(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0106975
(43) 공개일자 2024년07월08일

(51) 국제특허분류(Int. Cl.)
G06N 3/063 (2023.01) *G06F 15/80* (2006.01)
G06F 17/15 (2006.01) *G06N 3/0464* (2023.01)
(52) CPC특허분류
G06N 3/063 (2013.01)
G06F 15/8046 (2013.01)
(21) 출원번호 10-2023-0180084
(22) 출원일자 2023년12월12일
심사청구일자 2023년12월12일
(30) 우선권주장
1020220189775 2022년12월29일 대한민국(KR)

(71) 출원인
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50
(72) 발명자
이규호
울산광역시 울주군 언양읍 유니스트길 50
정주은
울산광역시 울주군 언양읍 유니스트길 50
(74) 대리인
특허법인 무한

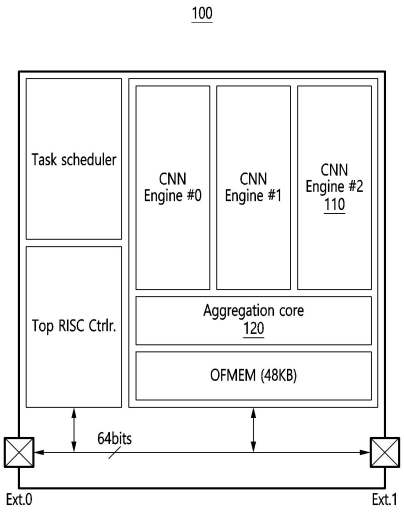
전체 청구항 수 : 총 16 항

(54) 발명의 명칭 **복수의 CNN 연산을 지원하는 통합 CNN 연산 가속 장치 및 방법**

(57) 요 약

일 실시예에 따른 CNN 연산 가속 장치는, 입력 데이터 및 가중치(weight) 간의 가중합 연산을 수행하는 컨볼루션 엔진(convolution engine); 및 상기 컨볼루션 엔진에 의해 수행된 상기 가중합 연산에 따른 연산 결과를 누적하는 축적 코어(aggregation core)를 포함할 수 있고, 상기 컨볼루션 엔진은, CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 개수의 시프트 레지스터(shift register)에 입력 데이터를 저장하는 시프트 레지스터 기반 입력 재구성 유닛(shift register based input reconfiguration unit, SIRU); 및 상기 SIRU에 저장된 입력 데이터 및 상기 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치(weight)에 기초하여, 가중합 연산을 수행하는, 시스톨릭 어레이(systolic array) 구조로 배열된, 복수의 프로세싱 엘리먼트들(processing elements, PEs)을 포함할 수 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 17/153 (2013.01)
G06N 3/0464 (2023.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711193811
과제번호 2021-0-00871-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 신개념PIM반도체기술개발
연구과제명 DRAM 기반 DNN 연산기 통합 PIM DRAM 칩 개발
기 여 율 50/100
과제수행기관명 한국과학기술원
연구기간 2023.01.01 ~ 2023.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711193271
과제번호 2022-0-01037-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 PIM인공지능반도체핵심기술개발(설계)
연구과제명 DRAM 기반 고성능 PIM 메모리 반도체 기술 개발
기 여 율 50/100
과제수행기관명 한국과학기술원
연구기간 2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

CNN(convolutional neural network) 연산 가속 장치에 있어서,

입력 데이터 및 가중치(weight) 간의 가중합 연산을 수행하는 컨볼루션 엔진(convolution engine); 및

상기 컨볼루션 엔진에 의해 수행된 상기 가중합 연산에 따른 연산 결과를 누적하는 축적 코어(aggregation core)

를 포함하고,

상기 컨볼루션 엔진은,

CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 개수의 시프트 레지스터(shift register)에 입력 데이터를 저장하는 시프트 레지스터 기반 입력 재구성 유닛(shift register based input reconfiguration unit, SIRU); 및

상기 SIRU에 저장된 입력 데이터 및 상기 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치(weight)에 기초하여, 가중합 연산을 수행하는, 시스톨릭 어레이(systolic array) 구조로 배열된, 복수의 프로세싱 엘리먼트들(processing elements, PEs)

을 포함하는

CNN 연산 가속 장치.

청구항 2

제1항에 있어서,

상기 SIRU는,

복수의 입력 채널 맵 각각의 동일한 위치에 배열된 입력 데이터를 미리 결정된 순서로 저장하는 복수의 시프트 레지스터들(shift registers); 및

상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 상기 CNN 모델의 해당 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 상기 복수의 프로세싱 엘리먼트들에 전달하는 멀티플렉서들(multiplexer)

을 포함하는,

CNN 연산 가속 장치.

청구항 3

제2항에 있어서,

상기 SIRU는,

상기 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들 중 제1 열(column)에 배열된 프로세싱 엘리먼트들에 상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 전달하는 멀티플렉서들

을 포함하는,

CNN 연산 가속 장치.

청구항 4

제2항에 있어서,

상기 SIRU는,

상기 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 출력 채널맵의 개수 및 상기 컨볼루션 연산 종류에 대응하는 커널맵의 가중치 간격에 대응하는 수치

에 기초하여 결정된 개수의 시프트 레지스터들을 활성화하는,

CNN 연산 가속 장치.

청구항 5

제1항에 있어서,

상기 복수의 프로세싱 엘리먼트들(processing elements, PEs)은,

복수의 곱셈기 및 복수의 가중치 홀더(weight holder)를 포함하고, 상기 축적 코어에 상기 가중합 연산의 결과를 누적하는 제1 프로세싱 엘리먼트; 및

복수의 곱셈기 및 복수의 가중치 홀더를 포함하고, 상기 축적 코어로부터 상기 제1 프로세싱 엘리먼트의 가중합 연산의 누적 결과를 수신 가능한 제2 프로세싱 엘리먼트

를 포함하는,

CNN 연산 가속 장치.

청구항 6

제5항에 있어서,

상기 제2 프로세싱 엘리먼트는 상기 시스톨릭 어레이 구조 중 제2열(column)에 배열되고,

상기 제2열(column) 이외의 열에 상기 제1 프로세싱 엘리먼트가 배열되는

CNN 연산 가속 장치.

청구항 7

제1항에 있어서,

상기 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들은,

상기 시스톨릭 어레이 구조의 각 열(column)에 배열된 프로세싱 엘리먼트로부터 행(row) 방향을 따라 배열된 프로세싱 엘리먼트로 상기 입력 데이터를 전파하는

CNN 연산 가속 장치.

청구항 8

CNN 연산 가속 장치에 의해 수행되는 CNN 연산 가속 방법에 있어서,

입력 데이터 및 가중치(weight) 간의 가중합 연산을 수행하는 컨볼루션 엔진(convolution engine)에 포함된 시프트 레지스터 기반 입력 재구성 유닛(shift register based input reconfiguration unit, SIRU)에, CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 개수의 입력 데이터를 저장하는 단계;

시스톨릭 어레이(systolic array) 구조로 배열되고, 상기 SIRU와 연결된 복수의 프로세싱 엘리먼트들(processing elements, PEs)에서, 상기 SIRU에 저장된 입력 데이터 및 상기 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치(weight)에 기초하여, 가중합 연산을 수행하는 단계;

상기 복수의 프로세싱 엘리먼트들로부터 상기 가중합 연산에 따른 연산 결과를 축적 코어(aggregation core)에 누적하는 단계;

를 포함하는,

CNN 연산 가속 방법.

청구항 9

제8항에 있어서,

상기 입력 데이터를 저장하는 단계는,

복수의 시프트 레지스터들에, 복수의 입력 채널 맵 각각의 동일한 위치에 배열된 입력 데이터를 미리 결정된 순서로 저장하는 단계

를 포함하고,

상기 가중합 연산을 수행하는 단계는,

멀티플렉서들(multiplexer)이 상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 상기 CNN 모델의 해당 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 상기 복수의 프로세싱 엘리먼트들에 전달하는 단계

를 포함하는

CNN 연산 가속 방법.

청구항 10

제9항에 있어서,

상기 복수의 프로세싱 엘리먼트들에 전달하는 단계는,

상기 멀티플렉서들이 상기 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들 중 제1 열(column)에 배열된 프로세싱 엘리먼트들에 상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 전달하는 단계

를 포함하는

CNN 연산 가속 방법.

청구항 11

제9항에 있어서,

상기 입력 데이터를 저장하는 단계는,

상기 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 출력 채널맵의 개수 및 상기 컨볼루션 연산 종류에 대응하는 커널맵의 가중치 간격에 대응하는 수치에 기초하여 결정된 개수의 시프트 레지스터들을 활성화하는 단계

를 포함하는

CNN 연산 가속 방법.

청구항 12

제8항에 있어서

상기 가중합 연산에 따른 연산 결과를 축적 코어(aggregation core)에 누적하는 단계는,

복수의 곱셈기 및 복수의 가중치 홀더를 포함하는 제1 프로세싱 엘리먼트에서의 가중합 연산 결과를 상기 축적 코어에 누적하는 단계; 및

상기 축적 코어에 누적된 상기 가중합 연산 결과를 제2 프로세싱 엘리먼트가 수신하는 단계

를 포함하는

CNN 연산 가속 방법.

청구항 13

제12항에 있어서,

상기 제2 프로세싱 엘리먼트는 상기 시스톨릭 어레이 구조 중 제2열(column)에 배열되고,

상기 제2열 이외의 열에 상기 제1 프로세싱 엘리먼트가 배열되는

CNN 연산 가속 방법.

청구항 14

제8항에 있어서,

상기 가중합 연산을 수행하는 단계는,

상기 시스톨릭 어레이 구조의 각 열(column)에 배열된 프로세싱 엘리먼트로부터 행(row) 방향을 따라 배열된 프로세싱 엘리먼트로 상기 입력 데이터를 전파하는 단계

를 포함하는

CNN 연산 가속 방법.

청구항 15

제8항에 있어서,

상기 입력 데이터를 저장하는 단계는,

컬러 이미지 또는 상기 컬러 이미지에 대응하는 깊이 이미지(depth image) 중 적어도 하나에 기초하는 입력 데이터를 상기 SIRU에 저장하는 단계

를 포함하고,

상기 가중합 연산을 수행하는 단계는

상기 시스톨릭 어레이 구조로 배열되고 상기 SIRU와 연결된 복수의 프로세싱 엘리먼트들에서, 상기 SIRU에 저장된 상기 깊이 이미지에 기초하는 입력 데이터 및 상기 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치에 기초하여, 상기 깊이 이미지의 문맥 정보(context information)에 대응하는 연산 결과를 산출하는 단계

를 포함하는,

CNN 연산 가속 방법.

청구항 16

하드웨어와 결합되어 제8항 내지 제15항 중 어느 하나의 항의 방법을 실행시키기 위하여 컴퓨터 판독 가능한 기록매체에 저장된 컴퓨터 프로그램.

발명의 설명

기 술 분 야

[0001] 이하, 서로 다른 복수의 CNN 연산 레이어를 포함하는 CNN 모델을 구현하기 위한 CNN 연산 장치에 대한 기술이 제공된다.

배 경 기 술

[0002] 최근 들어, 심층 신경망(DNN)이 의미론적 영상 분할, 객체 감지, 객체 추적 등 광범위한 인식 잡업에 활발히 사용되어 인간 수준의 성능을 뛰어 넘고 있다. DNN은 다양한 산업 분야에서 활용되어 지능형 시스템의 확장을 가속화하면서 인공 지능의 필수 구성요소로 볼 수 있다. 특히, 지능형 객체 인식은 승객 안전을 위한 정교한 상황 인식과 함께 완벽한 차량 제어를 요구하는 자율 주행 자동차(AEV)의 필수 구성요소이다. 객체 인식 및 상황 인식에 가장 적합한 신경망 구조로서 영상에 대한 특징을 가장 잘 분석할 수 있는 컨볼루션 신경망(CNN) 모델에 대한 연구가 계속되고 있다. 다만, 기존의 CNN 기반의 네트워크에서는 실시간 객체 인식을 수행하기 위해 전력의 소모가 크다는 문제가 있었다. 따라서, CNN 기반의 네트워크에서의 전력 소모를 줄이기 위해 네트워크를 경

량화 할 필요성이 제기된다. 그러므로, CNN을 이용한 의미론적 분할(semantic segmentation)을 구현하기 위해, 심층 신경망 구조의 CNN의 각 레이어에서 수행되는 서로 다른 컨볼루션 연산을 수행하기 위한 장치 및 해당 장치를 통해 구현 가능한 경량화된 CNN 모델 구조가 요구된다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0003] 일 실시예에 따른 CNN 연산 가속 장치는, 입력 데이터 및 가중치(weight) 간의 가중합 연산을 수행하는 컨볼루션 엔진(convolution engine); 및 상기 컨볼루션 엔진에 의해 수행된 상기 가중합 연산에 따른 연산 결과를 누적하는 축적 코어(aggregation core)를 포함하고, 상기 컨볼루션 엔진은, CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 개수의 시프트 레지스터(shift register)에 입력 데이터를 저장하는 시프트 레지스터 기반 입력 재구성 유닛(shift register based input reconfiguration unit, SIRU); 및 상기 SIRU에 저장된 입력 데이터 및 상기 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치(weight)에 기초하여, 가중합 연산을 수행하는, 시스톨릭 어레이(systolic array) 구조로 배열된, 복수의 프로세싱 엘리먼트들(processing elements, PEs)을 포함할 수 있다.

[0004] 상기 SIRU는, 복수의 입력 채널 맵 각각의 동일한 위치에 배열된 입력 데이터를 미리 결정된 순서로 저장하는 복수의 시프트 레지스터들(shift registers); 및 상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 상기 CNN 모델의 해당 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 상기 복수의 프로세싱 엘리먼트들에 전달하는 멀티플렉서들(multiplexer)을 포함할 수 있다.

[0005] 상기 SIRU는, 상기 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들 중 제1 열(column)에 배열된 프로세싱 엘리먼트들에 상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 전달하는 멀티플렉서들을 포함할 수 있다.

[0006] 상기 SIRU는, 상기 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 출력 채널맵의 개수 및 상기 컨볼루션 연산 종류에 대응하는 커널맵의 가중치 간격에 대응하는 수치에 기초하여 결정된 개수의 시프트 레지스터들을 활성화할 수 있다.

[0007] 상기 복수의 프로세싱 엘리먼트들(processing elements, PEs)은, 복수의 곱셈기 및 복수의 가중치 홀더(weight holder)를 포함하고, 상기 축적 코어에 상기 가중합 연산 결과를 누적하는 제1 프로세싱 엘리먼트; 및 복수의 곱셈기 및 복수의 가중치 홀더를 포함하고, 상기 축적 코어로부터 상기 제1 프로세싱 엘리먼트의 가중합 연산의 누적 결과를 수신 가능한 제2 프로세싱 엘리먼트를 포함할 수 있다.

[0008] 상기 제2 프로세싱 엘리먼트는 상기 시스톨릭 어레이 구조 중 제2열(column)에 배열되고, 상기 제2열(column) 이외의 열에 상기 제1 프로세싱 엘리먼트가 배열될 수 있다.

[0009] 상기 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들은, 상기 시스톨릭 어레이 구조의 각 열(column)에 배열된 프로세싱 엘리먼트로부터 행(row) 방향을 따라 배열된 프로세싱 엘리먼트로 상기 입력 데이터를 전파할 수 있다.

[0010] 일 실시예에 따르면, CNN 연산 가속 장치에 의해 수행되는 CNN 연산 가속 방법은, 입력 데이터 및 가중치(weight) 간의 가중합 연산을 수행하는 컨볼루션 엔진(convolution engine)에 포함된 시프트 레지스터 기반 입력 재구성 유닛(shift register based input reconfiguration unit, SIRU)에, CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 개수의 입력 데이터를 저장하는 단계; 시스톨릭 어레이(systolic array) 구조로 배열되고, 상기 SIRU와 연결된 복수의 프로세싱 엘리먼트들(processing elements, PEs)에서, 상기 SIRU에 저장된 입력 데이터 및 상기 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치(weight)에 기초하여, 가중합 연산을 수행하는 단계; 상기 복수의 프로세싱 엘리먼트들로부터 상기 가중합 연산에 따른 연산 결과를 축적 코어(aggregation core)에 누적하는 단계;

[0011] 를 포함할 수 있다.

[0012] 상기 입력 데이터를 저장하는 단계는, 복수의 시프트 레지스터들에, 복수의 입력 채널 맵 각각의 동일한 위치에

배열된 입력 데이터를 미리 결정된 순서로 저장하는 단계를 포함할 수 있고, 상기 가중합 연산을 수행하는 단계는, 멀티플렉서들(multiplexer)이 상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 상기 CNN 모델의 해당 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 상기 복수의 프로세싱 엘리먼트들에 전달하는 단계를 포함할 수 있다.

[0013] 상기 복수의 프로세싱 엘리먼트들에 전달하는 단계는, 상기 멀티플렉서들이 상기 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들 중 제1 열(column)에 배열된 프로세싱 엘리먼트들에 상기 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 전달하는 단계를 포함할 수 있다.

[0014] 상기 입력 데이터를 저장하는 단계는, 상기 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 출력 채널맵의 개수 및 상기 컨볼루션 연산 종류에 대응하는 커널맵의 가중치 간격에 대응하는 수치에 기초하여 결정된 개수의 시프트 레지스터들을 활성화하는 단계를 포함할 수 있다.

[0015] 상기 가중합 연산에 따른 연산 결과를 축적 코어(aggregation core)에 누적하는 단계는, 복수의 곱셈기 및 복수의 가중치 홀더를 포함하는 제1 프로세싱 엘리먼트에서의 가중합 연산 결과를 상기 축적 코어에 누적하는 단계; 및 상기 축적 코어에 누적된 상기 가중합 연산 결과를 제2 프로세싱 엘리먼트가 수신하는 단계를 포함할 수 있다.

[0016] 일 실시예에 따른 CNN 연산 가속 방법에서, CNN 연산 가속 장치에 포함된 상기 제2 프로세싱 엘리먼트는 상기 시스톨릭 어레이 구조 중 제2열(column)에 배열되고, 상기 제2열 이외의 열에 상기 제1 프로세싱 엘리먼트가 배열될 수 있다.

[0017] 일 실시예에 따른 CNN 연산 가속 방법은, 상기 시스톨릭 어레이 구조의 각 열(column)에 배열된 프로세싱 엘리먼트로부터 행(row) 방향을 따라 배열된 프로세싱 엘리먼트로 상기 입력 데이터를 전파하는 단계를 더 포함할 수 있다.

[0018] 상기 입력 데이터를 저장하는 단계는, 컬러 이미지 또는 상기 컬러 이미지에 대응하는 깊이 이미지(depth image) 중 적어도 하나에 기초하는 입력 데이터를 상기 SIRU에 저장하는 단계를 포함할 수 있고, 상기 가중합 연산을 수행하는 단계는 상기 시스톨릭 어레이 구조로 배열되고 상기 SIRU와 연결된 복수의 프로세싱 엘리먼트들에서, 상기 SIRU에 저장된 상기 깊이 이미지에 기초하는 입력 데이터 및 상기 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치에 기초하여, 상기 깊이 이미지의 문맥 정보(context information)에 대응하는 연산 결과를 산출하는 단계를 포함할 수 있다.

도면의 간단한 설명

[0019] 도 1은 일 실시예에 따른, 서로 다른 복수의 CNN 연산을 지원하는 CNN 연산 가속 장치에 관해 도시한다.

도 2는 일 실시예에 따른 컨볼루션 엔진의 구조에 대하여 도시한다.

도 3은 복수의 프로세싱 엘리먼트들 중 서로 다른 두 종류의 프로세싱 엘리먼트들에 대하여 도시한다.

도 4는 제안된 SIRU의 세부 구조 및 복수의 프로세싱 엘리먼트들과의 연결 방법을 도시한다.

도 5 내지 도 6은 일 실시예에 따른 SIRU의 입력 데이터 업데이트 모드(update mode) 및 재사용 모드(reuse mode)에 대하여 도시한다.

도 7은 일 실시예에 따른 SIRU에서의 스탠다드 컨볼루션 연산 수행에 따른 타이밍도를 나타낸다.

도 8a 내지 도 8e는 일 실시예에 따르는 CNN 연산 가속 장치에 의해 수행될 수 있는 서로 다른 컨볼루션 연산에 필요한, 입력 피처맵, 커널 맵 및 출력 채널맵을 나타낸다.

도 9a내지 도 9b는 일 실시예에 따른 CNN 연산 가속 장치에서 도 8a 내지 도 8e에 대응하는 컨볼루션 연산을 수행하기 위해, SIRU에서 복수의 프로세싱 엘리먼트들로의 입력 데이터 인가 방법에 대하여 도시한다.

도 10은 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 깊이별 분산가능 컨볼루션 연산의 데이터 흐름을 나타낸다.

도 11a 내지 도 11b는 일 실시예에 따른 CNN 연산 가속 장치에 포함된 컨볼루션 엔진의 동작 흐름도를 나타낸다.

