(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0145690
(43) 공개일자 2022년10월31일

(51) 국제특허분류(Int. Cl.)
*G06F 7/53* (2006.01) *G06F 7/501* (2006.01)
*G06F 7/544* (2017.01)
(52) CPC특허분류
*G06F 7/5312* (2013.01)
*G06F 7/501* (2021.08)
(21) 출원번호 10-2021-0052589
(22) 출원일자 2021년04월22일
심사청구일자 2021년04월22일

(71) 출원인
**연세대학교 산학협력단**
서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)
(72) 발명자
**정성욱**
서울특별시 서대문구 연세로 50, 제3공학관 C513 (신촌동, 연세대학교)
**정영석**
서울특별시 서대문구 연세로 50, 제3공학관 C421 (신촌동, 연세대학교)
*(뒷면에 계속)*
(74) 대리인
**특허법인우인**

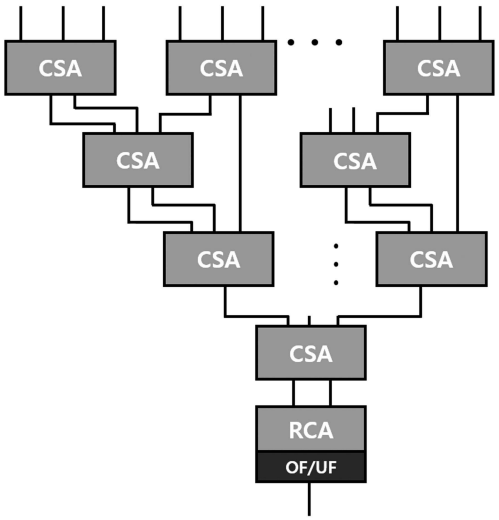
전체 청구항 수 : 총 5 항

(54) 발명의 명칭 **CNN 기반 AI 모델의 누산 방법 및 장치**

## *(57) 요 약*

본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 방법 및 장치는, CNN(convolution neural network) 기반 AI(artificial intelligence) 모델의 누산 동작을 캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)를 통해 수행함으로써, 멀티-비트(multi-bit) MAC(multiply-accumulate)의 연산 속도를 향상시킬 수 있다.

*대 표 도* - 도6



(52) CPC특허분류
*G06F 7/5443* (2013.01)
*G06N 3/04* (2013.01)
*G06N 3/063* (2013.01)
(72) 발명자
**안홍근**
서울특별시 서대문구 연세로 50, 제3공학관 C421(신촌동, 연세대학교)
**이수민**
서울특별시 서대문구 연세로 50, 제3공학관 C421(신촌동, 연세대학교)
**주성환**
서울특별시 서대문구 연세로 50, 제3공학관 C421(신촌동, 연세대학교)

# 명 세 서

## 청구범위

### 청구항 1

캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)에 입력값을 입력하는 단계;

상기 누산기(ACC)를 통해 상기 입력값으로부터 출력값을 획득하는 단계; 및

상기 누산기(ACC)를 통해 획득한 상기 출력값을 출력하는 단계;

를 포함하는 CNN 기반 AI 모델의 누산 방법.

### 청구항 2

제1항에서,

상기 누산기(ACC)는,

서로 계층적으로 연결된 복수의 캐리 세이브 가산기(CSA); 및

캐리 세이브 가산기(CSA)와 연결된 하나의 리플 캐리 가산기(ripple carry adder, RCA);

를 포함하는,

CNN 기반 AI 모델의 누산 방법.

### 청구항 3

제2항에서,

상기 출력값 획득 단계는,

상기 복수의 캐리 세이브 가산기(CSA)를 통해 상기 입력값으로부터 중간값을 획득하는 단계; 및

상기 하나의 리플 캐리 가산기(RCA)를 통해 상기 중간값으로부터 상기 출력값을 획득하는 단계;

를 포함하는,

CNN 기반 AI 모델의 누산 방법.

### 청구항 4

제2항에서,

상기 누산기(ACC)는,

상기 하나의 리플 캐리 가산기(RCA)에서만 오버플로우(overflow)와 언더플로우(underflow)의 센싱을 수행하는,

CNN 기반 AI 모델의 누산 방법.

### 청구항 5

캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)를 통해 누산하는 누산 장치로서,

상기 누산기(ACC)를 통해 누산하기 위한 하나 이상의 프로그램을 저장하는 메모리; 및

상기 메모리에 저장된 상기 하나 이상의 프로그램에 따라 상기 누산기(ACC)를 통해 누산하기 위한 동작을 수행하는 하나 이상의 프로세서;

를 포함하고,

상기 프로세서는,

상기 누산기(ACC)에 입력값을 입력하고, 상기 누산기(ACC)를 통해 상기 입력값으로부터 출력값을 획득하며, 상기 누산기(ACC)를 통해 획득한 상기 출력값을 출력하는,

CNN 기반 AI 모델의 누산 장치.

## 발명의 설명

### 기 술 분 야

[0001] 본 발명은 CNN 기반 AI 모델의 누산 방법 및 장치에 관한 것으로서, 더욱 상세하게는 CNN(convolution neural network) 기반 AI(artificial intelligence) 모델에서 누산하는, 방법 및 장치에 관한 것이다.

### 배 경 기 술

[0002] 최근 컴퓨터 비전(computer vision) 분야에서 이미지 슈퍼 레졸루션(super-resolution, SR)과 딥러닝(deep learning)을 접목시킨 기술들의 중요성이 커지는 추세이다.

[0003] 도 1은 종래의 SRCNN(super-resolution convolution neural network) 구조를 설명하기 위한 도면이다.

[0004] 도 1에서 확인할 수 있듯이, LR 이미지 -> HR 이미지로 업 스케일링(up scaling)하기 위해 패치 추출(patch extraction) 과정, 비선형 매핑(nonlinear mapping) 과정, 재구축(reconstruction) 과정이 필요하다. 각각의 과정에서 컨볼루션 레이어(convolution layer)를 구현하기 위해 많은 수의 MAC(multiply-accumulate) 연산을 빠르게 처리해야 한다. 즉, MAC(multiply-accumulate) 연산 속도가 슈퍼 레졸루션(SR) 시스템 성능에 큰 부분을 차지한다. 또한, 각각의 필터(filter)들은 부호가 있는 값(signed value)이므로 MAC(multiply-accumulate) 연산에서 이를 고려해야 한다.

[0005] 도 2는 종래의 컨볼루션(convolution) 연산 과정을 설명하기 위한 도면이고, 도 3은 도 2에 도시한 종래의 누산기(accumulation, ACC)의 구조를 설명하기 위한 도면이다.

[0006] 도 2 및 도 3을 참조하면, 종래의 CNN 기반 AI(artificial intelligence) 구조들의 경우, 1 클럭(clk) 안에 MUL 연산, ACC 연산과 부수적인 처리 과정이 완료되어야 한다. 그리고, 종래의 CNN 기반 AI 구조들은 가중치(weight)와 입력값(activation)의 MAC(multiply-accumulate) 연산을 할 때, ACC(accumulation) 과정에서 리플 캐리 가산기(ripple carry adder, RCA) 기반의 가산기 트리(adder tree) 구조를 사용한다.

