The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 20, No. 1, pp.193-198, Feb. 29, 2020. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2020.20.1.193
JIIBC 2020-1-27

XOR연산 기반의 데이터 재구성 기법을 활용한 컨볼루셔널 뉴럴 네트워크 성능 향상 기법

Techniques for Performance Improvement of Convolutional Neural Networks using XOR-based Data Reconstruction Operation

김영웅*

Young-Ung Kim*

요 약 컨볼루셔널 뉴럴 네트워크 기술의 다양한 활용은 컴퓨팅 분야의 발전을 가속화하고 있으나, 이에 대한 반대급부로 심각한 하드웨어 성능 부족을 초래하고 있다. 그 대응책으로 뉴럴 네트워크 가속기, 차세대 메모리 소자 기술, 그리고 고대역폭 메모리 구조 등이 제안되었으나, 이들은 각각 범용성, 기술 성숙도, 그리고 높은 비용의 문제를 야기하여 적극적으로 도입되기 어려운 실정이다. 따라서 현재의 하드웨어 범용성을 그대로 유지하면서도 컨볼루셔널 뉴럴 네트워크 기술의 성능을 증대시킬 수 있는 방안이 필요하다. 본 연구는 메인메모리 내부에서 리프레쉬 동작이 수행되는 상황에서도 미리 저장된 XOR 비트 값을 사용하여 리프레쉬 동작의 종료 시점까지 대기하지 않아도 읽기 동작을 완료할 수 있는 DRAM 기반 메인메모리 기술을 제안한다. 실험 결과 제안 기법은 5.8%의 수행 속도 향상 및 1.2%의 에너지 절감, 그리고 10.6%의 EDP 향상을 보여주었다.

Abstract The various uses of the Convolutional Neural Network technology are accelerating the evolution of the computing area, but the opposite is causing serious hardware performance shortages. Neural network accelerators, next-generation memory device technologies, and high-bandwidth memory architectures were proposed as countermeasures, but they are difficult to actively introduce due to the problems of versatility, technological maturity, and high cost, respectively. This study proposes DRAM-based main memory technology that enables read operations to be completed without waiting until the end of the refresh operation using pre-stored XOR bit values, even when the refresh operation is performed in the main memory. The results showed that the proposed technique improved performance by 5.8%, saved energy by 1.2%, and improved EDP by 10.6%.

Key Words : Convolutional neural network, Memory refresh, XOR operation

*정회원, 한성대학교 컴퓨터공학부(교신저자)
접수일자: 2019년 10월 11일, 수정완료: 2020년 1월 11일
게재확정일자: 2020년 2월 7일

Received: 11 October, 2019 / Revised: 11 January, 2020 / Accepted: 7 February, 2020
*Corresponding Author: yukim@hansung.ac.kr
Dept. of Computer Engineering, Hansung University, Korea

Ⅰ. 서 론

컨볼루셔널 뉴럴 네트워크(Convolutional Neural Network) 기술은 사물인식부터 차량의 자율주행까지 다양한 분야에서 급진적으로 사용되고 있으나, 뉴럴 네트워크의 긴 학습 시간과 높은 에너지 소모로 인하여 점차 한계를 맞이하고 있다. 이러한 한계를 극복하기 위해 뉴럴 네트워크 가속기(Neural network accelerator)[1][2], 차세대 메모리 소자의 활용[3][4], 그리고 고대역폭 메모리(High-bandwidth memory) 구조[5] 등이 제안되었으나, 이들은 각각 범용성, 기술 성숙도, 높은 비용 등의 이유로 도입이 어려운 상황이다. 전용 가속기는 뉴럴 네트워크 알고리즘의 반복 연산에 대한 효율성을 높일 수 있으나, 뉴럴 네트워크의 종류에 따라 다른 가속기 구조가 요구되며 더욱이 메모리 대역폭 부족의 문제는 해결하기 어렵다. STT-RAM(Spin-Torque Transfer RAM), PCM(Phase Change Memory) 등의 차세대 메모리 소자를 사용하면 에너지 절감 효과를 기대할 수 있으나, 기존 메모리 소자 대비 느린 속도가 문제가 된다. 고대역폭 메모리 구조는 뉴럴 네트워크의 높은 메모리 대역폭 문제를 해결할 수 있으나, 높은 생산 비용 및 낮은 범용성으로 인하여 적극적으로 도입하기 어렵다.