도 12는 의미론적 분할을 위한 CNN 기반 모델 아키텍처를 도시한다.

도 13은 일 실시예에 따른 CNN 연산 가속 장치를 통해 구현할 수 있는 CNN 기반 모델의 개략도다.

도 14는 일 실시예에 따른 CNN 연산 가속 장치를 통해 구현 가능한 DTN의 구체적인 구조를 도시한다.

도 15는 일 실시예에 따른 CNN 연산 가속 장치의 칩 레이아웃을 나타낸다.

도 16a는 레이어 별 DSC 연산과 DSC 레이어 융합을 수행하는 경우의 프로세싱 엘리먼트들의 활용도를 나타내는 그래프다.

도 16b는 DTN 연산을 위한 외부 메모리 액세스 비율을 나타낸다.

도 16c는 일 실시예에 따른 CNN 연산 가속 장치와 DTN 모델 알고리즘 최적화를 통한 프레임 속도 향상 결과를 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0020] 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.

[0021] 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.

[0022] 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.

[0023] 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 존재함으로 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.

[0024] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 해당 기술분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 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.

[0025] 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 일 실시예에 따른, 서로 다른 복수의 CNN 연산을 지원하는 CNN 연산 가속 장치에 관해 도시한다.

[0029] 일 실시예에 따르면, CNN 연산 가속 장치(100)는 서로 다른 복수의 컨볼루션(convolution) 연산을 수행할 수 있다. CNN 연산 가속 장치(100)는 예를 들어, 스탠다드 컨볼루션 (standard convolution), 깊이별 분리가능한 컨볼루션(depth-wise separable convolution), 확장 컨볼루션(dilated convolution), 위치별 컨볼루션(point-wise convolution), 트랜스포즈 컨볼루션(transposed convolution) 연산을 수행할 수 있다. CNN 연산 가속 장치(100)가 서로 다른 복수의 컨볼루션 연산을 수행함으로써, 복수의 레이어를 포함하고, 복수의 레이어 마다 다른 컨볼루션 연산이 수행되는 CNN 모델을 구현할 수 있다. CNN 연산 가속 장치(100)에 의해 수행되는 서로 다른 복수의 컨볼루션 연산 방법은 이하 도 9a 내지 도 9b에서 구체적으로 설명한다.

[0030] CNN 연산 가속 장치(100)는 입력 데이터 및 가중치(weight) 간의 가중합 연산을 수행하는 컨볼루션 엔진(convolution engine)(110), 및 컨볼루션 엔진(110)에 의해 수행된 가중합 연산에 따른 연산 결과를 누적하는 축적 코어(aggregation core)(120)를 포함할 수 있다. 예를 들어, CNN 연산 가속 장치(100)는, 3개의 컨볼루션 엔진(convolution engine)(110), 복수의 컨볼루션 엔진(110)과 연결된 축적 코어(aggregation core)(120),

출력 피처 메모리(output feature memory, OFMEM), 탑 RISC 컨트롤러, 및 작업 스케줄러(task scheduler)를 포함할 수 있다. CNN 연산 가속 장치(100)에 입력 데이터가 인가되면, 컨볼루션 엔진(110)에서 가중치와 입력 데이터 간의 곱셈 결과에 기초하여 가중합 연산이 수행될 수 있다. 컨볼루션 엔진(110)에서의 가중합 연산 결과는 축적 코어(120)에 누적될 수 있다. 컨볼루션 엔진(100)에서의 가중합 연산 결과 및 해당 가중합 연산 결과의 축적 코어(120)에서의 누적 결과는 입력 데이터(또는 입력 피처맵) 및 가중치 커널 간의 가중합 결과에 대응할 수 있다. 컨볼루션 엔진(110)은 서로 다른 컨볼루션 연산을 수행하기 위해, 입력 데이터가 인가되는 방법을 조절할 수 있다. 예를 들어, 컨볼루션 엔진(110)은 시프트 레지스터 기반의 입력 재구성 유닛(shift register based input reconfiguration unit, SIRU)(이하, SIRU)에 입력 데이터를 임시로 저장했다가, 컨볼루션 연산 종류에 기초하여 입력 데이터의 인가 시점을 조절하고, 가중합 연산을 수행할 수 있다. 또한, 컨볼루션 엔진(110)은 축적 코어(120)로부터 가중합 연산 결과의 누적 결과를 추가적으로 수신함으로써, 깊이별 분산가능 컨볼루션(depth-wise separable convolution)(이하, 'DSC')을 수행할 수 있다. 컨볼루션 엔진(110)에 포함된 SIRU의 구체적인 구조 및 동작 방법과 축적 코어(120)로부터의 누적 결과 수신에 대하여, 이하 도 4 내지 도 6에서 설명한다.

[0031] 축적 코어(120)는 컨볼루션 엔진(110)으로부터 입력 데이터 및 가중치 간의 가중합 연산 결과를 수신하고, 각 가중합 연산 결과에 대한 부분합을 생성할 수 있다. 축적 코어(120)는 생성된 부분합을 누적한 결과를 출력 피처 메모리에 저장할 수 있다.

[0032] 참고로, 작업 스케줄러는 탑 RISC 컨트롤러로부터 CNN 모델에 포함된 레이어의 컨볼루션 연산 종류에 대하여 전달받고, 가중치 인젝터와 컨볼루션 엔진(110)에 포함된 SIRU의 동작을 제어할 수 있다. 또한 컨볼루션 엔진(110)은 레이어의 컨볼루션 연산 종류에 따라 동적으로 클록 게이트됨으로서 동적 전력 소비를 줄일 수 있다.

[0034] 도 2는 일 실시예에 따른 컨볼루션 엔진의 구조에 대하여 도시한다.

[0035] 일 실시예에 따른 컨볼루션 엔진(110)은 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 개수의 시프트 레지스터(shift register)에 입력 데이터를 저장하는 시프트 레지스터 기반 입력 재구성 유닛(shift register based input reconfiguration unit)(이하 'SIRU')(210), 가중치 인젝터(weight injector), 가중치 메모리(예: WMEM) 및 SIRU(210)에 저장된 입력 데이터 및 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치(weight)에 기초하여, 가중합 연산을 수행하는, 시스톨릭 어레이(systolic array) 구조로 배열된, 복수의 프로세싱 엘리먼트들(processing elements, PEs)(220)을 포함할 수 있다.

[0036] SIRU(210)는 예를 들어, 입력 데이터를 입력 피처 메모리(예: IFMEM)에서 로드하고, CNN 모델에 포함된 레이어의 컨볼루션 연산 종류, 입력 채널의 수, 및 출력 채널의 수에 따라 입력 데이터를 미리 결정된 개수만큼의 시프트 레지스터에 저장할 수 있다. 예를 들어, CNN 모델에 포함된 레이어의 컨볼루션 연산이 스탠다드 컨볼루션에 대응하고, 해당 레이어의 출력 채널의 수가 32인 경우, SIRU(210)는 4개의 시프트 레지스터에 입력 데이터를 저장할 수 있다. SIRU(210)에 입력 데이터가 저장되는 구체적인 동작과 관련하여, 이하 도 5 및 도 6에서 설명한다.

[0037] 복수의 프로세싱 엘리먼트들(220)은 시스톨릭 어레이 구조로 배열될 수 있다. 복수의 프로세싱 엘리먼트들(220)은 예를 들어, 8×3의 시스톨릭 어레이 구조로 배열될 수 있다. 도 2에서는 총 8개의 행(row), 3개의 열(column)의 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들(220)에 대하여 도시하지만, 도 2가 시계방향으로 90도 회전한 경우를 가정하면, 총 8개의 열(column), 3개의 행(row)을 갖는 구조를 고려할 수 있다. 이하에서는 도 2의 8개의 행, 3개의 열의 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들(220)을 기준으로 설명한다.

[0038] 복수의 프로세싱 엘리먼트들(220)은 SIRU(210)로부터 입력 데이터를 수신할 수 있다. 예를 들어, 복수의 프로세싱 엘리먼트들(220) 중 제1행(예: Row 0)에 배열된 프로세싱 엘리먼트들이 SIRU(210)로부터 입력 데이터를 수신하고, 수신된 입력 데이터에 기초하여 가중합 연산을 수행할 수 있다. 제1행(예: Row 1)에서 SIRU(210)로부터 수신한 입력 데이터는 제2행(예: Row 2)으로 전파될 수 있고, 제1행에서의 SIRU(210)로부터 수신한 입력 데이터가 제2행으로 전파됨과 동시에, 제1행에 배열된 프로세싱 엘리먼트들은 SIRU(210)로부터 다른 입력 데이터를 수신할 수 있다.

[0039] 복수의 프로세싱 엘리먼트들(220)은 제2행(예: Row 1)을 제외한 행에 제1 프로세싱 엘리먼트(예: PE-A)가 배열되고, 제2행에 제2 프로세싱 엘리먼트(예: PE-B)가 배열된 시스톨릭 어레이 구조일 수 있다. 각 행에 배열된

프로세싱 엘리먼트들은 열방향을 따라 인접한 프로세싱 엘리먼트들과 연결될 수 있다. 예를 들어, 제1행에 배열된 프로세싱 엘리먼트들(예: Row 0의 PE-A)은 제2행에 배열된 프로세싱 엘리먼트들(예: Row 1의 PE-B)과 연결될 수 있다. 또한, 제n행에 배열된 프로세싱 엘리먼트들은 제n+1행에 배열된 프로세싱 엘리먼트들과 연결될 수 있다. 복수의 프로세싱 엘리먼트들(220) 중 제1 프로레싱 엘리먼트 및 제2 프로세싱 엘리먼트에 대하여 구체적으로 도 3에서 설명한다.

[0040] 복수의 프로세싱 엘리먼트들(220) 각각은 가중치 메모리(예: weight MEM)와 연결될 수 있고, 가중치 인젝터(weight injector)를 통해, 가중치 메모리로부터 각 프로세싱 엘리먼트들(220)에 가중치가 인가될 수 있다. 복수의 프로세싱 엘리먼트들(220)은 SIRU(210)로부터 수신한 입력 데이터와 가중치 메모리로부터 인가된 가중치 간의 가중합 연산을 수행할 수 있다. 예를 들어, 복수의 프로세싱 엘리먼트들(220)은 입력 데이터와 가중치를 곱하고, 복수의 프로세싱 엘리먼트들(220)과 연결된 가산기 트리(예: adder tree)를 통해 곱셈 결과를 각각 더함으로써 가중합 연산을 수행할 수 있다. 복수의 프로세싱 엘리먼트들(220)은 단일 입력 채널맵에 대한 컨볼루션 연산을 수행하고, 복수의 프로세싱 엘리먼트들(220) 중 하나의 행(row)에 배열된 프로세싱 엘리먼트들은 CNN 모델에 포함된 레이어의 컨볼루션 연산 종류에 따라 최대 3개의 출력 채널맵에 대응하는 가중합 결과를 생성할 수 있다.

[0041] 종합하면, 복수의 프로세싱 엘리먼트들(220) 중 제1행의 프로세싱 엘리먼트들은 SIRU(210)로부터 입력 데이터를 수신할 수 있다. 제1행의 프로세싱 엘리먼트들은 입력 데이터 및 가중치에 기초하여 가중합 연산을 수행할 수 있다. 복수의 프로세싱 엘리먼트들(220)은 제1행에 배열된 프로세싱 엘리먼트들에서부터 제n행에 배열된 프로세싱 엘리먼트들까지 인접한 프로세싱 엘리먼트 행에 입력 데이터를 전파할 수 있다. 입력 데이터가 전파됨에 따라, 해당 입력 데이터 및 해당 입력 데이터를 수신한 프로세싱 엘리먼트에 인가된 가중치 간의 가중합 연산이 수행될 수 있다. 각 행의 부분합은 가산기 트리를 통해 축적 코어로 전송될 수 있다. 축적 코어는 각 컨볼루션 엔진(110)의 부분합을 누적하고 최종 누적 출력을 출력 피처 메모리로 전달할 수 있다.

[0043] 도 3은 복수의 프로세싱 엘리먼트들 중 서로 다른 두 종류의 프로세싱 엘리먼트들에 대하여 도시한다.

[0045] 복수의 프로세싱 엘리먼트들은 복수의 곱셈기(312) 및 복수의 가중치 홀더(weight holder)(311)를 포함하고, 축적 코어에 가중합 연산 결과를 누적하는 제1 프로세싱 엘리먼트(310)를 포함할 수 있다. 예를 들어, 제1 프로세싱 엘리먼트(310)는 4개의 곱셈기(312) 및 4개의 가중치 홀더(311)를 포함할 수 있고, 4개 입력 채널맵의 입력 데이터에 대한 동시 연산을 수행할 수 있다.

[0046] 참고로 가중치 홀더(311)는 도 3에서와 같이 가중치 버퍼(예: weight buffer)로 명명될 수 있다. 가중치 홀더(311)는 가중치 인젝터로부터 인가된 가중치를 최대 16개 저장할 수 있다. 예를 들어, 가중치 홀더(3110)는 W^{0}_{0} 내지 W^{15}_{0}의 총 16개의 가중치를 저장할 수 있고, 가중치들은 재사용될 수 있다.

[0047] 제1 프로세싱 엘리먼트(310)는 SIRU 또는 이전 행(row)에 배열된 프로세싱 엘리먼트들로부터 입력 데이터를 수신할 수 있다. 예를 들어, 도 2의 제1행에 배열된 제1 프로세싱 엘리먼트(310)는 SIRU로부터 입력 데이터를 수신할 수 있다. 다른 예를 들어, 제3행 내지 제8행에 배열된 제1 프로세싱 엘리먼트(310)들은 각각 이전 행에 배열된 프로세싱 엘리먼트들로부터 입력 데이터를 수신할 수 있다. 제1 프로세싱 엘리먼트(310)는 SIRU로부터 수신한 입력 데이터 또는 이전 행에 배열된 프로세싱 엘리먼트들로부터 수신한 입력 데이터 중 하나와 가중치 홀더(311)에 저장된 가중치에 기초하여 곱셈기(312)를 통해 획득한 결과를 축적 코어에 저장할 수 있다. 예를 들어, 제1 프로세싱 엘리먼트(310)는 입력 데이터(예: A_0) 및 가중치 홀더(예: W buffer)(311)에 인가된 가중치들(예: W^{0}_{0} 내지 W^{15}_{0})간의 곱셈기(312)를 통한 연산 결과(예: O_0)를 축적 코어에 저장할 수 있다.

[0048] 복수의 프로세싱 엘리먼트들은 복수의 곱셈기 및 복수의 가중치 홀더를 포함하고, 축적 코어로부터 제1 프로세싱 엘리먼트(310)의 가중합 연산의 누적 결과를 수신 가능한 제2 프로세싱 엘리먼트(320)를 포함할 수 있다. 예를 들어, 제2 프로세싱 엘리먼트(320)는 4개의 곱셈기 및 4개의 가중치 홀더를 포함할 수 있고, 4개 입력 채널맵의 입력 데이터에 대한 동시 연산을 수행할 수 있다. 제2 프로세싱 엘리먼트(320)는 인접한 이전 행의 제1 프로세싱 엘리먼트로부터 입력 데이터를 수신할 수 있을 뿐만 아니라, 축적 코어 간의 추가 경로(330)를 통해 제1 프로세싱 엘리먼트의 연산 결과를 수신할 수 있다. 예를 들어, 제1 프로세싱 엘리먼트(310)가 제1행에 배

열되고, 제2 프로세싱 엘리먼트(320)가 제2행에 배열된 경우, 제1 프로세싱 엘리먼트(310)에 수신된 입력 데이터(예: A_0)는 제2 프로세싱 엘리먼트(320)로 전파될 수 있다. 제1 프로세싱 엘리먼트(310)의 제1 가중합 연산 결과(예: O_0)는 축적 코어에 누적될 수 있고, 제2 프로세싱 엘리먼트(320)는 추가 경로(330)를 통해 축적 코어로부터 제1 가중합 연산 결과(예: O_0)를 수신할 수 있다. 제2 프로세싱 엘리먼트(320)는 입력 데이터(예: A_0), 제1 가중합 연산 결과(예: O_0) 및 가주이 홀더에 인가된 가중치들 간의 곱셈기를 통한 연산 결과(예: O_1)를 축적 코어에 저장하고, 동시에 입력 데이터 및 제1 가중합 연산 결과에 기초하는 데이터를 다음 행에 배열된 프로세싱 엘리먼트로 전파할 수 있다. 제2 프로세싱 엘리먼트(320)는 축적 코어로부터 추가 경로(330)를 통해 제1 프로세싱 엘리먼트(310)의 가중합 연산 결과를 수신함으로서 DSC 연산을 지원할 수 있다.

[0050] 도 4는 제안된 SIRU의 세부 구조 및 복수의 프로세싱 엘리먼트들과의 연결 방법을 도시한다.

[0051] 일 실시예에 따르면, SIRU(420)는 복수의 입력 채널 맵 각각의 동일한 위치에 배열된 입력 데이터를 미리 결정된 순서로 저장하는 복수의 시프트 레지스터들(shift registers) 및 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나를 CNN 모델의 해당 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 복수의 프로세싱 엘리먼트들에 전달하는 멀티플렉서들(multiplexer)을 포함할 수 있다. SIRU(420)는, 예를 들어, 16개의 시프트 레지스터와 멀티플렉서(예: MUX)를 갖춘 3개의 동일한 모듈을 포함할 수 있다. 멀티플렉서는 16개의 시프트 레지스터에 저장된 입력 데이터 중 하나를 출력할 수 있다. 멀티플렉서는 이전 모듈(예: MUX #1을 포함하는 모듈에 대한 이전 모듈은 MUX #0을 포함하는 모듈에 해당함)의 출력을 현재 모듈의 출력으로 결정할 수 있다. 멀티플렉서는 입력 피처맵 메모리(예: IFMEM)(미도시)와 바이패스(bypass)를 통해 연결되어, 입력 피처맵 메모리에 저장된 입력 피처맵을 출력할 수 있다. 멀티플렉서는 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들(430) 중 제1 열(column)(또는 복수의 프로세싱 엘리먼트들을 시계방향으로 90도 회전시킨 경우 제1 행(row))에 배열된 프로세싱 엘리먼트들(430)에 복수의 시프트 레지스터들에 저장된 입력 데이터 중 적어도 하나, 이전 모듈의 출력, 또는 입력 피처맵 메모리로부터 수신한 입력 피처맵을 전달할 수 있다. 멀티플렉서에서 어떤 입력 데이터를 복수의 프로세싱 엘리먼트들(430)에 전달할지 여부는 작업 스케줄러(410)에 의해 조절될 수 있다.

[0052] SIRU(420)의 동작은 작업 스케줄러(task scheduler)(410)에 의해 조절될 수 있다. 작업 스케줄러(410)는 FSM(Finite State Machine) 컨트롤러를 포함할 수 있다. 작업 스케줄러(410)는 FSM 컨트롤러를 통해 멀티플렉서의 선택 신호(예: SEL0, SEL1, 또는 SEL2)를 조절할 수 있다. 작업 스케줄러(410)는 멀티플렉서에 대한 입력 데이터 중 어떤 입력을 출력할지 결정하기 위한 선택 신호를 조절할 수 있다. 예를 들어, 작업 스케줄러(410)는 멀티플렉서의 선택 신호를 조절함으로써, 멀티플렉서 각각에 대한 입력 데이터 중 1개를 선택할 수 있다. SIRU(420)는 예를 들어, 3개의 멀티플렉서들로부터 3개의 개별 출력(예: MUX #0 내지 #2 각각의 출력)을 복수의 프로세싱 엘리먼트들(430)에 전달할 수 있다. 추가적으로, 작업 스케줄러(410)는 클록 게이팅 컨트롤러(clock gating controller)를 포함할 수 있다. 작업 스케줄러(410)는 클록 게이팅 컨트롤러를 통해 SIRU(420) 및 복수의 프로세싱 엘리먼트들(430) 중 비활성 상태의 시프트 레지스터와 프로세싱 엘리먼트를 꺼서 전체 전력 소비 평균 20.7% 감소할 수 있다.

[0053] 참고로 FSM 컨트롤러는, 유한 상태 기계의 동작을 제어하는 장치를 포함할 수 있다. 유한 상태 기계는 특정 시스템이나 프로세서의 동작을 나타내는 수학적 모델로, 상태, 이벤트, 전이, 동작이 수행되는 하드웨어를 나타낼 수 있다. 예를 들어, 유한 상태 기계는 SIRU(420) 및 복수의 프로세싱 엘리먼트들(430)을 나타낼 수 있다.

[0054] 종합하면, 작업 스케줄러(410)는 컨볼루션 연산의 종류에 기초하여, SIRU(420)의 동작을 제어할 수 있다. SIRU(420)는 작업 스케줄러(410)로부터 컨볼루션 연산의 종류에 따른 입력 데이터를 선택하기 위한 선택 신호(예: SEL0, 1, 2)를 수신하고, 선택된 입력 데이터를 멀티플렉서로부터 복수의 프로세싱 엘리먼트(430)에 전파할 수 있다. 따라서, SIRU(420)는 유연한 데이터 흐름을 가능케 할 수 있다.

