논문 2004-41SD-5-8

재구성 가능한 뉴럴 네트워크 구현을 위한 새로운 저전력 내적연산 프로세서 구조

(The New Architecture of Low Power Inner Product Processor for Reconfigurable Neural Networks)

임 국 찬*, 이 현 수*

(Guk-Chan Lim and Hyon-Soo Lee)

요 약

뉴럴 네트워크는 동작 모드를 학습과 인지 과정으로 구분할 수 있다. 학습은 다양한 입력 패턴에 대하여 학습자가 원하는 결과값을 얻을 때까지 결합계수를 업데이트 하는 과정이고, 인지는 학습을 통해 결정된 결합계수와 입력 패턴과의 연산을 수행하는 과정이다. 기존의 내적연산 프로세서는 처리 속도를 개선하고 하드웨어 복잡도를 줄이는 다양한 구조가 연구되었지만, 뉴럴 네트워크의 학습과 인지모드에 대한 차별화된 구조는 없었다. 이를 위해, 본 논문에서는 재구성 가능한 뉴럴 네트워크 구현을 위한 새로운 저전력 내적연산 프로세서 구조를 제안한다. 제안한 구조는 학습모드에서 기존의 비트-시리얼 내적연산 프로세서와 같이 동작을 하여, 비트-레벨의 빠른 처리 및 하드웨어 구현에 적합하고 높은 수준의 파이프라인 적용이 가능하다는 장점을 가진다. 또한, 인지모드에서는 고정된 결합계수에 따라 연산을 수행할 활성화 유닛을 최소화시킴으로서 전력 소비를 줄일 수 있다. 시뮬레이션 결과 활성화 유닛은 결합계수에 의존적이기는 하지만 50% 내외까지 줄일 수 있음을 확인 하였다.

Abstract

The operation mode of neural network is divided into learning and recognition process. Learning is updating process of weight until neural network archives target result from input pattern. Recognition is arithmetic process of input pattern and weight. Traditional inner product process is focused to improve processing speed and hardware complexity. There is no hardware architecture to distinguish between learning and recognition mode of neural network. In this paper, we propose the new architecture of low power inner product processor for reconfigurable neural network. The proposed architecture is similar with bit-serial inner product processor on learning mode. It have several advantages which are fast processing base on bit-level, suitability of hardware implementation and pipeline architecture to compute data. And proposed architecture minimizes active units and reduces consumption power on recognition mode. Result of simulation shows that active units is depend on bit representation of weight, but we can reduce active units about 50 precent.

Keywords : Neural network, Inner product, Low power

I. 서 론

뉴럴 네트워크(neural network)는 생물학적 모델에 기반을 둔, 상호 연결된 뉴런(neurons)으로 구성되고, 특정한 대수 또는 논리 함수들을 계산할 수 있다. 각 뉴런의 자극되고 억제되는 동작은 시냅스(synapses)로부터 받아들여진 가중된 입력들을 합하는 동작으로 표현되고, 만일 그 합이 고정된 문턱 값보다 크면 활성화되는 동작 특성을 갖는다. 이러한 뉴럴 네트워크는 패턴 인식, 패턴 매칭, 분류작업, 최적화뿐만 아니라 다양한 신호처리 응용분야에서 새로운 해법을 제공한다.

뉴럴 네트워크는 맥컬리(McCulloch)와 피치(Pitts)에 의해서 1943년에 소개된 이래로 다양한 모델과 구현 방

* 정회원, 경희대학교 컴퓨터공학과
(Dept. of Computer Engineering, Kyunghee Univ.)
접수일자: 2003년2월3일, 수정완료일: 2004년4월17일

법이 연구되었다[1,2]. 최초의 뉴럴 네트워크 구현은 아날로그 기술에 기초하여 속도가 빠르고 통합이 가능하다는 장점을 가지지만, 정밀도(precision) 및 민감도(sen-sibility)의 관점에서 잡음 및 크로스토크(crosstalk)에 매우 약하다는 단점을 갖는다. 반면, 디지털 구현 방법은 뉴런을 임의의 정밀도로 표현이 가능하고, 잡음에 대한 강한 면역성(immunity) 및 다른 시스템 구성요소들에 대한 용이한 인터페이스(interface) 기능을 제공하기 때문에 많은 연구가 활발하게 진행되고 있다.

디지털 뉴럴 네트워크를 구현하는 방법에는 크게 범용 컴퓨터를 이용한 방법과 전용 프로세서를 구현하는 방법으로 나눌 수 있다. 범용 컴퓨터를 이용한 방법은 다양한 뉴럴 네트워크를 프로그램으로 모델링 하여 학습과 인지과정을 쉽게 다룰 수 있지만, 뉴럴 네트워크 고유의 병렬성을 다루지 못할 뿐만 아니라 실시간 응용에 요구되는 성능을 제공하기 위해서는 많은 자원이 요구된다. 이를 위해, 명령어 수준(instruction level)에서의 파이프라인(pipeline) 기법을 이용한 다양한 방법이 제안되었으나, 폰-뉴먼(Von-Neumann) 구조가 갖는 메모리와 프로세서간의 데이터의 병목현상(bottleneck)과 병렬성의 부재는 여전히 존재한다. 따라서 뉴럴 네트워크 자체의 병렬성과 복잡한 계산과정을 빠르게 처리하기 위해서는 전용 프로세서 구조를 갖고 실리콘 상에서 구현해야만 이러한 성능한계를 극복할 수 있다[3].