본 연구는 기존의 DRAM 기반 메인메모리의 범용성을 유지하면서도 성능과 에너지를 절감할 수 있는 기법을 제안한다. DRAM의 소자 특성상 시간이 지남에 따라 그 값을 잃어버려 저장된 값을 유지하기 위해서는 64ms 이내에 다시 충전해야 하는 리프레쉬(Refresh)[6] 과정을 거치는데, 리프레쉬 도중에는 읽기/쓰기 동작이 불가능하며 이는 메모리 접근 빈도가 높은 컨볼루셔널 뉴럴 네트워크 알고리즘의 성능을 저하시키는 요인이 된다. 특히, 읽기 동작은 수행 속도에 직접적인 요인이 되므로 리프레쉬 중에도 읽기 동작을 가능하게 하는 기법이 요구된다. 메인메모리 내 특정 부분에서 리프레쉬가 발생할 경우, 이 위치의 데이터를 재구성할 수 있는 추가 정보가 있다면 리프레쉬 종료까지 대기하지 않아도 된다.

제안 기법에서는 처리 속도가 빠른 XOR(Exclusive OR) 게이트를 사용하여 병렬적으로 저장되는 데이터들에 대한 XOR 비트를 추가 공간에 저장한 후, 이후 이 값을 데이터 재구성에 사용한다. USIMM 메모리 시뮬레이터를 통하여 제안 기법을 평가한 결과 9개 뉴럴 네트워크에 대하여 평균 5.8%의 성능 향상, 1.2%의 에너지 절감, 10.6%의 EDP(Energy-Delay Product)가 줄어들었다.

본 논문의 구성은 다음과 같다. 2절에서는 DRAM 기반 메인메모리 및 컨볼루셔널 뉴럴 네트워크에 대한 배경을 설명하고, 3절에서는 XOR 연산 기반의 데이터 재구성 방법 및 적용 방법을 소개한다. 4절에서는 USIMM 시뮬레이터를 통해 제안 기법의 효과를 검증하며, 마지막으로 5절에서 결론을 맺는다.

Ⅱ. 배 경

1. 컨볼루셔널 뉴럴 네트워크

뉴럴 네트워크는 다수의 뉴런(Neuron)과 레이어(Layer)로 구성된 네트워크이다[7]. 그림 1은 뉴럴 네트워크의 기본 구조를 나타낸다. 이 네트워크에서는 입력 레이어의 뉴런에 대해 입력 xn이 전달되며, 각각의 입력 값에 대한 가중치 wn을 곱하여 그 값이 한계점인 f(θ)를 넘으면 뉴런을 실행하고 그결과를 다음 뉴런으로 전달한다. 입력 레이어와 출력 레이어 사이에 한 개 이상의 히든 레이어(Hidden layer)가 존재한다.