[0056] 도 5 내지 도 6은 일 실시예에 따른 SIRU의 입력 데이터 업데이트 모드(update mode) 및 재사용 모드(reuse mode)에 대하여 도시한다.

[0057] SIRU(500, 600)는 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 출력 채널맵의 개수 및 컨볼루션 연산 종류에 대응하는 커널맵의 가중치 간격에 대응하는 수치에 기초하여, 결정된 개수의 시

프트 레지스터들을 활성화할 수 있다. 활성화된 시프트 레지스터들은 입력 데이터를 저장한 상태의 시프트 레지스터를 나타낼 수 있다. 예를 들어, 활성화된 시프트 레지스터의 수는 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류에 기초하여 결정된 타일 채널 계수(tile channel rate, TR) 및 확장 계수(dilated rate, DR)에 기초하여 결정된 홀더 길이(예: η)에 의해 정해질 수 있다. 예를 들어, 홀더 길이(η)는 타일 채널 계수(TR)X확장 계수(DR)를 나타낼 수 있다. 타일 채널 계수(TR)는 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류의 출력 채널맵의 개수를 연산 가능한 채널의 수로 나눈 값을 포함할 수 있다. 예를 들어, CNN 연산 가속 장치에 포함된 복수의 프로세싱 엘리먼트의 수가 행X열을 기준으로, 3X8에 대응하는 경우, 연산 가능한 채널의 수는 8일 수 있다. 예를 들어, CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류의 출력 채널맵의 개수가 32개이고, 연산 가능한 채널의 수는 8인 경우, 타일 채널 계수(TR)은 4를 나타낼 수 있다. 참고로, 이하에서는 타일 채널 계수와 재사용 계수를 동일한 의미로 사용하는 것으로 상정한다.

[0058] 확장 계수(DR)는 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산 종류의 커널 맵이 갖는 가중치간의 간격을 나타낼 수 있다. 예를 들어 9X9 커널 맵에 가중치가 2칸 간격으로 위치하는 경우 DR은 2를 나타낼 수 있다. 참고로, 확장 컨볼루션(dilated convolution)을 제외한 다른 컨볼루션 연산(예: 스탠다드 컨볼루션, DSC, 트랜스포즈 컨볼루션, 위치별 컨볼루션 등)에서 DR은 1로 고정될 수 있다.

[0059] 이하에서는, TR이 4이고, DR이 1로서, 홀더 길이(η)가 4인 경우의 스탠다드 컨볼루션(STD) 연산을 기준으로 SIRU(500, 600)의 업데이트 모드 및 재사용 모드 동작을 설명한다. 홀더 길이(η)가 4인 경우 멀티플렉서에 의한 선택 신호는 복수의 시프트 레지스터들 중 4번째 시프트 레지스터에 고정되어, 4번째 시프트 레지스터에 저장된 입력 데이터가 멀티플렉서에 의해 출력될 수 있다. 그러나, SIRU의 동작은 이에 제한되는 것은 아니다.

[0060] 도 5에서 일 실시예에 따른 SIRU(500)는 업데이트 모드에서 새로운 입력 데이터(501)를 업데이트 할 수 있다. SIRU(500)는 입력 피처맵 메모리(IFMEM)(미도시)로부터 입력 채널 순서로 새로운 입력 데이터(501)를 수신할 수 있다. SIRU(500)는 복수의 입력 채널맵 각각의 동일한 위치에 배열된 입력 데이터를 미리 결정된 순서로 저장하는 복수의 시프트 레지스터들(shift registers)을 포함할 수 있다. 예를 들어, SIRU(500)는 제1 입력 피처맵, 제2 입력 피처맵, 내지 제n 입력 피처맵이 각각 5X5 크기의 피처맵인 경우, 제1 입력 피처맵의 1행 1열에 위치한 입력 데이터, 제2 입력 피처맵의 1행 1열에 위치한 입력 데이터, 내지 제n 입력 피처맵의 1행 1열에 위치한 입력 데이터를 순차적으로 시프트 레지스터들에 저장할 수 있다. SIRU(500)는 제1 내지 제n 입력 피처맵 각각의 각 1행 1열에 위치한 입력 데이터를 시프트 레지스터들에 저장한 후, 제1 내지 제n 입력 피처맵 각각의 1행 2열에 위치한 입력 데이터를 시프트 레지스터들에 순차적으로 저장할 수 있다.

[0061] SIRU(500)는 새로운 입력 데이터(501)를 저장하면서, 저장된 기존 데이터(510a, 510b, 510c)를 이동할 수 있다. 예를 들어, SIRU(500)는 새로운 입력 데이터(501)(예: A_3)를 시프트 레지스터에 입력하면서, 기존 데이터를 동일한 멀티플렉서와 연결된 시프트 레지스터의 자리수가 증가하는 방향으로 이동할 수 있다. 예를 들어, SIRU(500)는 시프트 레지스터 1번째 자리에 저장된 기존 데이터를 시프트 레지스터 2번째 자리로 이동할 수 있다. 또한, SIRU(500)는 새로운 입력 데이터(501)를 시프트 레지스터에 입력하면서, 홀더 길이(η)에 대응하는 시프트 레지스터에 저장된 기존 데이터(510a, 510b, 510c)를 다음 멀티플렉서의 제1 시프트 레지스터로, 또는 복수의 프로세싱 엘리먼트들(520)로 이동할 수 있다. 예를 들어, SIRU(500)는 새로운 입력 데이터(501)를 시프트 레지스터에 저장하면서, 홀더 길이(η) 4에 대응하는 4번째 시프트 레지스터에 저장된 기존 데이터(510a, 510b)(예: A_2, A_1)를 다음 멀티플렉서의 1번째 시프트 레지스터 자리로 이동하고, 기존 데이터(510a, 510b, 510c)(예: A_2, A_1, A_0)를 복수의 프로세싱 엘리먼트들(520)에 전달할 수 있다.

[0062] 따라서, SIRU(500)는 업데이트 모드에서 홀더 길이(η) 번째의 시프트 레지스터에 선택 신호를 고정하고, 새로운 입력 데이터(501)를 업데이트 함으로써 홀더 길이 번째의 시프트 레지스터에 저장된 기존 데이터(510a, 510b, 510c)가 매번 이전 시프트 레지스터에 저장된 입력 데이터로 변화되도록 할 수 있다. SIRU(500)는 홀더 길이 번째의 시프트 레지스터에 저장된 입력 데이터를 멀티플렉서의 출력으로써 고정할 수 있으므로, 기존 4번째 시프트 레지스터에 저장된 기존 데이터(510a, 510b, 510c)가 출력 된 후, 이전 3번째 시프트 레지스터에 저장된 입력 데이터를 복수의 프로세싱 엘리먼트들(520)에 전달하는 패턴을 형성할 수 있다.

[0063] 도 6에서 일 실시예에 따른 SIRU(600)는 재사용 모드에서 시프트 레지스터에 저장된 입력 데이터를 재사용할 수

있다.

[0064] 일 실시예에 따른 SIRU(600)는 이전 모드(예: 업데이트 모드)를 통해 다음 동작(예: 커널 연산)을 위해 필요한 입력 피처맵들에 포함된 입력 데이터가 업데이트 되었을 때, 재사용 모드로 동작할 수 있다. 예를 들어, SIRU(600)는 홀더 길이(η) 번째의 시프트 레지스터에 저장된 입력 데이터부터 제1 번째 시프트 레지스터에 저장된 입력 데이터까지 스윕(sweep)(610)함으로써, SIRU(600)에 저장된 입력 데이터를 재사용 할 수 있다. 예를 들어, 홀더 길이(η)가 4인 경우, SIRU(600)는 4번째 시프트 레지스터에 저장된 입력 데이터를 멀티플렉서를 통해 출력하여 복수의 프로세싱 엘리먼트들에 전달할 수 있고, 그 다음에 3번째 내지 제1 번째 시프트 레지스터에 저장된 입력 데이터를 순차적으로 스윕(610)함으로써 멀티플렉서를 통해 출력하여 복수의 프로세싱 엘리먼트들에 전달할 수 있다.

[0066] 도 7은 일 실시예에 따른 SIRU에서의 스탠다드 컨볼루션 연산 수행에 따른 타이밍도를 나타낸다.

[0067] 일 실시예에 따른 SIRU는 업데이트 모드와 재사용 모드로 동작할 수 있다. SIRU는 입력 데이터를 모두 업데이트 한 후, 메모리로부터 다시 입력 데이터를 로딩하지 않고, 업데이트 된 입력 데이터를 재사용 할 수 있다. 예를 들어, 도 7은 확장 계수(DR)는 1, 재사용 계수(TR)는 4, 및 출력 채널맵의 개수가 32인 경우를 나타낸다. SIRU의 시프트 레지스터에 업데이트된 입력 데이터는 스윕(sweep)횟수 (즉, 재사용 계수)만큼 다시 사용될 수 있다. 예를 들어, SIRU는 업데이트 모드(710)에서 입력 데이터(예: A_3)를 시프트 레지스터로 업데이트할 수 있다. SIRU는 업데이트 모드(710)에서 시프트 레지스터에 저장된 기존 데이터(예: A_0 내지 A_2)를 멀티플렉서의 출력으로하여 복수의 프로세싱 엘리먼트들로 전달할 수 있다. 복수의 프로레싱 엘리먼트들은 SIRU의 업데이트 모드(710)에서 기존 데이터(예: A_0 내지 A_2) 및 커널 맵 가중치(예: kernel 24 내지 31) 간의 가중합 연산을 수행함으로써, 출력 피처맵(예: C24 내지 C31)을 출력할 수 있다. 재사용 계수(TR)가 4이므로, SIRU는 업데이트 모드(710)에서 업데이트된 입력 데이터를 포함하여 기존 데이터와 함께 입력 데이터를 4회 스윕(sweep)함으로써 재사용할 수 있다. 재사용 모드(720)에서 SIRU는 기존 데이터(예: A_1 및 A_2) 및 업데이트 된 데이터(예: A_3)를 4회 스윕함으로써 커널 맵 가중치(예: kernel 0 내지 7, kernel 8 내지 15, kernel 16 내지 23, kernel 24 내지 31)와 입력 데이터 간의 가중합 연산을 수행함으로써 출력 피처맵(예: C0 내지 C7, C8 내지 C15, C16 내지 C23, C24 내지 C31)을 출력할 수 있다. 특히, 재사용 모드(720)에서 재사용 계수에 대응하는 마지막 입력 데이터 스윕 동작은 업데이트 모드에서 동작할 수 있다. 예를 들어, 입력 피처맵에 포함된 입력 데이터는 컨볼루션 연산 시 출력 채널맵의 수에 따라 재사용될 수 있고, 재사용 계수는 CNN 모델에 포함된 해당 레이어와 복수의 프로세싱 엘리먼트들의 어레이에서 동시에 처리할 수 있는 출력 채널맵의 개수에 따라 조정될 수 있다. 예를 들어, 재사용 계수(TR)는 (출력 채널맵의 개수/연산 가능한 출력 채널맵의 수)로 정의될 수 있다. 예를 들어, 위에서 설명한 바와 같이 복수의 프로세싱 엘리먼트들이 행 X 열(8 X 3)의 시스톨릭 어레이 구조로 배열된 경우, 복수의 프로세싱 엘리먼트들은 8개의 행의 개수만큼의 서로 다른 커널 연산을 수행할 수 있으며, 복수의 프로세싱 엘리먼트들은 위치별 컨볼루션(pointwise convolution)에 대응하여 24개의 커널 연산을 수행할 수 있다. 도 7에서, 32개의 출력 채널 맵을 갖는 스탠다드 컨볼루션 연산은 최대 4의 재사용 계수를 가질 수 있다. 재사용 모드(720)에서 A_1 내지 A_3의 입력 데이터에 대한 가중합 연산이 완료되면 새로운 A_4 입력 데이터가 SIRU에 업데이트 될 수 있다. 그러나 입력 데이터를 로딩하기 위해 가중합 연산 작업을 중단하면 프로세싱 엘리먼트들의 활용도가 낮아질 수 있으므로, SIRU는 재사용 모드(720)의 마지막 입력 데이터 스윕에서 업데이트 모드(721)로 전환하고, SIRU의 멀티플렉서의 출력을 홀더 길이 번째의 시프트 레지스터로 고정할 수 있다. 새로운 입력 데이터(예: A_4)가 SIRU의 시프트 레지스터에 저장됨에 따라, SIRU 내에 저장된 기존의 입력 데이터는 각 사이클마다 인접한 시프트 레지스터로 순차적으로 이동할 수 있다. 선택 신호는 홀더 길이 번째의 시프트 레지스터에 고정되어 있으므로, 입력 데이터가 시프트 레지스터로 순차적으로 이동함에 따라 입력데이터를 재사용 모드에서와 같이 복수의 프로세싱 엘리먼트에 전달하는 패턴을 형성할 수 있다.

[0069] 도 8a 내지 도 8e는 일 실시예에 따르는 CNN 연산 가속 장치에 의해 수행될 수 있는 서로 다른 컨볼루션 연산에 필요한, 입력 피처맵, 커널 맵 및 출력 채널맵을 나타낸다.

[0070] 도 8a는 홀더 길이(η)가 4인 경우 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 스탠다드 컨볼루션

연산(standard convolution)(800a)을 도시한다. 홀더 길이(η)가 4인 경우에 대하여는 위의 설명과 중복되므로 생략한다. 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 스탠다드 컨볼루션 연산(800a)은 입력 피처맵 4개, 가중치 커널 맵 4개씩 총 8 묶음(예: kernel 0 내지 kernel 7) 간의 컨볼루션 연산을 나타낼 수 있다. 입력 피처 맵 4개와 가중치 커널 맵 4개씩 총 8묶음의 스탠다드 컨볼루션 연산(800a)을 통해 획득된 출력 피처 맵(예: O_0^0 내지 O_0^7)은 가중치 커널 맵 묶음 수(예: 8개)에 대응할 수 있다.

[0071] 도 8b는 홀더 길이(η)가 4인 경우 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 확장 컨볼루션 연산(dilated convolution, D-CONV)(800b)을 도시한다. 확장 컨볼루션 연산(800b)에서, 가중치 커널 맵에 포함된 가중치 간의 간격을 나타내는 확장 계수는 2에 대응하므로, 연산 가능한 입력 채널 맵의 수는 확장 계수 1에 대응하는 스탠다드 컨볼루션 연산(800a)의 경우와 다를 수 있다. 예를 들어, 확장 컨볼루션 연산(800b)은 입력 피처맵 2개, 가중치 커널 맵 2개씩 총 8묶음 간의 컨볼루션 연산을 나타낼 수 있다. 입력 피처 맵 2개와 가중치 커널 맵 2개씩 총 8묶음 간의 확장 컨볼루션 연산(800b)을 통해 획득된 출력 피처 맵(예: O_0^0 내지 O_0^7)은 가중치 커널 맵 묶음 수에 대응할 수 있다.

[0072] 도 8c는 홀더 길이(η)가 4인 경우 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 깊이별 컨볼루션 연산(depth-wise convolution, DWC)(800c)을 도시한다. 예를 들어, 깊이별 컨볼루션 연산(800c)은 입력 피처맵 1개, 가중치 커널 맵 1개 간의 컨볼루션 연산을 나타낼 수 있다.

[0073] 도 8d는 홀더 길이 (η)가 4인 경우 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 트랜스포즈 컨볼루션 연산(transposed convolution, T-CONV)(800d)을 도시한다. 예를 들어, 트랜스포즈 컨볼루션 연산(800d)은 제1 시간(예: t_1)에서, 입력 피처맵에 포함된 입력 데이터 중 A_0 내지 A_2를 가중치 커널 맵으로 스위핑함으로써 출력 피처맵(예: O_0^0, O_1^0, O_2^0)을 생성하고, 제2 시간(예: t_2)에서, 입력 피처맵을 스위핑 함으로써 출력 피처맵(예: O_2^0, O_3^0, O_4^0)을 생성하는 컨볼루션 연산을 나타낼 수 있다.

[0074] 도 8e는 홀더 길이 (η)가 4인 경우 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 위치별 컨볼루션 연산(point-wise convolution, PWC)(800e)을 도시한다. 위치별 컨볼루션 연산(800e)은 입력 피처맵들 각각의 동일한 위치에 포함된 입력 데이터에 1X1 크기의 커널 맵을 적용함으로써 출력 피처맵을 획득하는 컨볼루션 연산을 나타낼 수 있다. 예를 들어, 위치별 컨볼루션 연산(point-wise convolution, PWC)(800e)은 5X5의 입력 데이터를 포함하는 제1 입력 피처맵 내지 제4 입력 피처맵에 대하여 1X1 크기의 커널 맵(예: kernel 0 내지 kernel 23)을 스윕함으로써 출력 피처맵(예: O_0^1 내지 O_0^{23})을 획득하는 컨볼루션 연산을 나타낼 수 있다.

[0076] 도 9a내지 도 9b는 일 실시예에 따른 CNN 연산 가속 장치에서 도 8a 내지 도 8e에 대응하는 컨볼루션 연산을 수행하기 위해, SIRU에서 복수의 프로세싱 엘리먼트들로의 입력 데이터 인가 방법에 대하여 도시한다.

[0077] 도 9a는 도 8a 내지 도 8d의 컨볼루션 연산을 수행하기 위해, SIRU(900a)에서 복수의 프로세싱 엘리먼트 행들(910b)로의 입력 데이터 인가 방법을 나타낸다. 즉, 도 9a에 도시된 SIRU(900a)로부터 복수의 프로세싱 엘리먼트 행들(910b)로의 입력 데이터 인가 동작은, 도 8a 내지 8d의 스탠다드 컨볼루션 연산(800a), 확장 컨볼루션 연산(800b), 깊이별 컨볼루션 연산(800c), 및 트랜스포즈 컨볼루션 연산(800d)을 수행하기 위한 입력 데이터 인가 동작에 대응할 수 있다. 예를 들어, 도 8b의 확장 컨볼루션 연산(800b)의 가중합 연산(예: MAC 연산)의 수는, 가중치 커널 맵에 삽입된 0(예: 가중치 커널 맵의 제1 행에 배열된 데이터 [W_0, 0, W_1, 0, W_2] 중 0의 값에 대응하는 데이터)을 제외하고, 도 8a의 스탠다드 컨볼루션 연산(800a)에서의 가중합 연산의 수와 동일할 수 있다. 따라서, 도 9a에 도시된 SIRU(900a)로부터 복수의 프로세싱 엘리먼트들로의 입력 데이터 인가 동작은 도 8a의 스탠다드 컨볼루션 연산 및 도 8b의 확장 컨볼루션 연산 각각에서 동일할 수 있다.

[0078] 예를 들어, CNN 연산 가속 장치는 스탠다드 컨볼루션 연산 및 확장 컨볼루션 연산을 수행하는 경우, 8개의 커널(예: kernel 0 내지 kernel 7) 중 제1 행의 유효한 가중치(예: [W_0, W_1, W_2])를 가중치 인젝터를 통해 8개의 프로세싱 엘리먼트 행(예: PE Row 0 내지 PE Row 7)(910a)에 인가할 수 있다. 예를 들어, CNN 연산 가속 장치는

가중치 커널맵 중 kernel 0의 제1행에 포함된 가중치(예: [$W_{0,0}$, $W_{1,0}$, $W_{2,0}$])를 프로세싱 엘리먼트 행(910a) 중 제1행인 PE Row 0에 인가할 수 있다. 동시에, CNN 연산 가속 장치는 가중치 커널맵 중 kernel 1의 제1행에 포함된 가중치(예: [$W_{0,1}$, $W_{1,1}$, $W_{2,1}$])를 프로세싱 엘리먼트 행(910a) 중 제2행인 PE Row 1에 인가할 수 있다. 마찬가지로, CNN 연산 가속 장치는 가중치 커널맵 중 kernel 7의 제1행에 포함된 가중치(예: [$W_{0,7}$, $W_{1,7}$, $W_{2,7}$])를 프로세싱 엘리먼트 행(910a) 중 제8행인 PE Row 7에 인가할 수 있다.

