https://doi.org/10.7840/kics.2024.49.1.104

메모리 내 컴퓨팅의 연산 오류 정정 및 비-복호 감지를 위한 선택적 AN 부호 설계

김 명 인•, 하 정 석°

Selective AN Code Design for Correcting Computation Errors and Mis-Decoding Detection in Process-in-Memory

Myung-in Kim•, Jeong-seok Ha°

요 약

본 논문은 메모리 내 컴퓨팅 장치의 연산 과정에서 발생하는 오류를 정정하고 비-복호 (mis-decoding) 현상을 감지할 수 있는 선택적 AN 부호의 설계 방식을 제안하였다. AN 부호는 메시지 데이터에 양의 정수 A를 곱하여 부호화하는 오류정정부호이다. 이 부호는 오류가 발생한 부호화 데이터를 A로 나눈 나머지와 오류 패턴 사이의 순람표를 통해 오류를 색출하여 정정한다. 본 논문에서는 주어진 오류 환경에 따라 선택적으로 AN 부호를 설계하는 방법을 제안하였다. 먼저, 중요도가 높은 비트 라인들을 묶어서 정정 가능 집합으로 정의하고 이 집합 내에서 발생하는 오류 패턴들을 고칠 수 있으며 정정 가능 집합 외의 비트라인에서 발생한 오류 패턴의 비-복호를 감지할 수 있는 AN 부호 설계 조건을 제안하였다. 제안한 선택적 (selective) AN 부호 설계 방식은 기존의 정적 (static) AN 부호 방식과 비교하여 오류 환경에 따른 유연한 설계가 가능하다. 또한, 마찬가지로 유연한 부호 설계가 가능한 기존의 ABN 부호보다 저장공간 측면에서 더 효율적으로 복호기 구현이 가능하다.

키워드 : PIM, AN 부호, 가중치 행렬, 순람표, 정정 가능 집합, 비-복호
Key Words : PIM, AN codes, weight matrix, look-up-table, correctable set, mis-decoding

ABSTRACT

This work proposes a selective AN coding scheme for correcting errors which happen during computations in the process-in-memory devices. AN codes are error-correcting codes which encode the message data by multiplying a positive integer A to them. AN codes correct errors by mapping the division remainders of the erroneous coded data divided by A to the error patterns in the look-up-table. This paper proposes an AN coding scheme which selectively determines different error-correcting capabilities according to the significancy of computational results. At first, the proposed work defines the set of bit-lines with high significance as the correctable set. Then, it identifies conditions which ensure the correction and the mis-decoding detection for the error patterns inside and outside of the correctable set, respectively. The proposed AN coding scheme enables more flexible code design for a given noisy environment compared to the existing static AN coding scheme. Additionally, in terms of the storage capacity, a more efficient decoder implementation is possible compared to the ABN coding scheme which is known suitable for flexible code design.

※ 이 논문은 2023년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임.(NRF-2021R1A2B5B01002204)
• First Author : School of Electrical Engineering, Korea Advanced Institute of Science and Technology, kmi9212@kaist.ac.kr, 학생회원
° Corresponding Author : School of Electrical Engineering, Korea Advanced Institute of Science and Technology, jsha@ee.kaist.ac.kr, 종신회원
논문번호 : 202309-079-A-RN, Received September 13, 2023; Revised October 4, 2023; Accepted October 4, 2023

Ⅰ. 서 론

메모리 내 컴퓨팅 (Process-In-Memory, PIM)은 메모리 내에서 연산 처리의 일부를 수행할 수 있는 기술로 최근 인공지능 기술의 발전에 따른 고속 대용량 처리를 위한 해결책으로 주목받고 있다[1]. 이를 구현하기 위해 저항 변화 메모리 (resistive random access memory, ReRAM)와 같은 저항성 메모리 장치 (memristive device)를 크로스바 어레이 (crossbar array) 형태로 배치하여 다층 신경망 (multi-layer perceptron, MLP), 합성곱 신경망 (convolutional neural network, CNN)과 같은 다양한 인공지능 알고리즘에서 요구하는 행렬 연산을 수행할 수 있다[2]. 그러나, 메모리 셀에서 발생하는 열잡음 (thermal noise), 산탄 잡음 (shot noise), random telegraph noise (RTN), stuck-at-fault와 같은 다양한 오류 요인에 의해 행렬 연산 결과의 신뢰도가 저하되는 문제가 발생한다[3].

오류정정부호는 메모리 셀에 데이터를 부호화하여 저장함으로써 위와 같은 요인들로 인해 발생한 오류를 정정할 수 있는 기술이다[4]. 그러나 PIM 장치에서는 기존의 메모리 시스템과 달리 각 메모리 셀에 저장된 데이터가 아니라 행렬 연산 결과를 읽어온다. 따라서, 이 행렬 연산 결과가 부호어로 표현될 수 있어야 한다는 조건이 PIM을 위한 적절한 부호의 선택에 필요하다. PIM 장치 내부의 행렬 연산 결과는 크로스바 어레이의 각 행에 저장된 데이터의 선형 결합으로 표현할 수 있다. 따라서, 크로스바 어레이의 각 행에 선형성을 가지는 부호의 부호어를 저장하면 행렬 연산 결과 역시 부호어로 표현할 수 있으므로 연산 과정에서 발생한 오류를 정정할 수 있다. 그러나, 저장장치 시스템에서 일반적으로 사용되는 BCH (Bose-Chauduri-Hocquenghem)[5]나 저밀도 패리티 검사 부호 (low-density parity-check codes, LDPC)[6] 부호와 같은 선형 부호들은 유한체 (finite field) 위에서 모듈로 (modulo) 연산을 기반으로 설계되었다. 그러므로 이러한 유한체 위에서 정의된 부호를 사용하면, PIM의 행렬 연산 결과가 부호어가 되기 위해 모듈로 연산이 이루어지므로 연산 결과의 정보가 손실되는 문제가 발생한다.