[0007] 위와 같은 리플 캐리 가산기(RCA) 기반의 가산기 트리 구조를 사용하는 종래의 구조는 각 스테이지(stage)마다 오버플로우(overflow, OF)/언더플로우(underflow, UF) 센싱 동작을 수행하여야 한다.

[0008] 즉, 종래의 구조는 이전 스테이지(stage)의 가산기 연산이 끝나야 다음 스테이지(stage)의 연산이 시작할 수 있다. 이러한 구조는 실시간 CNN 기반 슈퍼 레졸루션(SR)과 같이 고속 연산 처리가 필요한 시스템을 구현하는 데에 어려움이 있다. 이에 따라, 종래의 구조는 연산 속도가 느린 문제가 있다.

## 발명의 내용

### 해결하려는 과제

[0009] 본 발명이 이루고자 하는 목적은, CNN(convolution neural network) 기반 AI(artificial intelligence) 모델의 누산 동작을 캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)를 통해 수행하는, CNN 기반 AI 모델의 누산 방법 및 장치를 제공하는 데 있다.

[0010] 본 발명의 명시되지 않은 또 다른 목적들은 하기의 상세한 설명 및 그 효과로부터 용이하게 추론할 수 있는 범위 내에서 추가적으로 고려될 수 있다.

### 과제의 해결 수단

[0011] 상기의 목적을 달성하기 위한 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 방법은, 캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)에 입력값을 입력하는 단계; 상기 누산기(ACC)를 통해 상기 입력값으로부터 출력값을 획득하는 단계; 및 상기 누산기(ACC)를 통해 획득한 상기 출력값을

출력하는 단계;를 포함한다.

[0012] 여기서, 상기 누산기(ACC)는, 서로 계층적으로 연결된 복수의 캐리 세이브 가산기(CSA); 및 캐리 세이브 가산기(CSA)와 연결된 하나의 리플 캐리 가산기(ripple carry adder, RCA);를 포함할 수 있다.

[0013] 여기서, 상기 출력값 획득 단계는, 상기 복수의 캐리 세이브 가산기(CSA)를 통해 상기 입력값으로부터 중간값을 획득하는 단계; 및 상기 하나의 리플 캐리 가산기(RCA)를 통해 상기 중간값으로부터 상기 출력값을 획득하는 단계;를 포함할 수 있다.

[0014] 여기서, 상기 누산기(ACC)는, 상기 하나의 리플 캐리 가산기(RCA)에서만 오버플로우(overflow)와 언더플로우(underflow)의 센싱을 수행할 수 있다.

[0016] 상기의 목적을 달성하기 위한 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 장치는, 캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)를 통해 누산하는 누산 장치로서, 상기 누산기(ACC)를 통해 누산하기 위한 하나 이상의 프로그램을 저장하는 메모리; 및 상기 메모리에 저장된 상기 하나 이상의 프로그램에 따라 상기 누산기(ACC)를 통해 누산하기 위한 동작을 수행하는 하나 이상의 프로세서;를 포함하고, 상기 프로세서는, 상기 누산기(ACC)에 입력값을 입력하고, 상기 누산기(ACC)를 통해 상기 입력값으로부터 출력값을 획득하며, 상기 누산기(ACC)를 통해 획득한 상기 출력값을 출력한다.

## *발명의 효과*

[0017] 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 방법 및 장치에 의하면, CNN(convolution neural network) 기반 AI(artificial intelligence) 모델의 누산 동작을 캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)를 통해 수행함으로써, 멀티-비트(multi-bit) MAC(multiply-accumulate)의 연산 속도를 향상시킬 수 있다.

[0018] 본 발명의 효과들은 이상에서 언급한 효과들로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

## *도면의 간단한 설명*

[0019] 도 1은 종래의 SRCNN(super-resolution convolution neural network) 구조를 설명하기 위한 도면이다.

도 2는 종래의 컨볼루션(convolution) 연산 과정을 설명하기 위한 도면이다.

도 3은 도 2에 도시한 종래의 누산기(accumulation, ACC)의 구조를 설명하기 위한 도면이다.

도 4는 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 장치를 설명하기 위한 블록도이다.

도 5는 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 방법을 설명하기 흐름도이다.

도 6은 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 구조를 설명하기 위한 흐름도이다.

도 7은 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 성능을 설명하기 위한 도면으로, 도 7의 (a)는 종래의 누산기(ACC)의 구조 일례를 나타내고, 도 7의 (b)는 본 발명에 따른 누산기(ACC)의 구조 일례를 나타낸다.

도 8은 도 7의 (b)에 도시한 본 발명에 따른 누산기(ACC)의 도트 다이어그램(dot diagram)을 나타낸다.

도 9는 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 성능을 설명하기 위한 도면으로, 도 9의 (a)는 종래의 누산기(ACC)의 부호 확장(sign extension) 구조를 나타내고, 도 9의 (b)는 본 발명에 따른 누산기(ACC)의 부호 확장(sign extension) 구조를 나타낸다.

도 10은 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 성능을 설명하기 위한 도면으로, 도 10의 (a)는 도 7의 (a)에 도시한 종래의 누산기(ACC)의 시뮬레이션 결과를 나타내고, 도 10의 (b)는 도 7의 (b)에 도시한 본 발명에 따른 누산기(ACC)의 시뮬레이션 결과를 나타낸다.

## *발명을 실시하기 위한 구체적인 내용*

[0020] 이하, 첨부된 도면을 참조하여 본 발명의 실시예를 상세히 설명한다. 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 그

러나, 본 발명은 이하에서 게시되는 실시예들에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시예들은 본 발명의 게시가 완전하도록 하고, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 발명은 청구항의 범주에 의해 정의될 뿐이다. 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성 요소를 지칭한다.

[0021] 다른 정의가 없다면, 본 명세서에서 사용되는 모든 용어(기술 및 과학적 용어를 포함)는 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 공통적으로 이해될 수 있는 의미로 사용될 수 있을 것이다. 또한, 일반적으로 사용되는 사전에 정의되어 있는 용어들은 명백하게 특별히 정의되어 있지 않는 한 이상적으로 또는 과도하게 해석되지 않는다.

[0022] 본 명세서에서 "제1", "제2" 등의 용어는 하나의 구성 요소를 다른 구성 요소로부터 구별하기 위한 것으로, 이들 용어들에 의해 권리범위가 한정되어서는 아니 된다. 예컨대, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고, 유사하게 제2 구성 요소도 제1 구성 요소로 명명될 수 있다.

[0023] 본 명세서에서 각 단계들에 있어 식별부호(예컨대, a, b, c 등)는 설명의 편의를 위하여 사용되는 것으로 식별부호는 각 단계들의 순서를 설명하는 것이 아니며, 각 단계들은 문맥상 명백하게 특정 순서를 기재하지 않는 이상 명기된 순서와 다르게 일어날 수 있다. 즉, 각 단계들은 명기된 순서와 동일하게 일어날 수도 있고 실질적으로 동시에 수행될 수도 있으며 반대의 순서대로 수행될 수도 있다.

