형 시냅스 어레이 및 아키텍처 개발
과제수행기관명	성균관대학교 산학협력단
연구기간	2024.01.01 ~ 2024.12.31

명세서

청구범위

청구항 1

BNN(binarized neural networks) 연산 또는 QNN(quantized neural networks) 연산을 선택적으로(selectively) 수행하는 뉴럴 네트워크 동작 방법에 있어서,

CPU(central processing unit)가 입력 데이터를 비트패킹(bit-packing)하는 단계;

상기 비트패킹된 입력 데이터를 메모리에 저장하는 단계;

NPU(neural processing unit)가 컴파일러(compiler)를 이용하여 상기 비트패킹된 입력 데이터 및 가중치 간의 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택하도록 결정하는, 선택 명령어를 생성하는 단계;

상기 NPU가, 상기 비트패킹된 입력 데이터 및 상기 가중치를 상기 메모리로부터 로딩하고, 상기 비트패킹된 입력 데이터 및 상기 가중치 간의 가중합 연산에 기초하여 BNN 연산 및 QNN 연산을 수행하는 단계;

상기 NPU가, 상기 컴파일러를 이용하여 생성한 상기 선택 명령어에 기초하여, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 하나를 선택(select)하는 단계; 및

상기 NPU가, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 선택된 연산 결과를 축적하는 단계를 포함하는 뉴럴 네트워크 동작 방법.

청구항 2

제1항에 있어서,

상기 CPU가 입력 데이터를 비트패킹(bit-packing)하는 단계는,

상기 NPU에서 상기 메모리로부터 비트패킹된 제1 입력 데이터 및 제1 가중치를 로딩하도록, 제1 로딩 전에 제1 비트패킹을 수행하는 단계; 및

상기 NPU에서 상기 메모리로부터 비트패킹된 제2 입력 데이터 및 제2 가중치를 로딩하도록, 제2 로딩 전에 제2 비트패킹을 수행하는 단계

를 포함하는 뉴럴 네트워크 동작 방법.

청구항 3

제1항에 있어서,

상기 CPU가 입력 데이터를 비트패킹(bit-packing)하는 단계는,

상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대응하는 BNN 연산 및 QNN 연산이 상기 NPU에서 수행되는 경우, 상기 해당 레이어의 다음 레이어에 대한 입력 데이터를 미리 비트패킹하는 단계

를 포함하는 뉴럴 네트워크 동작 방법.

청구항 4

제1항에 있어서,

상기 선택 명령어를 생성하는 단계는,

상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어의 출력이 BNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 하이(logic high) 값에 대응하는 제1 선택 명령어를 생성하는 단계; 및

상기 해당 레이어의 출력이 QNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 로우(logic low) 값에 대응하는 제2 선택 명령어를 생성하는 단계

를 포함하는 뉴럴 네트워크 동작 방법.

청구항 5

제1항에 있어서,

상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 하나를 선택(select)하는 단계는,

상기 컴파일러를 이용하여 로직 하이 값에 대응하는 제1 선택 명령어를 생성하는 경우, 상기 BNN 연산 결과를 선택하는 단계; 및

상기 컴파일러를 이용하여 로직 로우 값에 대응하는 제2 선택 명령어를 생성하는 경우, 상기 QNN 연산 결과를 선택하는 단계

를 포함하는 뉴럴 네트워크 동작 방법.

청구항 6

제1항에 있어서,

상기 컴파일러를 이용하여, 이진 컨볼루션(binary convolution) 연산에 대응하는 스케일 팩터(scale factor)에 기초하는 배치놈 정규화(batchnorm optimization) 연산을 수행하는 단계

를 더 포함하는 뉴럴 네트워크 동작 방법.

청구항 7

제1항에 있어서,

상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대한 상기 입력 데이터의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수보다 작은 경우,

상기 컴파일러를 이용하여 가중치 커널맵을 미리 결정된 크기로 재구성하고, 상기 재구성된 가중치 커널맵의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수의 배수가 되도록, 부족한 채널맵을 추가하는 단계; 및

상기 CPU가 상기 입력 데이터에 대한 상기 가중치 커널맵의 스위핑(sweeping)에 따른 최소 연산 가능 수에 기초하여, 상기 입력 데이터의 채널맵의 수를 재구성하는 단계

를 더 포함하는 뉴럴 네트워크 동작 방법.

청구항 8

하드웨어와 결합되어 제1항 내지 제7항 중 어느 하나의 항의 방법을 실행시키기 위하여 컴퓨터 판독 가능한 기록매체에 저장된 컴퓨터 프로그램.

청구항 9

뉴럴 네트워크 모델의 BNN(binanzed neural networks) 연산 또는 QNN(quantized neural networks) 연산을 선택적으로(selectively) 수행하는 전자 장치에 있어서,

입력 데이터를 비트패킹(bit-packing)하는 CPU(central processing unit);

상기 비트패킹된 입력 데이터를 저장하는 메모리;

컴파일러(compiler)를 이용하여 비트패킹된 입력 데이터 및 가중치 간의 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택하도록 결정하는, 선택 명령어를 생성하고, 상기 비트패킹된 입력 데이터 및 상기 가중치를 상기 메모리로부터 로딩하고, 상기 비트패킹된 입력 데이터 및 상기 가중치 간의 가중합 연산에 기초하여 BNN 연산 및 QNN 연산을 수행하며, 상기 컴파일러를 이용하여 생성한 상기 선택 명령어에 기초하여, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 하나를 선택(select)하고, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 선택된 연산 결과를 축적하는 NPU(neural processing unit)

를 포함하는 전자 장치.

청구항 10

제9항에 있어서,

상기 CPU는,

상기 NPU에서 상기 메모리로부터 비트패킹된 제1 입력 데이터 및 제1 가중치를 로딩하도록, 제1 로딩 전에 제1 비트패킹을 수행하고, 상기 NPU에서 상기 메모리로부터 비트패킹된 제2 입력 데이터 및 제2 가중치를 로딩하도록, 제2 로딩 전에 제2 비트패킹을 수행하는,

전자 장치.

청구항 11

제9항에 있어서,

상기 CPU는,

상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대응하는 BNN 연산 및 QNN 연산이 상기 NPU에서 수행되는 경우, 상기 해당 레이어의 다음 레이어에 대한 입력 데이터를 미리 비트패킹하는

전자 장치.

청구항 12

제9항에 있어서,

상기 NPU는,

상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어의 출력이 BNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 하이(logic high) 값에 대응하는 제1 선택 명령어를 생성하고, 상기 해당 레이어의 출력이 QNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 로우(logic low) 값에 대응하는 제2 선택 명령어를 생성하는

전자 장치.

청구항 13

제9항에 있어서,

상기 NPU는,

상기 컴파일러를 이용하여 로직 하이 값에 대응하는 제1 선택 명령어를 생성하는 경우, 상기 BNN 연산 결과를 선택하고, 상기 컴파일러를 이용하여 로직 로우 값에 대응하는 제2 선택 명령어를 생성하는 경우, 상기 QNN 연산 결과를 선택하는

전자 장치.

청구항 14

제9항에 있어서,

상기 NPU는,

상기 컴파일러를 이용하여, 이진 컨볼루션(binary convolution) 연산에 대응하는 스케일 팩터(scale factor)에 기초하는 배치놈 정규화(batchnorm optimization) 연산을 수행하는

전자 장치.

청구항 15

제9항에 있어서,

상기 NPU는,

상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대한 상기 입력 데이터의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수보다 작은 경우, 상기 컴파일러를 이용하여 가중치 커널맵을 미리 결정된 크기로 재구성하고, 상기 재구성된 가중치 커널맵의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수의 배수가 되도록, 부족한 채널맵을 추가하는

전자 장치.

청구항 16

제9항에 있어서,

상기 CPU는,

상기 입력 데이터에 대한 가중치 커널맵의 스위핑(sweeping)에 따른 최소 연산 가능 수에 기초하여, 상기 입력 데이터의 채널맵의 수를 재구성하는,

전자 장치.

발명의 설명

기술 분야

[0001] 이하의 개시는 BNN 연산 또는 QNN 연산이 포함된 뉴럴 네트워크 모델을 동작하기 위한 기술에 관한 것이다.

배경 기술

[0003] 저정밀 신경망(low-precision neural networks)은 모델 크기가 작고 계산시 요구되는 사항들이 적어서 효율적이다. 또한, 저정밀 신경망에 대한 관심은 최근 양자화 및 이진화 신경망 훈련이 발전함에 따라, 높아지고 다. 종래 3비트 또는 2비트 양자화 모델들은 ImageNet 분류 작업에서 기존 성능(즉, 부동 소수점 추론 정확도)을 달성하는 것으로 파악된다. 또한, 종래 이진화 신경망도 비슷한 결과를 얻은 것으로 파악된다. 다만, 이러한 저정밀 신경망을 구현하기 위한 하드웨어 아키텍처 및 컴파일러는 부족한 상황이다. 특히, BNN(이진 뉴럴 네트워크)은 리소스가 제한된 장치로 구현할 수 있는 가장 유망한 뉴럴 네트워크 모델이다. BNN은 비트와이즈(bitwise) 및 팝카운트 명령어들(pop-count instructions)을 사용하여 CPU에서 구현될 수 있으나, CPU에서는 비트 명령어들의 병렬 처리 정도가 제한되어 BNN 가속 정도가 제한될 수 있다.

[0004] BNN은 FPGA에서 구현됨으로써, BNN 가속화 시 매우 높은 수준의 병렬성을 제공받을 수 있으나, 종종 유연성이 없다. 즉, 해당 FPGA는 특정 BNN만 구현할 수 있는 하드와이어 설계를 생성할 수 있다. 또한, 해당 FPGA는 스케일 팩터(scale factors) 및 스킵 커넥션(skip connections) 기법을 활용함으로써 부동 소수점(floating point)에 가까운 정확도를 보여주는, BiRealNet 및 ReActNet과 같은 향상된(advanced) BNN을 구현하기 어렵다. 이하에서는, 스케일 팩터 및 스킵 커넥션 기법을 사용하는 최신 BNN을 구현하기 위한 BNN용 컴파일러(compiler) 및 프로그래밍 가능한 가속기(programmable accelerators)에 대하여 개시한다. 보다 구체적으로, 아래는 QNN 및 BNN을 동시에(시간적으로 동시가 아닌, 모두 구현할 수 있다는 의미) 구현할 수 있는 NPU(neural processing unit) 아키텍처 및 유연하고 효율적인 컴파일러를 이용한 BNN 네트워크 동작 방법에 대하여 개시한다.

발명의 내용

과제의 해결 수단

[0005] 일 실시예에 따른, BNN(binanzed neural networks) 연산 또는 QNN(quantized neural networks) 연산을 선택적으로(selectively) 수행하는 뉴럴 네트워크 동작 방법은, CPU(central processing unit)가 입력 데이터를 비트 패킹(bit-packing)하는 단계; 상기 비트패킹된 입력 데이터를 메모리에 저장하는 단계; NPU(neural processing unit)가 컴파일러(compiler)를 이용하여 상기 비트패킹된 입력 데이터 및 가중치 간의 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택하도록 결정하는, 선택 명령어를 생성하는 단계; 상기 NPU가, 상기 비트패킹된 입력 데이터 및 상기 가중치를 상기 메모리로부터 로딩하고, 상기 비트패킹된 입력 데이터 및 상기 가중치 간의 가중

합 연산에 기초하여 BNN 연산 및 QNN 연산을 수행하는 단계; 상기 NPU가, 상기 컴파일러를 이용하여 생성한 상기 선택 명령어에 기초하여, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 하나를 선택(select)하는 단계; 및 상기 NPU가, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 선택된 연산 결과를 축적하는 단계를 포함할 수 있다.

[0006] 상기 CPU가 입력 데이터를 비트패킹(bit-packing)하는 단계는, 상기 NPU에서 상기 메모리로부터 비트패킹된 제1 입력 데이터 및 제1 가중치를 로딩하도록, 제1 로딩 전에 제1 비트패킹을 수행하는 단계; 및 상기 NPU에서 상기 메모리로부터 비트패킹된 제2 입력 데이터 및 제2 가중치를 로딩하도록, 제2 로딩 전에 제2 비트패킹을 수행하는 단계를 포함할 수 있다.

[0007] 상기 CPU가 입력 데이터를 비트패킹(bit-packing)하는 단계는, 상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대응하는 BNN 연산 및 QNN 연산이 상기 NPU에서 수행되는 경우, 상기 해당 레이어의 다음 레이어에 대한 입력 데이터를 미리 비트패킹하는 단계를 포함할 수 있다.

[0008] 상기 선택 명령어를 생성하는 단계는, 상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어의 출력이 BNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 하이(logic high) 값에 대응하는 제1 선택 명령어를 생성하는 단계; 및 상기 해당 레이어의 출력이 QNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 로우(logic low) 값에 대응하는 제2 선택 명령어를 생성하는 단계를 포함할 수 있다.

[0009] 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 하나를 선택(select)하는 단계는, 상기 컴파일러를 이용하여 로직 하이 값에 대응하는 제1 선택 명령어를 생성하는 경우, 상기 BNN 연산 결과를 선택하는 단계; 및 상기 컴파일러를 이용하여 로직 로우 값에 대응하는 제2 선택 명령어를 생성하는 경우, 상기 QNN 연산 결과를 선택하는 단계를 포함할 수 있다.

[0010] 일 실시예에 따른, BNN(binarized neural networks) 연산 또는 QNN(quantized neural networks) 연산을 선택적으로(selectively) 수행하는 뉴럴 네트워크 동작 방법은, 상기 컴파일러를 이용하여, 이진 컨볼루션(binary convolution) 연산에 대응하는 스케일 팩터(scale factor)에 기초하는 배치놈 정규화(batchnorm optimization) 연산을 수행하는 단계를 더 포함할 수 있다.