디지털 하드웨어 구현 방법에는 각각의 뉴런 자체에 프로세서를 두거나, 빠른 처리를 갖는 전용 ALU(Ari-thmetic Logic Unit)을 갖는 구조[4,5], 시스톨릭 구조(systolic architecture)를 응용하여 병렬성을 최대한 이용하는 방법[6,7] 및 빈번한 데이터 계산에 소요되는 자원을 줄이기 위해 메모리에 추가 로직을 두어 구현 하는 방법(computational RAM, intelligent RAM)[8] 등 다양한 방법이 있다.

본 논문에서는 뉴럴 네트워크 구현에 필요한 막대한 산술 연산을 효율적으로 처리하기 위하여, 비트-레벨(bit-leale)에서 내적연산을 수행하는 프로세서 구조를 제안한다. 제안한 내적연산 프로세서는 각 비트를 처리하는 기본 덧셈 유닛(units)이 트리(tree) 형태로 배열되고, 뉴럴 네트워크의 학습(learning)과 인지(recognition) 모드에 따라 각각 다른 동작을 수행하는 특징을 갖는다. 학습모드에서는 모든 유닛이 활성화 되어 내적연산을 수행하고, 인지모드에서는 고정된 결합계수의 비트 값에 따라서 활성화 유닛(active unit)과 비활성화 유닛(non-active unit)으로 구분되어 동작한다. 시뮬레이션 결과, 제안한 내적연산 프로세서는 뉴럴 네트워크의 인지 동작 모드에서 소비되는 전력을 줄일 수 있고, 비트-레벨의 빠른 처리 속도의 장점을 갖으며 특히, 재학습을 통하여 결합계수의 업데이트(update)가 가능하므로 재구성 가능한 뉴럴 네트워크 응용에 적합하다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 뉴럴 네트워크의 구조 및 이를 구현하기 위한 기존의 내적연산 프로세서에 대하여 살펴보고, Ⅲ장에서는 본 논문에서 제안한 새로운 내적연산 프로세서의 구조에 대하여 설명한다. Ⅳ장에서는 시뮬레이션 결과에 대하여 기술하고 Ⅴ장에서 결론을 맺는다.

Ⅱ. 뉴럴 네트워크와 기존 내적연산 프로세서의 구조

1. 뉴럴 네트워크 구조

그림 1은 패턴 인식(pattern recognition)에 응용되는 다층 뉴럴 네트워크(multi-layer neural network)의 기본 구조이다. 입력층과 출력층 및 하나 이상의 중간층으로 구성되고, 각 층내의 연결은 출력층에서 입력층으로 직접적인 연결이 존재하지 않는 전방향(feedforward) 네트워크이다. 중간층의 수가 증가할수록 출력값의 특성은 더욱 고급화 되며, 일반적으로 각 뉴런 사이는 완전연결(full-connection)되어 있다.

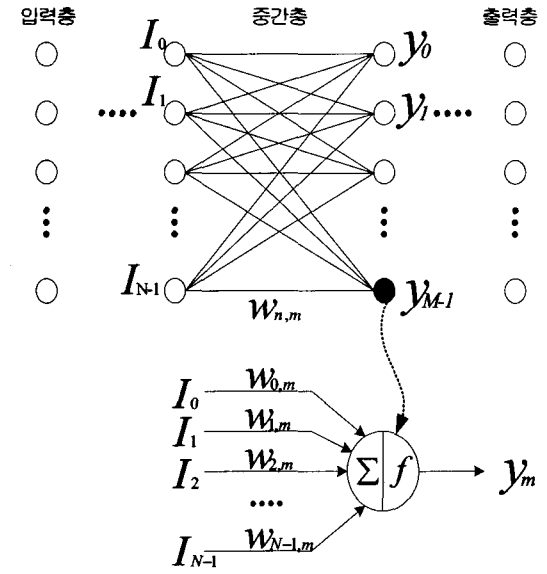


그림 1. 다층 뉴럴 네트워크의 구조
Fig. 1. The basic architecture of multi-layer neural network.