[0024] 본 명세서에서, "가진다", "가질 수 있다", "포함한다" 또는 "포함할 수 있다" 등의 표현은 해당 특징(예컨대, 수치, 기능, 동작, 또는 부품 등의 구성 요소)의 존재를 가리키며, 추가적인 특징의 존재를 배제하지 않는다.

[0027] 이하에서 첨부한 도면을 참조하여 본 발명에 따른 CNN 기반 AI 모델의 누산 방법 및 장치의 바람직한 실시예에 대해 상세하게 설명한다.

[0029] 먼저, 도 4를 참조하여 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 장치에 대하여 설명한다.

[0030] 도 4는 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 장치를 설명하기 위한 블록도이다.

[0031] 도 4를 참조하면, 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 장치(이하 '누산 장치'라 한다)(100)는 CNN(convolution neural network) 기반 AI(artificial intelligence) 모델의 누산 동작을 캐리 세이브 가산기(carry save adder, CSA) 기반의 누산기(accumulation, ACC)를 통해 수행할 수 있다.

[0032] 즉, CNN 기반 알고리즘에서는 각각의 레이어(layer)를 지날 때마다, 수많은 가중치(weight)와 입력값(activation)의 MAC(multiply-accumulate) 연산을 수행해야 한다. 따라서, MAC(multiply-accumulate) 연산 속도는 시스템 성능에 직접적인 영향을 미치게 된다. 종래의 리플 캐리 가산기(ripple carry adder, RCA)를 통한 가산기 트리(adder tree) 구조의 느린 연산 속도는 시스템 성능 저하의 원인이 되고 있다.

[0033] - 가산기 트리 연산 구조는 스테이지(stage)마다 오버플로우(overflow, OF)/언더플로우(underflow, UF) 센싱 동작이 필요함.

[0034] - 순차적인(sequential) 연산이므로 캐리(carry)를 기다려야 함.

[0035] 이와 같은 문제는 본 발명에 따른 캐리 세이브 가산기(CSA) 기반의 누산기(ACC) 구조를 통해 개선할 수 있다.

[0037] 이를 위해, 누산 장치(100)는 하나 이상의 프로세서(110), 컴퓨터 판독 가능한 저장 매체(130) 및 통신 버스(150)를 포함할 수 있다.

[0038] 프로세서(110)는 누산 장치(100)가 동작하도록 제어할 수 있다. 예컨대, 프로세서(110)는 컴퓨터 판독 가능한 저장 매체(130)에 저장된 하나 이상의 프로그램(131)을 실행할 수 있다. 하나 이상의 프로그램(131)은 하나 이상의 컴퓨터 실행 가능 명령어를 포함할 수 있으며, 컴퓨터 실행 가능 명령어는 프로세서(110)에 의해 실행되는 경우 누산 장치(100)로 하여금 누산기(ACC)를 통해 누산하기 위한 동작을 수행하도록 구성될 수 있다.

[0039] 컴퓨터 판독 가능한 저장 매체(130)는 누산기(ACC)를 통해 누산하기 위한 컴퓨터 실행 가능 명령어 내지 프로그

램 코드, 프로그램 데이터 및/또는 다른 적합한 형태의 정보를 저장하도록 구성된다. 컴퓨터 판독 가능한 저장 매체(130)에 저장된 프로그램(131)은 프로세서(110)에 의해 실행 가능한 명령어의 집합을 포함한다. 일 실시예에서, 컴퓨터 판독 가능한 저장 매체(130)는 메모리(랜덤 액세스 메모리와 같은 휘발성 메모리, 비휘발성 메모리, 또는 이들의 적절한 조합), 하나 이상의 자기 디스크 저장 디바이스들, 광학 디스크 저장 디바이스들, 플래시 메모리 디바이스들, 그 밖에 누산 장치(100)에 의해 액세스되고 원하는 정보를 저장할 수 있는 다른 형태의 저장 매체, 또는 이들의 적합한 조합일 수 있다.

[0040] 통신 버스(150)는 프로세서(110), 컴퓨터 판독 가능한 저장 매체(130)를 포함하여 누산 장치(100)의 다른 다양한 컴포넌트들을 상호 연결한다.

[0041] 누산 장치(100)는 또한 하나 이상의 입출력 장치를 위한 인터페이스를 제공하는 하나 이상의 입출력 인터페이스(170) 및 하나 이상의 통신 인터페이스(190)를 포함할 수 있다. 입출력 인터페이스(170) 및 통신 인터페이스(190)는 통신 버스(150)에 연결된다. 입출력 장치(도시하지 않음)는 입출력 인터페이스(170)를 통해 누산 장치(100)의 다른 컴포넌트들에 연결될 수 있다.

[0044] 그러면, 도 5 및 도 6을 참조하여 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 방법에 대하여 설명한다.

[0045] 도 5는 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 방법을 설명하기 흐름도이고, 도 6은 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 구조를 설명하기 위한 흐름도이다.

[0046] 도 5를 참조하면, 누산 장치(100)의 프로세서(110)는 누산기(ACC)에 입력값을 입력할 수 있다(S110).

[0048] 그러면, 프로세서(110)는 누산기(ACC)를 통해 입력값으로부터 출력값을 획득할 수 있다(S130).

[0049] 여기서, 누산기(ACC)는 도 6에 도시된 바와 같이, 서로 계층적으로 연결된 복수의 캐리 세이브 가산기(CSA) 및 캐리 세이브 가산기(CSA)와 연결된 하나의 리플 캐리 가산기(RCA)를 포함할 수 있다. 그리고, 누산기(ACC)는 하나의 리플 캐리 가산기(RCA)에서만 오버플로우(overflow)와 언더플로우(underflow)의 센싱을 수행할 수 있다. 도 6에 도시된 "OF/UF"는 "오버플로우(overflow)/언더플로우(underflow)의 센싱"을 나타내며, 도 6에 도시된 바와 같이, 본 발명에 따른 누산기(ACC)는 서로 계층적으로 연결된 복수의 캐리 세이브 가산기(CSA)에서는 오버플로우(overflow)와 언더플로우(underflow)의 센싱을 수행하지 않고, 캐리 세이브 가산기(CSA)와 연결된 하나의 리플 캐리 가산기(RCA)에서는 오버플로우(overflow)와 언더플로우(underflow)의 센싱을 수행할 수 있다.

[0050] 즉, 프로세서(110)는 복수의 캐리 세이브 가산기(CSA)를 통해 입력값으로부터 중간값을 획득할 수 있다.

[0051] 그리고, 프로세서(110)는 하나의 리플 캐리 가산기(RCA)를 통해 중간값으로부터 출력값을 획득할 수 있다.

[0053] 그런 다음, 프로세서(110)는 누산기(ACC)를 통해 획득한 출력값을 출력할 수 있다(S150).

[0055] 종래의 누산기(ACC)와 본 발명에 따른 누산기(ACC)의 연산 딜레이(delay)를 비교하면 아래의 [표 1]과 같은 결과를 얻을 수 있다.