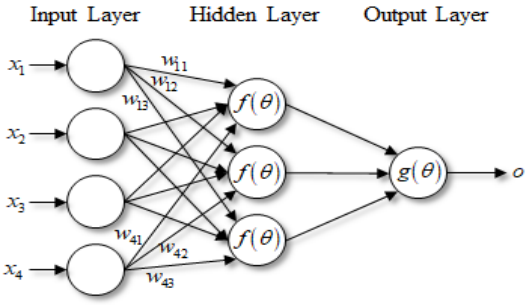

그림 1. 뉴럴 네트워크의 기본 구조
Fig. 1. Basic structure of the Neural network

컨볼루셔널 뉴럴 네트워크는 기본 뉴럴 네트워크에 여러 개의 컨볼루셔널 레이어 구조를 추가한 네트워크이다[7][8]. 이 네트워크는 컨볼루셔널 레이어를 통해서 입력 받은 이미지에 대한 특징을 추출하고, 추출한 특징을 기반으로 기존의 뉴럴 네트워크를 사용하여 이미지를 분류한다. 컨볼루셔널 뉴럴 네트워크는 학습 과정에서 많은 메모리 접근이 필요하기 때문에 많은 시간과 높은 파워/에너지를 요구한다. 특히, 네트워크에 많은 중간 레이어가 존재하는 점은 메인메모리에 대한 많은 접근을 유발시키는 주요 요인이 되며, 따라서 메인메모리에서 발생하는 병목현상은 컨볼루셔널 뉴럴 네트워크의 성능 및 에너지

에 영향을 미치는 주요한 원인이 된다.

2. 메인메모리

그림 2는 메인메모리 구조를 나타낸다. 메인메모리는 채널, 랭크, 뱅크, 로우, 컬럼으로 나뉜다. 채널을 통해 여러 랭크에 접근할 수 있으며, 각 랭크에는 여러 개의 뱅크가 존재하고, 뱅크 안에는 여러 개의 로우가 위치하고 있다. 그리고 하나의 로우는 여러 개의 컬럼으로 나뉜다. *DDRx*의 구조에 따르면 64 바이트의 데이터를 메모리로부터 가져오기 위해서는 하나의 뱅크에 묶여있는 여러 디바이스에서 병렬적으로 데이터에 접근한다[5].

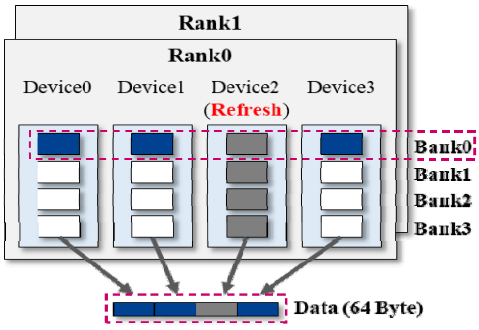

그림 2. DRAM 메인메모리 구조
Fig. 2. DRAM Main memory basic structure

III. 제안 기법

1. XOR 연산 기반 데이터 재구성

컨볼루셔널 뉴럴 네트워크는 그 구조와 알고리즘의 특성으로 인하여 많은 메모리 접근을 요구한다. 이 접근들 중 읽기 연산에 대한 지연이 발생할 경우 성능에 직접적인 영향을 끼치게 되며, 잦은 리프레쉬 동작이 하나의 주요한 원인이 된다. 따라서 본 연구에서는 메인메모리가 리프레쉬 동작을 수행하는 중에도 지연 없이 읽기 동작을 수행할 수 있는 기법을 제안한다.

그림 2에서 보는 바와 같이 하나의 뱅크는 다수의 로우를 포함하고 있다. 이 로우들은 물리적으로 여러 디바이스에 나뉘어져 있는데, 이 중 하나가 리프레쉬 중일 경우 종료되기 전까지 데이터를 읽을 수 없다. 그러나 만약 데이터를 저장할 때 특정 디바이스에 속한 데이터를 복원할 수 있는 정보를 함께 저장한다면 리프레쉬 중에도 데이터를 복원하여 읽기 동작을 수행할 수 있다.

그림 3의 위의 그림은 메인메모리에 데이터를 저장할 때 함께 저장되는 추가 정보의 생성 방법을 나타낸다. 이 추가 정보는 각 디바이스에 저장될 데이터 비트들에 대해 XOR 연산을 수행한 값이다. 예를 들어, 메인메모리의 한 뱅크에 네 개의 물리 디바이스가 존재하며 읽기/쓰기 데이터의 크기를 64-바이트라고 가정하면, 이 데이터는 네 개로 나뉘어져 각 디바이스에 16-바이트씩 저장되게 된다. 이 네 개의 16-바이트 부분 데이터들에 대해 XOR 연산을 하면 16-바이트의 XOR된 데이터가 나오게 된다. 이 XOR된 추가 데이터를 메인메모리의 추가 공간에 저장하여 추후 리프레쉬가 수행중인 디바이스의 부분 데이터를 재구성할 수 있다. 이 동작은 메인메모리에 쓰기 동작이 수행될 때만 발생한다.