AN 부호는 정수 형태의 데이터에 A라는 정수를 곱하여 부호화시키는 부호로 부호화된 데이터들을 더하더라도 A의 배수 형태, 즉 선형성이 유지되고 모듈로 연산이 아닌 정수 집합에서 정의된 연산이 이루어지기 때문에 연산 결과의 정보가 손실되지 않는다. 이러한 이유로 AN 부호는 PIM에서 이루어지는 행렬 연산 오류를 정정하는데 적합한 부호이다[7-9]. AN 부호는 오류가 발생한 부호화된 데이터가 A로 나누어떨어지지 않을 때 오류의 발생 여부를 탐지할 수 있다. 또한, 오류가 발생한 부호화 데이터를 A로 나눠서 나올 수 있는 나머지와 오류 패턴 사이의 관계를 정리한 순람표 (look-up table, LUT)를 통해 오류를 정정할 수 있다.

기존의 PIM 설계 방식에 적용하였던 AN 부호화 방식은 크게 static AN[8]과 ABN-X[9]로 구분할 수 있다. 먼저, static AN은 오류 요인이나 저장한 데이터와 상관없이 고정된 형태의 오류 패턴이 발생할 때 고칠 수 있도록 순람표를 설계한 AN 부호 기술이다[8]. ABN-X는 static 방식과 달리 크로스바 어레이에 저장된 데이터의 형태나 RTN과 같은 오류를 발생시키는 요인들을 고려하여 연산 결과에 크게 영향을 미치는 오류 패턴을 선택적으로 고칠 수 있도록 순람표를 설계한 부호이다[9]. 그러나 PIM 구조에서는 행렬 정보를 저장하는 크로스바 어레이마다 저장되는 데이터의 형태가 다르므로 연산 결과에 주로 영향을 미치는 오류 패턴도 달라진다. 따라서 ABN-X를 구현하기 위해서는 각 크로스바 어레이에 해당하는 순람표를 행렬 정보를 저장할 때마다 개별적으로 생성하여 저장해야 한다.

본 논문에서는 static AN 부호의 비탄력적인 설계 방식 및 ABN-X의 낮은 저장공간 효율성을 개선하기 위한 선택적 AN 부호 설계 방식을 제안하였다. 먼저, 정정하기 위한 정확도 범위의 오류 패턴의 집합을 정정 가능 집합으로 정의하고 이 오류 패턴을 고치기 위한 AN 부호 설계 조건을 명시하였다. 또한, 다른 부호어로 복호되는 현상인 비-복호 (mis-decoding) 현상을 감지하기 위한 조건을 수학적으로 정의하고 AN 부호 설계 조건에 추가하였다. 제안한 AN 부호 설계 방식은 오류 환경에 따라 요구되는 오류 정정 능력을 만족시킬 수 있는 A를 선택할 수 있어 탄력적인 설계가 가능하고 비-복호 현상을 방지할 수 있다. 또한, 크로스바 어레이에 저장된 데이터와 무관하게 AN 부호를 설계하므로 모든 크로스바 어레이에 같은 순람표를 사용할 수 있어 저장공간 효율성 문제를 개선할 수 있다.

Ⅱ. 저항성 메모리 기반 행렬 연산 방식

본 논문의 신경망 구조 내의 행렬 연산을 구현하기 위한 저항성 메모리를 기반으로 한 크로스바 어레이

구조는 ISAAC을 참고로 하였으며 그림 1과 같다[10]. 입력 벡터와 가중치 행렬 사이의 행렬 연산을 수행하기 위해서는 먼저, 각 데이터를 해당하는 인가전압 및 셀 상태로 변환시켜 주어야 한다. 예를 들어, (1)의 입력 행벡터와 가중치 행렬 사이의 연산을 비트 수준이 1인 셀로 구성된 크로스바 어레이 구조로 구현한다고 가정해 보자.