식 (1)은 그림 1에서 특정 뉴런의 산술연산과정을 나타낸 수식이다. 여기서, I_n은 뉴런의 입력이고, $w_{n,m}$은 각 입력에 대한 결합하중으로 뉴런과 뉴런사이의 시냅스(synapse) 연결을 나타낸다. n과 m은 각각 입력과 출력 뉴런을 나타내는 인덱스이다. O_m는 입력의 총합이고 f는 응답함수로 계단형 함수나 비선형(nonlinear) 함수 또는 시그모이드(sigmoid) 비선형 활성함수 등이 사용된다. y_m은 f에 의해 변형되어 나오는 뉴런의 출력이다[1,2].

$$y_m = f(O_m) = f(\sum_{n=0}^{N-1} I_n \cdot w_{n,m}) \quad (1)$$

다층 뉴럴 네트워크는 동작과정을 학습과 인지모드로 구분 할 수 있다. 학습모드는 뉴럴 네트워크가 입력된 벡터(vector)에 대하여 학습자가 원하는 결과를 출력하기 위하여 뉴런간의 결합계수를 조정하는 과정으로 대표적인 예로 백-프로퍼게이션(back-propagation) 학습 알고리즘이 있다. 인지모드는 학습이 완료된 후, 주어진 입력 벡터에 대하여 뉴런간의 학습된 결합계수에 의해 출력 벡터를 생성하는 과정이다.

뉴럴 네트워크에서는 학습뿐만 아니라 인지모드에서도 각 뉴런 마다 식 (1)과 같은 계산과정이 필요하기 때문에, 뉴런의 수가 증가할수록 복잡도는 지수적으로 증가한다. 따라서 적절한 산술연산 구조의 설계가 필수적이며, 뉴럴 네트워크의 동작 특성을 반영할 수 있도록 학습과 인지모드에 따른 차별화된 동작이 가능한 구조가 필요하다.

2. 기존의 내적연산 프로세서

다수의 곱셈과 이들의 합으로 구성되는 내적연산은 하드웨로 구현하는 경우, 곱셈에 대한 복잡도가 합에 대한 복잡도 보다 훨씬 크기 때문에, 곱셈연산에 대한 효율적인 하드웨어 구현 및 알고리즘 개발이 내적연산의 효율을 좌우하게 된다. 효율적인 곱셈에 대한 구조는 다양하게 연구되었으며, 크게 아래와 같이 네 가지 방법으로 분류할 수 있다[9].

- Traditional method (shift-adder architecture, Booth's algorithm, Wallace tree)
- Cellular array method
- Specialized VLSI array (systolic array, bit-serial multiplier, distributed arithmetic)
- Non-traditional method (canonical signed digital number system, logarithmic number system, residue number system)

기존의 내적연산 프로세서는 곱셈 처리 속도를 개선하거나 하드웨어 복잡도를 줄여 구현에 소요되는 크기를 줄이는데 초점이 맞춰져 있다. 일반적으로 곱셈의 구현은 프로세싱 속도와 하드웨어 복잡도가 반비례적인 상관관계가 있고, 프로세싱 단위는 비트(bit) 또는 워드(word) 단위로 곱셈 구현 방법은 매우 다양하다. 이런 다양한 곱셈기 구조는 두 입력 값에 대한 곱셈 프로세싱을 효율적으로 처리하는 것에만 초점이 맞춰져 있기 때문에 뉴럴 네트워크의 동작 특성을 완벽하게 반영하지 못한다. 즉, 기존의 내적연산 프로세서는 뉴럴 네트워크의 학습과 인지모드에 대한 동작 특성의 구분 없이 동일한 프로세싱 절차를 사용한다.

3. 기존의 비트-시리얼 내적연산 프로세서 구조

기존의 비트-시리얼(bit-serial) 내적연산 프로세서는 데이터가 비트-레벨로 순차적으로 입력되기 때문에 하드웨어 구현에 적합하고, 내부 인터페이스의 전환이 용이하며, 높은 수준의 파이프라인(pipeline) 설계가 가능하다. 비트-시리얼 내적연산 프로세서에서 가장 기본적인 계산 유닛은 BSA(Bit-Serial Adder)로, 하나의 FA(Full-Adder)와 FF(Flip- Flop)으로 구성된다.

BSA에 대한 구조가 그림 2에 나타나 있다. 여기서, 'In0'과 'In1'은 외부 입력이고, 'CLK'는 클럭(clock)을 나타내며, 'Sum'은 FA의 합을 나타낸다. 'En'은 인에이블(enable)로 하이 액티브(high active) 단자이다.

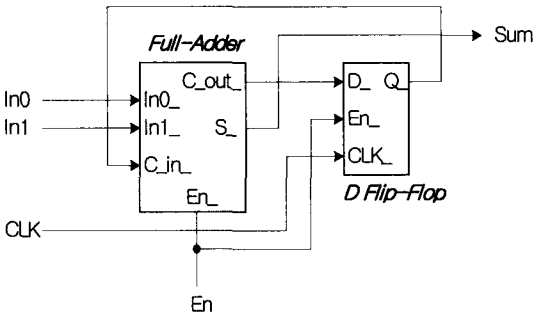


그림 2. 기존의 BSA의 구조
Fig. 2. An architecture of Bit-Serial Adder.

Ⅲ. 새로운 내적연산 프로세서 구조의 제안

본 장에서는 기존의 비트-시리얼 내적연산 프로세서와 비슷한 수준의 처리 속도 및 복잡도를 갖지만, 뉴럴 네트워크의 동작 모드에 따라서 활성화되는 BSA의 수를 줄임으로서 소비 전력을 줄일 수 있는 새로운 내적연산 프로세서 구조에 대하여 설명한다.