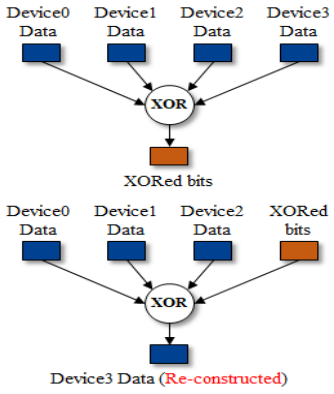

그림 3. XORed 비트 생성 및 데이터 재구성 방법
Fig. 3. XORed bits generation and Reconstructing data

그림 3의 밑의 그림은 메인메모리의 쓰기 동작 시 생성된 XOR된 추가데이터를 이용하여 원래의 부분 데이터를 재구성하는 방법을 나타낸다. 디바이스 3이 리프레쉬를 수행 중이라고 가정하면, 이 상황에서 읽기 동작에 대한 요청이 있을 경우 전통적인 메인메모리 구조에서는 디바이스 3의 리프레쉬가 끝날 때까지 대기하여야 하지만, XOR된 추가 데이터가 있다면 이를 통하여 디바이스 3에 위치한 부분 데이터를 재구성할 수 있다. 리프레쉬가 수행중인 디바이스 3의 데이터를 얻기 위하여 디바이스 0부터 디바이스 2의 부분 데이터 및 이전에 생성된 XOR된 비트를 함께 XOR시키면 원래의 디바이스 3에 대한 부분 데이터를 얻을 수 있다.

2. 제안 기법의 하드웨어 아키텍처

그림 4는 제안 기법을 적용한 메인메모리 구조를 나타

낸다. 기존 메인메모리 구조와는 달리 제안 기법을 적용한 구조에서는 XOR된 비트를 저장하기 위한 디바이스 4가 추가되었다. 이 디바이스도 데이터를 저장하는 디바이스와 물리적으로 동일한 구조이므로 이 디바이스에도 주기적인 리프레쉬가 수행되어야 한다. 이는 기존 구조 대비 추가적인 에너지 소모를 유발함을 의미하지만, 읽기 동작의 경우 리프레쉬가 끝난 후 원래 읽어야 할 로우를 읽을 필요가 없기 때문에 그만큼의 성능과 에너지를 감소할 수 있다.

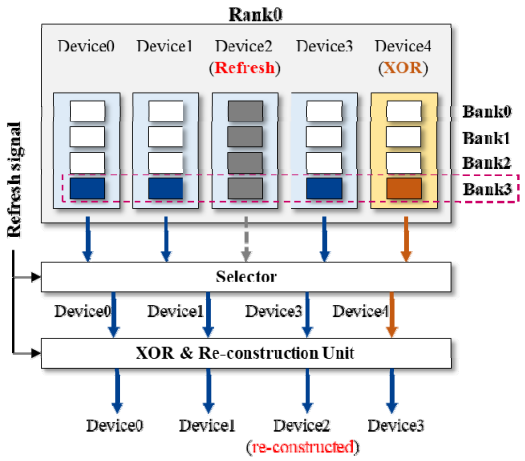

그림 4. 제안 기법의 하드웨어 아키텍처
Fig. 4. Hardware architecture of the proposed technique

제안 기법을 사용하기 위해서는 여분의 디바이스 외에 추가적으로 Selector와 XOR & Re-construction Unit이 필요하다. Selector는 여러 디바이스들로부터 부분 데이터를 선별하여 받아들이기 위하여 사용되며, 입력값으로 각 디바이스들에 대한 리프레쉬 여부를 사용한다. 예를 들어, 현재 랭크에 속한 어느 디바이스도 리프레쉬를 수행하고 있지 않다면 Selector는 기존의 메인메모리와 같이 디바이스 0부터 디바이스 3까지의 부분 데이터를 받아 하나의 64-바이트 데이터로 만든 후 프로세서로 전송한다. 만약 특정 디바이스가 리프레쉬 중이라면 Selector는 디바이스 0부터 디바이스 4중 리프레쉬가 발생한 디바이스를 제외한 부분 데이터와 XOR된 추가 데이터가 XOR & Re-construction Unit으로 전송된다. Selector는 (디바이스 개수-1)만큼의 MUX (Multiplexor)로 구현된다.