*표 1*

[0056]

| | **종래의 누산기 연산** | **본 발명에 따른 누산기 연산** |
|---|---|---|
| **구조** | 도 3에 도시된 누산기 구조 | 도 6에 도시된 누산기 구조 |
| **N=3** | $2((m-1)t_{carry}+t_{SUM}+t_{4:1MUX})$ | $t_{SUM}+(m-1)t_{carry}+t_{SUM}+t_{4:1MUX}$ |
| **N=4** | $2((m-1)t_{carry}+t_{SUM}+t_{4:1MUX})$ | $2t_{SUM}+(m-1)t_{carry}+t_{SUM}+t_{4:1MUX}$ |
| **N=5** | $3((m-1)t_{carry}+t_{SUM}+t_{4:1MUX})$ | $3t_{SUM}+(m-1)t_{carry}+t_{SUM}+t_{4:1MUX}$ |
| **N=6** | $3((m-1)t_{carry}+t_{SUM}+t_{4:1MUX})$ | $3t_{SUM}+(m-1)t_{carry}+t_{SUM}+t_{4:1MUX}$ |

| | | |
|---|---|---|
| N=2k-1,2k (k≥2) | $k((m-1)t_{carry}+t_{SUM}+t_{4:1MUX})$ | $kt_{SUM}+(m-1)t_{carry}+t_{SUM}+t_{4:1MUX}$ |

[0057] 여기서, N은 입력(input) 수를 나타낸다. m은 비트(bit) 수를 나타낸다.

[0060] 그러면, 도 7 내지 도 10을 참조하여 본 발명의 바람직한 실시예에 따른 CNN 기반 AI 모델의 누산 동작의 성능에 대하여 설명한다.

[0061] 도 7은 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 성능을 설명하기 위한 도면으로, 도 7의 (a)는 종래의 누산기(ACC)의 구조 일례를 나타내고, 도 7의 (b)는 본 발명에 따른 누산기(ACC)의 구조 일례를 나타낸다.

[0062] 도 7에 도시된 누산기(ACC)의 구조를 기반으로, 종래의 누산기(ACC)와 본 발명에 따른 누산기(ACC)의 연산 딜레이(delay)를 추정하면 아래의 [표 2]와 같은 결과를 얻을 수 있다. 여기서, 입력(input) 수인 N은 6이고, 비트(bit) 수인 m은 24이다.

*표 2*

[0063]

| | **종래의 누산기 연산** | **본 발명에 따른 누산기 연산** |
|---|---|---|
| **구조** | 도 7의 (a)에 도시된 누산기 구조 | 도 7의 (b)에 도시된 누산기 구조 |
| **딜레이 추정** | $23t_{carry} + t_{SUM} + t_{4:1MUX}$ [1 stage]<br>$+ 23t_{carry} + t_{SUM} + t_{4:1MUX}$ [2 stage]<br>$+ 23t_{carry} + t_{SUM} + t_{4:1MUX}$ [3 stage] | $t_{SUM} + t_{SUM} + t_{SUM}$ [1,2,3 stage]<br>$+ 24t_{carry} + t_{SUM} + t_{4:1MUX}$ [4 stage] |

[0064] 종래의 누산기(ACC)는 각 스테이지(stage)마다 오버플로우(overflow, OF)/언더플로우(underflow, UF) 센싱이 필요하므로 모든 스테이지(stage)에서 MSB(most significant bit)의 캐리(carry)를 기다려야 한다.

[0065] 이에 반면, 본 발명에 따른 누산기(ACC)는 마지막 스테이지(stage)에서만 오버플로우(overflow, OF)/언더플로우(underflow, UF) 센싱이 필요하므로 이전 스테이지(stage)에서는 캐리(carry)를 기다리지 않아도 된다.

[0067] 도 8은 도 7의 (b)에 도시한 본 발명에 따른 누산기(ACC)의 도트 다이어그램(dot diagram)을 나타낸다.

[0068] 도 8을 참조하면, 본 발명에 따른 누산기(ACC)는 총 4개의 스테이지(stage)를 거쳐 연산이 진행된다.

[0069] - 1, 2, 3 stage : CSA 구조 [$t_{Delay}$ : $t_{SUM} + t_{SUM} + t_{SUM}$]

[0070] - 4 stage : RCA 구조 [$t_{Delay}$ : $23t_{Carry} + t_{SUM}$]

[0071] 이때, 각 스테이지(stage)에서 부분 곱(partial product)은 부호가 있는 값(signed value)이다.

[0072] - S(Sum)과 C(Carry)의 비트(bit) 자리가 다르므로, 부호 확장(sign extension)이 필요하다.

[0074] 도 9는 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 성능을 설명하기 위한 도면으로, 도 9의 (a)는 종래의 누산기(ACC)의 부호 확장(sign extension) 구조를 나타내고, 도 9의 (b)는 본 발명에 따른 누산기(ACC)의 부호 확장(sign extension) 구조를 나타낸다.

[0075] 도 9의 (a)를 참조하면, 종래의 누산기(ACC)는 부호가 있는 수(signed number)를 더할 때, 확장 비트(extension bit)에 모두 MSB(most significant bit) 데이터를 넣어야 한다.

[0076] 이에 반면, 도 9의 (b)를 참조하면, 본 발명에 따른 누산기(ACC)는 입력 데이터에 확장(extension)을 하지 않고, 각 스테이지(stage)를 지날 때에만 하여 비트(bit) 자리를 맞추면서 덧셈이 진행된다.

[0077] 즉, 종래의 누산기(ACC)와 본 발명에 따른 누산기(ACC)의 확장(extension) 구조를 비교하면 아래의 [표 3]과 같

다.

표 3

[0078]

| | 종래의 누산기의 확장 구조 | 본 발명에 따른 누산기의 확장 구조 |
|---|---|---|
| Data path | extension bit만큼 증가 | extension과 무관 |
| Delay | extension bit만큼 증가 | unsigned 연산과 동일 |
| Area | extension bit만큼 증가 | unsigned 구조보다 1bit씩 추가 |

[0079] 종래의 CNN 구조에서 MAC(multiply-accumulate) 연산은 전체 계산(total computation)의 90% 이상을 소비하고 많은 면적을 차지한다.

[0080] 종래의 누산기(ACC)와 같은 부호 확장(sign extension) 구조를 사용할 경우, 실시간 구현(real-time implementation)에 적절하지 않는다.

[0081] - 모든 입력 데이터에 부호 확장(sign extension)을 해야 하고, 이를 모두 저장해야 하므로 큰 메모리 용량을 요구함.

[0082] - 각 스테이지(stage)마다 확장(extension)되는 데이터를 더해야 하므로 딜레이(delay)가 증가함.

[0083] 이에 반면, 본 발명에 따른 누산기(ACC) 구조를 사용할 경우, 모든 입력에 부호 확장(sign extension)을 하지 않고, 각 스테이지(stage)가 넘어갈 때에만 함.

[0084] - CNN 기반 시스템에서 면적을 많이 차지하는 MAC(multiply-accumulate) 구조의 면적을 줄임으로써 전체 면적의 감소(메모리 포함)에 큰 영향을 줄 수 있음.