[0079] SIRU(900a)에서 입력 데이터가 래치(latch)된 시프트 레지스터의 거리는 η를 나타낼 수 있다. CNN 연산 가속 장치는 SIRU(900a)에 포함된 시프트 레지스터에 래치된 입력 데이터 중 η번째 시프트 레지스터에 선택 신호를 고정함으로써, 입력 데이터에 포함된 0을 건너뛰고 유효한 입력 데이터를 프로세싱 엘리먼트 행(910a)에 인가할 수 있다. 예를 들어, 확장 계수(DR)는 1인 스탠다드 컨볼루션 연산을 CNN 연산 가속 장치가 수행하는 경우, SIRU(900a)에 래치된 입력 데이터는 입력 피처맵의 순서(예: 제1 입력 피처맵에서 제n 입력 피처맵의 순서)대로 저장된 입력 데이터일 수 있고, CNN 연산 가속 장치는 매 컨볼루션 연산 사이클 마다 각 입력 피처맵의 행(row)에 저장된 입력 데이터를 순차적으로 복수의 프로세싱 엘리먼트 행(910a)에 전달할 수 있다. PE Row 0에 전달된 입력 데이터는 프리패치(prefetch)된 가중치(예: [W_0, W_1, W_2])와 곱해질 수 있다. 입력 데이터와 가중치 간의 곱셈 결과는 합산된 출력 피처맵(예: O_0^0)을 생성하기 위해 가산기 트리에 입력될 수 있다. 참고로, 출력 피처 맵(예: O_N^C)에서 N은 출력 피처맵의 순서를 나타내고, C는 출력 채널을 나타낼 수 있다. CNN 연산 가속 장치는, 입력 피처맵에 포함된 입력 데이터의 행에 대하여 8주기(예: t_0 내지 t_7) 동안 8개의 채널맵에 대응하는 출력 피처맵(예: O_0^0 내지 O_0^7)을 생성할 수 있다.

[0080] CNN 연산 가속 장치는 깊이별 컨벌루션 연산(DWC)을 수행하는 경우, 위에서 설명한 복수의 입력 피처맵을 사용하는 스탠다드 컨볼루션 연산을 수행하도록 입력 데이터를 인가하는 동작과 유사하게, 단일 입력 피처맵을 사용하는 스탠다드 컨벌루션 연산처럼 동작할 수 있다.

[0081] CNN 연산 가속 장치는 출력 피처맵을 축적 코어(920a)에서 누적함으로써 부분합 결과를 생성할 수 있다.

[0082] CNN 연산 가속 장치는 트랜스포즈 컨볼루션 연산(T-CONV)을 수행하는 경우, 트랜스포즈 컨볼루션 연산에서는 입력 피처맵만 입력 데이터 간에 0이 추가됨으로써 확장되므로, 하나의 프로세싱 엘리먼트 행(910a)에서 3가지 다른 가중합을 생성할 수 있다. 예를 들어, 제1 시간(예: t_1)과 제2 시간(예: t_2)의 가중합을 누적한 결과는 도 8d에서와 같이 적어도 두 시간 사이클(예: t_1, t_1)에 걸쳐 생성될 수 있다. CNN 연산 가속 장치는 짝수 출력 피처맵(예: O_2^0)에 대하여 완전한 부분 합을 생성하고, 홀수 출력 피처맵(예: O_1^0, O_3^0)은 축적 코어(920a)에 한 주기 동안 래치(latch)(또는 일시 저장) 한 후 다음 주기에 누적함으로써 출력 피처 메모리(OFMEM)로 전달할 수 있다.

[0084] 도 9b는 도 8e의 위치별 컨볼루션 연산(PWC)을 수행하기 위해, SIRU(900b)에서 복수의 프로세싱 엘리먼트 행들(910b)로의 입력 데이터 인가 방법을 나타낸다. 참고로, 위치별 컨볼루션 연산에서는 1X1 크기의 커널 맵과 1X1 크기의 입력 피처맵 간에 가중합 연산이 수행된다. 따라서, 컨볼루션 엔진에 포함된 복수의 프로세싱 엘리먼트들 각각은 서로 다른 출력 채널맵에 대응하는 1X1 크기의 출력 피처맵을 생성할 수 있다. 예를 들어, 8X3의 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들은 최대 24개의 커널 연산에 대응하는 가중합 연산을 수행할 수 있다. CNN 연산 가속 장치는 SIRU(900b)에 포함된 3개의 멀티플렉서 모듈(예: MUX 및 시프트 레지스터들을 포함하는 모듈)에서 동일한 출력을 복수의 프로세싱 엘리먼트 행(910b)에 전달할 수 있다. 예를 들어, CNN 연산 가속 장치는 SIRU(900b)에 포함된 멀티플렉서 모듈 중 MUX #0을 포함하는 멀티플렉서 모듈의 4번째 시프트 레지스터에 저장된 입력 데이터를, MUX #0을 포함하는 멀티플렉서 모듈, MUX #1을 포함하는 멀티플렉서 모듈 및 MUX #2를 포함하는 멀티플렉서 모듈로부터 출력하여, 프로세싱 엘리먼트 행(910b)(예: PE Row 0)에 전달할 수 있다. CNN 모델에서 제1 레이어와 같이 얕은 네트워크 작업에서, SIRU(900b)에서 프로세싱 엘리먼트 행(910b)으로의 입력 데이터 인가 경로는 스탠다드 컨볼루션에서의 입력 데이터 인가 경로와 유사할 수 있고, 8개의 채널에 대해 3가지 다른 출력 피처맵을 생성할 수 있다.

[0085] 결론적으로, CNN 연산 가속 장치는, SIRU(900a, 900b)에서 피연산자(operand)의 복잡한 재배열 없이 입력 데이터의 전파(propagation)를 통해 입력 피처맵에 포함된 입력 데이터를 복수의 프로세싱 엘리먼트들에 제공함으로써 다양한 유형의 컨볼루션 연산을 수행할 수 있다.

[0087] 도 10은 일 실시예에 따른 CNN 연산 가속 장치에 의해 수행되는 깊이별 분산가능 컨볼루션 연산의 데이터 흐름을 나타낸다.

[0088] 깊이별 분산가능 컨볼루션 연산(depth-wise separable convolution, DSC)의 소프트웨어 수준 최적화(optimization)는 스탠다드 컨볼루션 연산에 비해 매개변수의 수와 연산 비용을 감소시켰다. 그러나 하드웨어에서 DSC를 가속화하려면 두 가지 주요 과제가 해결되어야 한다. 첫 번째 서로 다른 연산 특성을 갖는 깊이별 컨볼루션 연산(DWC) 및 위치별 컨볼루션 연산(PWC)에 대한 수행이 필요하다. 둘째, CNN 모델에 포함된 하나의 레이어를 두 개로 분할하여 DWC 및 PWC를 수행하면 분할된 레이어 간에 발생하는 중간 데이터로 인해 추가 외부 메모리 접근(external memory access, EMA)이 발생할 수 있다. 따라서, 이하에서는 DWC 및 PWC에 대한 병렬 작업 및 파이프라인 작업이 가능한 하드웨어가 제안된다.

[0089] 일 실시예에 따른 CNN 연산 가속 장치는 DWC와 PWC의 병렬 동작을 위해 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들을 포함할 수 있다. 복수의 프로세싱 엘리먼트들은 서로 다른 두 종류의 프로세싱 엘리먼트를 포함할 수 있다. 서로 다른 두 종류의 프로세싱 엘리먼트들에 대하여는 이미 도3에서 설명하였으므로 중복된 내용은 생략한다. 복수의 프로세싱 엘리먼트들은 시스톨릭 어레이 구조의 제2열(column)(1020)에 제2 프로세싱 엘리먼트를 포함할 수 있고, 제2 열(1020) 이외의 열에 제1 프로세싱 엘리먼트를 포함할 수 있다. 다만, 행 또는 열은 시계 방향(또는 반시계 방향)으로 90도 회전하는 경우 행은 열로, 열은 행으로 대체될 수 있으며, 이하에서는 도 10에 배열된 3행 8열의 시스톨릭 어레이 구조로 배열된 복수의 프로세싱 엘리먼트들을 기준으로 설명한다.

[0090] CNN 연산 가속 장치는 시스톨릭 어레이 구조의 제1열(1000)에 배열된 프로세싱 엘리먼트들에 SIRU로부터 입력 데이터를 인가하여, 단일 채널의 커널 맵에 대응하는 MAC연산을 생성할 수 있다. 예를 들어, CNN 연산 가속 장치는 복수의 프로세싱 엘리먼트들을 각각 포함하는 3개의 컨볼루션 엔진의 제1열(1000)에 배열된 프로세싱 엘리먼트들에 SIRU로부터 4개 채널맵에 대응하는 3X3 입력 피처맵을 수신함으로써 DWC 연산을 수행할 수 있다. 축적 코어는 제1열(1000)에서 생성된 출력을 누적함으로써, DWC의 최종 출력 피처 맵(즉, DWC 다음의 PWC에 대한 입력 피처맵)을 생성할 수 있다.

[0091] CNN 연산 가속 장치는 시스톨릭 어레이 구조의 각 열(column)에 배열된 프로세싱 엘리먼트로부터 행(row) 방향을 따라 배열된 프로세싱 엘리먼트로 입력 데이터를 전파할 수 있다. 예를 들어, CNN 연산 가속 장치는 SIRU로부터 입력된 입력 데이터를 제1열(1000)에 배열된 프로세싱 엘리먼트에서 DWC 연산 후 제2열(1020)에 배열된 프로세싱 엘리먼트로 전파할 수 있다. CNN 연산 가속 장치는 입력 데이터를 제1열(1000) 내지 제8열에 배열된 프로세싱 엘리먼트들로 전파하면서 가중합 연산을 수행하므로, 제1열(1000)에 배열된 프로세싱 엘리먼트들을 제외한 나머지 프로세싱 엘리먼트들은 PWC의 병렬 가속을 수행하기 위해 누적된 입력 피처맵을 제1열(1000)에서부터 제8열 까지 전파할 수 있다. 제2열(1020)에 배열된 제2 프로세싱 엘리먼트는, 축적 코어로부터 제1열(1000)에서의 연산 결과에 기초하는 값을 입력 받기위한 추가 입력 경로(1010)를 통해, 4채널의 피처맵을 수신할 수 있다. 제2열(1020)에 배열된 제2 프로세싱 엘리먼트에 의한 가중합 연산 결과는 제3 내지 제8열에 배열된 프로세싱 엘리먼트들에 행(row)방향을 따라 전파됨으로써, 3개의 컨볼루션 엔진에 포함된 63개의 프로세싱 엘리먼트들에 의한 PWC 연산을 수행할 수 있다. 일반적으로 CNN 모델에 포함된 컨볼루션 연산의 출력 채널맵의 수는 2의 거듭제곱이므로, 축적 코어는 하위 프로세싱 엘리먼트로 최종 64개의 출력 채널맵에 대한 피처맵 생성 동작을 완료할 수 있다. 결과적으로 CNN 연산 가속 장치는 36개의 DWC 및 256개의 PWC 연산을 병렬적으로 수행할 수 있다.

[0093] 도 11a 내지 도 11b는 일 실시예에 따른 CNN 연산 가속 장치에 포함된 컨불루션 엔진의 동작 흐름도를 나타낸다.

[0094] 도 11a는 일 실시예에 따른 CNN 연산 가속 장치에서 컨볼루션 엔진(1100a)의 동작 흐름을 나타낸다. 우선 가중치 인젝터(weight injector)는 가중치 메모리(예: WMEM)로부터 각 프로세싱 엘리먼트들로 서로 다른 커널 맵에 포함된 가중치를 인가할 수 있다. 예를 들어, 제1 가중치 메모리(예: WMEM #0)에 저장된 가중치는 시스톨릭 어

레이 구조로 배열된 복수의 프로세싱 엘리먼트들 중 제1 행(예: PE Row #0)에 배열된 프로세싱 엘리먼트들로 가중치 인젝터를 통해 인가될 수 있다. 다른 예를 들어, 제2 가중치 메모리(예: WMEM #1)에 저장된 가중치는 제2 행(예: PE Row #1)에 배열된 프로세싱 엘리먼트들에 가중치 인젝터를 통해 인가될 수 있다.

[0095] 컨볼루션 엔진(1100a)에 포함된 SIRU는 CNN 모델에 포함된 레이어에 대응하는 컨볼루션 연산에 기초하여 입력 데이터를 재정렬할 수 있다. SIRU는 컨볼루션 연산 종류에 맞게 재정렬된 입력 데이터를 제1 행에 배열된 프로세싱 엘리먼트들로 전달할 수 있다. 제1행에 배열된 프로세싱 엘리먼트들은 저장된 가중치 및 입력 데이터에 기초하여 가중합 연산을 수행할 수 있다. 제1행에 배열된 프로세싱 엘리먼트들에 공급된 입력 데이터는 8 사이클 동안 다음 행에 배열된 프로세싱 엘리먼트들로 전파됨으로써 재사용될 수 있다. 예를 들어, SIRU로부터 PE Row #0으로 인가된 입력 데이터는 PE Row #0에서 가중합 연산이 수행되고 동시에 PE Row #1으로 전파될 수 있다. 각 프로세싱 엘리먼트들에서의 가중합 연산 결과는 축적 코어에 누적될 수 있다. 예를 들어, PE Row #0에서의 가중합 연산 결과는 축적 코어에 누적될 수 있다. 동시에 PE Row #1에서의 가중합 연산 결과는 출적 코어에 누적될 수 있다.

[0096] 도 11b는 일 실시예에 따른 CNN 연산 가속 장치에 포함된 컨볼루션 엔진(1100b)에서 깊이별 분산가능 컨볼루션 연산(DSC)을 수행하기 위한 데이터 흐름도를 나타낸다.

[0097] 일 실시예에 따른 컨볼루션 엔진(1100b)에서 가중치 인젝터는 가중치 메모리로부터 복수의 프로세싱 엘리먼트들로 가중치를 인가할 수 있다. 가중치 인젝터를 통한 복수의 프로세싱 엘리먼트들로 가중치를 인가하는 동작은 위에서 설명하였으므로 생략한다. 컨볼루션 엔진(1100b)에서 SIRU는 시스톨릭 어레이 구조의 제1행에 배열된 프로세싱 엘리먼트들로 입력 데이터를 인가할 수 있다. 제1행에 배열된 프로세싱 엘리먼트들은 저장된 가중치 및 입력 데이터에 기초하여 깊이별 컨볼루션 연산(depth-wise convolution)을 수행할 수 있다. 제1행에 배열된 프로세싱 엘리먼트들에 의한 깊이별 컨볼루션 연산 결과는 축적 코어에 누적될 수 있다. 제1행에 배열된 프로세싱 엘리먼트들에 의한 깊이별 컨볼루션 연산 결과는 축적 코어로부터 제2행에 배열된 프로세싱 엘리먼트들로 인가될 수 있다. 즉, 깊이별 컨볼루션 연산 결과를 입력으로 하여 제2행에서 1X1 커널 연산이 수행될 수 있다. 다시 말해, 깊이별 컨볼루션 연산 결과에 대한 위치별 컨볼루션 연산이 제2행에 배열된 프로세싱 엘리먼트들에 의해 수행될 수 있다. 축적 코어로부터 제2행에 배열된 프로세싱 엘리먼트들로 인가된 깊이별 컨볼루션 연산 결과는 제2행에서 제8행까지 전파될 수 있다.

[0099] 도 12는 의미론적 분할을 위한 CNN 기반 모델 아키텍처를 도시한다.

[0100] 자율주행차(AEV) 기술은 높은 정밀도를 입증한 DNN 기반 인식(DNN-based perception)의 발전과 함께 폭넓게 연구되고 있다. AEV에 대한 핵심 기술은 1) 지능형 기계 인식, 2) 경로 계획(path planning), 및 3) 차량 제어의 세 가지 주요 기술을 포함할 수 있다. 특히, 지능형 기계 인식은 AEV의 전체 시스템 지연 시간의 99%를 차지한다. 이때, 기계 인식의 시스템 지연 중 98.9%는 DNN 연산의 수행과 관련된다. 따라서, 지능형 기계 인식은 AEV에서 가장 중요한 핵심 기술로서, 실시간 동작을 위해 DNN의 가속화가 필수적이다.

[0101] 입력 영상에 대한 의미론적 분할(semantic segmentation)은 입력 영상에 대해 픽셀 단위의 정보를 제공함으로써, 자동차가 운전 가능 지역, 보행자, 차량, 장애물 등의 운전 환경을 시각적으로 이해할 수 있도록 한다. 따라서, 기계 인식 기술의 기본 작업으로 의미론적 분할이 요구된다.

[0102] CNN(convolutional neural network)은 운전 환경에서 의미론적 분할을 위해 적합한 신경망으로 평가된다. 그러나, 실시간으로 입력 영상에 대한 의미론적 분할을 수행하는 것은 운전 중에 상당한 에너지를 소비하는 GPGPU를 갖춘 강력한 컴퓨팅 시스템에 의존한다. 거대한 전력 소모 컴퓨팅 시스템은 작동 중 지속적으로 에너지를 공급받지 못하므로, AEV의 주행 거리를 감소시킨다. 따라서, 다양한 AEV 환경에서 의미론적 분할을 수행할 수 있는 것은 다소 어렵다. AEV는 여러 센서를 활용하여 각 센서에서 누락되거나 잘못된 정보를 보상할 수 있다. 특히 컬러 영상(예: RGB 영상)과 깊이 영상(예: depth image)을 융합한 CNN 기반의 멀티 모달 네트워크(multimodal network)는 다양한 주행 환경에서 높은 정확도의 솔루션을 제공할 수 있다. 그러나 AEV에서 RGB-D와 같은 네트워크를 실시간으로 초당 30프레임 동작시키는 것은 몇 가지 어려움이 있다. 첫번째, 주행 환경에서 넓은 시야각을 커버하기 위해서는 입력 영상의 넓고 높은 해상도가 요구된다. 예를 들어, 드라이빙 데이터세트인 KITTI(1248X384) 및 Cityscapes(2048X1024)은 기존 이미지 분류를 위한 대표적인 데이터 세트인 ImageNet(416X416)에 비해 최대 12배 더 큰 해상도를 가질 수 있다. 둘째, 서로 다른 모달리티를 결합하기 위해 깊이 피처들과 RGB 피처들을 레이어별로 융합하려면 방대한 연산과 외부 메모리 대역폭을 필요로 한다. 따

라서, CNN 기반 네트워크(또는 모델)를 구현하기 위해 엣지 컴퓨팅 노드의 실시간 동작을 위한 강력한 솔루션이 요구된다. 특히 네트워크의 계산 복잡성을 줄이기 위한 CNN 기반 모델 아키텍처(1200)가 제안되었다.

[0103] CNN 기반 모델 아키텍처(1200)는 인코더-디코더(encoder-decoder)구조를 포함할 수 있다. 이러한 구조를 기반으로 스탠다드 컨볼루션 연산(STC)의 경량화 및 더 높은 정확도를 위해 네트워크 모델에 다양한 컨볼루션 연산 종류가 적용될 수 있다. 예를 들어, CNN 기반 모델 아키텍처(1200)는 컬러 영상(예: large-scale field of view image)에 대한 의미론적 분할(semantic segmentation)을 수행하는 모델을 포함할 수 있다. CNN 기반 모델 아키텍처(1200)는 예를 들어, 깊이별 분리가능 컨볼루션(DSC), 확장 컨볼루션(D-CONV), 트랜스포즈드 컨볼루션(T-CONV) 연산 각각을 수행하는 레이어를 포함할 수 있다. 그러나, CNN 기반 모델 아키텍처(1200)는 컬러 영상만을 입력으로 하여 의미론적 분할 결과를 출력하므로, AEV에서의 실시간 영상 처리와 같이 고전력이 요구되는 상황에서의 전력 사용량을 감소하기 위한 경량화된 모델이 요구된다. 경량화된 모델에 대하여는 이하 도 13 및 도 14에서 구체적으로 설명한다.

[0105] 도 13은 일 실시예에 따른 CNN 연산 가속 장치를 통해 구현할 수 있는 CNN 기반 모델의 개략도다.

[0106] 자율주행환경은 기후, 시간, 위치 등에 따라 주행 조건이 매우 가변적이다. RGB 기반 분할 네트워크는 정적이고 밝은 조건에서 의미론적 분할에 좋은 결과를 얻을 수 있지만, 다용도 주행(versatile driving)(예: 그늘, 터널 내부, 어두운 환경 등)에서는 정확도고 떨어질 수 있다. 조명 변화 등과 같은 환경에서 취약한 인식은 안전이 중요한 AEV에서 해결되어야 할 과제다. 결과적으로 AEV에서 다중 센서 시스템을 활용하는 다중 모드 네트워크 연구가 제안되었고, RGB-D 네트워크는 RGB네트워크보다 더 높은 정확도를 달성한다. 그러나 RGB 피처 및 깊이(depth) 피처들의 서로 다른 모달리티(modality)를 효과적으로 융합하는 것이 중요한 과제였다. 따라서, AEV의 실시간 다중 객체 분할을 위한 경량화된 깊이 융합 삼중 분할 네트워크(depth-fused trilateral segmentation network, DTN)를 제안한다. DTN은 기존 양자 네트워크 아키텍처(bilateral network architecture)를 기반으로 할 수 있다. 고품질의 예측을 획득하기 위해 의미론적 분할에 문맥 정보(context information)와 공간 정보(spatial information)가 모두 필요할 수 있다. 제안된 DTN은 RGB-D와 양자 네트워크(bilateral network)를 활용한다. 우선 깊이 영상(depth image)에 포함된 객체는 센서로부터의 거리에 따라 본질적으로 구별되어 경계 정보와 객체까지의 깊이 정보를 제공할 수 있다. 깊이 기반 문맥 정보(context information)(1300)는 RGB 데이터만으로는 얻을 수 없는 의미 있는 상황별 정보를 제공할 수 있다. 따라서, 깊이 기반 문맥 정보(1300)를 RGB에 통합하면 개체에 대한 보다 전체적이고 상세한 이해가 가능할 수 있다. 결론적으로 RGB 영상에 대한 공간 정보 및 문맥 정보에 깊이 영상에 대한 깊이 기반 문맥 정보(1300)를 융합하여 DTN 구조를 형성할 수 있다.