1. 내적연산의 비트-레벨 표현

식 (1)에서 뉴럴 네트워크의 출력함수를 제외한 내적연산은 식 (2)과 같이 표현된다.

$$O_m = \sum_{n=0}^{N-1} I_n \cdot w_{n,m} \tag{2}$$

결합계수와 입력값을 각각 비트-레벨로 확장하면 식 (3)과 식 (4)와 같다.

$$I_n = \sum_{l=0}^{L-1} I_{n,l} \cdot 2^l \tag{3}$$

$$w_{n,m} = \sum_{t=0}^{T-1} w_{n,m,t} \cdot 2^t \tag{4}$$

여기서 L과 T는 각각 입력값(I_n)과 결합계수($w_{n,m}$)의 비트 폭(bit width)을 나타낸다. 식 (3)과 식 (4)을 이용하여 식 (2)를 비트-레벨로 확장하면, 식 (5)와 같이 표현된다.

$$O_m = \sum_{l=0}^{L-1}\left(\sum_{t=0}^{T-1}\left(\sum_{n=0}^{N-1}(I_{n,l} \cdot w_{n,m,t})2^t\right)2^l\right. \tag{5}$$

2. NBSA 구조 설계

본 논문에서는 뉴럴 네트워크의 동작 모드에 따른 소비전력을 줄이기 위하여 BSA을 변형한 NBSA(New Bit-Serial Adder) 구조를 제안한다.

그림 3은 FA의 구조이다. 두 개의 반가산기(Half-Adder)와 하나의 OR 게이트로 구성된다. 'In0', 'In1', 'C_in'은 각각 FA의 입력을 나타내고, 'C_out'과 'S'는 각각 캐리(carry)와 합(sum)을 나타낸다.

그림 3에서 'Half-Adder_1'의 XOR 게이트를 하드웨어 구조의 변화 없이 추출하여, 'Modified FA'를 만든다. 그림 4는 'Modified FA'의 구조와 블록도를 나타낸다. 'Modified FA' 는 기존의 FA에서 XOR 게이트 하나가 제외되고, FA 모듈 외부에서 동작 하는 구조이다.

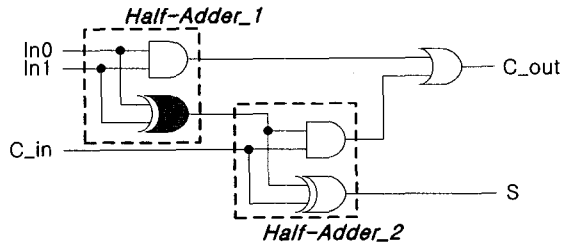


그림 3. FA의 구조
Fig. 3. An architecture of Full-Adder.

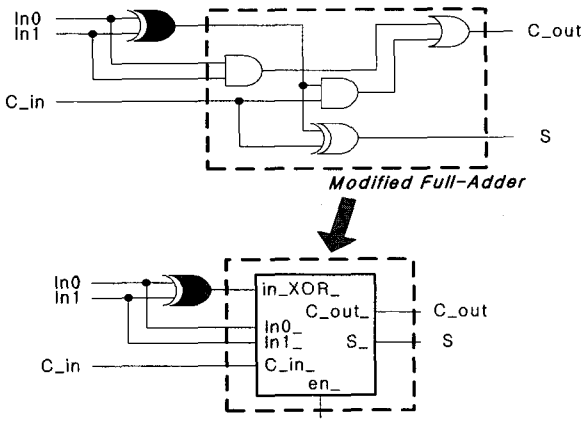


그림 4. 'Modified FA'의 구조와 블록도
Fig. 4. An architecture and block diagram of 'Modified Full-Adder'.

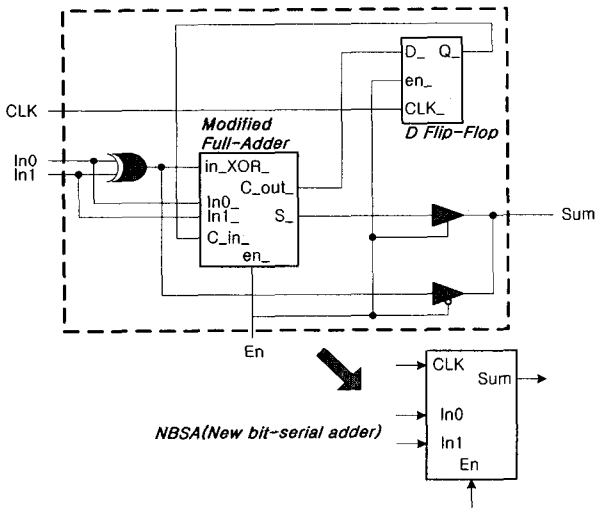


그림 5. NBSA의 구조와 블록도
Fig. 5. The architecture and block diagram of new bit-serial adder.

표 1. 'En'에 따른 NBSA의 출력값 및 동작 게이트
Table. 1. Output of 'Sum' and operating gate for NBSA depend on 'En' state.