[0085] - 연산 비중인 큰 MAC(multiply-accumulate) 연산 속도를 향상시킴에 따라 더 빠른 클럭(clk)으로 동작시킬 수 있으며, 이로 인해 더 높은 fps(frame per second)를 만족시킬 수 있음.

[0087] 도 10은 본 발명의 바람직한 실시예에 따른 누산기(ACC)의 성능을 설명하기 위한 도면으로, 도 10의 (a)는 도 7의 (a)에 도시한 종래의 누산기(ACC)의 시뮬레이션 결과를 나타내고, 도 10의 (b)는 도 7의 (b)에 도시한 본 발명에 따른 누산기(ACC)의 시뮬레이션 결과를 나타낸다.

[0088] 도 7에 도시된 누산기(ACC)의 구조를 기반으로, 종래의 누산기(ACC)와 본 발명에 따른 누산기(ACC)의 면적(area) 및 딜레이(delay)를 시뮬레이션한 결과는 도 10 및 아래의 [표 4]와 같다. 여기서, 입력(input) 수인 N은 6이고, 비트(bit) 수인 m은 24이다.

표 4

[0089]

| | | 종래의 누산기 구조 | 본 발명에 따른 누산기 구조 |
|---|---|---|---|
| Tr 개수 (개) | 1bit FA / HA | 115 / 5 | 121 / 2 |
| | 4:1 MUX | 120 | 50 |
| Area($\mu m^2$) | | 1014.95 | 650.03 |
| Pre-Sim delay(ns) | | 2.28 | 0.865 |

[0090] - 오버플로우(overflow, OF)/언더플로우(underflow, UF) 센싱을 하기 위해 2:1 MUX 3개를 이용하여 4:1 MUX를 설계함.

[0091] - 28nm에서 배치(layout)한 결과를 토대로 각각의 면적(area)을 측정함.

[0092] 1bit FA : 3.11$\mu m^2$ / 1bit HA : 1.86$\mu m^2$ / 4:1 MUX : 5.4$\mu m^2$

[0093] - 종래의 누산기 구조와 본 발명에 따른 누산기 구조 모두 임계 경로(critical path)의 최악의 케이스(worst case) 조건에서 측정한 값임.

[0094] Worst case : 000...001 + 111...110 -> 000...001 + 111...111

[0095] LSB 데이터가 0 -> 1로 전이(transition)되어 모든 캐리(carry)가 전파(propagation)되는 경우

[0096] 이와 같이, 본 발명에 따른 누산기(ACC) 구조는 종래의 누산기(ACC) 구조 대비 면적(area), 딜레이(delay) 면에서 각각 35.9%, 62.1% 향상된다.

[0097] 또한, 누산기(ACC)의 입력(input) 개수가 증가할 수록 종래의 누산기(ACC) 구조는 MSB(most significant bit)의 캐리(carry)를 기다려야 하는 스테이지(stage) 수가 증가한다. 따라서, 종래의 누산기(ACC) 구조 대비 딜레이(delay) 차이가 더 극명할 것으로 예상된다.

[0098] 본 발명에 따른 누산기(ACC) 구조를 사용함으로 인해, 다음과 같은 이점이 있다.

[0099] 1) CNN 기반 SR(super-resolution)에서 면적을 많이 차지하는 MAC(multiply-accumulate) 구조의 면적을 줄임으로써 전체 면적 감소에 큰 영향을 줄 수 있음.

[0100] 2) 연산 비중이 큰 MAC(multiply-accumulate) 연산 속도를 향상시킴에 따라 더 빠른 클럭(clk)으로 동작시킬 수 있음. 또한, 이로 인해 더 높은 fps(frame per second)를 만족 시킬 수 있음.

[0103] 본 실시예들에 따른 동작은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능한 저장 매체에 기록될 수 있다. 컴퓨터 판독 가능한 저장 매체는 실행을 위해 프로세서에 명령어를 제공하는데 참여한 임의의 매체를 나타낸다. 컴퓨터 판독 가능한 저장 매체는 프로그램 명령, 데이터 파일, 데이터 구조 또는 이들의 조합을 포함할 수 있다. 예컨대, 자기 매체, 광기록 매체, 메모리 등이 있을 수 있다. 컴퓨터 프로그램은 네트워크로 연결된 컴퓨터 시스템 상에 분산되어 분산 방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수도 있다. 본 실시예를 구현하기 위한 기능적인(Functional) 프로그램, 코드, 및 코드 세그먼트들은 본 실시예가 속하는 기술 분야의 프로그래머들에 의해 용이하게 추론될 수 있을 것이다.

[0104] 본 실시예들은 본 실시예의 기술 사상을 설명하기 위한 것이고, 이러한 실시예에 의하여 본 실시예의 기술 사상의 범위가 한정되는 것은 아니다. 본 실시예의 보호 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 실시예의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