[0107] 도 14는 일 실시예에 따른 CNN 연산 가속 장치를 통해 구현 가능한 DTN의 구체적인 구조를 도시한다.

[0108] 일 실시예에 따른 CNN 연산 가속 장치는 컬러 이미지 또는 컬러 이미지에 대응하는 깊이 이미지(depth image) 중 적어도 하나에 기초하는 입력 데이터를 SIRU에 저장할 수 있다. 예를 들어, CNN 연산 가속 장치는 RGB 이미지에 기초하는 입력 데이터를 SIRU에 저장할 수 있다. SIRU에 저장된 RGB 이미지에 기초하는 입력 데이터에 대하여, CNN 연산 가속 장치는 RGB 이미지에 대한 공간 정보를 추출하기 위한 공간 경로(spatial path) 상의 레이어들에 대응하는 컨볼루션 연산을 수행할 수 있다. 예를 들어, RGB 이미지에 대하여, 공간 경로 상의 제1 레이어에서 rate 4인 확장 컨볼루션(D-CONV) 연산이 수행될 수 있다. RGB 이미지에 대하여, 공간 경로 상의 제2레이어에서 rate 2인 확장 컨볼루션 연산이 수행될 수 있다. RGB 이미지에 대하여 공간 경로 상의 마지막 레이어에서 위치별 컨볼루션 연산이 수행될 수 있다.

[0109] 또한, CNN 연산 가속 장치는 RGB 이미지에 기초하는 입력 데이터에 대하여, RGB 이미지에 대한 문맥 정보(context information)를 추출하기 위한 문맥 경로(context path) 상의 레이어들에 대응하는 컨볼루션 연산을 수행할 수 있다. 예를 들어, RGB 이미지에 대하여, 문맥 경로 상의 레이어들에 대응하도록, rate 2인 D-CONV, rate 4인 D-CONV, rate 8인 D-CONV, rate 16인 D-CONV 연산이 수행될 수 있다.

[0110] CNN 연산 가속 장치는, 컬러 이미지에 대응하는 깊이 이미지(depth image)에 기초하는 입력 데이터를 SIRU에 저장할 수 있다. CNN 연산 가속 장치는 제안된 DTN 구조의 깊이 이미지 문맥 경로(1400)에서의 컨볼루션 연산을 구현할 수 있다. 예를 들어, CNN 연산 가속 장치는 시스톨릭 어레이 구조로 배열되고 SIRU와 연결된 복수의 프로세싱 엘리먼트들에서, SIRU에 저장된 깊이 이미지에 기초하는 입력 데이터 및 컨볼루션 연산 종류에 대응하는 가중치 커널 맵의 가중치에 기초하여, 깊이 이미지의 문맥 정보(context information)에 대응하는 연산 결과를

산출함으로써 가중합 연산을 수행할 수 있다. 예를 들어, CNN 연산 가속 장치는 깊이 이미지(depth image)에 기초하는 입력 데이터에 대하여, 깊이 이미지 문맥 경로(1400) 상에 배열된 레이어에 대응하는 연산으로, rate 2의 D-CONV, rate 4의 D-CONV, rate 8의 D-CONV, 및 rate 16의 D-CONV 연산을 수행할 수 있다.

[0111] 종합하면, 일 실시예에 따른 CNN 연산 가속 장치는 DTN을 구현하기 위해, DTN에 포함된 복수의 레이어에 대응하는 컨볼루션 연산을 수행할 수 있다. DTN은 RGB 이미지의 정보에 대한 공간 경로 및 문맥 경로를 포함할 수 있다. 또한, DTN은 RGB 이미지에 대응하는 깊이 이미지의 정보에 대한 문맥 경로(1400)를 포함할 수 있다. DTN에서 공간 경로는 4개의 컨볼루션 레이어를 통해 RGB 이미지에서 하위 수준 공간 정보를 추출할 수 있다. RGB 이미지 문맥 경로 및 깊이 이미지 문맥 경로(1400)는 스템 블록(stem block)을 포함할 수 있고, 스탬 블록을 통해 피처 맵을 축소할 수 있다. 축소된 특징 맵은 ASPP(atrous spatial pyramid pooling)가 제공하는 대규모 수용가능한 필드를 통해 풍부한 문맥 정보를 추출할 수 있다. ASPP에서 추출된 문맥 정보는ARM(Attention Refinement Module)을 통해 정제될 수 있다. 공간 경로 및 문맥 경로에서 추출된 피처맵은 깊이 이미지에 대한 정보 및 RGB 이미지에 대한 정보를 포함하므로 FFM(feature fusion mouule)을 통해 서로 다른 특징을 결합하고 업샘플링링 함으로써 의미론적 분할된 영상을 획득할 수 있다. 문맥 경로는 전체 네트워크 내 연산량의 82%를 차지하는 다양한 규모의 특징을 추출하기 때문에, 상당한 계산 리소스를 요구할 수 있다. 그러나 RGB 이미지에 대한 문맥 경로 및 깊이 이미지에 대한 문맥 경로(1400)를 함께 활용하면, 보완적인 상황 정보를 제공함으로써 추가적인 네트워크 압축(또는 경량화)이 가능할 수 있다. 단일 문맥 경로가 아닌 두 개의 문맥 경로를 사용하더라도 매개변수 수와 연산 복잡성이 각각 42% 또는 36% 감소할 수 있다.

[0112] 일 실시예에 따른 CNN 연산 가속 장치를 통해 제안된 DTN을 구현하는 경우 정확한 의미론적 분할을 위한 대규모 수용 필드 연산(large receptive field computation)을 수행할 수 있다. 하이 퀄리티 의미론(high-quality semantics)을 획득하기 위해 풍부한 공간 정보 및 문맥 정보가 필요하고, 이는 충분한 수용 필드 연산을 통해 획득할 수 있다. 그러나, 큰 커널을 사용하는 것은 매개변수의 수를 증가시키고, 연산량을 커널 사이즈의 제곱만큼 증가시킬 수 있다. 따라서, 큰 커널의 연산 제약을 극복하기 위해 다양한 유형의 컨볼루션 레이어를 지원할 수 있는 하드웨어로서 CNN 연산 가속 장치와 함께 D-CONV를 DTN 네트워크에 적용할 수 있다.

[0113] 제안된 DTN의 공간 경로에서는 넓은 수용 필드와 함께 공간 정보를 인코딩하기 위해 다중 D-CONV 레이어가 적용될 수 있다. 아래 표 1은 대규모 STC 커널과 공간 경로의 순차적 D-CONV 구성 간의 성능 비교를 나타낸다.

표 1

[0114]

Network Configuration of Spatial Path	Val-mIOU [%] @Cityscapes
(Conv 7 x 7) – (Conv 3 x 3) – (Conv 3 x 3)	59.10
Sequential D-CONV with {3, 1, 1}	59.10
Sequential D-CONV with {4, 2, 1}	59.31

[0115] 제안된 CNN 연산 가속 장치를 통해 구현되는 D-CONV는 3X3 커널과 동등한 유효 매개변수의 수 및 연산 기능을 제공할 수 있다. 공간 경로에 대한 초기 대규모 수용 필드 및 축소 인자를 고려하면, 공간 경로는 확장 계수 {4, 2, 1}, 스트라이드(stride) 2, 단일 1X1 컨볼루션 레이어로 구성될 수 있다. 결과적으로 일련의 3X3레이어와 동일한 연산량으로 정확도가 0.21% 증가할 수 있다. 또한, 동일한 크기의 대형 커널 연산에 비해 공간 경로에서 두 개의 D-CONV 레이어의 총 매개변수의 수와 연산을 각각 67.3% 및 73.4% 감소할 수 있다. 제안한 CNN 연산 가속 장치 하드웨어와 알고리즘 간의 공동 최적화를 통해 네트워크는 D-CONV를 최대한 활용하는 동시에 연산 오버헤드를 최소화하고 풍부한 공간 정보 및 문맥 정보를 제공할 수 있다.

[0116] 자원이 제한된 임베디드 시스템의 연산 복잡성을 효과적으로 감소하기 위해 다양한 네트워크 압축 방법이 연구되고 잇다. 특히, DSC는 연산과 매개변수를 줄이는 방법으로 네트워크에 포함될 수 있다. DSC의 효율성은 커널 쿠기에 의해 결정되며, 커널 컨볼루션이 클수록 압축 효과도 클 수 있다. 제안된 DTN 구조에서는 D-CONV와 DSC를 동시에 적용하는 확장 계수-DSC(dilation rate-DSC)가 적용된다. 아래 수학식 1을 통해 연산량 및 매개변수의 수를 줄일 수 있다.

수학식 1

$$MAC_{DSC}, Params_{DSC} \propto \left(1+\frac{9}{N}\right)\frac{1}{(2\times DR+1)^2}$$

[0117]

[0118] 수학식 1에서, MAC_{DSC}는 DSC에서의 MAC연산수를 나타낸다. $Params_{DSC}$는 DSC에서의 매개변수의 수를 나타낸다. N은 출력 채널의 수를 나타낸다. DR은 확장 계수를 나타낸다.

[0120] 도 15는 일 실시예에 따른 CNN 연산 가속 장치의 칩 레이아웃을 나타낸다.

[0121] CNN 연산 가속 장치(1500)는 입력 피처맵 메모리(예: IFMEM), 가중치 메모리(예: WMEM), 출력 피처맵 메모리(예: OFMEM), 탑 컨트롤러(예: Top Ctrl), 축적 코어(예: Aggr), 및 컨볼루션 엔진을 포함할 수 있다. CNN 연산 가속 장치(1500)는 28nm CMOS 기술 공정을 통해 생산될 수 있다. CNN 연산 가속 장치(1500)의 칩 레이아웃에 따르면, CNN 연산 가속 장치(1500)는 가로 1.65nm, 세로 0.9nm 크기로, 1.485nm^2의 면적을 차지할 수 있다. 표 2는 CNN 연산 가속 장치(1500)의 성능 및 스펙에 대하여 개시한다.

표 2

	This Work
Technology	28nm CMOS
Operating Condition	0.9V, 200MHz
Area	1.485 mm^2
Power	43.6 mW
Performance	115.2 GOPS (STC) 215.6 GOPS (DTN)
Framerate	72.2 fps @960x280 (KITTI) 40.07 fps @1024x512 (Cityscapes)
Power Efficiency	4.94 TOPS/W
On-Chip SRAM Size	264 KB

[0122]

[0123] CNN 연산 가속 장치(1500)는, 264KB 온-칩 SRAM을 탑재하고, 0.9V의 공급 전압에서 최대 200MHz의 주파수로 동작할 수 있다. CNN 연산 가속 장치(1500)를 통해 제안된 DTN을 실행할 때, 215.6GOPS를 달성할 수 있다. CNN 연산 가속 장치(1500)는 평균 43.6 mW를 소비할 수 있고, 4.94 TOPS/W의 에너지 효율을 달성할 수 있다. 또한, CNN 연산 가속 장치(1500)는 72.2fps와 40.07fps를 달성할 수 있다. 즉, AEV의 주행 중 DTN을 통한 실시간 영상 분할은 각각 72.2프레임과 40.07프레임을 달성할 수 있다.

[0124] 아래 표 3은 일 실시예에 따른 CNN 연산 가속 장치(1500)와 비교 실시예들의 스펙을 항목에 따라 비교한다.

표 3

	비교 실시예 1	비교 실시예 2	비교 실시예 3	비교 실시예 4	This Work
Technology [nm]	28	40	40	65	28
Operating Condition	1,449MHz	0.85V, 100MHz	1,250 MHz	1.2V, 200MHz	0.9V, 200MHz
Optimized Types of CONV	STC, D/T-CONV	PWC	STC, PWC, T-CONV	STC, PWC, D/T-CONV	STC, PWC, D/T-CONV, DSC
Performance [GOPS]	374.7	79.3	1,440	*664.4 / **149.9	*760.6 / **215.6
Area [mm^2]	-	1.03	2.29	6.8	1.485
Power [mW]	201.1	25.3	526.1	206	43.6
Energy Efficiency [TOPS/W]	1.86	3.17	2.74	*3.22 / **0.73	*17.44 / **4.94
On-chip SRAM	114.7 KB	-	286 KB	256 KB	264 KB

*Based on Logical Throughput with 288×288 on ENET
**Based on Logical Throughput with 1024×512 on DTN

[0125]

[0126] 표 3에서, 비교 실시예 1은 W. Liu, J. Lin and Z. Wang, "USCA: A Unified Systolic Convolution Array Architecture for Accelerating Sparse Neural Network," 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1-5. 문헌의 기재에 따라 구현된 장치일 수 있다. 비교 실시예 2는 Y. S. Chong, W. L. Goh, Y. S. Ong, V. P. Nambiar and A. T. Do, "An Energy-Efficient Convolution Unit for Depth-wise Separable Convolutional Neural Networks," 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1-5. 문헌의 기재에 따라 구현된 장치일 수 있다. 비교 실시예 3은 C. -H. Hu, I. -H. Tseng, P. -H. Kuo and J. -D. Huang, "An SoC Integration Ready VLIW-Driven CNN Accelerator with High Utilization and Scalability," in Proc. IEEE 4th Int. Conf. Artif. Intell. Circuits Syst. (AICAS), 2022, pp. 246-249. 문헌 기재에 따라 구현된 장치일 수 있다. 비교 실시예 4는 D. Im, D. Han, S. Choi, S. Kang and H. -J. Yoo, "DT-CNN: An Energy-Efficient Dilated and Transposed Convolutional Neural Network Processor for Region of Interest Based Image Segmentation," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 10, pp. 3471-3483, Oct. 2020. 문헌 기재에 따라 구현된 장치일 수 있다. 비교 실시예 1내지 4는 D/T-CONV 또는 PWC와 같은 제한된 컨볼루션 연산을 수행에 특화된 장치일 수 있다. 비교 실시예 2는 DSC 연산을 지원하기 위해 PWC에 최적화된 구조를 구현하여 PWC의 경우 96.5%의 활용률을 달성했지만, DWC의 경우 4.1%의 활용률을 달성했다. 비교 실시예 3은 STC, PWC 및 T-CONV를 가속화할 수 있지만, DSC 및 D-CONV은 가속화할 수 없다. 비교실시예 1 내지 4와 달리, 제안된 CNN 연산 가속 장치(1500)는 STC, DSC, D/T-CONV, PWC를 포함하는 총 5가지의 서로 다른 컨볼루션 연산을 지원할 수 있다. 따라서, 제안된 CNN 연산 가속 장치(1500)를 통해 구현된 DTN은 비교 실시예 2 및 비교 실시예3에 비해 각각 1.56배, 1.8배의 높은 에너지 효율을 달성할 수 있다. 비교 실시예 4와 비교하여 제안된 CNN 연산 가속 장치(1500)는 1.43배 높은 성능, 6.76배 높은 에너지 효율을 달성할 수 있다.

[0127] 아래 표 4는 복수의 컨볼루션 연산 레이어를 포함하는 세 가지 네트워크를 CNN 연산 가속 장치(1500)를 통해 구현하는 경우 CNN 연산 가속 장치(1500)의 논리적 처리량(logical throughput)을 나타낸다.

표 4

Throughput [GOPS]	STC	D-CONV Rate 2	D-CONV Rate 4	D-CONV Rate 8	D-CONV Rate 16	T-CONV	PWC	DSC
ENET (FSP 288x288)	115.2	320	1097	4161	19300	460.8	102.4	-
DTN-Large (KITTI 960x280)	112.4	329	1067	4315	18764	-	112	-
DTN (KITTI 960x280)	-	*116.8	*161.5	*488	*1768	-	112	97.5

*Dilation Rate-DSC

[0128]

[0129] 표 4에서, DTN은 STC 및 D-CONV 레이어에 DSC를 적용하기 때문에, 기존 STC, PWC 및 D-CONV를 사용한 DTN-Large 모델도 전체 5가지 유형의 컨볼루션에 대한 논리적 처리량을 리포트하드로고 평가된다. DTN에는 T-CONV가 포함되지 않으므로, FSP(Freiburg Sitting people) 데이터 세트를 가진 ENET도 평가될 수 있으며, D-CONV 및 T-CONV에 대한 논리적 처리량은 가상의 0 데이터에 대한 연산이 포함될 수 있다. 제안된 CNN 연산 가속 장치(1500)는 확장 계수(DR)가 증가함에 따라 D-CONV의 처리량을 극대화할 수 있다. 요약하면, 표 4는 DSC는 83.4$이고, 다른 유형의 경우 95.8%를 초과하여 모든 유형의 컨볼루션 연산에서 DTN에 대한 높은 프로세싱 엘리먼트(PE) 활용도를 보여준다. DSC가 다른 유형의 컨볼루션에 비해 처리량이 적은 이유는 DTN이 극도로 압축(또는 경량화)되어 있고, DSC 레이어 융합 아키텍처가 PWC의 64개 출력 채널을 병렬로 처리할 수 있기 때문이다. 따라서, CNN 연산 가속 장치(1500) DTN의 네트워크가 깊어질수록 더 높은 처리량을 달성할 수 있다.

[0131] 도 16a는 레이어 별 DSC 연산과 DSC 레이어 융합을 수행하는 경우의 프로세싱 엘리먼트들의 활용도를 나타내는 그래프다.

[0132] 그래프(1600a)는 레이어 별 DSC(layer-by-layer DSC) 연산 및 DSC 레이어 융합(DSC layer fusion) 연산을 위한 제안된 DTN에서의 프로세싱 엘리먼트(PE) 활용도를 비교한다. 그래프(1600a)에서 기존의 레이어 별 DSC 연산은 DWC 동작 특성으로 인해 평균 PE 활용도가 44%를 나타낸다. 이와 달리, 그래프(1600a)에서 제안된 CNN 연산 가속 장치에 의해 지원되는 DSC 레이어 융합 연산은 PE 활용도가 평균 83.4%에 달하므로, 기존 대비 1.89배 향상된다.

[0134] 도 16b는 DTN 연산을 위한 외부 메모리 액세스 비율을 나타낸다.

[0135] 그래프(1600b)는 비교 실시예(예: DTN)에 비해, DSC 레이어 융합 연산을 수행함으로써 외부 메모리로의 액세스 비율이 감소하는 것을 나타낸다. DSC 레이어 융합 연산을 수행하는 제안된 CNN 연산 가속 장치는 DSC 레이어 융합 연산의 병렬 가속을 통해 DWC 및 PWC 간의 중간 특징맵을 감소하여 34.7%의 불필요한 외부 메모리 액세스 비율을 제거할 수 있다.

[0137] 도 16c는 일 실시예에 따른 CNN 연산 가속 장치와 DTN 모델 알고리즘 최적화를 통한 프레임 속도 향상 결과를 나타낸다.

[0138] 그래프(1600c)는 DTN Large 모델, 하드웨어 없이 최적화된 DTN 모델, 하드웨어 SIRU를 포함하고 최적화된 DTN 모델, 및 하드웨어 SIRU를 포함하고 모델에 DSC 레이어 융합 연산을 포함하는 최적화된 DTN 모델 각각의 프레임 속도(Framerate, FPS)를 포함할 수 있다.

[0139] 그래프(1600c)에서 DTN Large 모델에 비하여, 하드웨어 없이 DSC만을 적용하여 최적화한 DTN은 1.75배의 프레임 속도 향상을 나타낼 수 있다. 반면, 그래프(1600c)에서 D-CONV 또는 T-CONV에서의 입력 피처맵 또는 가중치 커널 맵에 포함된 0에 대한 연산을 생략하기 위해 제안된 SIRU를 포함하여 D-CONV의 연산량을 크게 줄인 DTN + SIRU가 DTN에 비해 프레임 속도를 11.16배 향상한다. SIRU 하드웨어 추가에 더하여 DSC 레이어 융합 연산을 포함하는 경우 프레임 속도는 1.61배 더 향상될 수 있다. 결론적으로, 제안한 CNN 연산 가속 장치 및 DTN 모델 구조는 고해상도 주행 장면 데이터세트(driving scenes dataset)로 실시간 의미론적 분할을 시연하는 40.07fps를 달성할 수 있다.

[0140] 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.

[0141] 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.

[0142] 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), 플롭티컬 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.