'high'	normal operation	all gates
'low'	in0 ⊕ in1	XOR
'Z'	'Z'	-

그림 5는 NBSA의 구조와 블록도를 나타낸다. NBSA는 출력 'Sum'을 제어하기 위하여 두 개의 출력 버퍼가 추가된다. 'En'의 상태에 따른 NBSA의 출력값(Output of 'Sum') 및 동작 게이트(Operating gates)는 표 1과 같이 세 가지 상태로 나눌 수 있다. 여기서 'Z'는 하이 임피던스(high impedance)상태 이다. 'En'이 'high' 상태에서는 NBSA 내부의 모든 게이트가 활성화되어 출력값인, 'Sum'을 계산하고, 'En'이 'low' 상태에서는 XOR 게이트만을 활성화 시켜 'Sum'을 계산한다.

3. 새로운 내적연산 프로세서의 구조

식 (5)은 I와 w의 비트 곱셈연산의 집합으로 생각할 수 있으므로, 그림 6과 같이 표현할 수 있다. '$I_{n,l} \cdot w_{n,m,t}$'을 중심으로 각각의 비트 인덱스(l, n, t)로 방향을 가진다. 이러한 비트 곱셈은 $(L-1) \cdot (N-1) \cdot (T-1)$만큼 존재하며, 이들을 모두 합한 값이 식 (5)의 결과값, O_m이 된다. 여기서, 가중치를 갖는 인덱스는 l과 t이므로 이들에 대한 덧셈은 2^l과 2^t의 가중치를 각각 고려한 연산이 이뤄져야 한다.

그림 7은 그림 6에 표현된 비트 곱셈에 대하여 덧셈연산을 수행하기 위한 구조이다. 그림 7의 (a)는 n-방향 연산 구조로 가중치가 없는 N개의 비트 곱셈을 AND 게이트로 구하고, 이들에 대한 합은 BSA를 트리(tree)형태로 배열하여 결과값을 얻는다. BSA의 FA에서 발생한 캐리는 FF에 저장된 후, 다음 클럭에 연산이 수행된다. 이때, 모든 BSA의 'En'은 '1'인, 액티브(active) 상태여야 한다. 그림 7의 (b)는 t -방향 연산 구조이다. t는 2^t의 가중치를 가짐으로 (a)에서 계산된 결과값이 가중치에 따라 비트 폭이 T인 워드를 생성한다. 그림 7의 (c)는 l-방향 연산 구조로 지연(delay)요소를 갖고, 클럭에 의해 동기화 되어 (b)의 결과 값이 입력된다. 이때, l의 가중치인 2^l을 반영하기 위하여, 이전의 결과값은 한 비트가 우측으로 시프트(shift)된 후 덧셈이 이뤄지게 된다.

그림 7의 (a)에서 하나의 BSA의 입력은 '$I_{n-1,l} \cdot w_{n-1,m,t}$'와 '$I_{n,l} \cdot w_{n,m,t}$'로 표현할 수 있다.

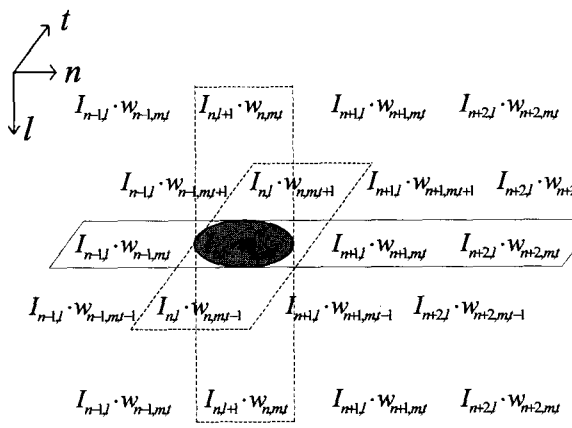
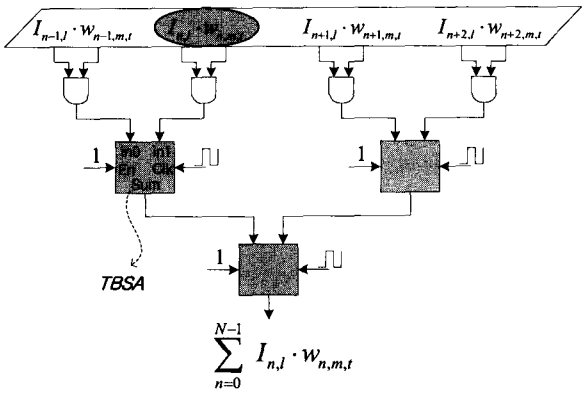
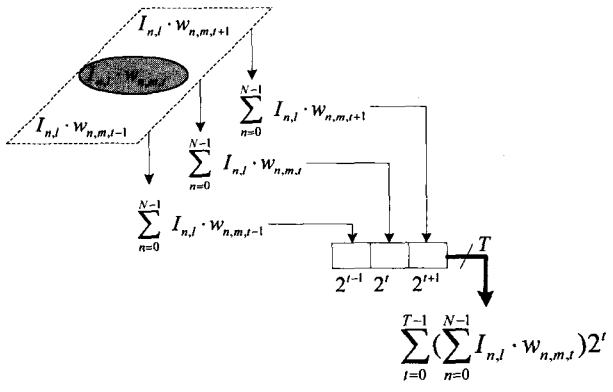