$$[1\,1\,0\,1]\times\begin{bmatrix}3&0\\0&3\\3&3\\2&1\end{bmatrix} \tag{1}$$

입력 벡터는 그림 1의 디지털-아날로그 변환기에 의해 데이터에 비례하는 전압으로 변환되어 각 워드 라인(word line, WL)에 인가된다. 가중치 행렬의 원소의 최댓값은 3이므로 각 원소를 비트 수준이 1인 셀 하나에 저장할 수 없다. 이를 해결하기 위해, 비트-슬라이싱 (bit-slicing) 기법을 이용하여 각 행렬 원소를 두 개의 비트로 분할하여 두 개의 셀에 각각 저장한다. 비트-슬라이싱 기법이란 가중치 행렬의 각 원소가 하나의 셀로 표현할 수 없는 경우 이를 워드 라인 내의 여러 개의 연속된 셀에 중요도에 따라 나누어 저장하는 방식이다[10]. 각 비트 라인 (bit line, BL) 하단에서 읽어온 전류의 데이터는 그 비트 라인에 존재하는 저항에 흐르는 전류의 합에 비례한다. 따라서, 아날로그-디지털 변환기의 양자화 간격을 1이 저장된 cell에 1의 데이터에 해당하는 전압을 가해줬을 때 흐르는 전류의 세기로 두고 각 비트라인 하단에 흐르는 전류를 측정하여 양자화시키면 입력 벡터와 해당 비트라인에 존재하는 셀에 저장된 데이터의 내적값을 그림 1과 같이 계산할 수 있다. 그 이후, shift & add unit을 통해 각 비트라인에서 연산된 결과를 비트 중요도를 고려하여 가중치를 두고 결합하여 원하는 행렬 연산의 결과를 얻을 수 있다.

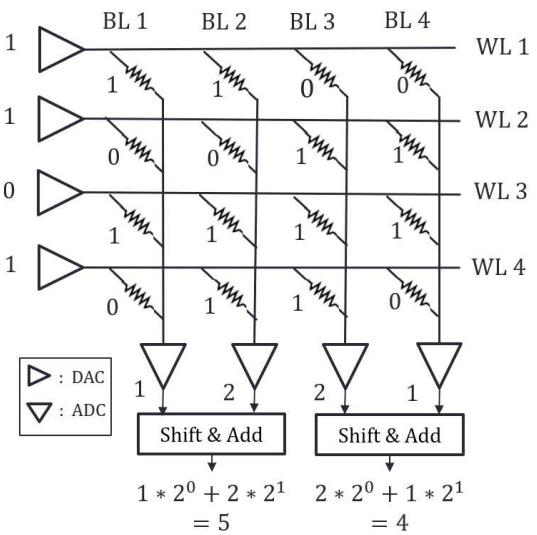

그림 1. 저항성 메모리 기반 행렬 연산 방식
Fig. 1. Matrix computation based on the memristive devices

Ⅲ. 저항 변화 메모리 오류 모델

저항 변화 메모리로 이루어진 크로스바 어레이에서는 열잡음, 산탄 잡음, RTN과 같은 오류 요인들이 행렬 연산 결과에 영향을 미친다. 열잡음의 경우 각 셀에 흐르는 전하의 열 교란 현상 때문에 발생하는 오류 요인이다. 이러한 열잡음은 평균이 0, 분산이 $4K_BTfG$과 같은 가우시안 (Gaussian) 분포를 따르는 전류의 세기로 모델링 가능하다[3]. 산탄 잡음도 동일하게 가우시안 분포를 따르는 전류의 세기로 모델링 가능하고, 평균은 0, 분산은 $2qGVf$와 같다[3,11]. 여기서 K_B는 볼츠만 (Boltzmann) 상수, T는 온도, f는 동작 빈도수, G는 셀의 컨덕턴스, q는 기본 전하량, 그리고 V는 워드 라인에 가해주는 인가전압을 의미한다. RTN은 전류가 셀에 흐를 때 전자가 일시적으로 소자에 갇히는 현상 때문에 발생한다. 이는 셀의 유효 저항값을 감소시키는 효과를 발생시키므로 전압을 가해주었을 때 셀에 흐르는 전류의 세기를 일시적으로 증가시킨다[12,13]. 이 외에도 stuck-at-fault나 IR drop과 같은 다른 오류 요인들도 존재하지만 본 논문에서는 기존의 AN 부호를 PIM에 적용한 결과와[9] 제안한 방식과의 성능 비교를 위해서 열잡음, 산탄 잡음, RTN 만을 고려하여 셀에 흐르는 전류의 오류를 모델링하였다. 이러한 오류 요인들로 인한 전류 잡음은 그림 1과 같은 크로스바 어레이에서 각 비트라인 하단에서 합해진 후 아날로그-디지털 변환기에 의해 양자화되어 정수 형태의 오류로 변환된다.

Ⅳ. AN 부호 설계 방식

4.1 AN 부호의 부호화 및 복호 과정

AN 부호는 정수 형태의 메시지 데이터에 정수 A를 곱하여 부호화하기 때문에, 부호화된 데이터를 더하더라도 A의 배수 형태를 유지하여 선형성을 가지는 부호로 알려져 있다[8]. AN 부호에 의해 부호화된 데이터에 오류가 발생했을 때, 이를 정정하기 위한 복호