## *부호의 설명*

[0105] 100 : 누산 장치,

110 : 프로세서,

130 : 컴퓨터 판독 가능한 저장 매체,

131 : 프로그램,

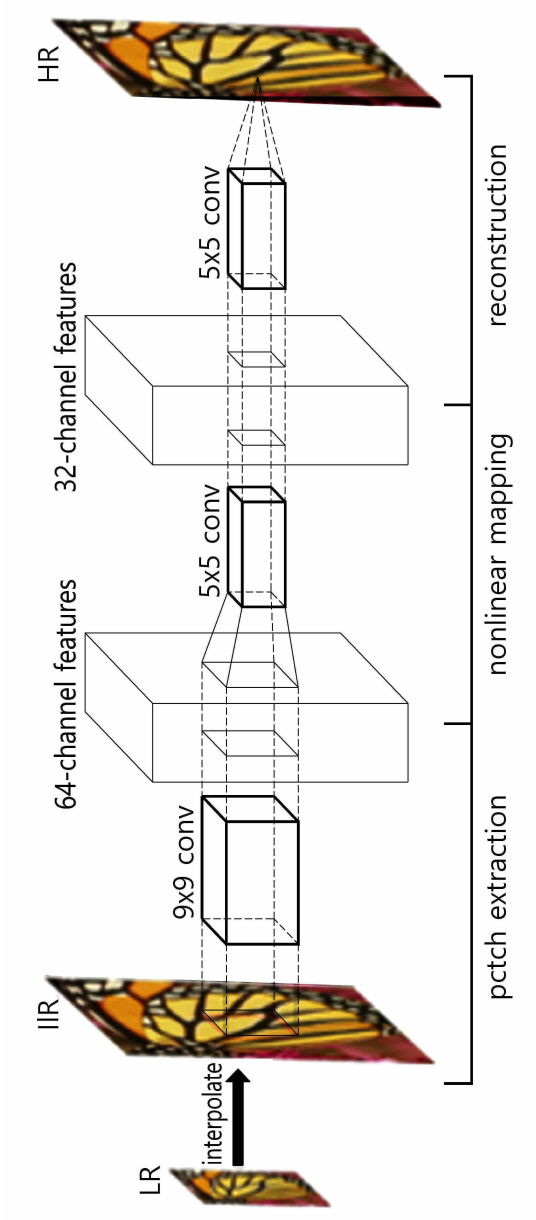
150 : 통신 버스,

170 : 입출력 인터페이스,

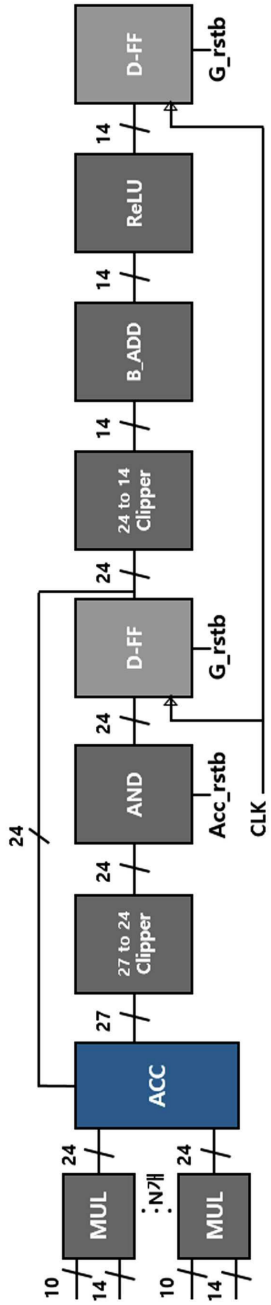
190 : 통신 인터페이스

도면

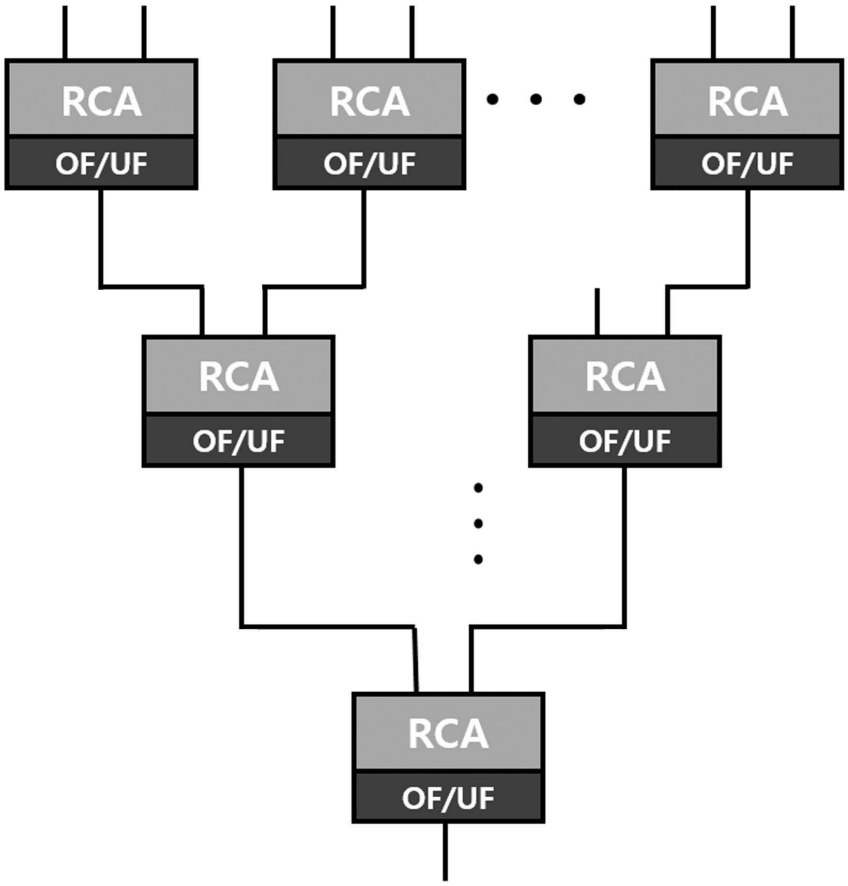
도면1


HR
5x5 conv
32-channel features
5x5 conv
64-channel features
9x9 conv
IIR
interpolate
LR
reconstruction
nonlinear mapping
pctch extraction

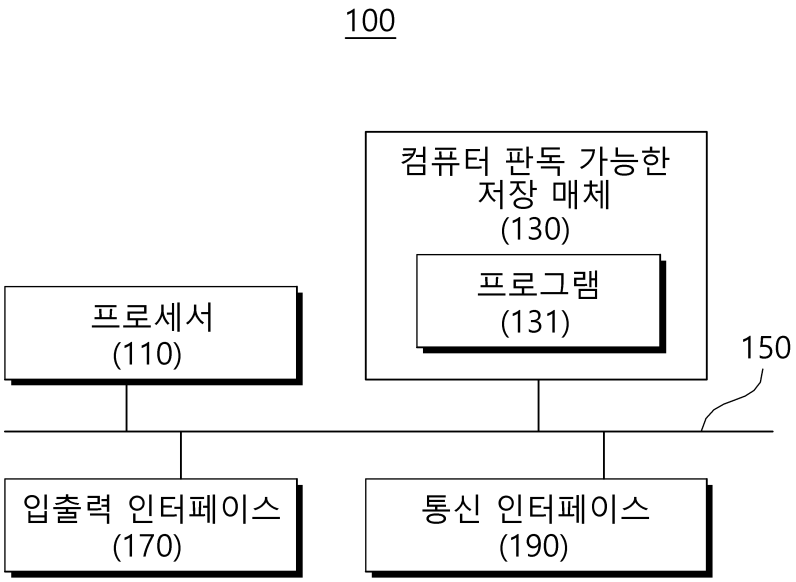
도면2


D-FF
G_rstb
14
ReLU
14
B_ADD
14
24 to 14 Clipper
24
D-FF
G_rstb
24
AND
Acc_rstb
CLK
24
24
27 to 24 Clipper
27
ACC
24
24
MUL
MUL
10
14
10
14