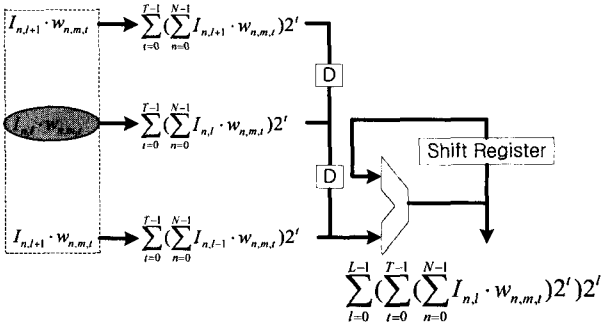
그림 6. 내적연산의 비트-레벨 표현
Fig. 6. The bit-level representation of inner product.



(a) n-방향 연산 구조



(b) t-방향 연산 구조



(c) l-방향 연산 구조

그림 7. 내적연산을 위한 프로세서의 구조
Fig. 7. The processor architecture for inner product.

여기서, l와 w의 상태에 대한 BSA 입력 상태는 표 2와 같다. 표 2에서 두 결합계수($w_{n-1,m,t}$, $w_{n,m,t}$)가 모두 '1'일 때만, BSA의 두 입력이 모두 '1'이 되는 경우가 발생하고, 그 외 조건에서는 BSA의 두 입력이 모두 '1'인 상태가 발생하지 않는다. 즉, BSA에서 캐리가 발생하는 경우는 $w_{n,m,t}$와 $w_{n,m,t+1}$ 모두 '1'일 때만 발생한다. 그 외의 경우에는 캐리가 발생하지 않으므로 FF을 활성화 시킬 필요 없이 최소 게이트만을 활성화시켜 연산을 수행할 수 있다.

뉴럴 네트워크가 학습모드일 때는 결합계수가 변하지만, 인지모드에서는 결합계수가 상수로 고정된다. 따라서 인지과정에서 고정된 결합계수에 따라 BSA의 내부 FF 및 불필요한 게이트의 동작을 최소화시키면 소비되는 전력을 줄일 수 있다. 이를 위하여, 그림 7의 (a)에서 BSA를 NBSA로 바꾸면, 그림 8과 같이 나타낼 수 있다. 그림 8은 NBSA를 이용한 내적연산 프로세서의 구조이다. 여기서, 'En'단자는 두 결합계수의 AND 게이트의 출력으로 묶어, 표 1과 같이 'En'의 상태에 따라 NBSA의 동작을 제어한다. 또한 OR 게이트의 출력은 아래의 AND 게이트에 입력되어, 다음 단의 NBSA의 'En'을 제어한다.

임의의 연산과정을 예로 살펴보자. n이 '0'부터 N까지 일때, $I_{n,l}$, $I_{n,l+1}$, $w_{n,m,t}$을 각각 "10011110", "00010110", "11011000"으로 가정하고, $I_{n,l} \cdot w_{n,m,t}$와 $I_{n,l+1} \cdot w_{n,m,t}$의 관계를 살펴본다. 그림 9는 BSA를 이용한 경우를 나타낸 그림으로 'C'(carry)와 'S'(Sum)의 상태를 나타낸다. 이때, 모든 BSA는 연산을 위해 활성화되어야 한다. 회색은 'En'이 '1'로 되어 해당 BSA가 활성화되었음을 의미한다.

반면에, 그림 10은 NBSA를 이용한 경우의 활성화되는 경우와 비활성화 되는 경우를 나타낸 그림이다. 활성화되는 경우는 'En'이 '1'인 상태로 덧셈기 내부의 전체 게이트가 활성화되었음을 의미하고, 비활성화 되는 경우에는 'En'이 '0'인 경우로 NBSA의 XOR 게이트 부분만 활성화됨을 의미한다. 이때, 'En'은 그림 7과 같이 결합계수(w)의 비트표현에 대한 AND와 OR 게이트 조합으로 결정된다. 비활성화된 NBSA의 'C'는 출력이 없으므로 하이임피던스 상태인, 'Z'가 된다.

그림 9와 그림 10을 비교해보면 각 유닛의 'S'의 결과값은 동일하다. 또한, 'C'의 결과값이 '1'인 곳의 위치가 모두 동일하고, 그림 9에서 최종 연산결과에 영향을

표 2.. l와 w 상태에 따른 BSA의 입력값(in0, in1)
Table 2.. Input data of BSA depend on l and w.