[0011] 일 실시예에 따른, BNN(binarized neural networks) 연산 또는 QNN(quantized neural networks) 연산을 선택적으로(selectively) 수행하는 뉴럴 네트워크 동작 방법은, 상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대한 상기 입력 데이터의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수보다 작은 경우, 상기 컴파일러를 이용하여 가중치 커널맵을 미리 결정된 크기로 재구성하고, 상기 재구성된 가중치 커널맵의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수의 배수가 되도록, 부족한 채널맵을 추가하는 단계; 및 상기 CPU가 상기 입력 데이터에 대한 상기 가중치 커널맵의 스위핑(sweeping)에 따른 최소 연산 가능 수에 기초하여, 상기 입력 데이터의 채널맵의 수를 재구성하는 단계를 더 포함할 수 있다.

[0012] 일 실시예에 따른, 뉴럴 네트워크 모델의 BNN(binarized neural networks) 연산 또는 QNN(quantized neural networks) 연산을 선택적으로(selectively) 수행하는 전자 장치는, 입력 데이터를 비트패킹(bit-packing)하는 CPU(central processing unit); 상기 비트패킹된 입력 데이터를 저장하는 메모리; 컴파일러(compiler)를 이용하여 비트패킹된 입력 데이터 및 가중치 간의 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택하도록 결정하는, 선택 명령어를 생성하고, 상기 비트패킹된 입력 데이터 및 상기 가중치를 상기 메모리로부터 로딩하고, 상기 비트패킹된 입력 데이터 및 상기 가중치 간의 가중합 연산에 기초하여 BNN 연산 및 QNN 연산을 수행하며, 상기 컴파일러를 이용하여 생성한 상기 선택 명령어에 기초하여, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 하나를 선택(select)하고, 상기 BNN 연산 결과 또는 상기 QNN 연산 결과 중 선택된 연산 결과를 축적하는 NPU(neural processing unit)를 포함할 수 있다.

[0013] 상기 CPU는, 상기 NPU에서 상기 메모리로부터 비트패킹된 제1 입력 데이터 및 제1 가중치를 로딩하도록, 제1 로딩 전에 제1 비트패킹을 수행하고, 상기 NPU에서 상기 메모리로부터 비트패킹된 제2 입력 데이터 및 제2 가중치를 로딩하도록, 제2 로딩 전에 제2 비트패킹을 수행할 수 있다.

[0014] 상기 CPU는, 상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대응하는 BNN 연산 및 QNN 연산이 상기 NPU에서 수행되는 경우, 상기 해당 레이어의 다음 레이어에 대한 입력 데이터를 미리 비트패킹할 수 있다.

[0015] 상기 NPU는, 상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어의 출력이 BNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 하이(logic high) 값에 대응하는 제1 선택 명령어를 생성하고, 상기 해당 레이어의 출력이 QNN 연산 결과에 기초하는 출력인 경우, 상기 컴파일러를 이용하여 로직 로우(logic low) 값에

대응하는 제2 선택 명령어를 생성하는 전자 장치.

- [0016] 상기 NPU는, 상기 컴파일러를 이용하여 로직 하이 값에 대응하는 제1 선택 명령어를 생성하는 경우, 상기 BNN 연산 결과를 선택하고, 상기 컴파일러를 이용하여 로직 로우 값에 대응하는 제2 선택 명령어를 생성하는 경우, 상기 QNN 연산 결과를 선택하는 전자 장치.
- [0017] 상기 NPU는, 상기 컴파일러를 이용하여, 이진 컨볼루션(binary convolution) 연산에 대응하는 스케일 팩터(scale factor)에 기초하는 배치범 정규화(batchnorm optimization) 연산을 수행하는 전자 장치.
- [0018] 상기 NPU는, 상기 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대한 상기 입력 데이터의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수보다 작은 경우, 상기 컴파일러를 이용하여 가중치 커널맵을 미리 결정된 크기로 재구성하고, 상기 재구성된 가중치 커널맵의 채널맵 수가 상기 NPU의 최소 연산 가능 데이터의 수의 배수가 되도록, 부족한 채널맵을 추가하는 전자 장치.
- [0019] 상기 CPU는, 상기 입력 데이터에 대한 가중치 커널맵의 스위핑(sweeping)에 따른 최소 연산 가능 수에 기초하여, 상기 입력 데이터의 채널맵의 수를 재구성하는, 전자 장치.

도면의 간단한 설명

- [0020] 도 1은 뉴럴 네트워크 모델의 동작을 구현하기 위한 비교실시예를 도시한다.
- 도 2는 BNN 또는 QNN 연산을 선택적으로 수행할 수 있는 비교실시예를 도시한다.
- 도 3은 일 실시예에 따른 뉴럴 네트워크 모델의 BNN(binarized neural network) 연산 또는 QNN(quantized neural network) 연산을 선택적으로(selectively) 수행하는 전자 장치를 도시한다.
- 도 4는 일 실시예에 따른 전자 장치를 동작하기 위한 명령어 세트 아키텍처를 도시한다.
- 도 5는 일 실시예에 따른 전자 장치의 동작 흐름도이다.
- 도 6은 복수의 레이어를 포함하는 뉴럴 네트워크 모델과 일 실시예에 따른 전자 장치의 동작을 매핑하기 위한 도면이다.
- 도 7은, 일 실시예에 따른 전자 장치가 BNN 컨볼루션 노드를 인식하는 방법을 도시한다.
- 도 8a 내지 도 8b는 일 실시예에 따른 전자 장치에서 수행되는 데이터 재구성 기법을 나타낸다.
- 도 9는 일 실시예에 따른 전자 장치로 구현할 수 있는 작업 수준 파이프라인 병렬 처리(task-level pipeline parallelism, TLPP)를 도시한다.
- 도 10 내지 도 11은 일 실시예에 따른 전자 장치의 성능을 나타내는 그래프이다.

발명을 실시하기 위한 구체적인 내용

- [0021] 실시예들에 대한 특정한 구조적 또는 기능적 설명들은 단지 예시를 위한 목적으로 개시된 것으로서, 다양한 형태로 변경되어 구현될 수 있다. 따라서, 실제 구현되는 형태는 개시된 특정 실시예로만 한정되는 것이 아니며, 본 명세서의 범위는 실시예들로 설명한 기술적 사상에 포함되는 변경, 균등물, 또는 대체물을 포함한다.
- [0022] 제1 또는 제2 등의 용어를 다양한 구성요소들을 설명하는데 사용될 수 있지만, 이런 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 해석되어야 한다. 예를 들어, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.
- [0023] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다.
- [0024] 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 설명된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함으로써 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0025] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 해당 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는

는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

- [0026] 이하, 실시예들을 첨부된 도면들을 참조하여 상세하게 설명한다. 첨부 도면을 참조하여 설명함에 있어, 도면 부호에 관계없이 동일한 구성 요소는 동일한 참조 부호를 부여하고, 이에 대한 중복되는 설명은 생략하기로 한다.
- [0027] 도 1은 뉴럴 네트워크 모델의 동작을 구현하기 위한 비교실시예를 도시한다.
- [0028] 비교실시예(100)는 딥러닝 연산을 수행하기 위하여, 진화하는 워크로드에 맞춰 확장할 수 있는 프로그래밍 가능한 딥러닝 아키텍처에 대응할 수 있다. 예를 들어, 비교실시예(100)는 다목적 텐서 가속기(versatile tensor accelerator)에 대응할 수 있다. 비교실시예(100)는 복수의 레이어를 포함하는 뉴럴 네트워크 모델의 하나의 레이어에서의 연산 동작을 수행하는 장치에 대응할 수 있다. 예를 들어, 2개의 레이어를 포함하는 뉴럴 네트워크 모델을 가정할 때, 비교실시예(100)가 1회 동작하는 것은, 뉴럴 네트워크 모델에 포함된 레이어 중 제1 레이어에서의 연산에 대응할 수 있다. 마찬가지로, 비교실시예(100)가 1회 동작 후, 2번째 동작하는 것은, 뉴럴 네트워크 모델에 포함된 레이어 제1레이어의 출력을 입력으로 받는 제2레이어에서의 연산에 대응할 수 있다. 따라서, 비교실시예(100)가 N회 반복 동작하는 것은, 뉴럴 네트워크 모델에 포함된 N개의 레이어의 동작에 대응할 수 있다.
- [0029] 비교실시예(100)는 메모리(예: DRAM) 및 NPU(neural processing unit)를 포함할 수 있다. 도1에 도시되지는 않았으나, 비교실시예(100)는 CPU(central processing unit)(미도시)를 포함할 수 있다.
- [0030] 비교실시예(100)에 포함된 NPU는 로드 모듈(load module)(110), 계산 모듈(compute module)(120) 및 저장 모듈(store module)(130)을 포함할 수 있다. 비교실시예(100)는 메모리에서 NPU의 로드 모듈(110)로 뉴럴 네트워크 모델의 해당 레이어에 대한 입력 데이터 및 가중치를 로딩할 수 있다. 비교실시예(100)는 로딩된 입력 데이터 및 가중치를 로드 모듈(110)에서 계산 모듈(120)로 전달할 수 있다. 계산 모듈(120)은 전달받은 입력 데이터 및 가중치에 기초하여, 뉴럴 네트워크의 해당 레이어에 대응하는 가중합 연산을 수행할 수 있다. 예를 들어, 계산 모듈(120)은 입력 데이터 및 가중치를 곱하고, 대응하는 바이어스를 더해줌으로써 가중합 결과를 생성할 수 있다. 계산 모듈(120)은 생성된 가중합 결과를, 저장 모듈(130)에 저장할 수 있다. 저장 모듈(130)은 계산 모듈(120)로부터 전달받은 가중합 결과를 축적함으로써, 해당 레이어의 출력에 대응하는 컨볼루션 연산 결과를 메모리로 저장할 수 있다.
- [0031] 계산 모듈(120)은 GEMM(General matrix multiply) 코어(140)를 포함할 수 있다. GEMM 코어(140)는 로드 모듈(110)로부터 전달받은, 입력 데이터 및 가중치에 기초하여 가중합 연산을 수행할 수 있다. 다만, 비교실시예(100)의 GEMM 코어(140)는 가중합 연산으로서, 8-bit 기반의 행렬곱을 지원할 수 있을 뿐, 1-bit 기반의 행렬곱은 진행하지 못할 수 있다. 예를 들어, GEMM 코어(140)에서는 int8 GEMM 연산을 수행할 수 있으나, 이진 가중합 연산은 수행하지 못할 수 있다. 따라서, 확장된 BNN을 구현하기 위해서는, 8-bit 기반 행렬곱 뿐만 아니라, 1-bit 기반 행렬곱을 모두 지원할 수 있는 GEMM 코어 아키텍처가 요구될 수 있다. 이하에서는 요구되는 GEMM 코어의 스펙을 충족하는 새로운 GEMM 코어 구조를 포함하는 전자 장치들에 대하여 설명한다.
- [0032] 도 2는 BNN 또는 QNN 연산을 선택적으로 수행할 수 있는 비교실시예를 도시한다.
- [0033] 비교실시예(200)는 도 1에 도시된 비교실시예(100)에 하드웨어를 추가한 아키텍처일 수 있다. 예를 들어, 비교실시예(200)는 도 1에 도시된 비교실시예(100)에 추가로 MICRO-OP SRAM 및 제어기(예: controller)를 추가한 하드웨어 아키텍처를 포함할 수 있다. 비교실시예(200)는 도 1에 도시된 비교실시예(100)의 계산 모듈에 GEMM 코어(210)를 추가한 하드웨어 아키텍처를 포함할 수 있다. 비교실시예(200)는 기존 GEMM 코어를 이용하여 로드 모듈로부터 전달받은 입력 데이터 및 가중치 간의 기존 연산(예: int8 multiplication)을 수행하고, 추가 GEMM 코어(210)를 이용하여 로드 모듈로부터 전달받은 입력 데이터 및 가중치 간의 BNN에 대응하는 연산을 수행할 수 있다. 예를 들어, 비교실시예(200)는 기존 GEMM 코어를 이용하여 QNN 연산에 대응하는 8-bit 기반의 int8 multiplication을 수행할 수 있고, 추가 GEMM 코어(210)를 이용하여 BNN 연산에 대응하는 Xnor 및 popcount 기반의 비트와이즈(bitwise) 곱셈(multiplication)을 수행할 수 있다. 따라서, 비교실시예(200)는 BNN 및 QNN을 제어기를 통해 선택적으로 수행할 수 있는 듀얼 모드 아키텍처에 대응할 수 있다.
- [0034] 비교실시예(200)는 도 1의 비교실시예(100)와 비교하여, 계산 모듈에 추가적으로 GEMM 코어(210)를 추가함으로써 구현될 수 있다. 즉, 비교실시예(200)는 하드웨어를 추가함으로써 쉽게 구현될 수 있고, 추가 GEMM 코어

(210) 이외의 또 다른 GEMM 코어를 추가함으로써, 뉴럴 네트워크 연산에 포함된 다른 다양한 연산을 구현할 수 있다. 그러나, 비교실시예(200)는 추가 하드웨어가 많이 소모되고, 차지하는 부피가 크다는 단점이 있을 수 있다. 또한, 비교실시예(200)는 기존 GEMM 코어 및 추가 GEMM 코어(210)가 동시에 실행될 수 있더라도, 동일한 입력/출력 버퍼를 공유하기 때문에, 하나의 GEMM 코어로부터 유용한 계산이 수행될 수 있다.