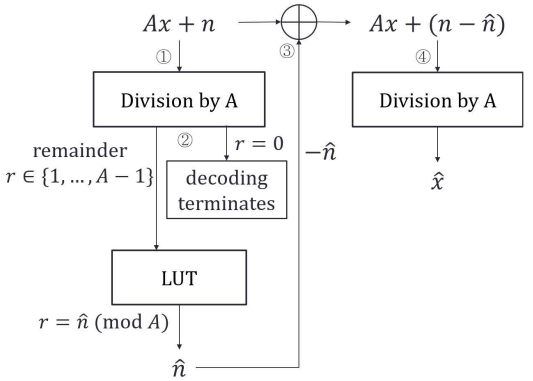

그림 2. AN 부호 복호 과정
Fig. 2. Decoding process of the AN code

과정은 그림 2와 같이 총 4가지 단계로 진행된다. 복호하고자 하는 데이터를 x, 부호화에 사용한 정수를 A, 그리고 발생한 오류를 n이라고 하자. 먼저, 읽어온 데이터 $Ax+n$을 A로 나눠서 얻은 나머지 r을 구한다. r이 0인 경우에는 오류가 발생하지 않았다고 선언하고 복호 과정을 종료한다. 만약, r이 0이 아닌 경우에는 순람표를 통해 예측된 오류 패턴 $\hat{n}$과 대응시킨 후 $Ax+n$에서 대응된 $\hat{n}$을 빼준다. 마지막으로, 본래의 데이터 x를 얻기 위해 최종 결과 $Ax+(n-\hat{n})$을 A로 나누어준다.

4.2 기존 AN 부호화를 통한 행렬 연산 오류 정정

AN 부호화된 가중치 행렬의 각 원소는 한 셀에 저장할 수 있는 비트의 수를 고려하여, 비트-슬라이싱 기법을 이용하여서 한 워드 라인 내의 여러 개의 셀에 나누어 연속적으로 저장한다. 이 때, 각 비트라인에서의 연산 결과의 중요도가 차이가 나기 때문에 그 중요도를 반영한 형태의 오류를 정정할 수 있는 AN 부호의 설계가 필요하다. 가령, static AN 부호의 경우에는 각 비트라인에서 발생한 ± 1 오류 패턴에 중요도를 반영한 형태의 오류를 고칠 수 있도록 순람표를 설계한다[8]. 예를 들어, 가중치 심벌을 $A=19$인 AN 부호로 부호화하여 9비트로 변환하고 비트 수준이 1인 셀에 저장하였다고 가정하자. ± 1 오류 패턴에 중요도를 반영한 $\pm 2^k\ (k\in\{0,1,\dots,8\})$ 오류 패턴은 19로 나누었을 때 나올 수 있는 0이 아닌 모든 나머지와 일대일 대응이 가능하다. 따라서, $A=19$인 static AN 부호는 표 1과 같이 순람표를 설계하여 모든 단일 ± 1 오류 패턴을 정정할 수 있다. 그러나 상대적으로 중요도가 떨어지는 비트라인에서 발생하는 ± 1 오류의 경우 실제 연산 결과에 영향을 미치는 정도가 낮다. 또한, 기존의 static AN 부호 관련 논문에서는 두 개 이상의 비트라인에서 발생하는 ± 1 오류를 고칠 수 있는 부호를 구체적으로 설계하는 방법에 대한 설명이 없다[8].

표 1. 비트 수준이 1인 셀에서 $A=19$인 static AN 부호의 순람표
Table 1. Look-up table for the static AN codes for $A=19$ and the single-bit level cell

r	1	2	3	...	17	18
error	2^0	2^1	-2^4	...	-2^1	-2^0

ABN-X 부호는 이를 해결하기 위해 크로스바 어레이에 저장된 데이터, 비트의 중요도, 그리고 각 비트라인의 오류 발생 확률을 고려하여 다수의 비트라인에서 발생하는 오류 패턴도 정정하도록 설계한 AN 부호 방식이다[9]. ABN-X는 먼저, 단일 및 다수의 비트라인에서 발생하는 오류 패턴들의 연산 결과에 미치는 영향력을 측정할 수 있는 점수 함수 (score function)을 정의하였다. 그리고, 이 점수 함수가 큰 오류 패턴부터 순차적으로 순람표에 추가하여 AN 복호기가 연산 결과에 크게 영향을 미칠 수 있는 오류 패턴을 특정화하여 고칠 수 있도록 하였다.

Static AN 부호와 ABN-X 부호는 모두 발생한 오류 패턴이 다른 오류 패턴으로 고쳐지는 비-복호 현상이 발생할 수 있다. 비-복호 현상이란 발생한 오류 패턴이 A로 나누었을 때 같은 나머지 값을 가지는 순람표에 존재하는 다른 오류 패턴으로 복호되는 현상을 의미한다. 이러한 비-복호 현상은 오류를 정정하지 않았을 때와 비교하여 오히려 연산 오류의 정도를 증가시킬 가능성이 존재한다. 기존의 AN 부호화 방식에서는 비-복호로 인한 오류를 감지하기 위해 정수 A뿐만 아니라 B라는 정수를 곱하여 저장한다[9]. 복호과정에서는 연산 결과를 A로 나눈 나머지로 복호를 진행한 후 결과를 B로 나눈 나머지 값이 0이 나오지 않는 경우 비-복호가 발생했다고 판단한다. 이 경우에는 복호 결과를 원래 오류가 발생했던 연산 결과로 되돌린다.