[0143] 위에서 설명한 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 또는 복수의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

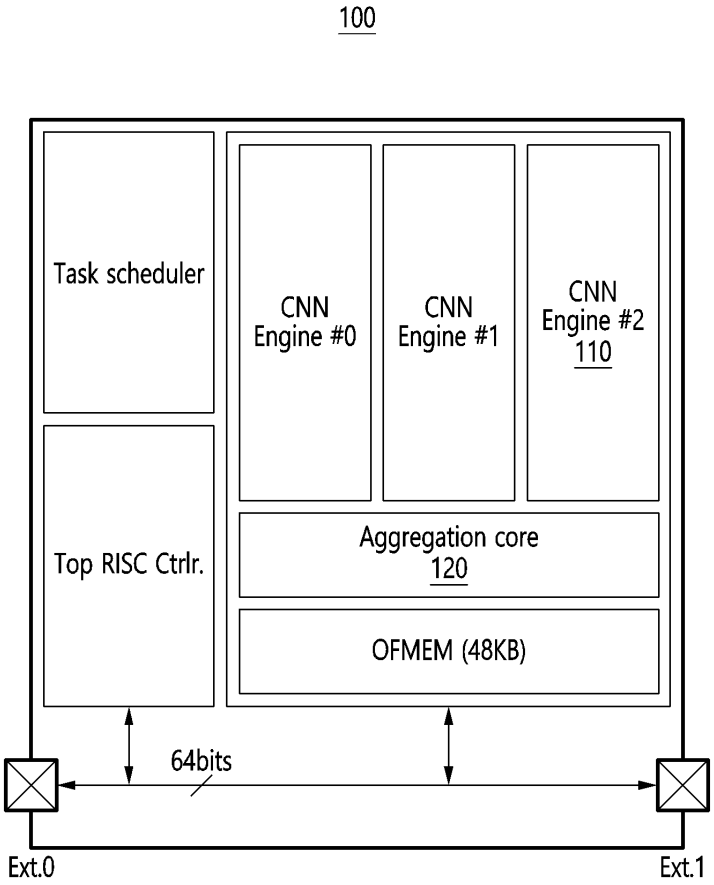
[0144] 본 문서에서, "A 또는 B", "A 및 B 중 적어도 하나", "A 또는 B 중 적어도 하나", "A, B 또는 C", "A, B 및 C 중 적어도 하나", 및 "A, B, 또는 C 중 적어도 하나"와 같은 문구들 각각은 그 문구들 중 해당하는 문구에 함께 나열된 항목들 중 어느 하나, 또는 그들의 모든 가능한 조합을 포함할 수 있다.

[0145] 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.

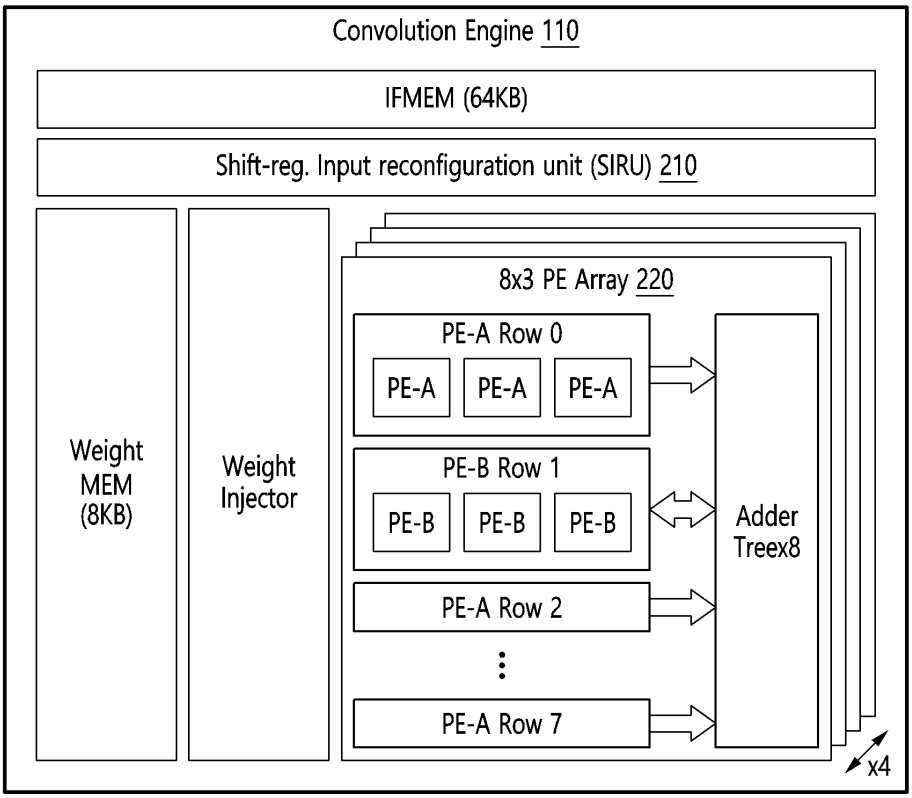
[0146] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

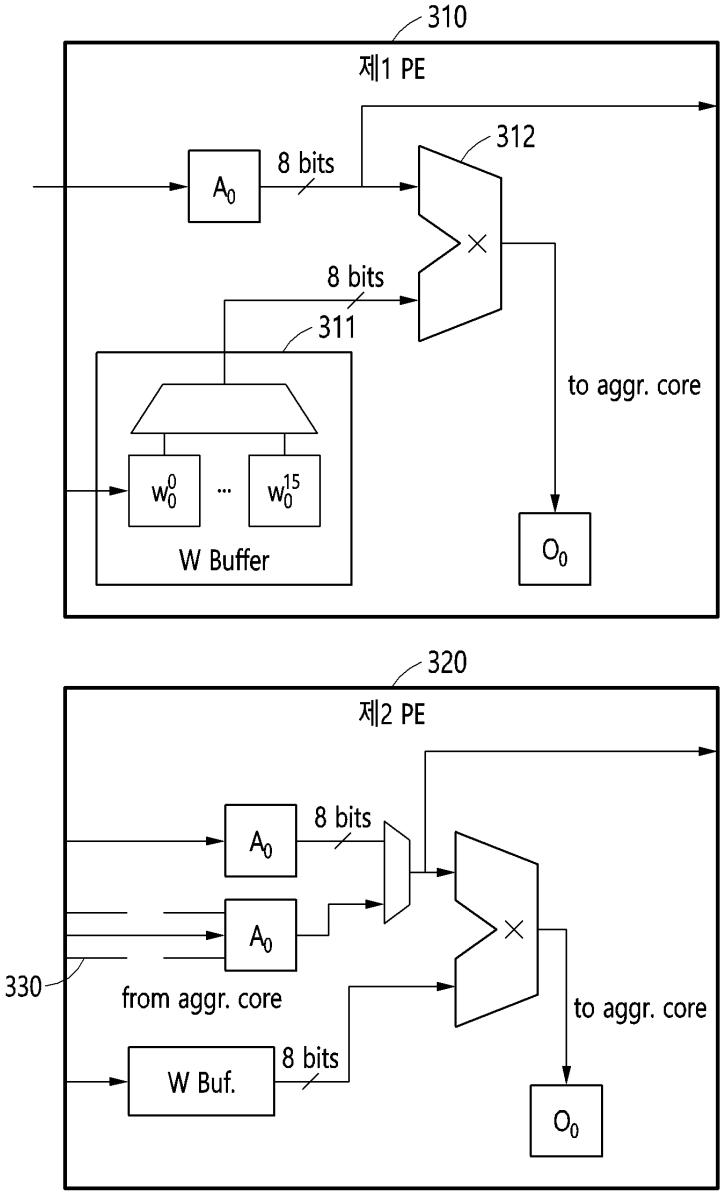
도면1


100
Task scheduler
CNN Engine #0
CNN Engine #1
CNN Engine #2
110
Top RISC Ctrlr.
Aggregation core
120
OFMEM (48KB)
64bits
Ext.0
Ext.1

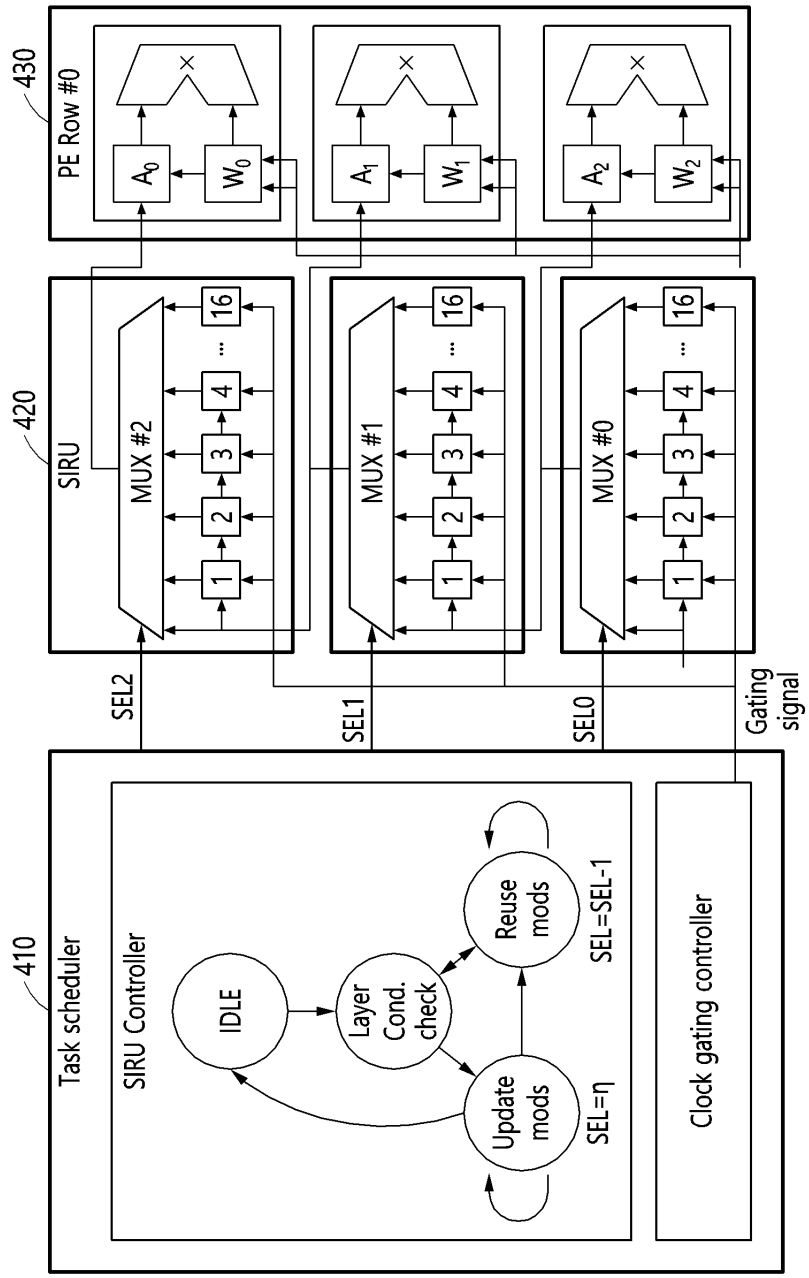
도면2


Convolution Engine 110
IFMEM (64KB)
Shift-reg. Input reconfiguration unit (SIRU) 210
Weight MEM (8KB)
Weight Injector
8x3 PE Array 220
PE-A Row 0
PE-A
PE-A
PE-A
PE-B Row 1
PE-B
PE-B
PE-B
PE-A Row 2
PE-A Row 7
Adder Treex8
x4

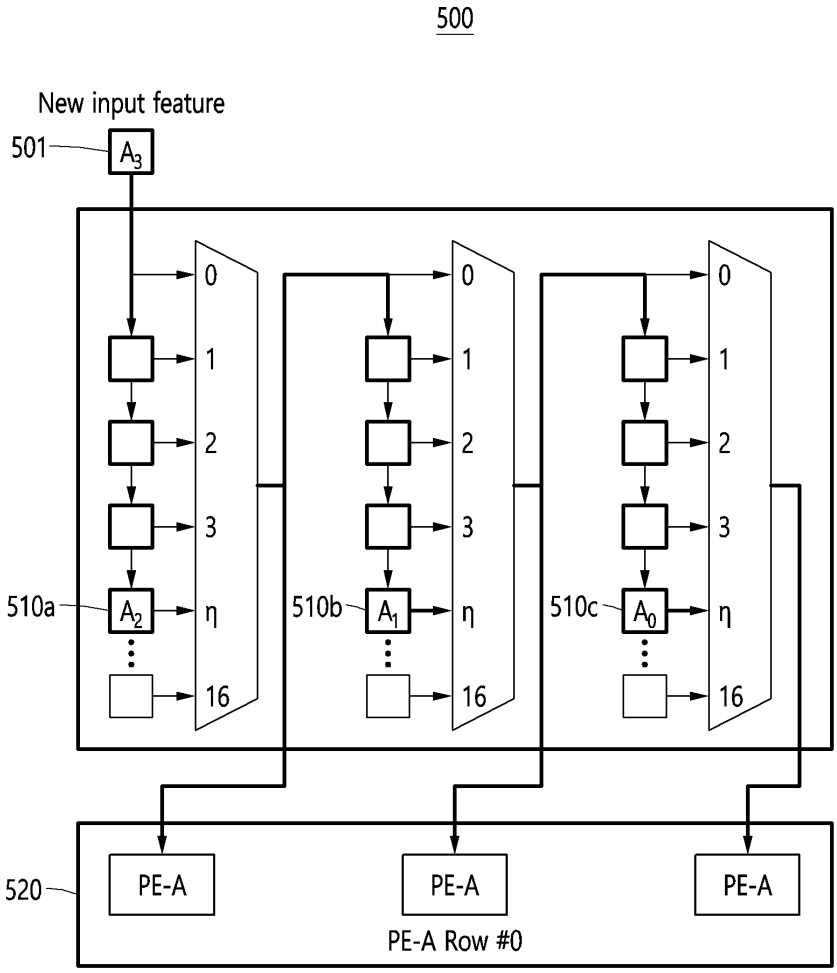
도면3


310
제1 PE
312
8 bits
A0
8 bits
311
to aggr. core
w0 0
...
w0 15
W Buffer
O0
320
제2 PE
8 bits
A0
A0
330
from aggr. core
to aggr. core
W Buf.
8 bits
O0

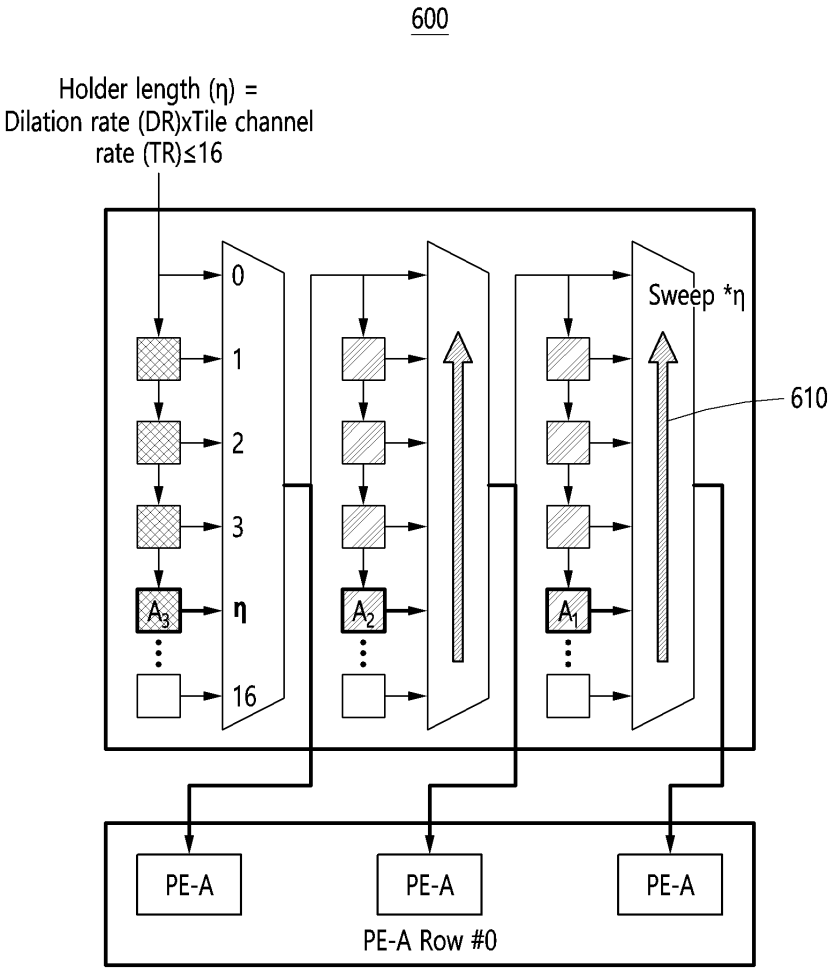
도면4


410
Task scheduler
SIRU Controller
IDLE
Layer Cond. check
Reuse mods
Update mods
SEL=η
SEL=SEL-1
Clock gating controller
420
SIRU
MUX #2
MUX #1
MUX #0
1
2
3
4
...
16
SEL2
SEL1
SEL0
Gating signal
430
PE Row #0
A0
W0
A1
W1
A2
W2
×

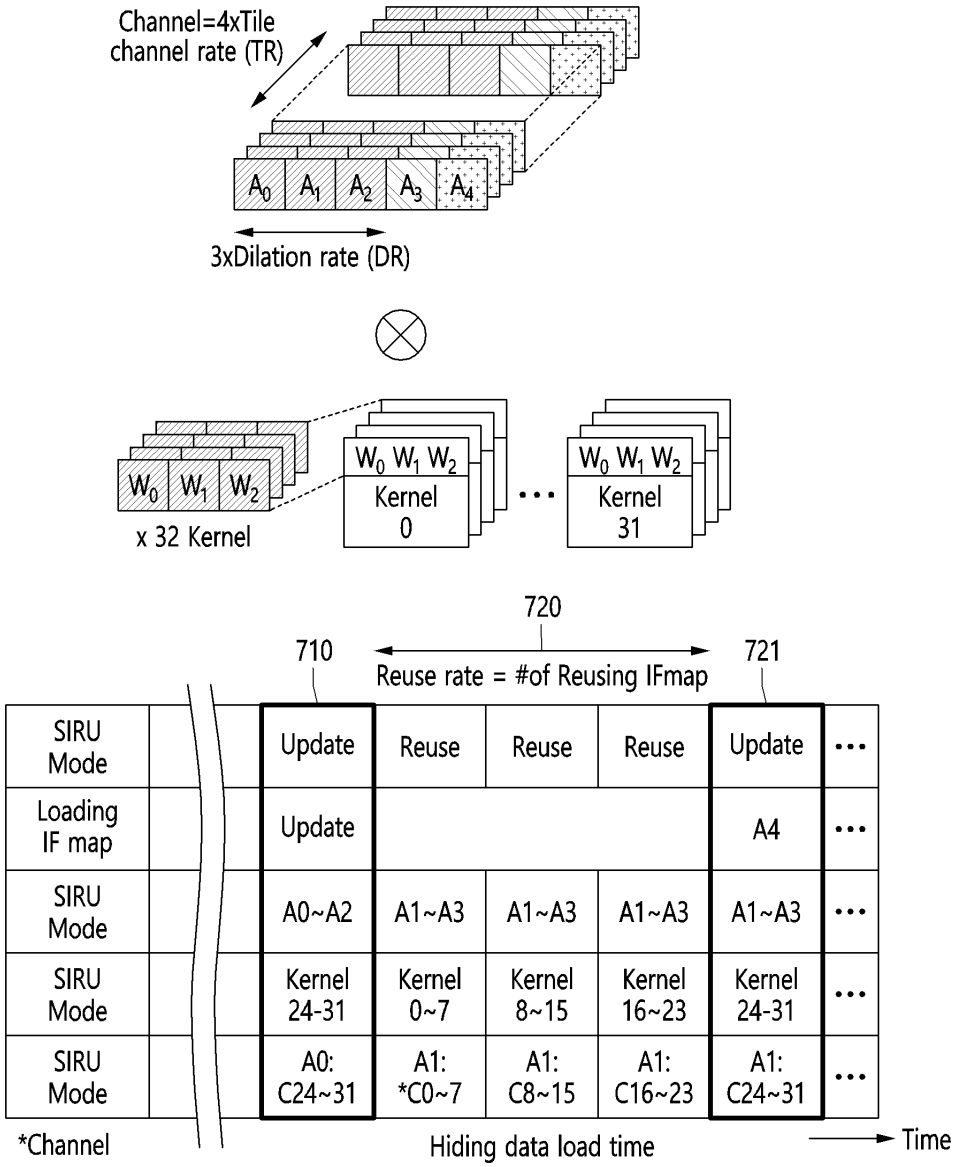
도면5


500
New input feature
501
A3
0
1
2
3
η
16
510a
A2
510b
A1
510c
A0
520
PE-A
PE-A
PE-A
PE-A Row #0

도면6

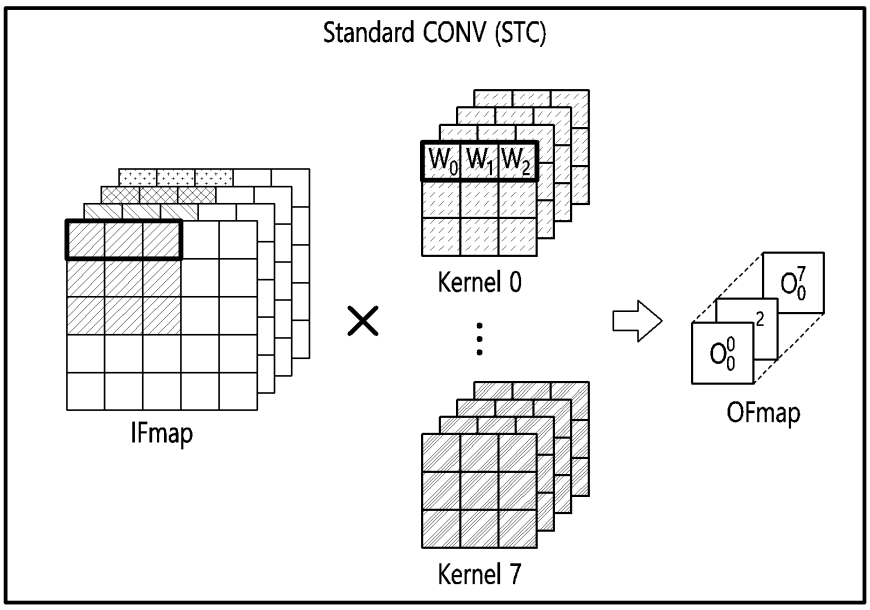

600
Holder length (η) =
Dilation rate (DR)xTile channel
rate (TR)≤16
0
1
2
3
η
16
Sweep *η
610
A3
A2
A1
PE-A
PE-A
PE-A
PE-A Row #0