[0035] 따라서, 비교실시예(200)에 비해 하드웨어를 적게 사용함으로써 차지하는 부피를 줄이고 생산 비용을 줄임과 동시에, 비교실시예(200)처럼 QNN 연산 뿐만 아니라 BNN 연산을 지원하는 하드웨어 아키텍처가 요구될 수 있다. 이하에서는, 추가 GEMM 코어(210)를 부가하는 방식이 아닌, 기존의 GEMM 코어에 기능 유닛(function unit, FU)를 부가함으로써 차지하는 부피 및 비용을 줄이는 방안에 대해 설명한다.

[0036] 도 3은 일 실시예에 따른 뉴럴 네트워크 모델의 BNN(binarized neural network) 연산 또는 QNN(quantized neural network) 연산을 선택적으로(selectively) 수행하는 전자 장치를 도시한다.

[0037] 전자 장치(300)는 복수의 레이어를 포함하는 뉴럴 네트워크 모델을 구현하는 전자 장치를 포함할 수 있다. 예를 들어 전자 장치(300)는 뉴럴 네트워크 모델의 복수의 레이어 각각에서 요구되는 연산을 모두 지원할 수 있는 장치에 대응할 수 있다. 예를 들어, 뉴럴 네트워크 모델의 제1레이어에서는 입력 데이터 및 가중치에 대한 BNN에 대응하는 컨볼루션 연산이 요구될 수 있고, 이와 달리 제2레이어에서는 입력 데이터 및 가중치에 대한 부동 소수점(floating-point) 컨볼루션 연산이 요구될 수 있다. 따라서, 전자 장치(300) 해당 예시에 대응하여, BNN에 대응하는 컨볼루션 연산 및 부동 소수점 컨볼루션 연산을 모두 지원하는 장치에 대응할 수 있다. 다만, 이는 전자 장치(300)에 의해 지원되는 연산 방식에 대한 일 예시에 불과하며, 전자 장치(300)가 지원할 수 있는 연산 방식은 이에 제한되는 것은 아니다. 따라서, 전자 장치(300)는 8-bit 기반 행렬곱을 지원할 뿐만 아니라, 1-bit 기반 행렬곱을 지원하는 전자 장치를 포함할 수 있다.

[0038] 참고로, BNN은 입력 액티베이션(activation), 출력 액티베이션 및 가중치가 각각 $\{-1, 1\}$ 로 제한되는 뉴럴 네트워크 모델을 나타낼 수 있다. BNN에 포함된 이진화된 레이어에서의 연산은 아래 수학적 식 1과 같이 계산될 수 있다(단, 아래 수학적 식 1은 바이어스에 의한 영향은 생략한다).

수학적 식 1

$$y_j = \sum_{i=1}^N w_{ji} \cdot x_i = 2 \cdot \text{popcount}(\text{XNOR}(w_j, x)) - N \quad (1)$$

[0039]

[0040] 수학적 식 1에서, w 와 x 는 각각 이진화된 가중치 및 입력 액티베이션을 나타내고, y 는 출력 액티베이션을 나타낼 수 있다. 수학적 식 1에서, w 와 x 의 값인 $\{-1, 1\}$ 은 각각 비트 값 0 및 1로 표현될 수 있다. 즉, BNN에서 수학적 식 1에 기초하여 효율적인 비트 단위 연산이 수행될 수 있다. 논문 제목 "M. Rastegari et al., "XNOR-net: Imagenet classification using binary convolutional neural networks," in European conference on computer vision. Springer, 2016, pp. 525-542."에 기초하여 구현된 XNOR-Net에 의해 BNN에 포함된 이진화된

레이어는 이진화로 인한 정보 손실을 줄이기 위해 채널별 스케일 팩터 α_j 를 사용하여 아래 수학적 식 2와 같이 계산될 수 있다.

수학적 식 2

$$y_j = 2 \cdot \alpha_j \cdot \text{popcount}(\text{XNOR}(w_j, x)) - N \quad (2)$$

[0041]

[0043] 전자 장치(300)는 입력 데이터를 비트패킹(bit-packing)하는 CPU(central processing unit)(미도시)를 포함할 수 있다. 전자 장치(300)는 입력 데이터를 CPU를 통해 입력 데이터를 비트패킹함으로써 입력 데이터를 조밀하게 배열하여 메모리 공간을 절약할 수 있다. 전자 장치(300)는 CPU를 통해 입력 데이터를 여러 비트로 이루어진 블록으로 비트패킹할 수 있다. 참고로, 비트패킹은 데이터를 저장하거나 전송할 때, 데이터를 밀집하는 기

법을 나타낼 수 있다. 예를 들어, 비트패킹은 입력 데이터를 바이트(8비트)나 워드의 전체 크기로 제한하지 않고, 필요한 만큼의 비트만 사용하여 저장하는 기법에 대응할 수 있다. 전자 장치(300)에 포함된 CPU에 기초하여 수행되는 입력 데이터의 비트패킹에 대하여, 이하 도 9에서 구체적으로 설명한다.

[0044] 전자 장치(300)는 비트패킹된 입력 데이터를 저장하는 메모리를 포함할 수 있다. 예를 들어, 전자 장치(300)는 DRAM에 대응하는 메모리를 포함할 수 있으나, 메모리의 종류는 이에 제한되는 것은 아니다. 전자 장치(300)에 포함된 메모리는, 입력 데이터, 가중치, 및 일 실시예에 따른 NPU 또는 CPU를 동작하기 위한 명령어들(instructions)을 포함할 수 있다.

[0045] 전자 장치(300)는 컴파일러(compiler)를 이용하여 비트패킹된 입력 데이터 및 가중치 간의 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택하도록 결정하는, 선택 명령어(340)를 생성하는 NPU(neural processing unit)를 포함할 수 있다. 전자 장치(300)는 컴파일러에 기초하여 NPU가 생성한 선택 명령어(340)에 대응하는 값에 기초하여, NPU에 의한 BNN 연산 결과 및 QNN 연산 결과 중 하나를 선택할 수 있다. 선택 명령어(340)에 대해서는 이하 도 4에서 구체적으로 설명한다.

[0046] 전자 장치(300)에 포함되는 NPU는 비트패킹된 입력 데이터 및 가중치를 메모리로부터 로딩하며, 비트패킹된 입력 데이터 및 가중치 간의 가중합 연산에 기초하여 BNN 연산 및 QNN 연산을 수행할 수 있다. 예를 들어, NPU는 GEMM 코어(310)를 포함할 수 있다. GEMM 코어(310)는 입력 버퍼, 가중치 버퍼, 출력 버퍼와 연결될 수 있다. GEMM 코어(310)는 입력 버퍼로부터 입력 데이터(또는 액티베이션)를, 가중치 버퍼로부터 가중치를 입력 받을 수 있다. 또한, GEMM 코어(310)는 출력 버퍼와 입력 데이터 및 가중치 간의 컨볼루션 연산 결과에 대응하는 출력 데이터(또는 액티베이션)를 주고받을 수 있다. GEMM 코어(310)는 입력 데이터와 가중치에 기초하여 QNN 연산 및 BNN 연산을 수행할 수 있다. 예를 들어, GEMM 코어(310)는 8-bit 기반의 QNN 연산을 수행하는 멀티플라이어(multiplier)(320) 및 1-bit 기반의 BNN 연산을 수행하는 Xnor-popcount(330)를 포함할 수 있다. 따라서, NPU는 GEMM 코어(310)에 기초하여, 입력 데이터 및 가중치 간의 BNN 연산 및 QNN 연산을 모두 지원하는 장치에 대응할 수 있다. 참고로, GEMM 코어(310)에 기초하여 수행되는 BNN 연산은 바이너리(binary) GEMM으로도 표현될 수 있다. 또한, GEMM 코어(310)에 기초하여 수행되는 QNN 연산은 정수 GEMM으로도 표현될 수 있으며, 이하에서는 설명하는 BNN 연산은 바이너리 GEMM으로도 대체될 수 있으며, QNN 연산도 정수 GEMM으로 대체될 수 있다.

[0047] 이하 알고리즘 1은 일 실시예에 따른 GEMM 코어(310)의 연산 동작 슈도 코드(pseudo code)를 나타낼 수 있다.

Data: W, H = GEMM core width, height
Data: $weight[W][H], input[H][1]$

```

1 begin
2   GEMM_res = 0
3   foreach  $w$  in range(0,  $W$ ) do
4     MUL_res = sum( $weight[w] * input$ )
5     BNN_res = popcount(XNOR( $weight[w]$ ,
6        $input$ )) * 2 -  $H$ 
7     if  $binary\_mode$  then
8       | GEMM_res += BNN_res
9     else
10      | GEMM_res += MUL_res

```

[0048]

[0049] 알고리즘 1에서, W 및 H 는 GEMM 코어(310)의 너비와 높이를 나타낼 수 있다. 참고로, GEMM 코어(310)의 너비 및 높이 각각은 입력 데이터와 가중치 간의 행렬-벡터곱 연산이 수행될 수 있는 최소 단위의 행 및 열 배열의 수에 대응할 수 있다. GEMM 코어(310)에 기초하여, 가중치의 크기는 $weight[W][H]$ 로 결정되고, 입력 데이터는 $[H][1]$ 의 크기를 가질 수 있다. 예를 들어, GEMM 코어(310)의 크기가 3×3 의 크기를 가지는 경우, 가중치의 크기는 $weight[3][3]$ 로 결정되고, 입력 데이터는 $[3][1]$ 의 크기를 가질 수 있으나, 이는 일 예시에 불과할 뿐 이에 제한되는 것은 아니다. 알고리즘 1에서, GEMM 코어(310)는 4행의 곱셈 및 sum 함수에 의하여 정수형 데이터의 행렬-벡터곱 연산을 수행하고 해당 연산 결과를 MUL_res라는 임시변수에 저장할 수 있다. 알고리즘 1에서,

GEMM 코어(310)는 5행의 XNOR 및 popcount 함수에 의하여 binary 데이터 타입의 행렬-벡터곱 연산을 수행하고 해당 연산 결과를 BNN_res라는 임시 변수에 저장할 수 있다. 알고리즘 1의 6행 내지 9행에서, GEMM 코어(310)는 binary mode 인 경우 BNN_res 결과를 누적하고, binary mode가 아닌 경우 MUL_res 결과를 누적할 수 있다. 참고로, binary mode인지 여부는, NPU의 GEMM 코어(310)가 동작하기 전에, 전자 장치가 컴파일러를 이용하여 미리 생성한 선택 명령어(예: USE_BIN)의 값에 따라 결정될 수 있다. 미리 생성(또는 결정)되는 선택 명령어에 대하여는 이하 도 4에서 구체적으로 설명한다.

[0050] 전자 장치(300)는 컴파일러를 이용하여 생성한 선택 명령어(340)에 기초하여, BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택(select)하고, BNN 연산 결과 또는 QNN 연산 결과 중 선택된 연산 결과를 축적기(예: accumulator)에 누적할 수 있다. 예를 들어, 전자 장치(300)의 NPU는 GEMM 코어(310)에서의 QNN 연산 및 BNN 연산이 수행되기 이전에, GEMM 코어(310)에서 수행되는 연산에 대응하는 뉴럴 네트워크 모델의 레이어에 대응하여 선택 명령어(340)를 미리 지정할 수 있다. 예를 들어, 선택 명령어(340)는 복수의 레이어를 포함하는 뉴럴 네트워크 모델에 대하여, 뉴럴 네트워크 모델에 포함된 제1 레이어가 int8 convolution 연산을 수행하는 레이어인 경우, 선택 명령어(340)는 제1 선택 명령어로 미리 지정될 수 있다. 또한, 뉴럴 네트워크 모델에 포함된 제2 레이어가 binary convolution 연산이 수행되는 레이어인 경우, 선택 명령어(340)는 제2 선택 명령어로 미리 지정될 수 있다. 전자 장치(300)는 미리 지정된 선택 명령어(340)에 기초하여, BNN 연산 결과 또는 QNN 결과 중 하나를 선택할 수 있다. 전자 장치(300)는 선택된 연산 결과를 축적기에 누적할 수 있다. 전자 장치(300)가 선택된 연산 결과를 축적기에 누적하는 동작은 컨볼루션 연산에서 입력 데이터와 가중치를 곱한 후 해당 결과를 누적하여 더하는(add) 동작에 대응할 수 있다. 전자 장치(300)는 축적기에 누적된 최종 결과 값을 저장 모듈(예: store module)에 저장할 수 있다.

[0051] 정리하면, 도 1에 도시된 비교실시예(100)와 같은 오픈 소스로 프로그래밍 가능한 NPU 아키텍처인 VTA(versatile tensor accelerator)는 로드(load) 명령어, 저장(store) 명령어, GEMM(General matrix multiply) 명령어, ALU(Arithmetic Logic Unit) 명령어라는 4개의 128 비트 명령어로 동작할 수 있다. 특히, 비교실시예(100)에서의 GEMM의 성능이 가장 중요하며, 실제 컨볼루션 및 완전 연결 레이어 작업을 수행할 수 있다. 새로운 1비트 필드에 표시된 대로 정수 GEMM 또는 이진 GEMM 연산을 수행하도록 GEMM 명령어를 확장할 필요가 있다. 도 3의 전자 장치(300)는, Xnor-popcount(330) 및 선택 명령어(340)를 추가함으로써, GEMM 명령어의 확장을 달성할 수 있다.

[0052] 도 4는 일 실시예에 따른 전자 장치를 동작하기 위한 명령어 세트 아키텍처를 도시한다.

[0053] 일 실시예에 따른 명령어 세트 아키텍처(instruction set architecture)(이하, 'ISA')(400)는 도 3에 도시된 전자 장치(300)가 동작하기 위해 필요한 명령어들을 포함할 수 있다.