실제 PIM 구조에서는 인공 신경망의 한 계층의 가중치 행렬의 행 또는 열의 크기가 각각 단위 크로스바 어레이의 행 또는 열의 크기보다 큰 경우가 발생한다. 이 경우에는 계층별 단위 가중치 행렬을 다수의 크로스바 어레이에 나누어 저장한다[10]. 또한, 열잡음 및 RTN이 발생한 셀의 전류의 잡음의 세기는 셀에 저장된 데이터의 값에 따라 다르다. 그러므로 ABN-X의 부호화 방식을 실제 PIM 구조에 구현하기 위해서는 각 크로스바 어레이마다 저장되는 가중치 행렬의 데

이터 특성을 고려하여 순람표도 개별적으로 생성되어야 한다. 또한, 이렇게 생성된 개별적인 순람표를 저장할 수 있는 공간이 필요하다. 그리고 모든 패턴의 비-복호 현상을 항상 B로 나눈 나머지로 감지하는 것은 불가능하므로 평균적으로 연산 오류에 영향을 크게 미치는 비-복호 오류 패턴을 선택적으로 감지할 수 있도록 A를 설정해야 한다.

4.3 제안한 AN 부호화 방식

본 논문에서는 비트라인의 중요도를 고려하여 오류를 정정할 수 있는 비트라인을 미리 설정하고, 그 비트라인 내에서 발생하는 단일 ± 1 오류 및 다중 ± 1 오류를 정정할 수 있는 AN 부호를 설계하였다. 아래에서는 제안한 부호의 설계 조건을 설명하기 위해 사용할 기본적인 용어를 정의하려고 한다. 먼저, 비트라인의 위치를 나타내는 숫자들의 집합에서 중요도가 높은 비트라인을 나타내는 일부 원소들을 택하여 모은 부분집합을 정정 가능 집합 (correctable set) C로 정의한다. 그리고, C의 여집합 즉, 정정 대상이 아닌 비트라인의 위치를 나타내는 숫자들의 집합을 N이라 하자. 그리고 오류 패턴을 나타내는 집합 $E_1^C, E_2^C, E_2^{C,N}, E_1^N, E_2^N$을 다음과 같이 정의한다.

$$E_1^C = \{2^i : i \in C\} \cup \{-2^i : i \in C\} \tag{2}$$

$$\begin{aligned} E_2^C = & \{2^i + 2^j : i \neq j \in C\} \\ & \cup \{2^i - 2^j : i \neq j \in C\} \\ & \cup \{-2^i - 2^j : i \neq j \in C\} \end{aligned} \tag{3}$$

$$\begin{aligned} E_2^{C,N} = & \{2^i + 2^j : i \in C,\ j \in N\} \\ & \cup \{2^i - 2^j : i \in C,\ j \in N\} \\ & \cup \{-2^i + 2^j : i \in C,\ j \in N\} \\ & \cup \{-2^i - 2^j : i \in C,\ j \in N\} \end{aligned} \tag{4}$$

$$E_1^N = \{2^i : i \in N\} \cup \{-2^i : i \in N\} \tag{5}$$

$$\begin{aligned} E_2^N = & \{2^i + 2^j : i \neq j \in N\} \\ & \cup \{2^i - 2^j : i \neq j \in N\} \\ & \cup \{-2^i - 2^j : i \neq j \in N\} \end{aligned} \tag{6}$$

여기서 (2), (5)의 E_1^C, E_1^N은 각각 C와 N내에서 정의된 단일 ± 1 오류 패턴을 나타낸 집합이고, (3), (6)의 E_2^C, E_2^N은 각각 C와 N내에서 정의된 2개의 ± 1 오류 패턴을 나타낸 집합이다. (4)의 $E_2^{C,N}$은 2개의 ± 1 오류 패턴 중에서 하나는 C, 하나는 N내에서 정의된 집합이다.

4.3.1 단일 ± 1 오류 정정 AN 부호

단일 ± 1 오류를 정정할 수 있는 부호의 경우 부호화에 사용하는 A를 선택 및 그에 적합한 순람표를 설계할 때, 다음 두 가지 조건을 만족하도록 한다. 먼저, E_1^C내의 원소들은 모두 고칠 수 있는 오류 패턴이어야 한다. 그렇게 하기 위해서는 E_1^C내의 오류 패턴들 사이에는 복호과정에서 비-복호가 발생하지 않아야 한다. 이를 수식으로 표현하면 아래의 조건 1과 같다.

조건 1. E_1^C내의 원소들은 모듈로 A로 같지 않다.

두 번째로, 비-복호 방지를 위해 사용하는 B가, 정정 대상이 아닌 N의 비트라인에서 발생하는 오류 패턴이 C에서 발생하는 오류 패턴으로 비-복호되는 것을 전부 감지할 수 있어야 한다. 이는 조건 2와 같이 표현할 수 있다.

조건 2. E_1^N의 임의의 원소 e_n에 대해, 만약 $e_n = e_c \pmod A$을 만족시키는 e_c가 E_1^C에 존재하면, $e_n \neq e_c \pmod B$을 만족시켜야 한다.