	(0,0)	(0,0)	(0,0)	(0,1)
	(0,0)	(0,1)	(0,0)	(0,1)
	(0,0)	(0,0)	(1,0)	(1,0)
	(0,0)	(0,1)	(1,0)	(1,1)

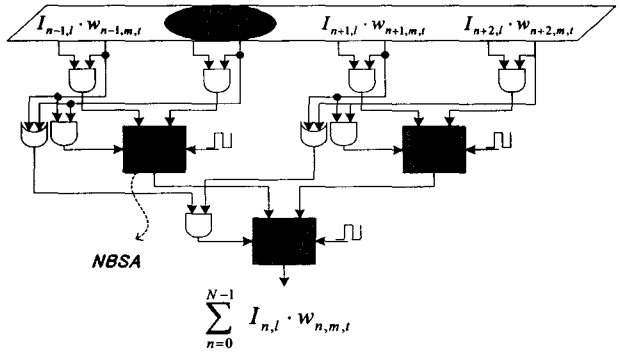


그림 8. NBSA를 이용한 내적연산의 구조
Fig. 8. The architecture of inner product using New Bit-Serial Adder.

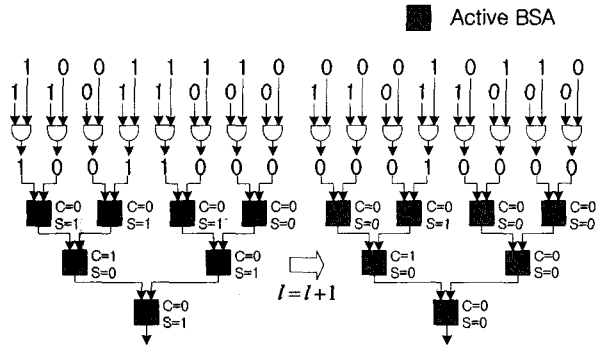


그림 9. 기존의 BSA를 이용한 내적연산 처리과정 예
Fig. 9. An example of inner product processing usage traditional Bit-Serial Adder.

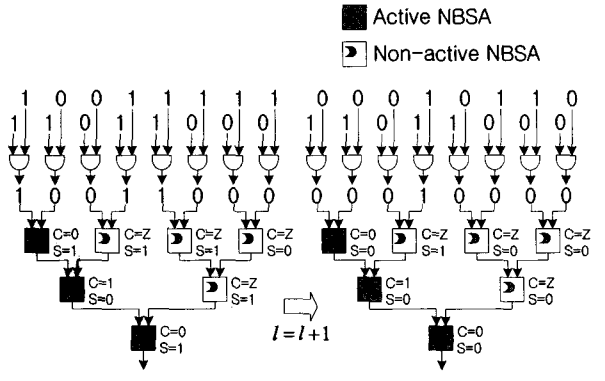


그림 10. NBSA을 이용한 내적연산 처리과정의 예
Fig. 10. An example of inner product processing using New Bit-Serial Adder.

주지 않는 부분의 '0'만 그림 10에서 'Z'가 됨을 알 수 있다. 제안된 내적연산 프로세서의 구조에서 활성화되는 전체 게이트의 수는 $w_{n,m,l}$에 따라 제어될 수 있으므로, 전체 소비되는 전력을 줄일 수 있다.

뉴럴 네트워크의 학습모드에서는 I와 w 모두 변수이므로, 그림 10과 같은 구조에서는 매번 w을 로드(load)해야 하고, 각 덧셈기의 'En' 상태를 빈번하게 업데이트해야 하므로, 오히려 동작속도가 느려질 수 있다. 따라서 학습모드에서는 모든 NBSA의 상태를 활성화 시킨 상태에서 내적연산을 처리한다. 뉴럴 네트워크의 인지모드에서는 w가 상수로 고정되므로 그림 10과 같은 구조로 동작시키면, 기존 구조와 대비하여 성능의 저하 없이, 소비되는 전력을 효율적으로 줄일 수 있다.

Ⅳ. 시뮬레이션 결과

본 논문에서는 두 입력 데이터, I와 w의 비트 폭은 16비트로 가정하고($L, T=16$), 데이터 표현은 부호가 없는 이진수로 하였다. 시뮬레이션을 위한 환경은 아래와 같다.

- Design : Renoir(2000.3)
- Simulation : ModelSim SE/EE PLUS E5.4d
- Synthesis : LeonardoSpectrum(v20001a2.75)
- Library : FPGA Xilinx XC9500

식 (2)에서 $N=16$으로 가정하고, 시뮬레이션을 위한 수식은 식 (6)과 같다. 식 (6)에서 내적연산 결과 값은 '3030522423'임을 알 수 있다.

$$\sum_{n=0}^{N-1} I_n \cdot w_{n,m} = \left\{ \begin{array}{ll} w_{n,m} & I_n \\ 32783 & \cdot\ 783 \\ 13107 & \cdot\ 3359 \\ 12996 & \cdot\ 3133 \\ 47163 & \cdot\ 10482 \\ 1748 & \cdot\ 16370 \\ 39253 & \cdot\ 883 \\ 10196 & \cdot\ 2795 \\ 31 & \cdot\ 3764 \\ 51146 & \cdot\ 5201 \\ 25627 & \cdot\ 15 \\ 25816 & \cdot\ 4522 \\ 32646 & \cdot\ 4526 \\ 21845 & \cdot\ 2730 \\ 6485 & \cdot\ 898 \\ 6495 & \cdot\ 4495 \\ 65280 & \cdot\ 26169 \end{array} \right\} = 3030522423 \quad (6)$$