도면7


Channel=4xTile
channel rate (TR)
A0 A1 A2 A3 A4
3xDilation rate (DR)
W0 W1 W2
x 32 Kernel
W0 W1 W2
Kernel
0
W0 W1 W2
Kernel
31
720
710
721
Reuse rate = #of Reusing IFmap
SIRU Mode
Update
Reuse
Reuse
Reuse
Update
Loading IF map
Update
A4
SIRU Mode
A0~A2
A1~A3
A1~A3
A1~A3
A1~A3
SIRU Mode
Kernel 24-31
Kernel 0~7
Kernel 8~15
Kernel 16~23
Kernel 24-31
SIRU Mode
A0: C24~31
A1: *C0~7
A1: C8~15
A1: C16~23
A1: C24~31
*Channel
Hiding data load time
Time

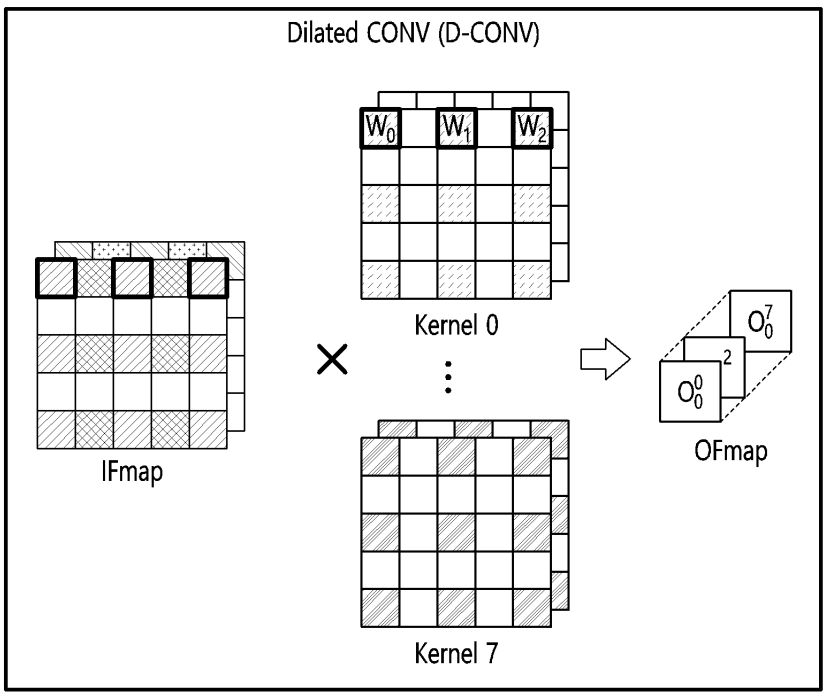
도면8a

800a



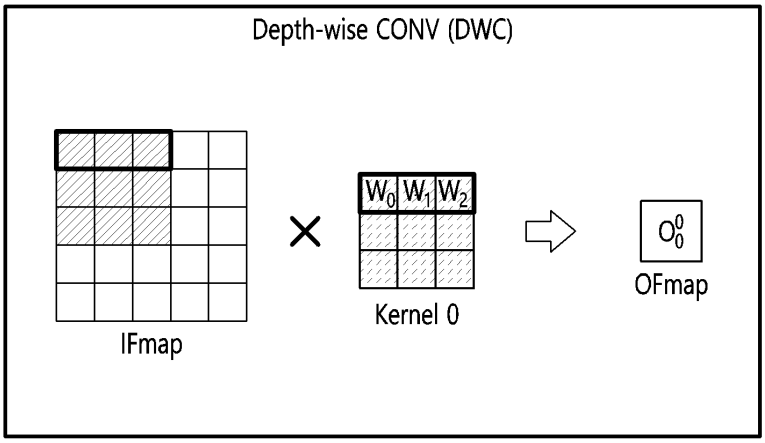
도면8b

800b

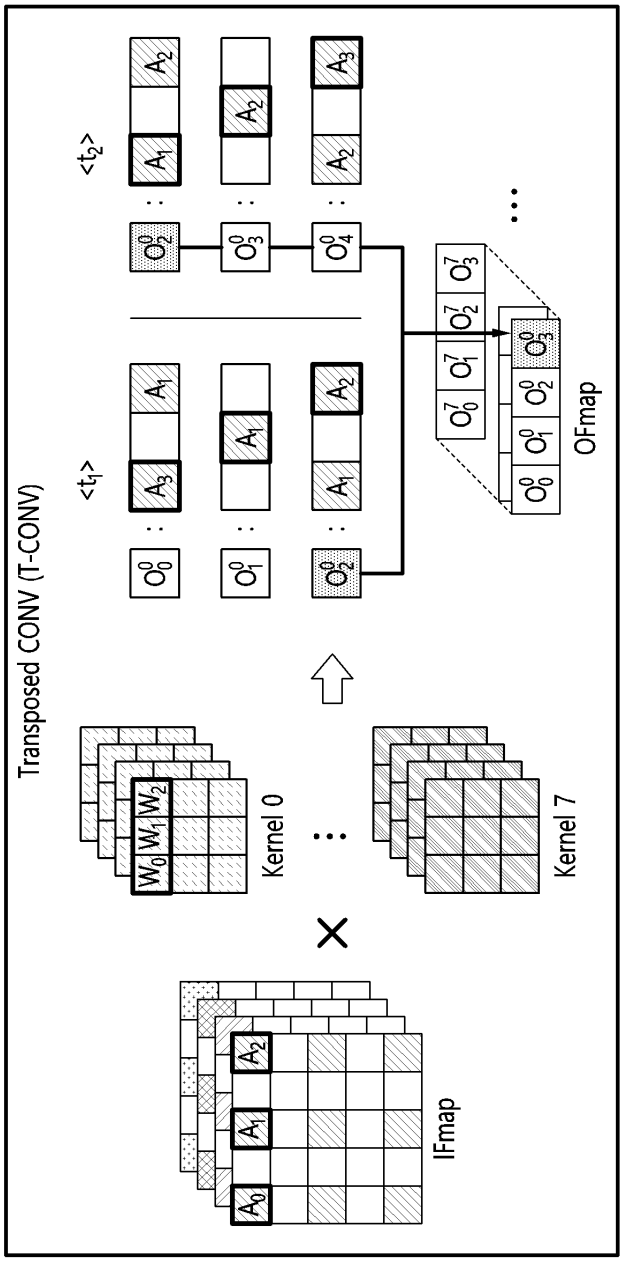


도면8c

800c

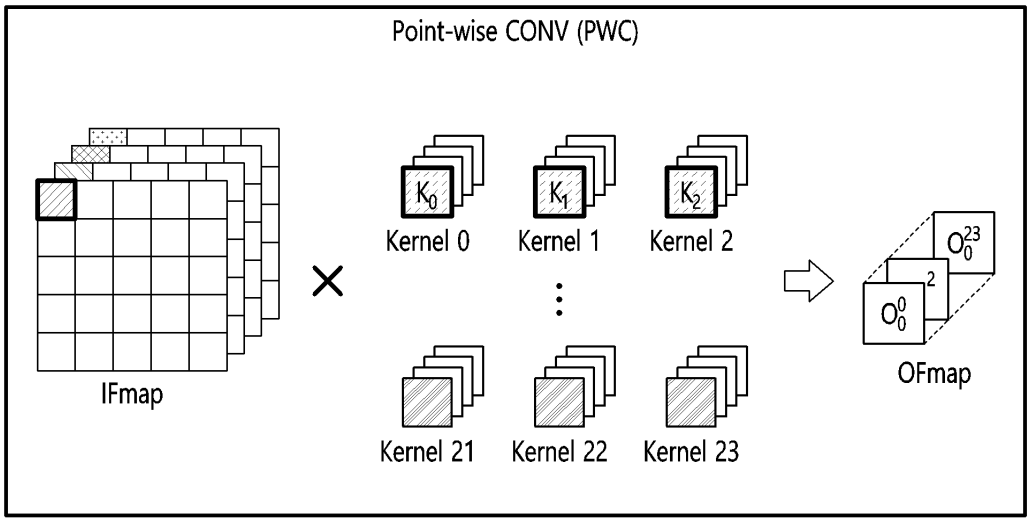


도면8d

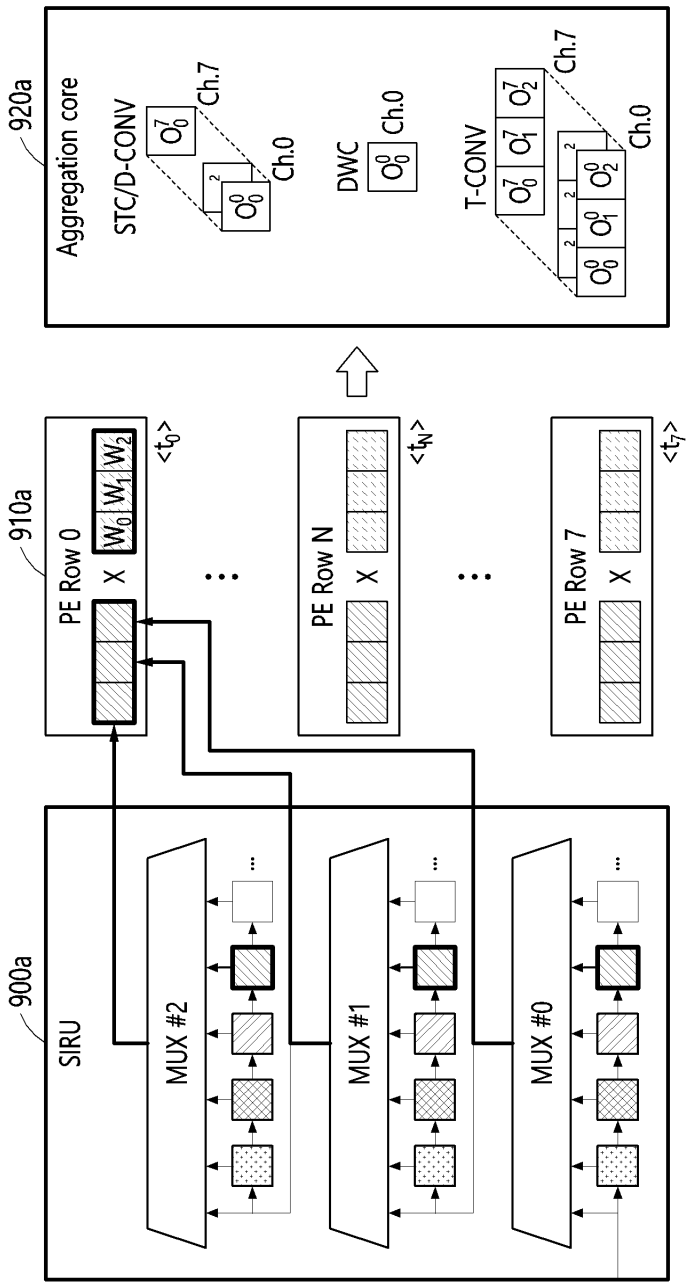

800d
Transposed CONV (T-CONV)
<t1>
<t2>
Kernel 0
Kernel 7
IFmap
OFmap

도면8e

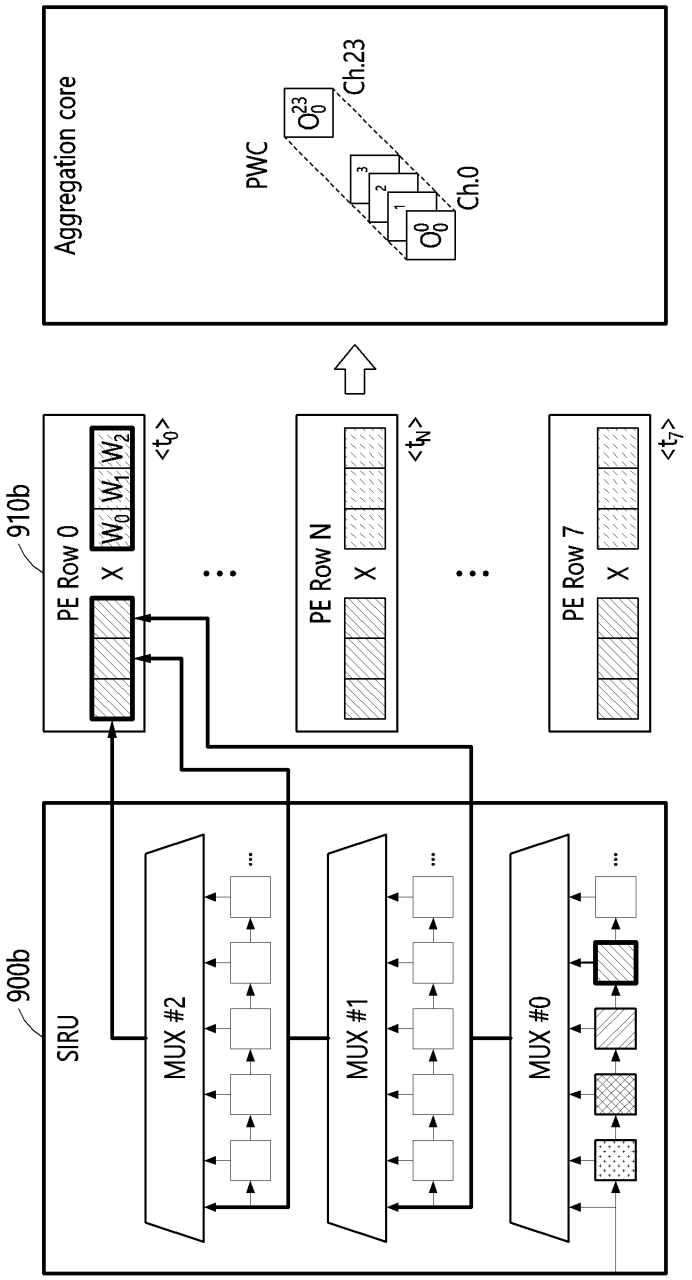
800e


Point-wise CONV (PWC)
IFmap
Kernel 0
Kernel 1
Kernel 2
Kernel 21
Kernel 22
Kernel 23
OFmap

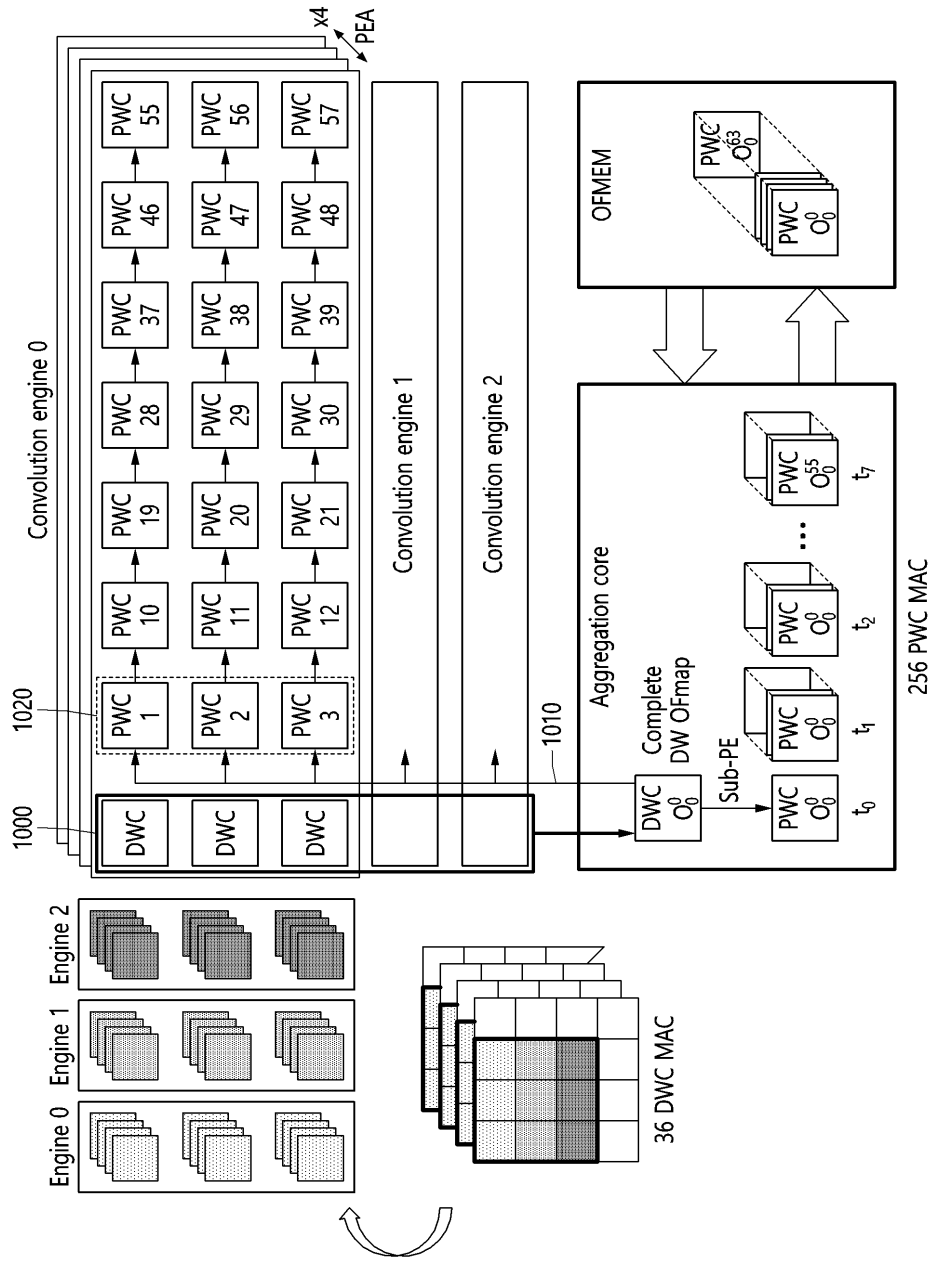
도면9a


920a
Aggregation core
STC/D-CONV
Ch.7
Ch.0
DWC
Ch.0
T-CONV
Ch.7
Ch.0
910a
PE Row 0
PE Row N
PE Row 7
900a
SIRU
MUX #2
MUX #1
MUX #0