예를 들어, 16비트의 메시지를 $A=37$, $B=3$으로 부호화하여 생성된 총 23비트를 비트 수준이 1인 셀들에 연달아 저장한다고 가정하자. 정정 가능 집합 C를 $\{6,7, \ldots, 22\}$로 정했을 때, 나올 수 있는 $\pm 2^i$ 오류 패턴들을 $A=37$로 나누면 그 나머지들이 서로 다르므로 첫 번째 조건을 만족한다. 또한, 정정 가능 집합의 여집합 $N=\{0,1,2,3,4,5\}$에서 나올 수 있는 +1 오류 패턴 중 하나인 2^5은 모듈로 A로 같은 오류 패턴이 순람표에 없어서 비-복호가 일어나지 않고, -2^5 역시 대칭성에 의해 모듈로 A로 같은 오류 패턴이 순람표에 존재하지 않는다. 나머지 +1 오류 패턴들의 집합인 $\{2^4, 2^3, 2^2, 2^1, 2^0\}$이 C에서 나올 수 있는 오류 패턴 $\{-2^{22}, -2^{21}, -2^{20}, -2^{19}, -2^{18}\}$과 모듈로 A로 같으나 오류 정정 시 발생하는 비-복호 오류 패턴인 $\{2^4+2^{22}, 2^3+2^{21}, 2^2+2^{20}, 2^1+2^{19}, 2^0+2^{18}\}$가 전부 B로 나누어떨어지지 않기 때문에 이를 감지할 수 있다. -1 오류 패턴인 $\{-2^4, -2^3, -2^2, -2^1, -2^0\}$도 대칭성 때문에 $\{2^{22}, 2^{21}, 2^{20}, 2^{19}, 2^{18}\}$와 모듈로 A로 같으나 나올 수 있는 모든 비-복호 오류 패턴이 전부 B로 나누어떨어지지 않기 때문에 이를 감지할 수 있다.

4.3.2 다중 ±1 오류 정정 AN 부호

다중 ±1 오류 정정 AN 부호의 경우에는 2개의 비트라인에서 발생한 ±1 오류를 고칠 수 있는 부호를 설계하였다. 다중 ±1 오류 정정 AN 부호도 단일 ±1 오류 정정 AN 부호와 마찬가지로 첫 번째 조건으로는, 발생하는 모든 단일 ±1 및 2개의 ±1 오류 패턴은 모듈로 A로 같지 않아야 한다. 2개의 ±1 오류 패턴 중에서 하나는 C, 하나는 N에서 발생한 $E_2^{C,N}$내의 오류 패턴의 경우, C에서 선택한 단일 ±1 오류 패턴의 중요도가 더 높으므로 이 값으로 근사시킬 수 있다. 따라서, $E_2^{C,N}$내의 오류 패턴도 단일 ±1 오류 패턴으로 간주하고 정정 대상으로 포함한다. 예를 들어, 3.1절의 예제에서 2개의 오류 패턴이 2^4+2^{17}으로 발생하였을 때 17번째 비트라인에서 발생한 ±1 오류의 중요도가 더 높기 때문에 2^{17}으로 근사시킬 수 있다. 고쳐야 할 오류 패턴을 모아놓은 집합인 E^C을 $E_1^C \cup E_2^C \cup E_2^{C,N}$로 정의하면 첫 번째 조건은 다음과 같이 표현할 수 있다.

조건 1. E^C내의 원소들은 모듈로 A로 같지 않다.

두 번째 조건도 단일 ±1 오류 정정 AN 부호와 마찬가지로 N의 비트라인에서 발생하는 오류 패턴이 C에서 발생하는 오류 패턴으로 비-복호되는 것을 B로 감지할 수 있어야 한다. N의 비트라인에서 발생하는 오류 패턴의 집합인 E^N을 $E_1^N \cup E_2^N$으로 정의하면 이 조건은 다음과 같이 표현할 수 있다.

조건 2. E^N의 임의의 원소 e_n에 대해, 만약 $e_n = e_c \pmod A$을 만족시키는 e_c가 E^C에 존재하면, $e_n \neq e_c \pmod B$을 만족시켜야 한다.

4.3.3 AN 부호 구성 방법

각 비트라인의 중요도는 하나의 셀에 저장하는 비트의 수 및 부호화된 데이터의 크기에 따라 결정된다. 따라서, 셀 레벨 및 데이터의 크기에 따라 조건 1, 조건2를 만족시키는 A,B 및 정정 가능 집합의 조합이 달라진다. 정해진 데이터의 크기 및 셀 레벨에서 조건 1, 조건2를 만족시키는 A와 B 및 정정 가능 집합을 찾은 경우 조건 1의 오류 패턴 집합 (단일 ±1 오류 정정 부호의 경우 E_1^C, 다중 ±1 오류 정정 부호의 경우 E^C) 내의 원소들을 A로 나눈 나머지와 각각 대응시켜 복호기의 순람표를 형성한다.

±1 오류를 정정하는 데 필요한 A의 크기는 ±1 오류의 개수에 따라 기하급수적으로 증가한다. ($A = O(n^k)$, k: ±1 오류 발생한 비트라인 개수, n: 단위 부호화 가중치 심벌당 필요한 셀의 개수) 메모리 저장공간은 일반적으로 높은 부호율을 가지는 부호를 요구하기 때문에 정정할 수 있는 ±1 오류의 개수에는 한계가 있다. 반면, 오류 발생 확률의 경우에는 ±1 오류 개수가 증가함에 따라 기하급수적으로 감소한다. ($O(p^k)$, k: ±1 오류 발생한 비트라인 개수, p: 단위 비트라인에서 ±1 오류 발생 확률) 따라서, 본 논문에서는 발생 빈도수가 높은 단일 혹은 2개의 비트라인에서 발생한 ±1 오류를 고칠 수 있는 AN 부호만 고려하였다.