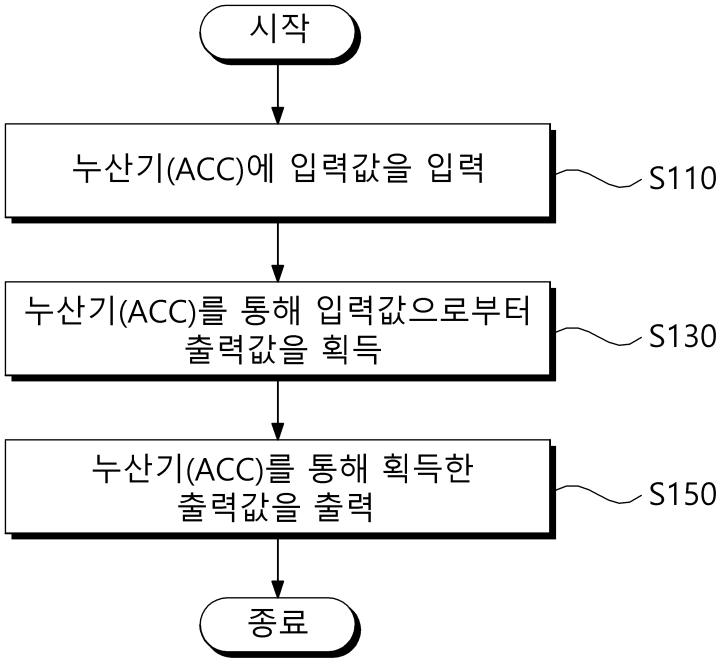
도면3


RCA
OF/UF
RCA
OF/UF
RCA
OF/UF
RCA
OF/UF
RCA
OF/UF
RCA
OF/UF

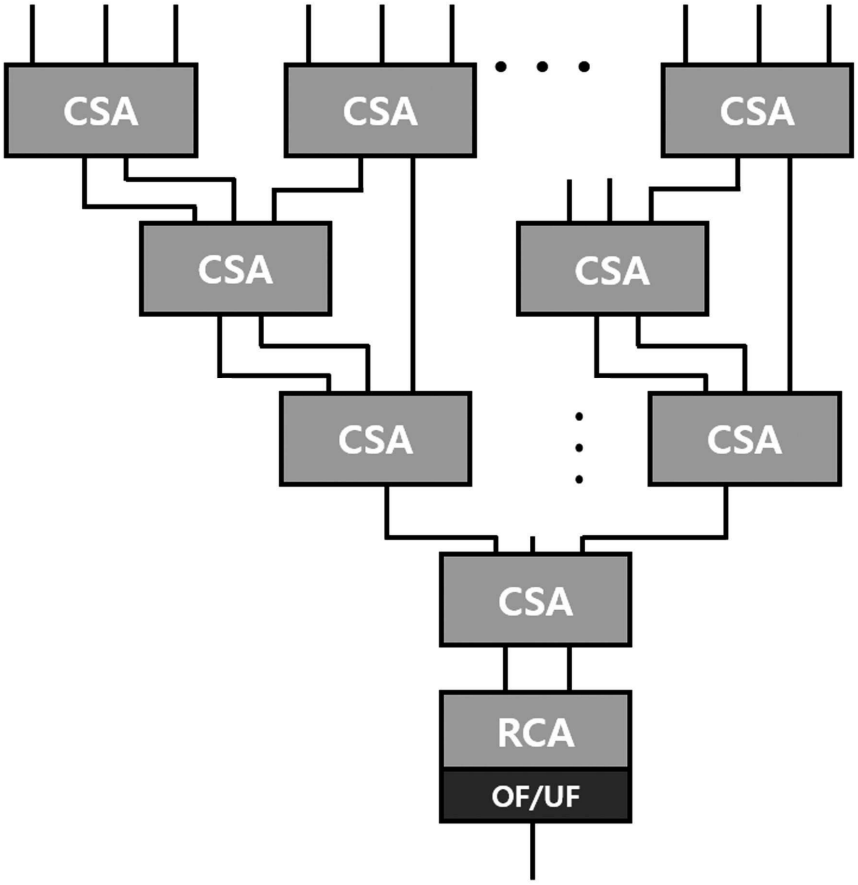
도면4


100
컴퓨터 판독 가능한
저장 매체
(130)
프로그램
(131)
프로세서
(110)
150
입출력 인터페이스
(170)
통신 인터페이스
(190)

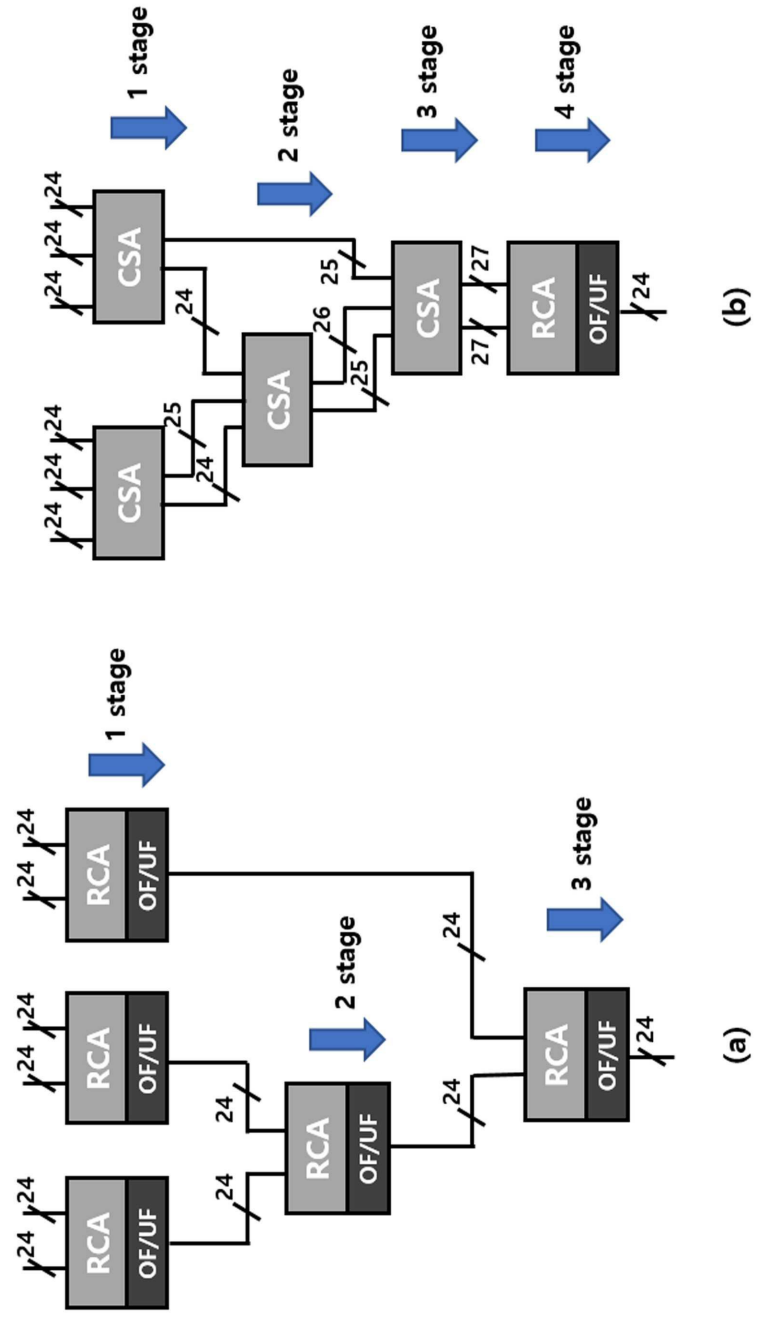
도면5


시작
누산기(ACC)에 입력값을 입력
S110
누산기(ACC)를 통해 입력값으로부터 출력값을 획득
S130
누산기(ACC)를 통해 획득한 출력값을 출력
S150
종료