XOR & Re-construction Unit 역시 디바이스들의 리프레쉬 정보 및 Selector로부터 전달받은 부분 데이터와 XOR된 추가 데이터를 입력으로 받는다. 만약 디바이스 1에서 디바이스 3중 리프레쉬 중인 디바이스가 있다면 이에 해당하는 부분 데이터를 재구성하기 위하여 XOR된 추가 데이터를 사용하여 데이터를 복원한다. 이후 복원된 데이터는 원래의 위치에 배치되어 64-바이트 전체 데이터를 구성하는데 사용된다. XOR & Re-construction Unit은 (디바이스의 개수 -1) x (한번에 가져오는 부분데이터의 비트 수)만큼의 XOR 게이트가 필요하며, 또한 부분 데이터를 재배치시키는데 필요한 Shifter가 필요하다.

Selector와 XOR & Re-construction Unit을 구성하는 하드웨어는 메인메모리 전체 하드웨어와 비교하였을 때 매우 작은 영역만을 차지하므로 이에 대한 비용은 본 연구에서는 무시할만한 수준이라고 가정하였다.

IV. 실험 결과

1. 실험 방법

제안 기법의 효과를 검증하기 위하여 Instrumentation 기반 시뮬레이터인 Snipersim[9]를 통하여 컨볼루셔널 뉴럴 네트워크에 대한 트레이스를 생성하였고, 이를 cycle-accurate 메모리 시뮬레이터인 USIMM 시뮬레이터[10]의 입력으로 사용하여 메모리 동작의 효율성을 평가하였다. 메모리 용량은 16GB, 채널, 랭크, 뱅크의 수는 각각 2개, 2개, 그리고 8개로 설정하였다. 또한 하나의 뱅크는 65535개의 로우를 포함하고 있다. 표 1은 이러한 시뮬레이터의 전체 구성을 나타낸다.

표 1. 시뮬레이터 설정
Table 1. Simulator Settings

Configuration	Value
Number of cores	4
Processor clock speed	3.2GHz
Processor ROB size	192
Processor fetch/retire width	4
Last level cache (shared)	1MB, 8-way, 64B lines
Memory size	16GB
Memory bus speed	800MHz
Banks, Ranks, Channels	8, 2, 2
Rows per bank	64K
Cache lines per row	128
Page allocation policy	Random

컨볼루셔널 뉴럴 네트워크들은 Darknet에 포함되어 있는 네트워크 구성(Topology) 중 9개를 선정하여 평가하였다. 이들은 모두 서로 다른 뉴런의 개수, 레이어의 개수, 학습 알고리즘을 사용한다. 표 2는 이들 네트워크에

대한 MPKI(Memory accesses Per Kilo Instruction) 및 사용 메모리 공간(Footprint)을 나타낸다. 9개의 뉴럴 네트워크 모두 일반 어플리케이션 대비 매우 높은 메모리 접근 횟수를 가지고 있으며, 높은 읽기 동작 비율을 가지고 있다.