V. 시뮬레이션 결과

제안한 AN 부호의 오류 정정 성능을 평가하기 위해 MNIST 데이터 집합의 분류 정확도를 평가 기준으로 사용하였다. 학습 모델은 계층의 개수가 3인 MLP1[14]을 사용하였다. 해당 학습 모델의 입력 유닛의 개수는 MNIST 데이터의 각 샘플의 픽셀 수인 784, 2개의 은닉 계층의 유닛의 개수는 각각 500, 150이고, 출력 유닛의 개수는 숫자를 분류하기 위해 10으로 설정되어 있다. 입력 벡터의 각 원소는 MNIST 데이터가 0과 255 사이에 분포하고 있으므로 8비트로 양자화하였고, 이를 통해 입력 벡터를 8개의 이진 입력 벡터로 나누었다. 나머지 계층에서는 이전 계층에서의 연산 결과 중 중요도가 높은 순으로 16비트의 정보만을 가져와서 행렬 연산에 사용하였다. 각 계층에서의 가중치 행렬의 경우 ISAAC 및 기존의 AN 부호에서 수행한 방식과 같게 16비트로 양자화하였다[10]. 학습된 가중치 행렬은 AN 부호로 부호화하여 저항변화 메모리로 이루어진 크로스바 어레이에 저장하도록 시뮬레이션하였다. 오류 모델링에 사용된 변수들은 표 2의 값들을 사용하였다. 여기서, R_{LO}와 R_{HI}는 각각 셀이 가질 수 있는 최소 저항값과 최대 저항값을 의미한다. $\frac{\Delta R}{R}$은 RTN에 의해서 감소되는 저항값의 변화량과 본래 저항값 사이의 비율을 나타낸다. $\frac{\Delta R}{R}$는 R의 선형함수 형태로 나타나므로 $aR+b$와 같이 표현할 수 있다[3,13]. R_{LO}와 R_{HI}의 $\frac{\Delta R}{R}$값을 이용해서 a와 b를 구하고 R_{LO}와 R_{HI}사이에 있는 레벨의 저

표 2. 오류 모델링 매개 변수
Table 2. Parameters of the error modeling

Parameter		Value
R_{LO}		$2k\Omega$
R_{HI}		$5M\Omega$
V_{app}		$0.3V$
T		$350K$
$\frac{\Delta R}{R}$	R_{LO}	0.028
	R_{HI}	0.5
RTN error probability		0.27

항값의 $\frac{\Delta R}{R}$를 구하여 RTN에 의해 영향받은 셀의 저항값을 구하였다. 비트 수준이 1 또는 2 정도로 낮은 셀에서는 저항 마진 (resistance margin)이 충분히 크기 때문에 분류 정확도에 영향을 줄 정도의 오류가 발생하지 않아 각 셀에 저장할 수 있는 비트의 수는 3으로 설정하였다.

표 3은 static AN 부호, 제안한 AN 부호, 부호화를 사용하지 않은 경우, 그리고 오류가 발생하는 메모리 장치가 아닌 software에서 구한 MNIST 데이터 분류 실패율을 시뮬레이션한 결과의 값들이다. 표 3의 제안한 AN 부호의 경우 $A=395$, $B=3$을 이용하여 각 계층의 가중치 행렬을 부호화하였고 $C=\{6,7,8\}$, $N=\{0,1,2,3,4,5\}$으로 설정하여 2개의 ±1오류까지 정정할 수 있도록 설계하였다. 표 3을 보면 static 부호는 비-복호 현상에 의해 발생하는 오류로 인해 부호를 사용하지 않았을 때 결과보다 더 높은 분류 실패율을 보여준다. 제안한 AN 부호의 경우 부호를 걸지 않았을 때와 static 부호의 결과보다 더 낮은 분류 실패율을 얻으면서 ABN-X의 경우보다도 더 낮은 분류 실패율을 보인다. ABN-X는 제안한 AN 부호화 방식과 동일한 A 및 B를 사용하였으므로 부호화된 데이터를 저장하는데 필요한 셀의 개수는 같다. 그러나, 제안한 AN 부호화 방식은 순람표를 크로스바 어레이마다 생성하여 저장할 필요가 없이 하나의 순람표만 사용하기 때문에 ABN-X 부호화 방식과 비교해서 저장 효율 면에서 더 우수한 측면을 보인다.

표 3. MNIST 데이터 분류 실패율
Table 3. Misclassification rate of the MNIST data

	Misclassification Rate (%)
Static AN	3.5
ABN-X	3.0
Proposed AN	2.6
Uncoded Scheme	3.1
Software	2.1

표 4는 표 2의 매개 변수에서 더 높은 잡음 환경을 만들기 위해 R_{LO}에 대한 $\frac{\Delta R}{R}$ 값을 0.042까지 올리고 RTN 발생 확률을 0.37까지 올렸을 때 구한 MNIST 데이터 분류 실패율이다. 표 4의 제안한 AN 부호의 경우에는 더 강한 오류 정정 성능의 AN 부호를 사용하기 위해 $A=533$로 증가시키고, $C=\{1,2,3,4,5,6,7,8\}$, $N=\{0\}$으로 설정하였다. 표 4의 결과를 통해 제안한 AN 부호는 중요도가 상대적으로 낮은 비트 라인들도 정정 가능 집합에 포함하여 더 강한 오류정정부호를 설계함으로써 더 잡음이 심한 환경에서도 표 3의 결과와 비슷한 정도의 분류 실패율을 얻을 수 있다. 또한 오류에 크게 영향을 미치는 비-복호 패턴을 확정적으로 감지할 수 있으므로 동일한 A 및 B를 사용한 ABN-X 부호보다도 더 낮은 분류 실패율을 보인다.