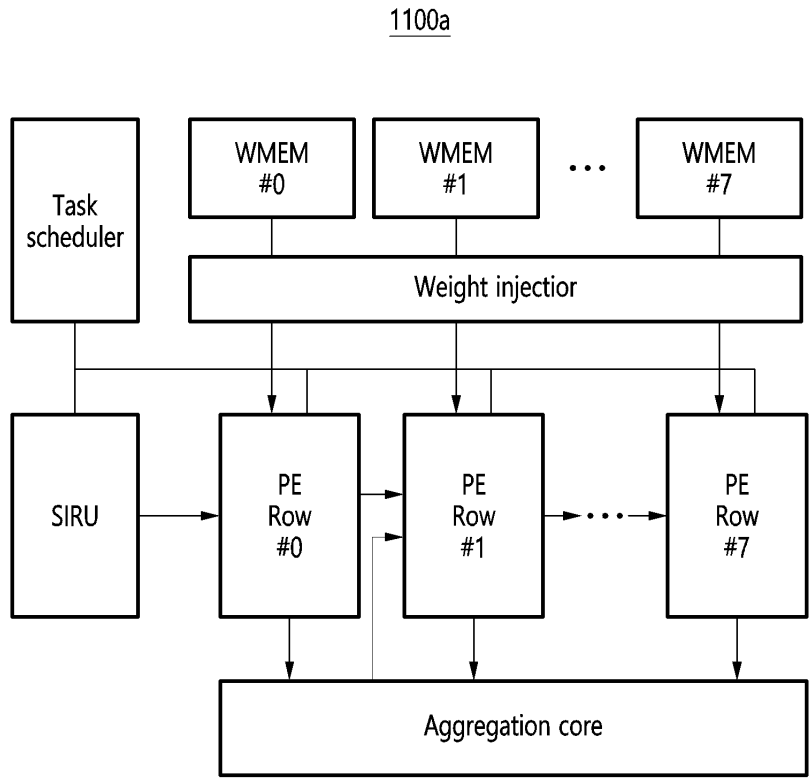
도면9b


Aggregation core
PWC
Ch.23
Ch.0
910b
PE Row 0
PE Row N
PE Row 7
900b
SIRU
MUX #2
MUX #1
MUX #0

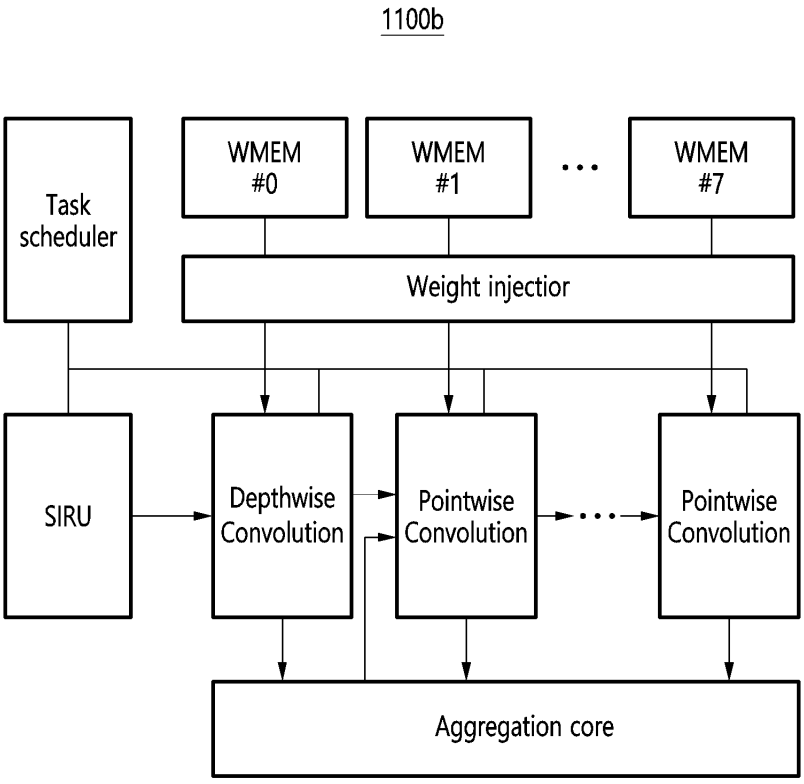
도면10


x4
PEA
Convolution engine 0
Convolution engine 1
Convolution engine 2
1020
1000
1010
DWC
PWC 1
PWC 2
PWC 3
PWC 10
PWC 11
PWC 12
PWC 19
PWC 20
PWC 21
PWC 28
PWC 29
PWC 30
PWC 37
PWC 38
PWC 39
PWC 46
PWC 47
PWC 48
PWC 55
PWC 56
PWC 57
OFMEM
Aggregation core
Complete DW OFmap
Sub-PE
256 PWC MAC
Engine 0
Engine 1
Engine 2
36 DWC MAC

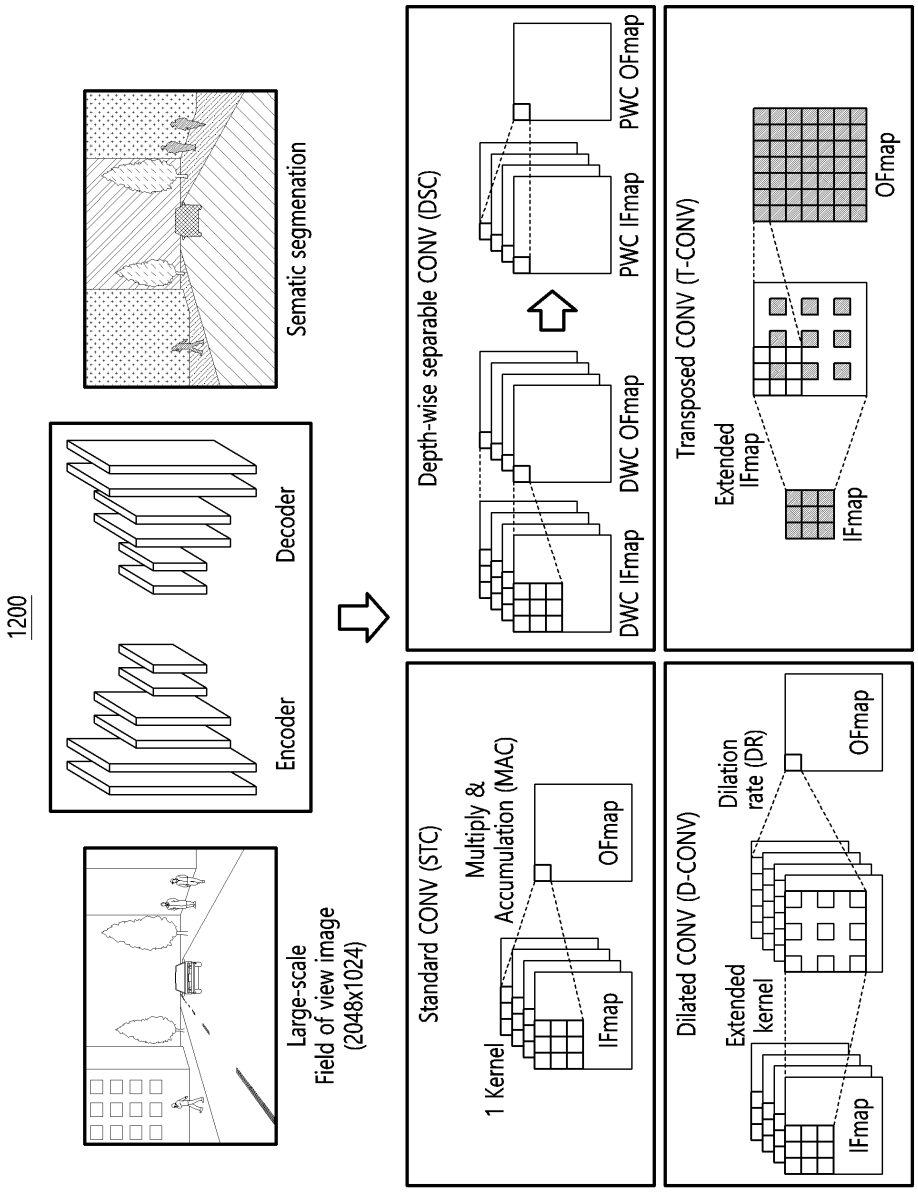
도면11a


1100a
Task scheduler
WMEM #0
WMEM #1
WMEM #7
Weight injectior
SIRU
PE Row #0
PE Row #1
PE Row #7
Aggregation core

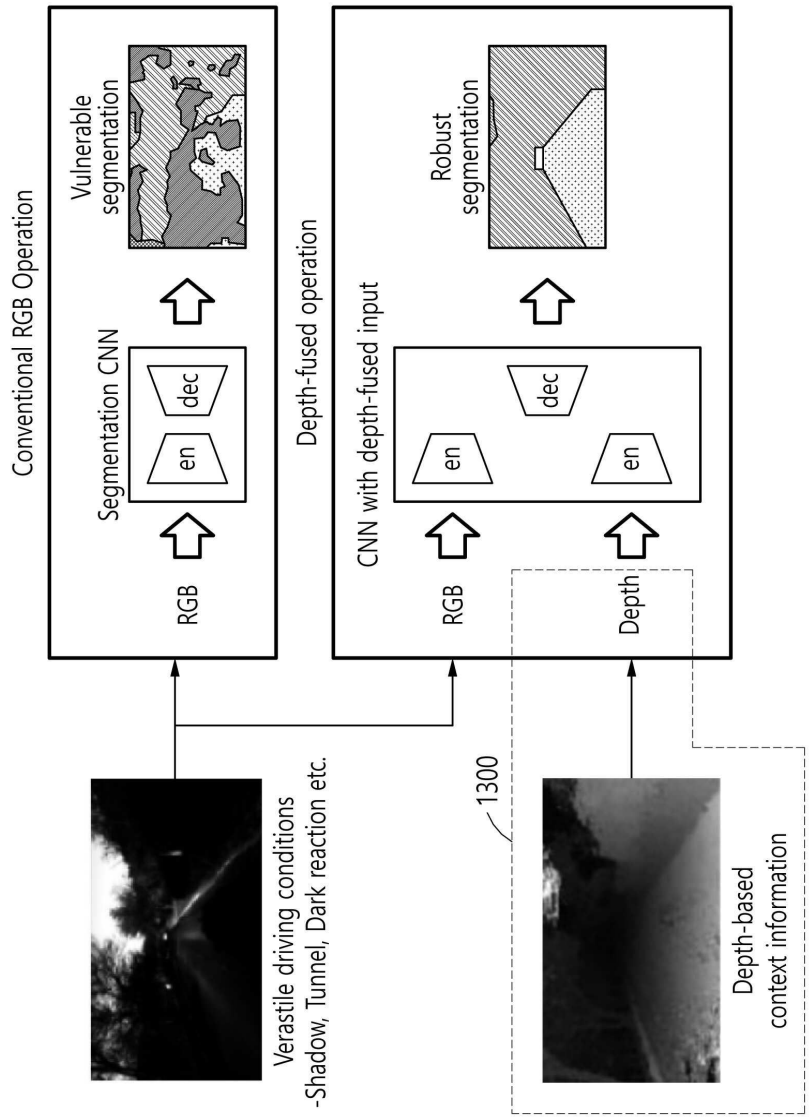
도면11b


1100b
Task scheduler
WMEM #0
WMEM #1
WMEM #7
Weight injectior
SIRU
Depthwise Convolution
Pointwise Convolution
Pointwise Convolution
Aggregation core

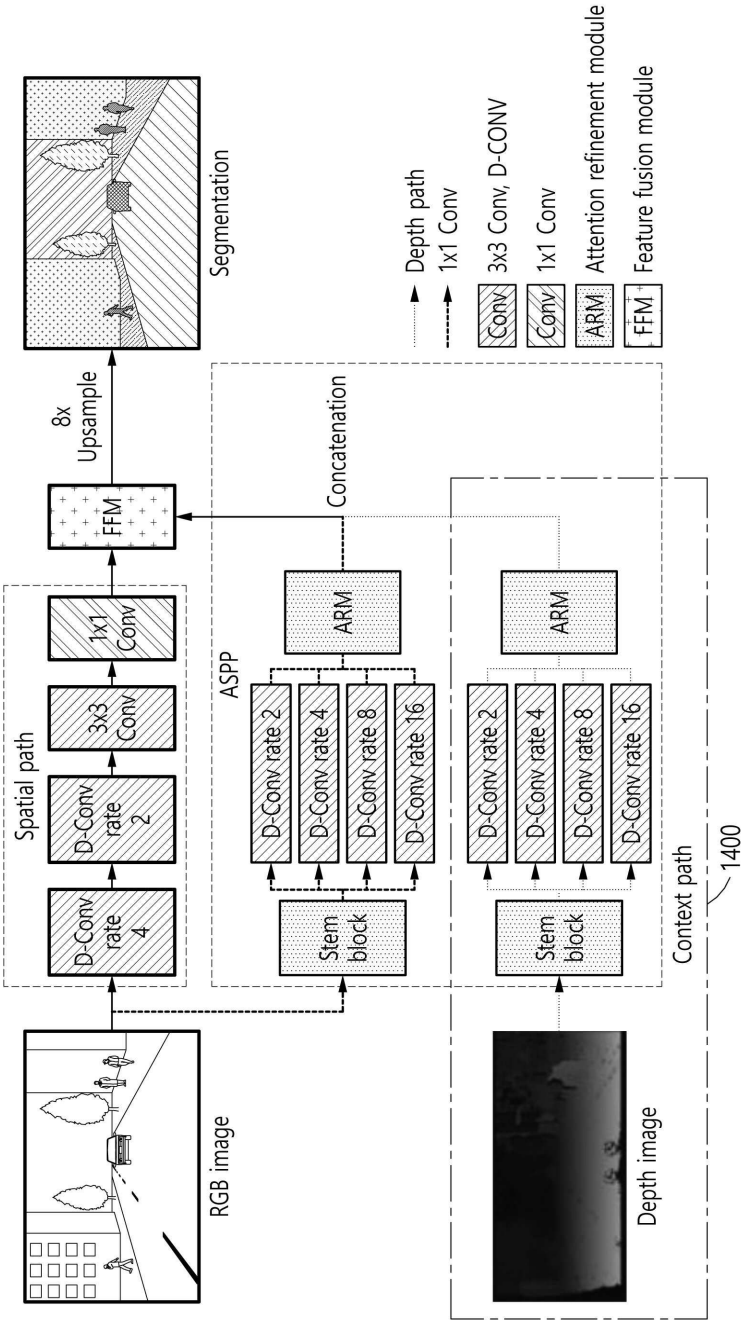
도면12


1200
Large-scale
Field of view image
(2048x1024)
Encoder
Decoder
Sematic segmenation
Standard CONV (STC)
Multiply &
Accumulation (MAC)
1 Kernel
IFmap
OFmap
Depth-wise separable CONV (DSC)
DWC IFmap
DWC OFmap
PWC IFmap
PWC OFmap
Transposed CONV (T-CONV)
Extended
IFmap
IFmap
OFmap
Dilated CONV (D-CONV)
Extended
kernel
Dilation
rate (DR)
IFmap
OFmap

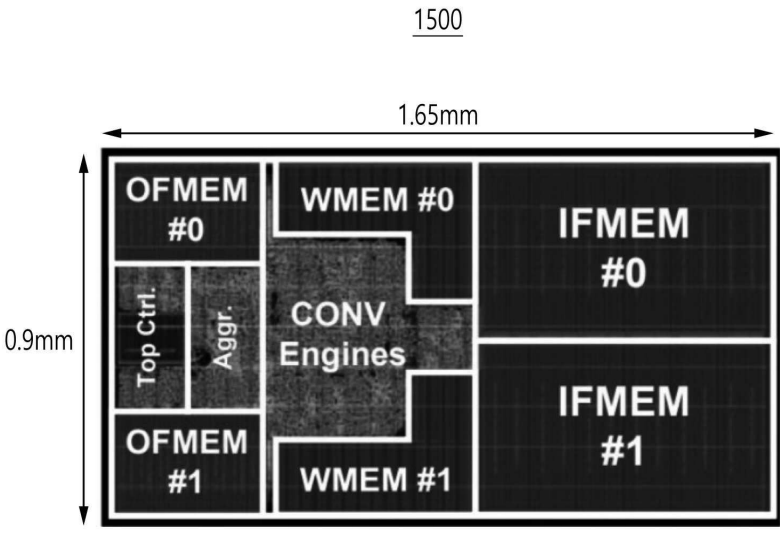
도면13


Conventional RGB Operation
Vulnerable segmentation
Segmentation CNN
en
dec
RGB
Depth-fused operation
Robust segmentation
CNN with depth-fused input
en
dec
en
RGB
Depth
1300
Verastile driving conditions
-Shadow, Tunnel, Dark reaction etc.
Depth-based
context information

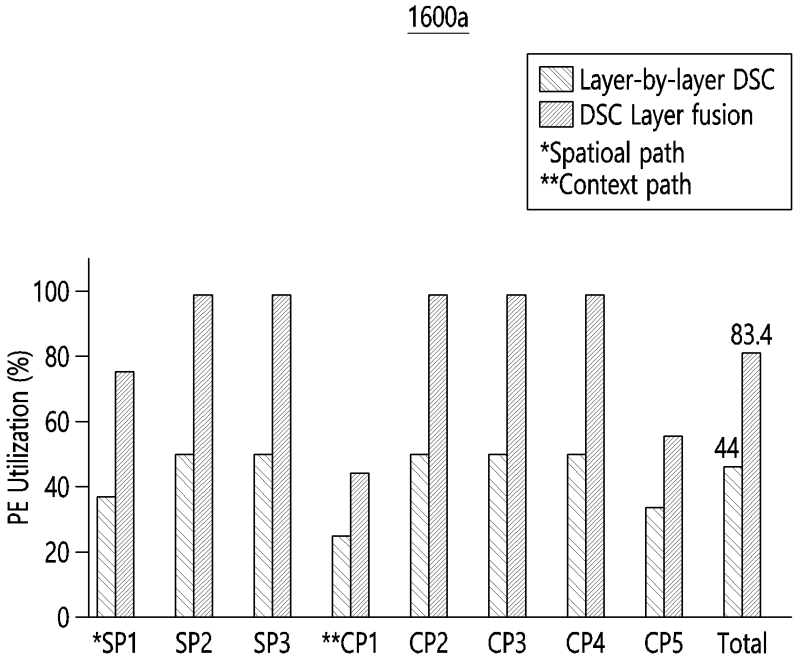
도면14


Segmentation
8x Upsample
FFM
1x1 Conv
3x3 Conv
D-Conv rate 2
D-Conv rate 4
Spatial path
Concatenation
ARM
ASPP
D-Conv rate 2
D-Conv rate 4
D-Conv rate 8
D-Conv rate 16
Stem block
ARM
D-Conv rate 2
D-Conv rate 4
D-Conv rate 8
D-Conv rate 16
Stem block
Context path
1400
RGB image
Depth image
Depth path
1x1 Conv
3x3 Conv, D-CONV
1x1 Conv
Attention refinement module
Feature fusion module
Conv
Conv
ARM
FFM

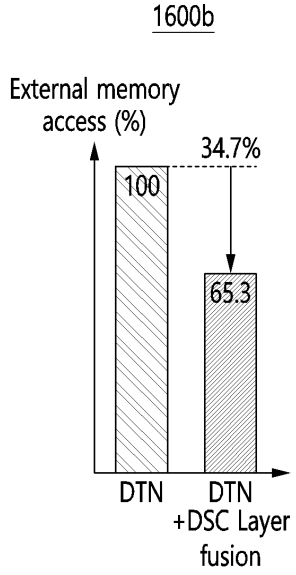
도면15


1500
1.65mm
0.9mm
OFMEM #0
WMEM #0
IFMEM #0
Top Ctrl.
Aggr.
CONV Engines
IFMEM #1
OFMEM #1
WMEM #1

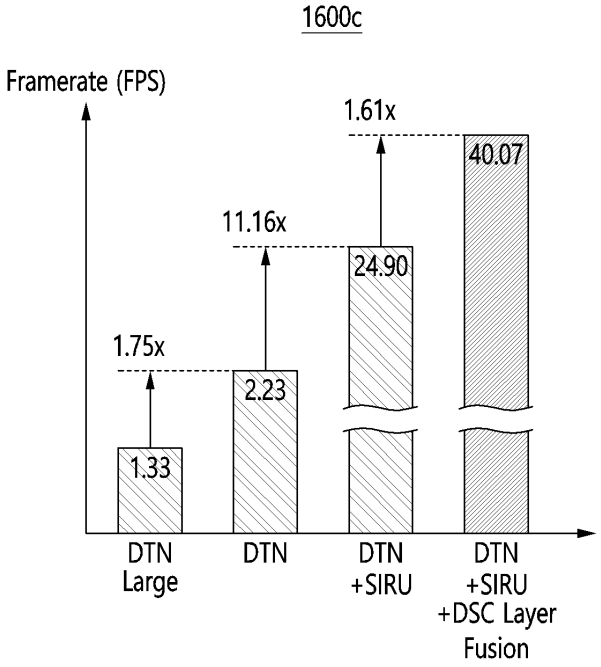
도면16a


1600a
Layer-by-layer DSC
DSC Layer fusion
*Spatioal path
**Context path
PE Utilization (%)
100
80
60
40
20
0
83.4
44
*SP1
SP2
SP3
**CP1
CP2
CP3
CP4
CP5
Total

도면16b


1600b
External memory access (%)
34.7%
100
65.3
DTN
DTN +DSC Layer fusion

도면16c


1600c
Framerate (FPS)
1.61x
40.07
11.16x
24.90
1.75x
2.23
1.33
DTN Large
DTN
DTN +SIRU
DTN +SIRU +DSC Layer Fusion