[0054] 참고로, ISA(400)는 컴퓨터 아키텍처의 일부로서, 프로그래밍 언어의 기계어 수준에서 컴퓨터와 상호작용하는 방법을 정의하기 위한 명령어들의 집합일 수 있다. 예를 들어, ISA(400)는 산술 연산, 데이터 이동, 로직 연산, 제어 연산 등의 연산 명령어를 포함할 수 있다.

[0055] 일 실시예에 따른, ISA(400)는 선택 명령어(410)를 저장하는 저장 공간을 포함할 수 있다. 예를 들어, ISA(400)의 선택 명령어(410)는 1 비트의 정보를 나타내는 명령어에 대응할 수 있다. 예를 들어, 선택 명령어(410)는 1 또는 0의 값에 대응할 수 있으나 이에 제한되는 것은 아니다. ISA(400)는 전자 장치가 생성한 선택 명령어(410)를 저장할 수 있다. ISA(400)는 더 나아가 전자 장치가, 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어의 출력이 BNN 연산 결과에 기초하는 출력인 경우, 컴파일러를 이용하여 로직 하이(logic high) 값에 대응하여 생성한 제1 선택 명령어를 저장할 수 있다. 예를 들어, 전자 장치가 컴파일러를 통해 미리 선택 명령어(410) 값을 활성화 값에 대응하는 1로 생성한 경우, ISA(400)의 선택 명령어(410)는 1의 값을 저장할 수 있다.

[0056] 다른 예를 들어, ISA(400)는 전자 장치가, 해당 레이어의 출력이 QNN 연산 결과에 기초하는 출력인 경우, 컴파일러를 이용하여 생성한 로직 로우(logic low) 값에 대응하는 제2 선택 명령어를 선택 명령어(410)에 저장할 수 있다. 예를 들어, 전자 장치가 컴파일러를 통해 미리 선택 명령어(410) 값을 비활성화 값에 대응하는 0으로 생성한 경우, ISA(400)의 선택 명령어(410)는 0의 값을 저장할 수 있다.

[0057] 일 실시예에 따른 전자 장치는 ISA(400)에 저장된 선택 명령어(410)의 값에 대응하여, BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택할 수 있다. 예를 들어, 전자 장치가 컴파일러를 이용하여 로직 하이 값(예: 1)에 대응하는 제1 선택 명령어를 생성하는 경우, ISA(400)는 1의 값을 선택 명령어(410)에 저장할 수 있다. 전자 장

치가 NPU를 통해 BNN 및 QNN 연산을 모두 수행하는 경우, 전자 장치는 1의 값에 대응하는 선택 명령어(410)에 기초하여, BNN 연산 결과를 선택하여 누적할 수 있다. 다른 예를 들어, 전자 장치가 컴파일러를 이용하여 로직로우 값(예: 0)에 대응하는 제2 선택 명령어를 생성하는 경우, ISA(400)는 0의 값을 선택 명령어(410)에 저장할 수 있다. 전자 장치가 NPU를 통해 BNN 및 QNN 연산을 모두 수행하는 경우, 전자 장치는 0의 값에 대응하는 선택 명령어(410)에 기초하여, QNN 연산 결과를 선택하여 누적할 수 있다.

[0058] 도 5는 일 실시예에 따른 전자 장치의 동작 흐름도이다.

[0059] 단계(510)에서, 일 실시예에 따른 전자 장치의 CPU(central processing unit)는 입력 데이터를 비트패킹(bit-packing)하고, 비트패킹된 입력 데이터를 메모리에 저장할 수 있다. 예를 들어, 전자 장치는 입력 데이터를 비트패킹하고, DRAM에 해당 데이터를 저장할 수 있다.

[0060] 단계(520)에서, 일 실시예에 따른 전자 장치의 NPU(neural processing unit)는 컴파일러를 이용하여 비트패킹된 입력 데이터 및 가중치 간의 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택하도록 결정하는, 선택 명령어를 생성할 수 있다. 선택 명령어는 도 4에 도시된 ISA(400)에 포함된 선택 명령어(410)에 대응할 수 있으며, 중복되는 내용에 대하여는 설명을 생략한다.

[0061] 단계(530)에서, 일 실시예에 따른 전자 장치의 NPU는 비트패킹된 입력 데이터 및 가중치를 메모리로부터 로딩하고, 비트패킹된 입력 데이터 및 가중치 간의 가중합 연산에 기초하여 BNN 연산 및 QNN 연산을 수행할 수 있다. 예를 들어, NPU는 비트패킹된 입력 데이터 및 가중치를 DRAM으로부터 로드 모듈(load module)을 거쳐 계산 모듈(compute module)로 전달할 수 있다. NPU는 비트패킹된 입력 데이터 및 가중치를 계산 모듈에 포함된 멀티플라이어(multiplier) 및 Xnor-popcount에 입력할 수 있다. NPU의 멀티플라이어는 비트패킹된 입력 데이터 및 가중치 간의 QNN 연산을 수행할 수 있고, 동시에 Xnor-popcount는 BNN 연산을 수행할 수 있다. 참고로, BNN 연산은 binary convolution에 대응할 수 있고, QNN 연산은 int8 convolution에 대응할 수 있다.

[0062] 단계(540)에서, 일 실시예에 따른 전자 장치의 NPU는 컴파일러로부터 전달받은 선택 명령어에 기초하여, BNN 연산 결과 또는 QNN 연산 결과 중 하나를 선택할 수 있다. 예를 들어, NPU의 멀티플라이어를 통해 비트패킹된 입력 데이터 및 가중치 간의 QNN 연산 결과가 생성되고, 동시에 NPU의 Xnor-popcount를 통해 입력 데이터 및 가중치 간의 BNN 연산 결과가 생성될 수 있다. 전자 장치의 NPU는 컴파일러로부터 전달받은 선택 명령어가 예를 들어, 활성화화를 나타내는 1의 값에 대응하는 경우, Xnor-popcount를 통해 생성된 BNN 연산 결과를 선택할 수 있다. 다른 예를 들어, 전자 장치의 NPU는 컴파일러로부터 전달받은 선택 명령어가 예를 들어, 비활성화를 나타내는 0의 값에 대응하는 경우, 멀티플라이어를 통해 생성된 QNN 연산 결과를 선택할 수 있다.

[0063] 단계(550)에서, 일 실시예에 따른 전자 장치의 NPU는 BNN 연산 결과 또는 QNN 연산 결과 중 선택된 연산 결과를 축적할 수 있다. 예를 들어, 전자 장치의 NPU는 선택 명령어의 값에 기초하여, 선택된 BNN 연산 결과 또는 QNN 연산 결과 중 하나를 축적기(예: accumulator)에 누적할 수 있다. NPU는 누적되는 연산 결과들을 주기적으로 합산해줄 수 있다. 전자 장치의 NPU는 축적기의 최종 누적 결과를 저장 모듈(store module)에 저장할 수 있다.

[0064] 도 6은 복수의 레이어를 포함하는 뉴럴 네트워크 모델과 일 실시예에 따른 전자 장치의 동작을 매핑하기 위한 도면이다.

[0065] 일 실시예에 따른 뉴럴 네트워크 모델(600)은 복수의 레이어를 포함할 수 있다. 예를 들어, 뉴럴 네트워크 모델(600)은 컨볼루션 레이어, 배치놈 정규화(예: batch norm) 레이어, 및 부호(sign) 레이어를 포함할 수 있다. 뉴럴 네트워크 모델(600)의 각 레이어에서는 컨볼루션 연산, 배치놈 정규화 연산이 각각 수행될 수 있다. 레이어들(610)에서 수행되는 연산은 일 실시예에 따른 전자 장치의 CPU 및 NPU에서 수행될 수 있다.

[0066] 예를 들어, 전자 장치는 NPU 연산에 앞서, 컴파일러를 이용하여 뉴럴 네트워크 모델(600)에 포함된 배치놈 정규화 레이어에서의 배치놈 정규화(batch norm) 연산에 대응하는 연산을 수행할 수 있다. BN(Batch Normalization) 레이어는 컨볼루션 레이어의 바로 다음에 배치될 수 있다. 전자 장치의 모델(600)을 통한 추론을 위해, BN 레이어의 파라미터는 동결되어 아핀 변환(affine transformation)으로 축소될 수 있다. 따라서, 컨볼루션 레이어와 뒤따르는 아핀 변환된 레이어는 BN 폴딩(BN folding)을 통해 단일 컨볼루션 레이어로 결합될 수 있다. 그러나 기존 BN 폴딩 방식은 가중치의 정밀도가 충분할 경우에만 동작 가능하며, 이전 컨볼루션 레이어의 가중치와 바이어스를 변경하거나, BNN의 경우 BN 레이어의 출력이 즉시 이진화된다는 가정에 따라 바이어스만 변경하는 방식이었다(즉, 부호 연산이 뒤따름). 그러나, 뉴럴 네트워크 모델(600)을 참조하면, 레이어들(610)은 이진 컨볼루션을 포함하지만, 부호 레이어를 포함하지 않는다. 전자 장치는 레이어들(610)이 부호 레

이어를 포함하지 않는 경우에도, 컴파일러를 이용하여, 이진 컨볼루션(binary convolution) 연산에 대응하는 스케일 팩터(scale factor)에 기초하는 배치놈 정규화(batchnorm optimization) 연산을 수행할 수 있다. 예를 들어, 전자 장치는 BN 파라미터로 가중치와 바이어스가 아닌 스케일 팩터와 바이어스에 기초하여 BN 폴딩을 수행할 수 있다. 이는 이하의 수학적식 3 및 4를 통해, 도출된 수학적식 5로 설명될 수 있다.

수학적식 3

$$y = W \cdot x \approx \alpha \cdot \text{sign}(W) \cdot x = \alpha \cdot (W_b \cdot x) \quad (3)$$

수학적식 3에서, W_b 는 이진화된 가중치이고 α 는 채널 별 스케일 팩터를 나타낼 수 있다.

수학적식 4

$$z = \gamma \cdot \frac{y - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \quad (4)$$

수학적식 4에서, z 는 BN 레이어의 출력을 나타낼 수 있고, y 는 컨볼루션 연산의 결과값을 나타낼 수 있다.

수학적식 5

$$\begin{aligned} z &= \gamma \cdot \frac{\alpha \cdot W_b \cdot x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \\ &= \frac{\alpha \cdot \gamma}{\sqrt{\sigma^2 + \epsilon}} (W_b \cdot x) - \gamma \frac{\mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \end{aligned}$$

수학적식 5는 수학적식 3 및 수학적식 4를 통해 도출될 수 있으며, 수학적식 5에서, γ 및 β 는 배치놈 정규화 한 값을 스케일과 시프트(shift)하기 위한 배치놈 연산에서의 학습 가능한 가중치(weight)를 나타낼 수 있다. 따라서, BN 폴딩 후 이진화된 레이어에 대한 새로운 스케일 팩터와 바이어스는 아래 수학적식 6 및 수학적식 7을 통해 계산될 수 있다.

수학적식 6

$$Scale_{BinConv} = \frac{\alpha \cdot \gamma}{\sqrt{\sigma^2 + \epsilon}}$$

수학식 7

$$Bias_{\text{BinConv}} = \beta - \gamma \frac{\mu}{\sqrt{\sigma^2 + \epsilon}}$$

[0074]

[0076]

레이어들(610)에서 이진 컨볼루션 연산(예: binary conv2d 또는 conv2d)이 수행되기에 앞서, 레이어들(610)에서 입력 데이터(또는 입력 액티베이션) 및 가중치들이 먼저 이진화되고, 비트패킹되어야 한다. 참고로, 부호 레이어(예: Sign layer)에서의 부호 연산(예: Sign)은 이진화에 사용될 수 있으나, 부호 연산의 결과는 각 배열 요소가 1 또는 -1인 정수 배열로 메모리에 저장될 수 있다. 전자 장치의 GEMM 코어는 입력 데이터로 비트 벡터를 포함할 수 있으므로, 정수 배열을 비트 벡터로 변환해야 한다. 전자 장치가 입력 데이터 중 정수 배열을 비트 벡터로 변환하는 것이 비트패킹(bit-packing)에 대응할 수 있다. 입력 데이터를 이진화 하고, 입력 데이터를 비트패킹하는 것은 런타임 이전에 수행되어야 하므로, 레이어들(610) 중 이진 컨볼루션 전에 CPU에 의해 수행될 수 있다.

[0077]

종합하면, 전자 장치는 레이어들(610)에서 CPU를 통해 입력 데이터(예: feature)에 대하여 부호 연산(예: Sign) 및 비트패킹 연산(예: Bit-pack)을 수행할 수 있다. 전자 장치의 NPU는 비트패킹된 입력 데이터 및 가중치들(예: W1, W2)에 기초하여 binary conv2d 또는 Conv2d와 같은 이진 컨볼루션 연산을 수행함으로써, 뉴럴 네트워크를 동작할 수 있다.

[0078]

도 7은, 일 실시예에 따른 전자 장치가 BNN 컨볼루션 노드를 인식하는 방법을 도시한다.

[0079]

일 실시예에 따른 전자 장치는, BNN 컨볼루션 레이어(또는 노드)(710, 720)를 인식하기 위한 컴파일러로 동작할 수 있다. 전자 장치는 DNN 프레임워크에 고유한 이진 컨볼루션 및 완전 연결 레이어(Fully connected layer) 연산자를 포함함으로써, 컴파일러의 프런트-엔드에서 이진화된 컨볼루션 레이어(또는 노드)(710, 720)를 인식할 수 있다. 예를 들어 전자 장치는, BNN 컨볼루션 노드(710, 720)의 바로 앞에, 부호 노드(예: Sign node)가 배치된다는 점을 이용하여, 부호 노드의 바로 다음에 나오는 레이어를 BNN 컨볼루션 노드(710, 720)로 인식할 수 있다. 이러한 방식을 통해 ONNX(open neural network exchange) 형식을 변경할 필요가 없을 수 있다. 참고로, ONNX 형식은 다른 DNN 프레임 워크 환경(예: Tensorflow, PyTorch)에서 만들어진 모델들을 서로 호환되게 사용할 수 있도록 만들어진 공유 플랫폼을 포함할 수 있다. 따라서, 이진화된 계산 노드를 생성할 때 스케일 팩터를 저장하기 위해 채널 단위로 스케일 팩터라는 파라미터를 추가할 수 있다.