표 2. Kilo Instruction 당 메모리 접근 횟수
Table 2. Number of memory accesses per Kilo Installation

Workloads	Read-PKI	Write-PKI	Footprint(GB)
AlexNet	17.9	16.7	0.3
DarkNet19	33.7	24.2	0.2
DartNet53	34.9	28.5	0.4
Extraction	35.6	22.6	0.2
ResNet152	27.0	24.8	0.7
ResNet18	39.3	26.2	0.1
ResNet50	28.8	22.9	0.3
ResNext50	16.4	9.6	0.4
ResNext152	22.1	19.5	1.0

2. 성능 오버헤드

그림 5는 기본 메모리 구조 대비 제안 기법에 대한 상대적인 실행 시간을 나타낸다. Baseline은 기존 메모리 구조를 나타내며, Re-construction은 제안 기법을 나타낸다. 대부분의 트레이스들에서 기본 메모리 구조 대비 더 적은 실행 시간을 보였다. 특히, Extraction과 ResNext50의 경우 거의 10%에 가까운 성능 향상을 보였는데, 이는 표 2에서 볼 수 있듯이 메모리 읽기/쓰기 동작 중 읽기 동작이 더 많아 메모리 접근 중 리프레쉬로 인한 읽기 지연을 피할 수 있는 경우가 더 많기 때문이다. 이와 반대로 ResNet152와 ResNext152의 경우는 읽기와 쓰기 동작의 비율이 크게 차이가 나지 않기 때문에 성능 향상 역시 큰 폭으로 차이가 나지 않는다. 평균적으로는 9개 트레이스에 대하여 약 5.8%의 성능 향상이 나타났다.

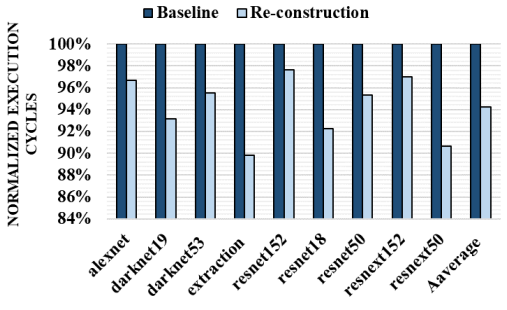

그림 5. 제안 기법의 성능 오버헤드
Fig. 5. Performance overhead of proposed technique

3. 에너지 오버헤드

그림 6은 제안 기법에 대한 에너지 소모를 나타낸다. 제안 기법은 추가적인 디바이스를 필요로 하기 때문에 기존 메모리 구조 대비 평균 4.9%의 더 높은 파워를 요구한다. 그러나 메인메모리에서의 읽기 동작이 많을수록 더 적은 실행 사이클을 나타내기 때문에 에너지 측면에서는 오버헤드가 상쇄되거나 더 나은 효율을 보인다. Extraction 및 ResNext50의 경우 각각 2.5%와 3%의 더 나은 에너지 효율을 보인 반면, ResNext152와 ResNet50의 경우 각각 0.1%와 0.5% 정도의 추가 에너지를 요구한다. 평균적으로는 제안 기법을 통해 1.2%의 에너지를 절감할 수 있는 것으로 나타났다.

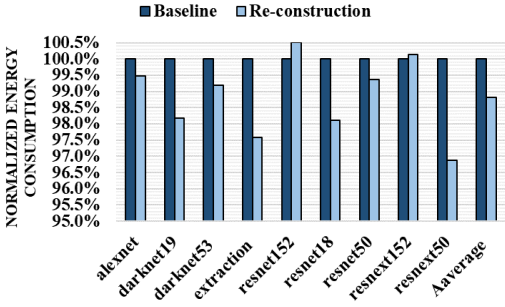

그림 6. 제안 기법의 에너지 오버헤드
Fig. 6. Energy overhead of proposed technique

4. EDP

그림 7은 기본 메모리 구조 대비 제안 기법의 상대적 EDP를 나타낸다. 비록 일부 트레이스에서 제안 기법이 더 높은 에너지 소모를 나타내었으나, 제안 기법을 통한 실행 시간 감소로 인하여 EDP의 효율성은 모든 트레이스에 대해 증가되었다. Extraction, ResNext50에 대하여 각각 18.3% 및 17.2%의 EDP 감소가 나타냈으며 평균적으로 10.6%의 EDP 절감 효과가 나타났다.