표 4. MNIST 데이터 분류 실패율
Table 4. Misclassification rate of the MNIST data

	Misclassification Rate (%)
Static AN	4.5
ABN-X	3.5
Proposed	2.5
Uncoded	4.1
Software	2.1

Ⅵ. 결 론

본 논문에서는 저항 변화 메모리로 이루어진 PIM 구조에서 디바이스 환경에서 요구되는 오류 정정 능력에 따라 선택적으로 설계할 수 있는 AN 부호를 제안하였다. 제안한 AN 부호 설계 방식은 한정된 저장 공간에서 연산장치를 구현해야 하는 PIM 구조 내부에서 효율적인 오류 정정 방식으로 이용할 수 있을 것으로 기대된다. 또한, 본 AN 부호 설계 방식은 저항 변화 메모리뿐만 아니라 스핀 전달 토크형 자기 저항 메모리 (STT-MRAM)와 같이 다른 저항성 메모리로 구성된 PIM 구조에도 응용할 수 있다. 제안한 부호화 방식은 현재 저항성 메모리 기반 PIM 구조의 한계점 중 하나인 낮은 신뢰도의 연산 결과를 효율적으로 정정할 수 있는 오류정정부호로 사용될 수 있을 것으로

기대된다.

References

[1] M. S. Koo, “PIM and artificial intelligence,” *Mag. IEEK*, vol. 48, no. 6, pp. 43-50, 2021.

[2] Y. Xi, et al., “In-memory learning with analog resistive switching memory: A review and perspective,” in *Proc. IEEE*, vol. 109, no. 1, pp. 14-42, Jan. 2021. (https://doi.org/10.1109/JPROC.2020.3004543)

[3] Z. He, et al., “Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping,” *2019 56th ACM/IEEE DAC*, pp. 1-6, Las Vegas, NV, USA, Jun. 2019. (https://doi.org/10.1145/3316781.3317870)

[4] D. Niu, Y. Xiao, and Yuan Xie, “Low power memristor-based ReRAM design with error correcting code,” *17th Asia and South Pacific Design Automat. Conf.*, pp. 79-84, Sydney, NSW, Australia, Feb. 2012. (https://doi.org/10.1109/ASPDAC.2012.6165062)

[5] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes,” *Inf. Control*, vol. 3, no. 1, pp. 68-79, Mar. 1960. (https://doi.org/10.1016/S0019-9958(60)90287-4)

[6] R. Gallager, “Low-density parity-check codes,” in *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21-28, Jan. 1962. (http://doi.org/10.1109/TIT.1962.1057683)

[7] U. Schiffel, “Hardware error detection using AN-codes,” Ph.D. dissertation, Technische University Nat Dresden, 2011.

[8] W. W. Peterson and E. J. W. Jr., *Error-Correcting Codes*, 2nd Ed., Cambridge, Massachusetts: MIT Press, 1972.

[9] B. Feinberg, et al,, “Making memristive neural network accelerators reliable,” *2018 IEEE Int. Symp. HPCA*, pp. 52-65, Vienna, Austria, Feb. 2018. (http://doi.org/10.1109/HPCA.2018.00015)

[10] A. Shafiee, et al., “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” *2016 ACM/IEEE 43rd Annual ISCA*, pp. 14-26, Seoul, Korea, Jun. 2016. (https://doi.org/10.1109/ISCA.2016.12.)

[11] L. B. Kish, et al., “Noise in nanotechnology,” *Microelectronics Reliability*, vol. 40, no. 11, pp. 1833-1837, Oct. 2000. (https://doi.org/10.1016/S0026-2714(00)00063-9)

[12] F. M. Puglisi et al., “A complete statistical investigation of RTN in HfO2-based RRAM in high resistive state,” *IEEE Trans. Electron Devices*, vol. 62, no. 8, pp. 2606-2613, Aug. 2015. (https://doi.org/10.1109/TED.2015.2439812)

[13] D. Ielmini, F. Nardi, and C. Cagli, “Resistance-dependent amplitude of random telegraph-signal noise in resistive switching memories,” *Applied Physics Lett.*, vol. 96, no. 5, 2010.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998. (https://doi.org/10.1109/5.726791)

김 명 인 (Myung-in Kim)

2016년 8월 : 한국과학기술원 수리과학과 학사

2018년 8월 : 한국과학기술원 전기 및 전자공학과 석사

2018년 9월~현재 : 한국과학기술원 전기 및 전자공학과 박사과정

<관심분야> 전자공학, 통신공학, 오류정정부호

[ORCID:0000-0002-0720-2671]

하 정 석 (Jeong-seok Ha)

1992년 2월 : 경북대학교 전자공학과 학사
1994년 2월 : 포항공과대학교 전자 및 전기공학과 석사
2003년 12월 : GeorgiaTech electrical engineering 박사
2018년 9월~현재 : KAIST 전기 및 전자공학과 교수

<관심분야> 전자공학, 통신공학, 오류정정부호
[ORCID:0000-0003-1262-151X]