[0080]

도 8a 내지 도 8b는 일 실시예에 따른 전자 장치에서 수행되는 데이터 재구성 기법을 나타낸다.

[0081]

예를 들어, 전자 장치는 입력 데이터의 채널 단위로 타일링(tiling) 하여 순차적으로 계산 모듈에서 GEMM 연산을 수행하는 NPU를 포함할 수 있다. 해당 NPU에 인가되는 입력 데이터의 수가, NPU에 포함된 계산 모듈의 수보다 작은 경우 효율적으로 컨볼루션 연산이 수행되지 못할 수 있다. 예를 들어, 이미지 처리 CNN의 첫 번째 컨볼루션 레이어는 일반적으로 3개와 같이 매우 낮은 입력 채널수를 가질 수 있다. 더 나아가, 예를 들어 뉴럴 네트워크 모델의 입력 레이어의 경우 입력 데이터의 채널 수가 3개이면, NPU에 포함된 GEMM 코어를 통해 연산될 때 효율성이 떨어질 수 있다. 구체적으로, GEMM 코어는 두 가지 차원(binary 또는 int8)을 입력 및 출력 채널 차원에 매핑하므로, 채널 수가 GEMM 하드웨어 차원(예: 16 X 16)보다 적으면, GEMM 코어의 전체 컴퓨팅 기능이 활용되지 못할 수 있다. 예를 들어 3개의 입력 채널만을 사용하면 컴퓨팅 성능의 약 81%가 낭비될 수 있다. 낮은 활용도를 해결하기 위해 컨볼루션을 포인트와이즈 컨볼루션으로 변환할 수 있다.

[0082]

따라서, 도 8a 내지 도 8b에서는 기존의 방식인 입력 데이터의 채널 수를 계산 모듈의 크기에 맞게 zero padding 하는 대신, 새로운 재구성 기법에 대하여 설명한다.

[0083]

도 8a는 전자 장치에 의해 수행되는 가중치의 재구성 기법(800)을 나타낸다.

[0084]

전자 장치는, 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대한 입력 데이터의 채널맵 수가 NPU의 최소 연산 가능 데이터의 수보다 작은 경우, 컴파일러를 이용하여 가중치 커널맵을 미리 결정된 크기로 재구성하고, 재구성된 가중치 커널맵의 채널맵 수가 NPU의 최소 연산 가능 데이터의 수의 배수가 되도록, 부족한 채널맵을 추가할 수 있다. 예를 들어, 가중치가 [64, 3, 7, 7]의 크기를 나타낸다고 가정할 수 있다. 이 때, 가중치의

크기를 나타내는 수치들은 예를 들어, [출력 채널수, 입력 채널수(810), 높이(820), 너비(830)]를 나타낼 수 있다. 전자 장치는 재구성 기법(800)을 통해, 포인트와이즈 컨볼루션(pointwise convolution)을 구현하기 위해, 가중치 커널의 높이(820) 및 너비(830)를 1로 만들고, 해당 높이(820) 및 너비(830)에 대응하도록 나머지 데이터를 서로 다른 가중치 채널 맵으로 재구성할 수 있다. 예를 들어, 가중치 커널맵에 포함된 가중치가 $X_{0,0}$, $X_{0,1}$, 내지 $X_{6,6}$ 에 대응하는 경우, 전자 장치는 재구성 기법(800)을 통해, $X_{0,0}$ 부터 $X_{6,6}$ 까지를 서로 다른 채널 방향(840)으로 일렬로 배치할 수 있다. 따라서, [64, 3, 7, 7] 크기의 가중치는 재구성 기법(800)을 통해 [64, 147, 1, 1]의 크기로 변환될 수 있다. 이 때, 전자 장치에 포함된 제안된 계산 모듈(compute module)의 최소 연산 단위가 16을 나타내는 경우, 가중치의 채널 수를 16의 배수로 변환할 수 있다. 예를 들어, 전자 장치는 [64, 147, 1, 1] 크기의 변환된 가중치에 추가로 채널 방향(850)으로 zero padding을 수행할 수 있다. 따라서, 전자 장치는, NPU에 의한 컨볼루션 연산 전에, 뉴럴 네트워크 모델에 포함된 레이어에 대한 입력 데이터의 채널수가, 계산 모듈의 크기에 비하여 적다고 판단되는 경우, NPU에 의한 컨볼루션 연산 전에 가중치를 재구성 기법(800)으로 재구성할 수 있다. 전자 장치는 재구성 기법(800)으로 재구성된 가중치를 NPU 연산시 메모리로부터 로딩하여 컨볼루션 연산을 수행할 수 있다.

[0085] 도 8b는 전자 장치에 의해 수행되는 입력 데이터의 재구성 기법(860)을 나타낸다.

[0086] 전자 장치의 CPU는 입력 데이터에 대한 가중치 커널맵의 스위핑(sweeping)에 따른 최소 연산 가능 수에 기초하여, 입력 데이터의 채널맵의 수를 재구성할 수 있다. 예를 들어, 전자 장치에 대한 입력 데이터의 크기가 [1, 3, 244, 244]에 대응하는 경우를 가정한다. 참고로, 입력 데이터의 크기는 [배치 크기(batch size), 채널수(861), 높이(862), 너비(863)]에 대응할 수 있다. 이 경우, 가중치의 커널과 입력 데이터가 연산되는 최소 단위는 [1, 3, 7, 7]에 대응할 수 있는데, 이는 가중치의 크기(본 예에서는 [64, 3, 7, 7])에서 출력 채널수에 대응하는 첫 번째 원소 값인 64를 1로 만든 값에 대응할 수 있다. 해당 예는, 입력 데이터와 가중치 간의 연산에 기초하여 출력 데이터의 채널맵의 수가 최소인 경우를 나타낼 수 있으나, 해당 연산 최소 단위는 이에 제한되는 것은 아니다.

[0087] 전자 장치는 가중치 커널의 스트라이드(stride)(예: 2), 패딩(padding) 그리고 하나의 가중치 커널의 입력 데이터에 대한 연산 최소 단위를 고려하여, 입력 데이터의 너비(863)가 224인 경우, 해당 너비(863)에서 가중치와 입력 데이터 간에 112번의 연산을 수행할 수 있다. 따라서, 전자 장치의 CPU는 입력 데이터에 대하여 가중치의 재구성 기법(800)과 유사한 재구성 기법(860)을 적용함으로써, 채널수(861)는 늘리고, 높이(862) 및 너비(863)는 감소시킬 수 있다. 예를 들어, 전자 장치의 CPU는 [1, 3, 244, 244] 크기의 입력 데이터를 [1, 147, 112, 112] 크기의 입력 데이터로 변환할 수 있다. CPU는 입력 데이터 채널맵의 너비(863)를 따라 배열된 입력 데이터 들을 채널 방향(870)으로 동일한 위치에 배열함으로써 입력 데이터를 변환할 수 있다. 이 때, 전자 장치에 포함된 제안된 계산 모듈(compute module)의 최소 연산 단위가 16을 나타내는 경우, 입력 데이터의 채널맵의 수를 16의 배수로 변환할 수 있다. 즉, CPU는 채널 방향(870)을 따라 배열된 입력 데이터와 동일한 위치에 채널 방향(850)으로 zero padding 함으로써, [1, 160, 112, 112] 크기의 입력 데이터로 기존 입력 데이터를 재구성할 수 있다.

[0088] 종합하면, 재구성 기법(800) 및 재구성 기법(860) 없이, [64, 3, 7, 7] 크기의 가중치와 [1, 3, 244, 244] 크기의 입력 데이터 간의 연산이 스트라이드의 크기가 2로 연산이 수행되는 경우, 전자 장치에 직접 매핑되면 $4 \times 7 \times 7 = 196$ 번 GEMM 코어가 동작함으로써 (16,3) 하위 배열만 활용될 수 있다. 따라서, 전자 장치는, 재구성 기법(800)을 통해 재구성된 가중치 및 재구성 기법(860)을 통해 재구성된 입력 데이터에 기초하여 포인트와이즈 컨볼루션(pointwise convolution)(또는 위치별 컨볼루션)을 수행할 수 있다. 즉, 전자 장치는 GEMM 연산을 $4 \times 10 = 40$ 회 호출하여, GEMM 코어 활용도를 4.9배 향상시킬 수 있다. 전자 장치의 CPU는 NPU 연산 중에 입력 데이터에 대한 재구성 기법(860)을 수행함으로써, 전자 장치에서 효율적으로 컨볼루션 연산이 수행될 수 있다. 전자 장치는 재구성 기법(800, 860)을 통해, 아래 표 1과 같이 하드웨어 측정 결과에 따라 BiRealNet-18의 첫 번째 컨볼루션 레이어에 대해 3.44배 속도 향상을 달성할 수 있다.

표 1

TABLE I: Effect of our FIC convolution mapping method

Layer	Data type	TVM (GOPS)	Ours (GOPS)
1st conv layer (BiRealNet-18)	uint8	15.14	52.09

[0089]

[0091] 도 9는 일 실시예에 따른 전자 장치로 구현할 수 있는 작업 수준 파이프라인 병렬 처리(task-level pipeline parallelism, TLPP)를 도시한다.

[0092] 일 실시예에 따른 전자 장치는 작업(task)을 파이프라이닝(900, 950)할 수 있다. 전자 장치는, 로드 모듈, 계산 모듈에 포함된 GEMM 코어, 및 저장 모듈 간에 작업 수준 파이프라인 병렬 처리를 수행할 수 있다.

[0093] 참고로, 파이프라인 병렬 처리 또는 파이프라이닝은 복수의 명령어들을 중첩하여 실행하는 기술을 나타낼 수 있다. 예를 들어, 전자 장치는 명령어를 실행하는데 사용되는 프로세서의 동작을 복수의 독립적인 단계들로 분할하고 분할된 복수의 독립적인 단계들이 서로 다른 명령어들을 처리하도록 함으로써 프로세서(예: NPU)의 성능을 높일 수 있다.

[0094] 위의 개념을 확장하여, 뉴럴 네트워크를 동작하기 위한 복수의 명령어들을 분할하여 복수의 하드웨어 유닛들에 의해 파이프라이닝이 수행되도록 할 수 있다. 예를 들어, 전자 장치는, 파이프라이닝(900)에서 CPU로 입력 데이터를 NPU의 연산에 앞서 비트패킹(910)할 수 있다. 전자 장치는 파이프라이닝(900)에서 비트패킹(910)된 제1 입력 데이터 및 제1 가중치를 로드 모듈로 로딩할 수 있다. 로드 모듈은 로딩한 제1 입력 데이터 및 제1 가중치를 계산 모듈에 포함된 GEMM 코어로 전달함과 동시에, 다른 제2 입력 데이터 및 제2 가중치를 로드 모듈로 로딩할 수 있다. 계산 모듈의 GEMM 코어는 로드 모듈이 제2 입력 데이터 및 제2 가중치를 로딩하는 동안, 제1 입력 데이터 및 제1 가중치간의 제1 행렬곱연산을 수행할 수 있다. 계산 모듈의 GEMM 코어는 제1 행렬곱연산 결과를 저장 모듈에 전달함으로써, 저장 모듈은 제1 행렬곱연산 결과를 저장할 수 있다. 저장 모듈이 제1 행렬곱연산 결과를 저장할 때, 계산 모듈의 GEMM 코어는 제2 입력 데이터 및 제2 가중치간의 제2 행렬곱연산을 수행할 수 있다. 위의 예시와 같이, 전자 장치는 파이프라이닝(900)을 달성할 수 있다. 다만, 파이프라이닝(900)에서 비트 패킹은 NPU 연산 전에 1회 수행될 수 있다.

[0095] 전자 장치의 파이프라이닝(950)은 파이프라이닝(900)과 달리, 복수의 비트패킹(960)을 수행할 수 있다. 전자 장치의 CPU는 NPU에서 메모리로부터 비트패킹된 제1 입력 데이터(970) 및 제1 가중치를 로딩하도록, 제1 로딩 전에 제1 비트패킹을 수행할 수 있다. 또한, 전자 장치의 CPU는 NPU에서 메모리로부터 비트패킹된 제2 입력 데이터(980) 및 제2 가중치를 로딩하도록, 제2 로딩 전에 제2 비트패킹을 수행할 수 있다. 전자 장치의 CPU는 파이프라이닝(950)에서 NPU가 메모리로부터 입력 데이터를 로딩하기 전에 입력 데이터에 대한 비트패킹(960)을 수행함으로써, 비트 패킹 수행 시간 단축(예: execution savings)을 달성할 수 있다.