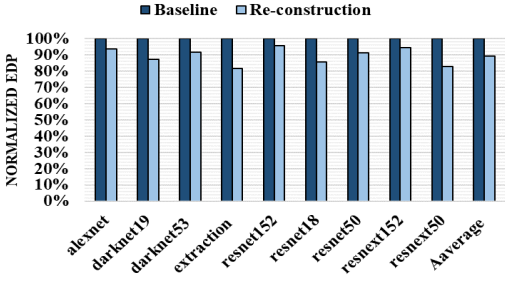

그림 7. 제안 기법의 EDP
Fig. 7. EDP of the proposed technique

5. 공간 오버헤드

제안 기법은 XOR된 추가 데이터를 저장하기 위하여 하나의 디바이스를 메모리 내에 추가해야 한다. 메인메모리가 데이터를 저장하는 디바이스만 내장하고 있다면 공간 오버헤드는 (*1/전체 디바이스의 수*)가 된다. 그러나 근래의 메인메모리는 높은 집적률로 인하여 잦은 에러가 발생하는 상황이며, 이를 위해 메인메모리 내부에 ECC (Error Correcting Code)를 저장하는 ECC-DIMM 구조를 사용한다. 따라서 ECC를 저장하는 공간에 제안 기법의 XOR된 추가 데이터를 저장할 경우 공간 오버헤드는 발생하지 않는다.

V. 결 론

본 연구는 범용 메인메모리 구조 기반 하에서 리프레쉬 동작으로 인하여 읽기 동작에 대한 대역폭 저하를 방지할 수 있는 DRAM 기반 메인메모리 기술을 제안하였다. 제안 기법에서는 리프레쉬 상황에서도 미리 저장된 XOR 비트 값을 사용하여 접근 불가능한 영역의 데이터를 복원할 수 있다. 실험 결과에 의하면 제안 기법은 5.8%의 수행 속도 향상 및 1.2%의 에너지 절감, 그리고 10.6%의 EDP 향상을 보여주었다.

References

[1] T. Ham et al., "Graphicionado: A High-Performance and Energy-Efficient Accelerator for Graph Analytics", MICRO 2014.
DOI: 10.1109/MICRO.2016.7783759

[2] Y. Wang et al., "Re-architecting the On-chip memory Sub-system of Machine-Learning Accelerator for Embedded Devices", ICCAD 2016.
DOI: 10.1145/2966986.2967068

[3] D. Xue et al., "Adaptive Memory Fusion: Towards Transparent, Agile Integration of Persistent Memory", HPCA 2018.
DOI: 10.1109/HPCA.2018.00036

[4] Lee, Jong Chern, et al. "High bandwidth memory (HBM) with TSV technique." 2016 International SoC Design Conference (ISOCC). IEEE, 2016.
DOI: 10.1109/ISOCC.2016.7799847

[5] K. Chang et al., "Improving DRAM Performance by Parallelizing Refreshes with Accesses", HPCA 2014.
DOI: 10.1109/HPCA.2014.6835946

[6] J. Kotra et al., "Hardware-Software Co-design to Mitigate DRAM Refresh Overheads: A Case for Refresh-Aware Process Scheduling", ASPLOS 2017.

[7] Y. Liang et al., "Deep learning based inference of private information using embedded sensors in smart devices," IEEE Network, 2018.
DOI: 10.1109/MNET.2018.1700349

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[9] T. E. Carlson et al., "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in SC, 2011.
DOI: 10.1145/2063384.2063454

[10] N. Chatterjee et al., "Usimm: the utah simulated memory module," University of Utah, Tech. Rep, 2012.

저 자 소 개

김 영 웅(정회원)

- 1993년 KAIST 전산학과 박사
- 1984 ~ 1997년 KT 통신망연구소
- 1997년 ~ 현재 한성대학교 컴퓨터공학과 교수
- 주관심분야 : 시스템 신뢰도, 소프트웨어 설계, 임베디드 시스템

※ 본 연구는 한성대학교 교내연구비 지원 과제임.