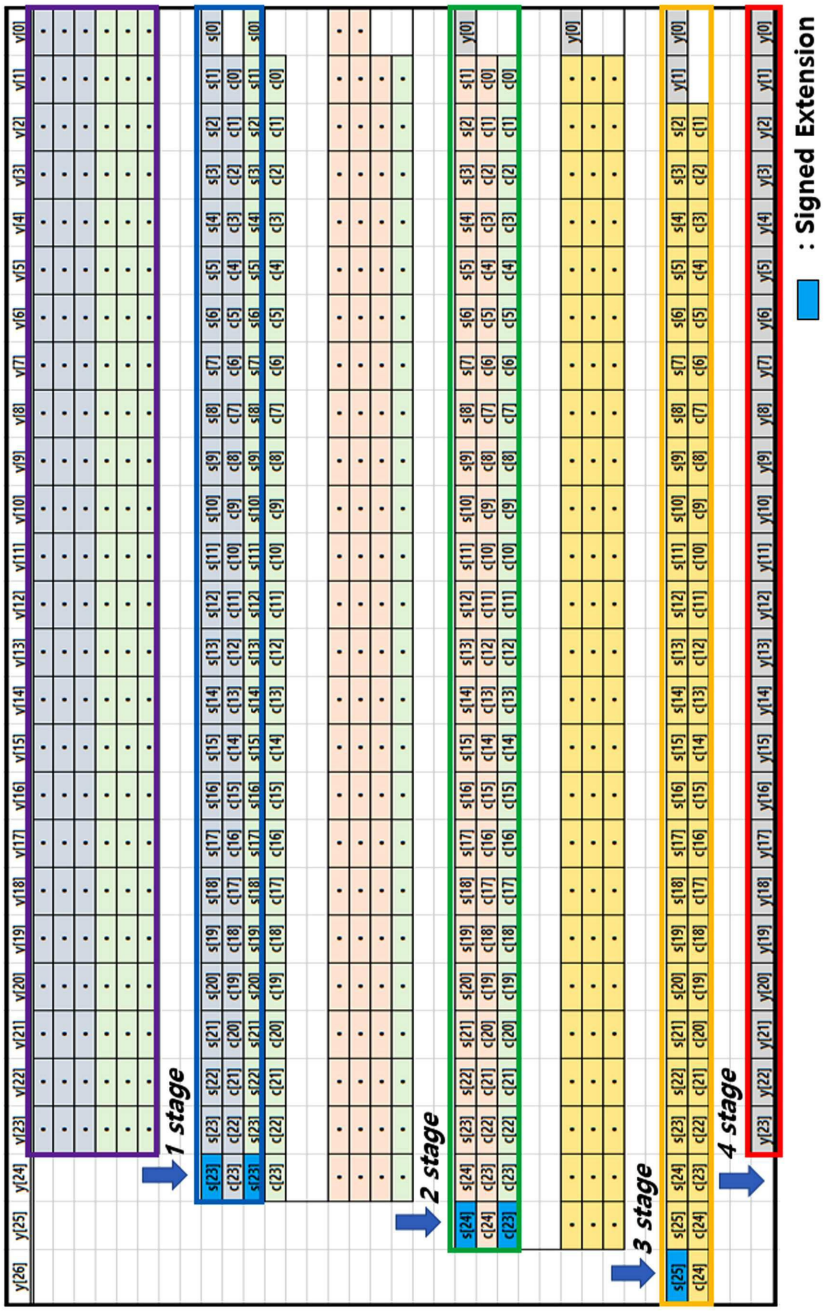
도면6


CSA
CSA
CSA
CSA
CSA
CSA
CSA
CSA
RCA
OF/UF

도면7

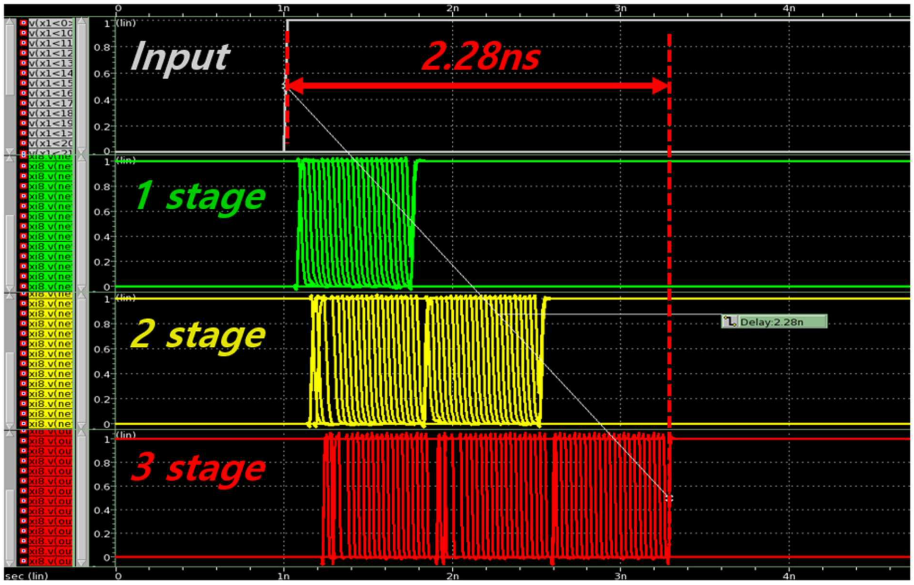


도면8

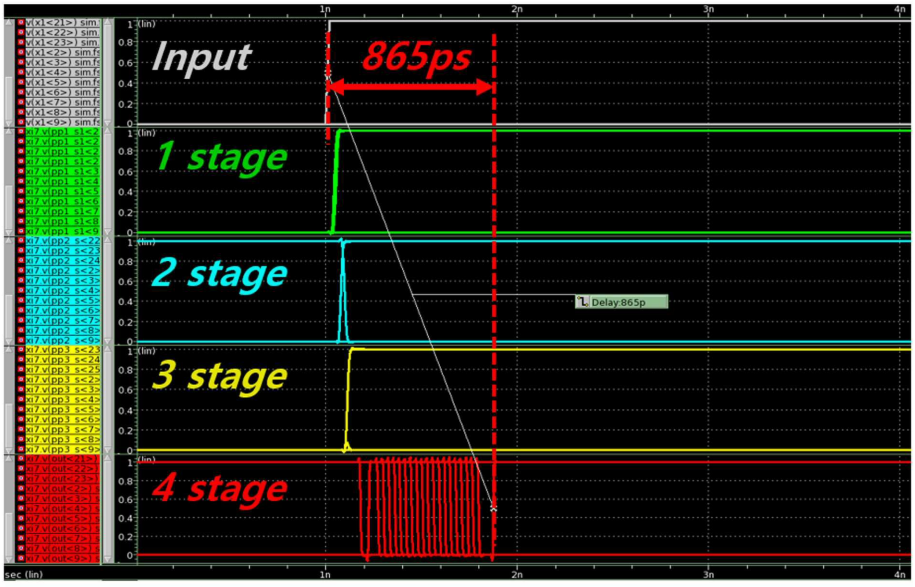


도면9

도면10


Input
2.28ns
1 stage
2 stage
Delay:2.28n
3 stage

(a)


Input
865ps
1 stage
2 stage
Delay:865p
3 stage
4 stage

(b)