[0096] 전자 장치의 CPU는, 파이프라이닝(950)을 통해 뉴럴 네트워크의 복수의 레이어 중, 해당 레이어에 대응하는 BNN 연산 및 QNN 연산이 NPU에서 수행되는 경우, 해당 레이어의 다음 레이어에 대한 입력 데이터를 미리 비트패킹할 수 있다. 예를 들어, 전자 장치의 CPU는 해당 레이어에 대응하는 행렬곱연산이 수행되는 동안 입력 데이터를 비트패킹하여 메모리에 저장할 수 있다. 전자 장치의 NPU에서 해당 레이어의 다음 레이어에 대응하는 연산이 수행되는 경우(예: 도 9의 next tile features에 대응하는 부분), 미리 비트패킹된 입력 데이터(990, 991)를 로딩할 수 있다. 전자 장치는 다음 레이어에 대응하는 연산이 수행될 때, 미리 비트패킹된 입력 데이터(990, 991)를 로딩함으로써, 복수의 비트패킹으로 인해 발생하는 비트 패킹 대기 시간을 부분적으로 숨김으로써 처리량을 늘릴 수 있다.

[0097] 도 10 내지 도 11은 일 실시예에 따른 전자 장치의 성능을 나타내는 그래프이다.

[0098] 제안된 뉴럴 네트워크 모델의 프레임워크와 전자 장치를 평가하기 위해 Resnet-18의 이진화된 버전인 BiRealNet-18 모델을 사용하여 이미지 분류 작업을 수행하였다. 전자 장치의 GEMM 코어는 8비트 정밀도로 16×16의 크기로 구성된다. 바이어스 추가는 32비트로 구현된다. 이하에서 확인한 모든 성능들은 FPGA 보드 상에

실제 측정된 것으로서, Xilinx ZCU102 FPGA 보드는 ARM Cortex-A53 CPU를 포함하는 Zynq UltraScale+ 디바이스를 포함할 수 있다. 다만, 확인된 성능은 일 구현 방법에 따른 성능일 뿐, 제안된 전자 장치의 성능이 이에 제한되는 것은 아니다.

[0099] 아래 표 2는 ZCU102에서 제안된 전자 장치인 확장된 VTA의 FPGA 리소스 활용도를 비교한 것이다.

표 2

	LUT	FF	BRAM	DSP
Original VTA (DSP)	29 465	39 123	139	315
Original VTA (LUT)*	49 613	48 406	139	59
Extended VTA – new GEMM core	55 580 (+12.0%)	55 147 (+13.9%)	139 (0%)	65 (+10.2%)
Extended VTA – new FU within GEMM	53 367 (+7.6%)	53 233 (+9.1%)	139 (0%)	59 (0%)

[0100]

[0101] 표 2에서 클럭 주파수는 오리지널 VTA와 동일한 333Mhz로 설정한다. 표 2에서 오리지널 VTA는 두가지 방식으로 구현될 수 있다. 예를 들어, LUT를 이용하는 방식을 기준으로 제안된 전자 장치와 비교한다. 새로운 GEMM 코어를 추가하는 방식(Extended VTA-new GEMM core)을 사용하더라도, 영역 오버헤드가 작아서 이진 GEMM 동작의 하드웨어 친화성을 확인할 수 있다. 그러나, 새로운 FU(function unit)을 GEMM 에 추가하는 방식은 오버헤드가 LUT/FF의 사용량과 비교하여 8~9%만 증가했을 뿐이다. BRAMs 및 DSP의 사용량은 전혀 증가하지 않았음을 확인할 수 있다.

[0103] 아래 표 3은 BiRealNet-18의 모든 바이너리 컨볼루션 레이어(예: 도 6에 포함된 레이어들(610))를 사용하여, 비교실시예의 TVM 컴파일러 성능과 제안된 전자 장치의 컴파일러 성능을 비교한 것이다.

표 3

H,W	#In, out channels	Kernel size	Pad, Stride	TVM-VTA (GOPS)	Ours (GOPS)	Speed-up
56,56	64,64	3,3	1,1	124.46	294.21	2.36
56,56	64,128	3,3	1,2	103.84	251.21	2.42
28,28	128,128	3,3	1,1	138.80	555.40	4.00
28,28	128,256	3,3	1,2	109.14	462.82	4.20
14,14	256,256	3,3	1,1	148.08	740.63	5.00
14,14	256,512	3,3	1,2	77.25	578.28	7.49
7,7	512,512	3,3	1,1	113.74	812.40	7.14

[0104]

[0105] 표 3은 표 2에서와 마찬가지로, 클럭 주파수를 333Mhz로 설정하여, 제안된 BN 폴딩 기법이 비교실시예의 TVM 케이스에서도 적용되는 경우의 성능을 비교한다. 비교실시예의 TVM의 경우 이진 값($\{-1, 1\}$)이 int8 값으로 표현되므로 연산과 메모리 액세스 오버헤드가 추가될 수 있다. 제안된 전자 장치의 경우 8비트를 하나의 int8 변수에 담기 때문에 이상적인 성능 향상은 8배에 대응할 수 있다. 반면에, CPU에서 실행되는 비트패킹에도 오버헤드가 있을 수 있다. 표 3에서 입력 채널 수가 128개 미만이면 바이너리 GEMM 코어를 완전히 활용할 수 없기 때문에, 표 3의 처음 두 행에서는 속도 향상이 더딜 수 있다. 표 3에서 출력 채널들이 가장 많은 마지막 두 행은 가장 큰 성능향상을 보이며, 이상적인 속도 향상인 8배에 근접할 수 있다. 이는 비트패킹 오버헤드가 많은 수의 출력 채널들로 상쇄될 수 있기 때문이다.

[0106] 도 10은 제안된 전자 장치가 활용하는 컴파일러 최적화의 효과를 확인하기 위해 비교실시예에 해당하는 컴파일러 최적화 방식의 이진 컨볼루션 레이어에서의 레이턴시를 비교한다. 그래프(1000)는 위의 표 3의 제1 행에 대응하는 BiRealNet-18의 Binary Conv 1 레이어를 사용한 결과를 나타낸다. 그래프(1000)는 활용된 컴파일러 최적화 방식에 따른 이진 컨볼루션 레이턴시(1010), 비트패킹 레이턴시(1020) 및 배치넘 레이턴시(1030)를 포함할

수 있다. 그래프(1000)에서, 제안된 전자 장치가 활용하는 컴파일러 최적화가 없는 그래프(A)는 레이턴시(1010, 1020, 1030) 모두 감소하지 못한다. BN 폴딩 기법을 적용한 그래프(B)는 레이턴시(1030)을 크게 감소하는 효과를 얻을 수 있다. 이는 컨볼루션 연산이 GEMM 코어에 의해 가속화되는 반면 배치 정규 작업은 ARM CPU의 소프트웨어를 통해 수행되기 때문이다. 또한, BN 폴딩 기법 및 비트패킹을 통한 작업 수준 파이프라이닝을 적용한 그래프(C)는 약 10%의 레이턴시(1020) 감소를 달성할 수 있다.

[0107] 도 11은, 비교실시예들과 제안된 전자 장치의 성능을 비교하는 그래프이다.

[0108] 그래프(1100)를 통해 3X3 Binary 컨볼루션 레이어 연산에 대한 비교실시예들과 제안된 전자 장치의 성능을 비교할 수 있다. 그래프(1100)에 포함된 그래프(A 내지 D)는 도 6의 Binary Conv 1 내지 4에 대응할 수 있다. 비교실시예(1110, 1120) 및 제안된 방식(1130)을 비교하여, 그래프(1100)는 이진 컨볼루션 레이어의 레이턴시를 보여준다. 그래프(A 내지 D)를 통해, 제안된 방식(1130)은 비교실시예들(1110 및 1120)에 비해 6~12배 레이턴시 감소를 달성할 수 있다.

[0109] 아래 표 4는 이전 BNN 컴파일러 및 TVM-VTA와 비교하여, 제안된 전자 장치의 엔드-투-엔드 네트워크 성능을 비교한다.

표 4

[0110]

	비교실시예 1	비교실시예 2	비교실시예 3	전자 장치
Device	Cortex A53	Cortex A53	Ultra96	ZCU102
$F_{\max}$ (MHz)	-	-	333	333
FPS	4.42	6.76	15.38	24.29

[0111] 표 4에서 비교실시예1 및 비교실시예 2의 경우 오픈 소스 컴파일러를 사용하여 FPGA 보드에서 BiRealNet-18을 컴파일하고 성능을 측정하였다. 비교실시예들과 비교하여, 전자 장치는 BiRealNet-18에서 3.6~5.5배 더 나은 엔드-투-엔드 성능을 제공할 수 있다.

[0112] 이상에서 설명된 실시예들은 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치, 방법 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0113] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 또는 컴퓨터 저장 매체 또는 장치에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0114] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있으며 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와

같은 광기록 매체(optical media), 플로옵티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0115] 위에서 설명한 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 또는 복수의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

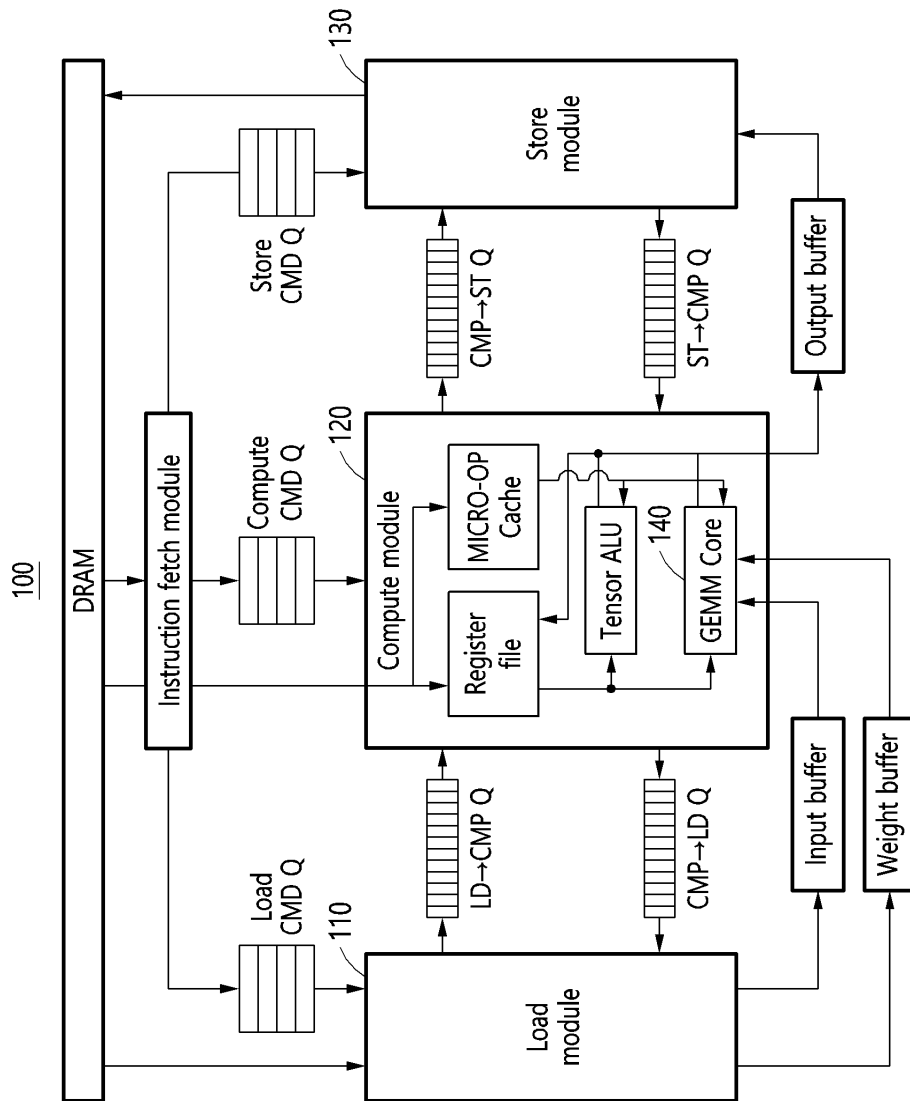
[0116] 본 문서에서, "A 또는 B", "A 및 B 중 적어도 하나", "A 또는 B 중 적어도 하나", "A, B 또는 C", "A, B 및 C 중 적어도 하나", 및 "A, B, 또는 C 중 적어도 하나"와 같은 문구들 각각은 그 문구들 중 해당하는 문구에 함께 나열된 항목들 중 어느 하나, 또는 그들의 모든 가능한 조합을 포함할 수 있다.

[0117] 이상과 같이 실시예들이 비록 한정된 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 이를 기초로 다양한 기술적 수정 및 변형을 적용할 수 있다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

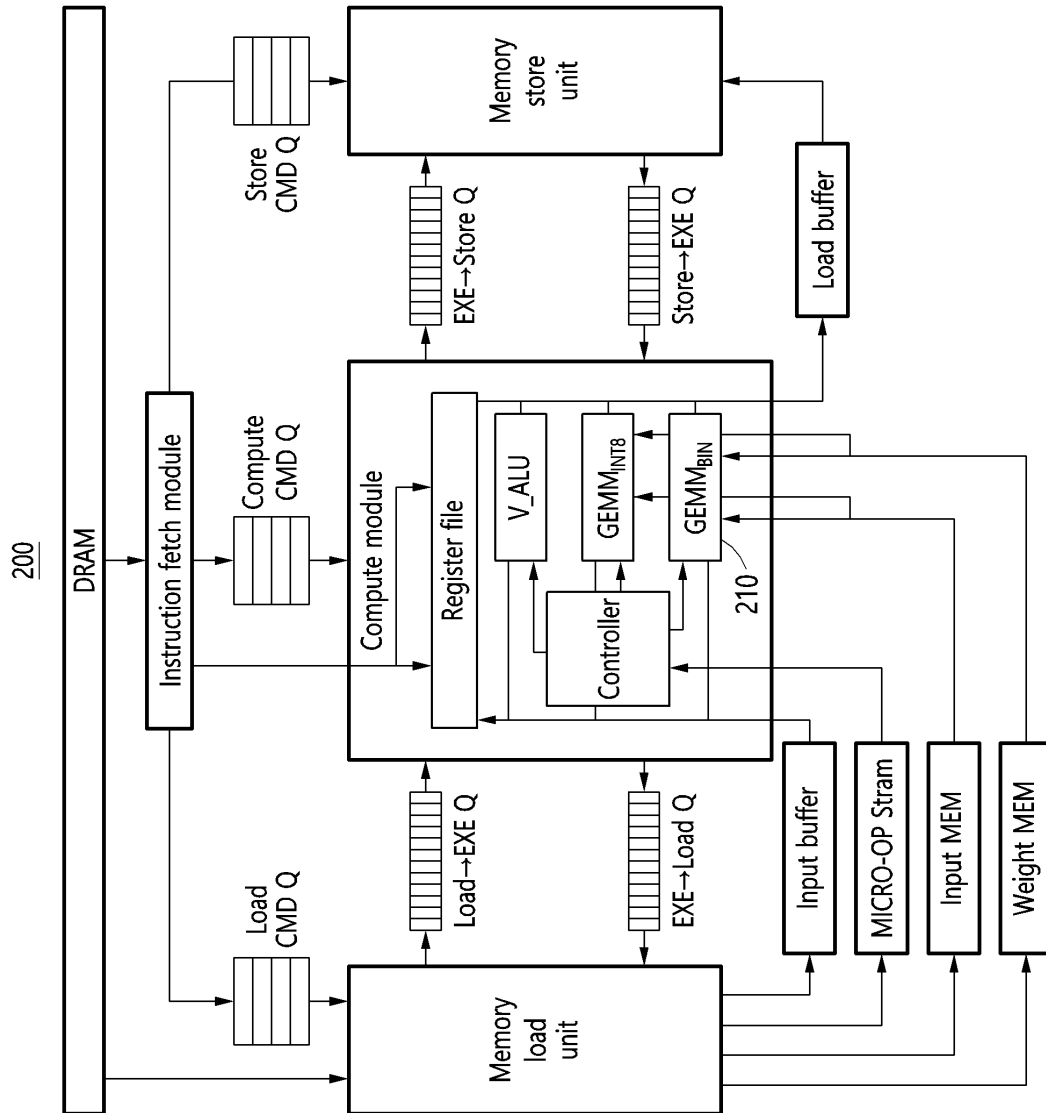
[0118] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

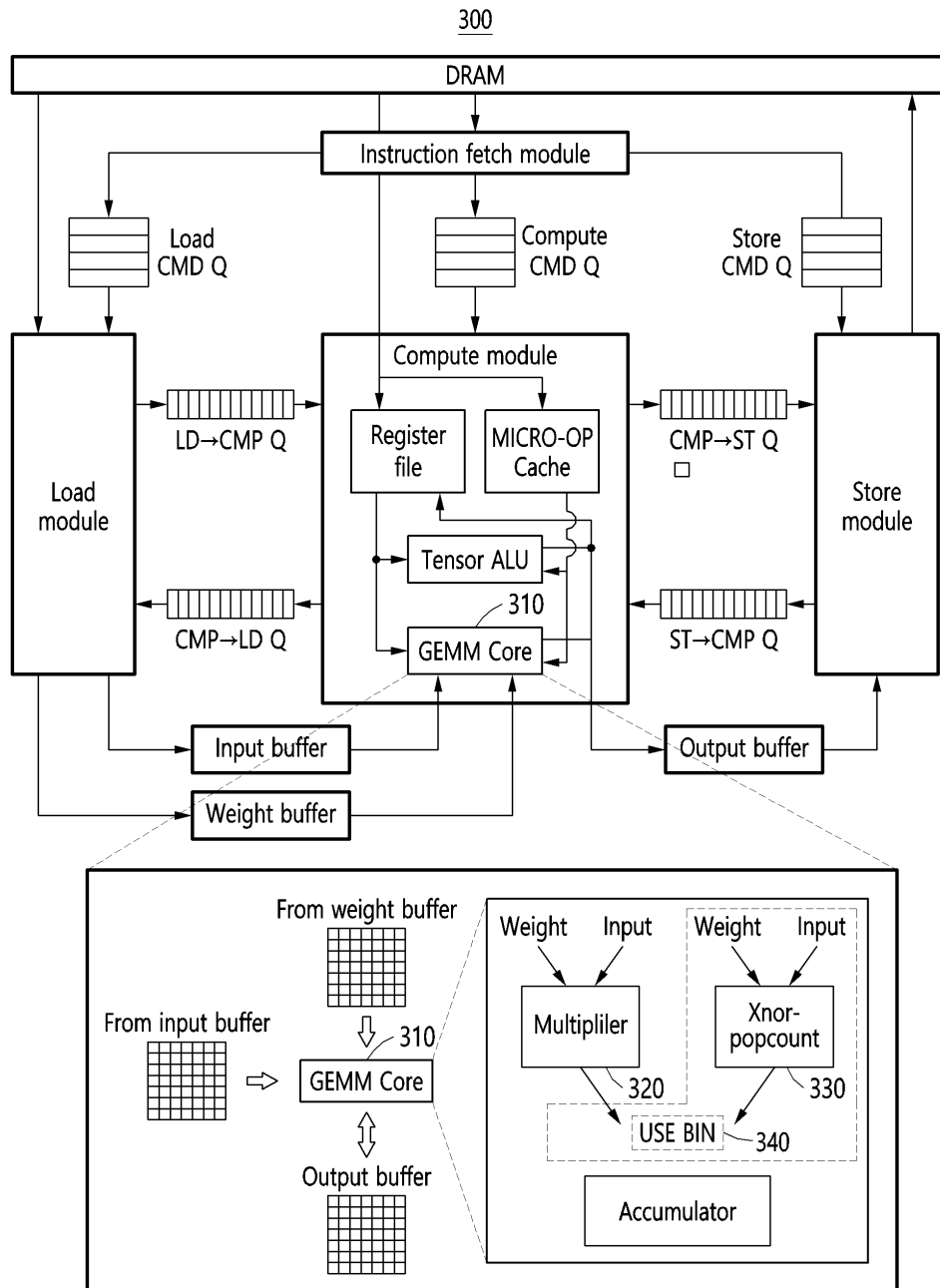
도면1



도면2

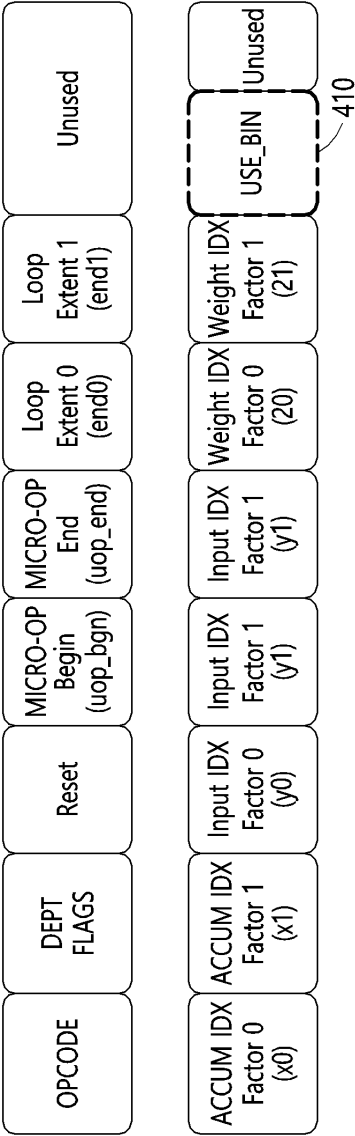


도면3

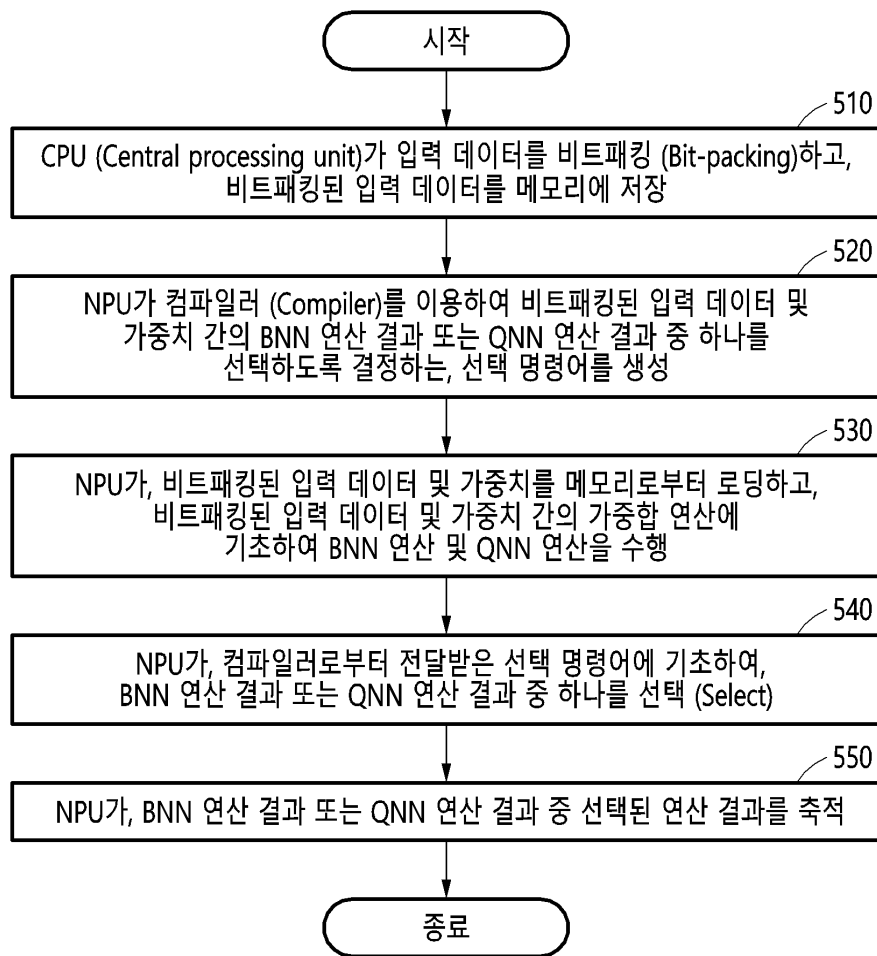


도면4

400

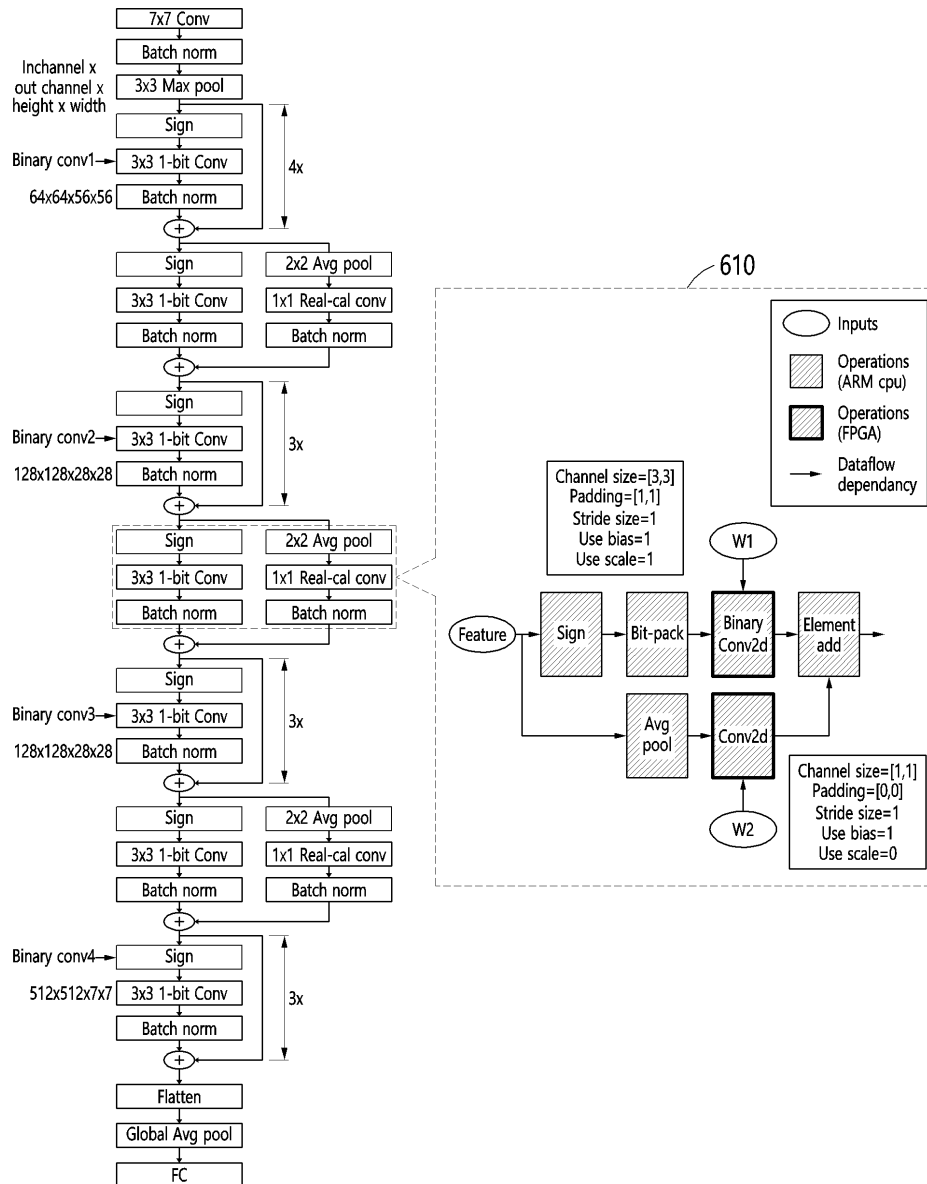


도면5

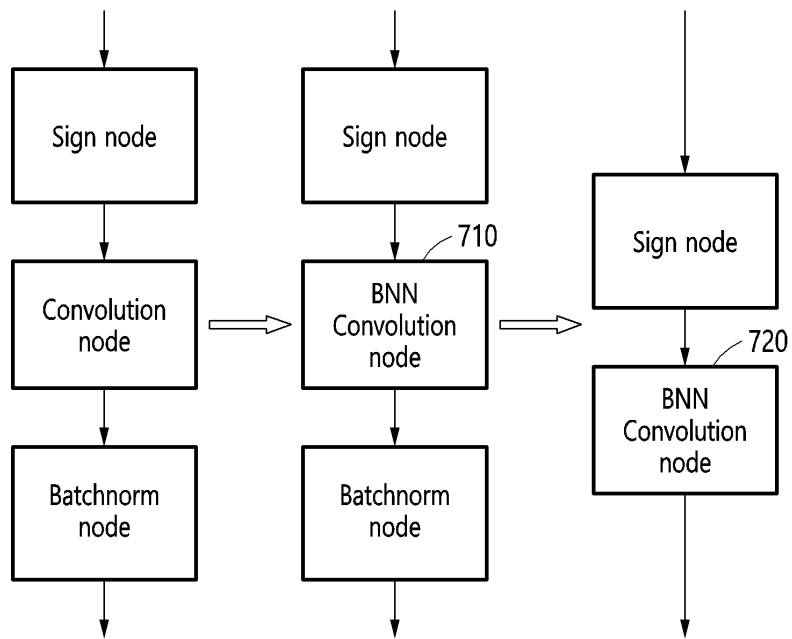


도면6

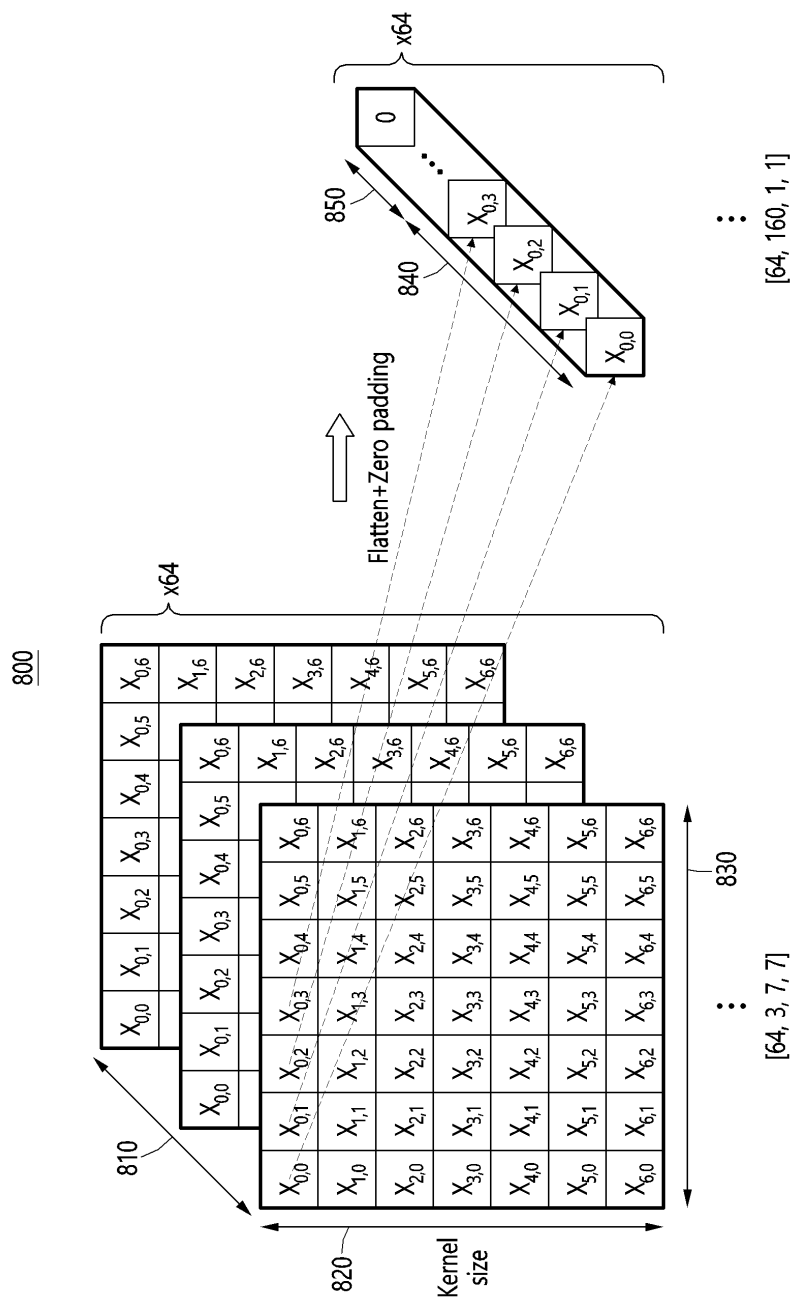
600



도면7

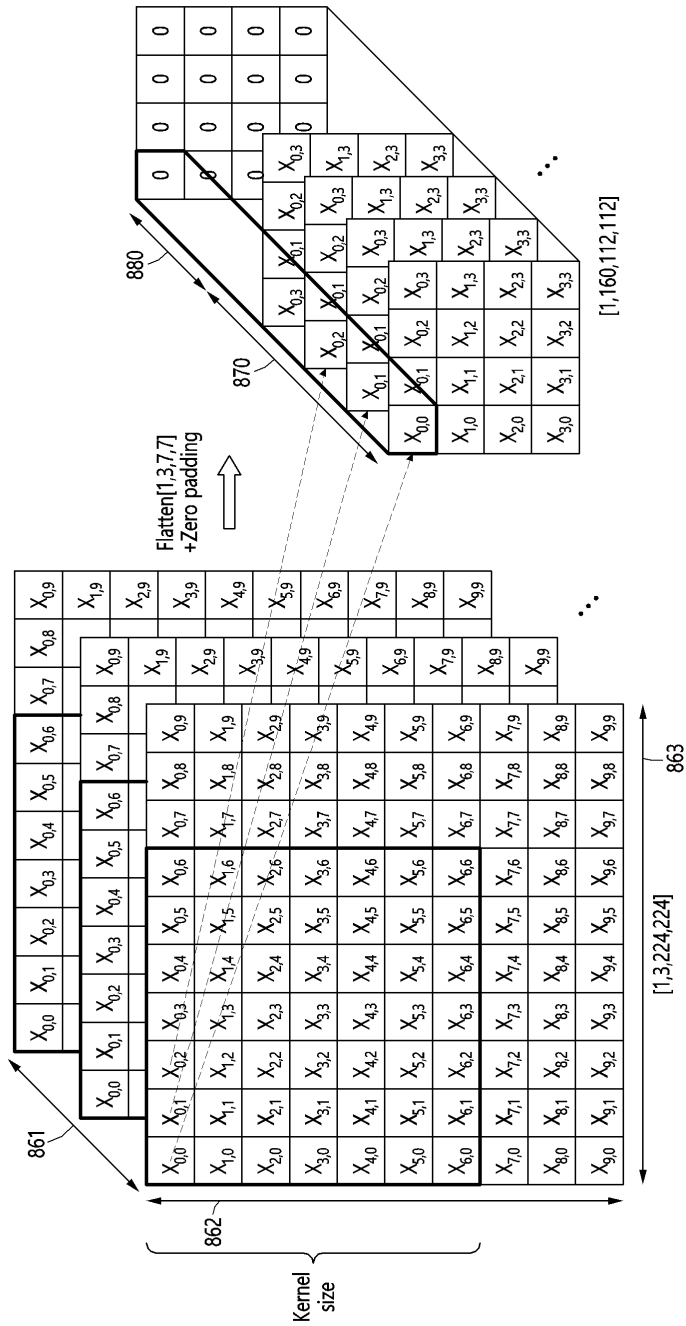


도면 8a

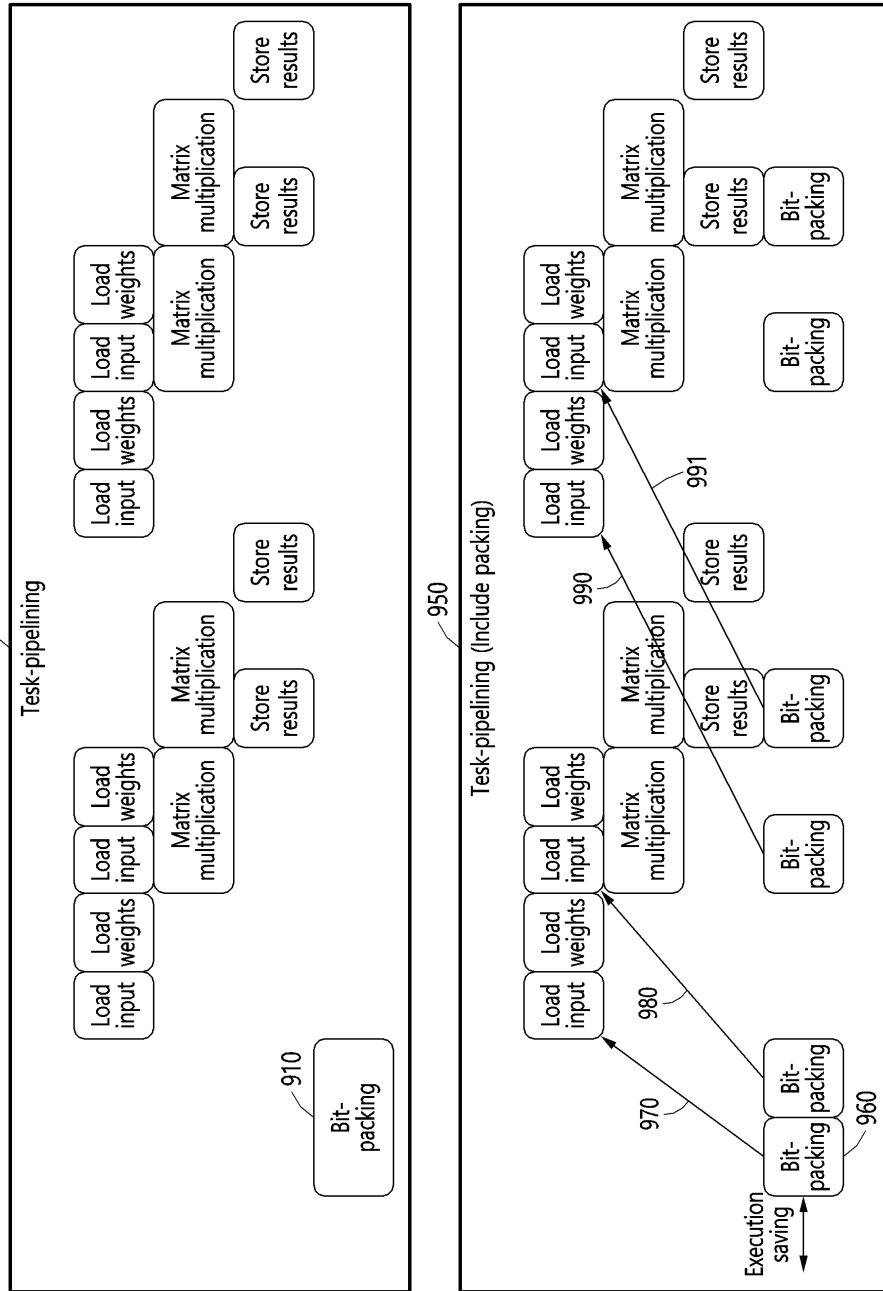


도면8b

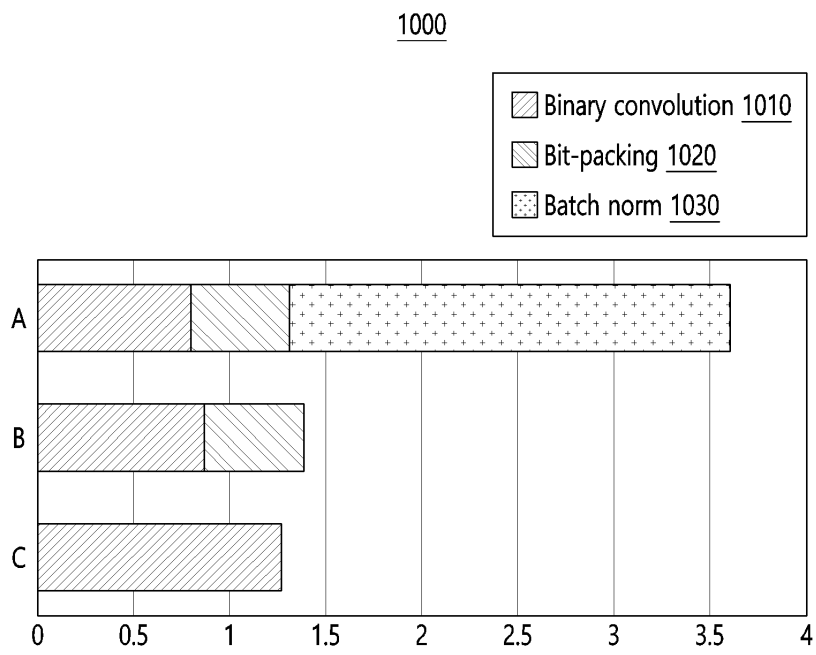
860



도면9



도면10



도면11