그림 11은 시뮬레이션 파형이다. I_n은 클럭에 동기화되어 $I_{n,l}$, $I_{n,l+1}$, $I_{n,l+2}$의 순으로 입력되고, $w_{n,m}$와의 연산이 수행되어 결과 값이 출력되며, 이에 대해 시프트 덧셈을 수행하면 원하는 최종 결과 값을 얻을 수 있

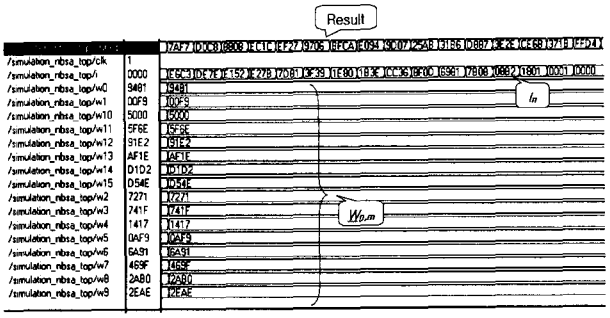


그림 11. 시뮬레이션 파형
Fig. 11. Waveform of simulation result.

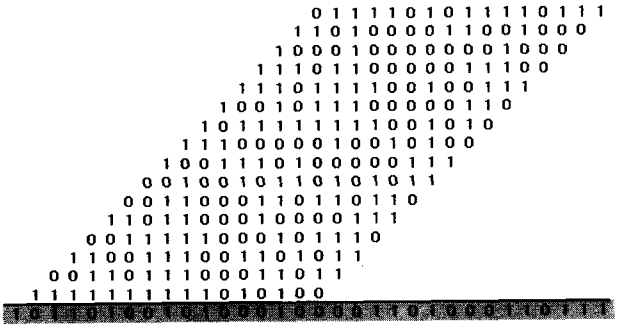
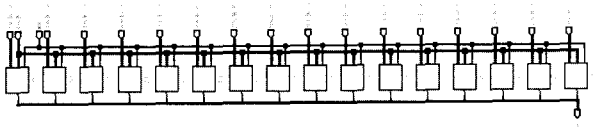
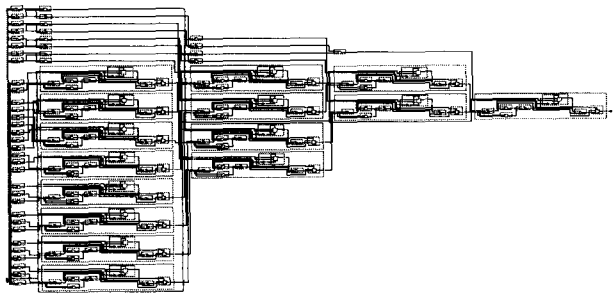


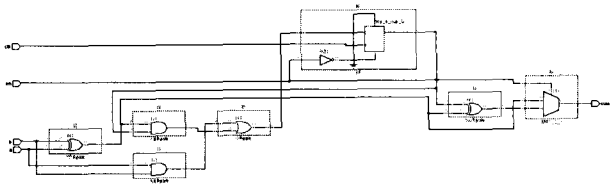
그림 12. 그림 11 출력의 시프트 덧셈 결과
Fig. 12. Result of shift adding from output of Fig 11.



(a) TOP view



(b) TREE view



(c) NBSA view

그림 13. 합성 그림
Fig. 13. Schematic of synthesis.

다. 그림 12는 그림 11의 출력인 'Result'에 대하여 시프트 덧셈을 수행한 결과 값으로 식 (6)의 계산 결과인 '3030522423'이 됨을 알 수 있다.

그림 13은 제안한 구조의 합성 결과를 나타낸다. 여기서, 시프트 덧셈기는 기존과 동일하므로, 합성결과에서 제외시킨다. 그림 13의 (a) TOP에서 $N=16$이므로 16개의 TREE로 구성되고, 각 (b) TREE는 $T-1$개의 NBSA와 인에이블 단자를 제어할 추가 로직을 갖는다. (c) NBSA는 그림 5와 같이 기존의 BSA에 출력 버퍼가 추가된다.

표3은 BSA와 NBSA의 합성결과이다. FF는 두 구조 모두 '1'개를 사용하고, NBSA에 추가된 출력버퍼 때문에 BSA보다 합성에 필요한 게이트가 조금 증가한다.

표 4은 BSA와 NBSA로 내적연산을 구현한 합성 결과

표 3. BSA와 NBSA의 합성 결과 비교
Table 3. Synthesis result comparison of BSA and NBSA architecture.

	1	*1*	0
	16	*19*	18.8
	133.3MHz	*133.3MHz*	0

표 4. BSA와 NBSA로 구현한 내적연산 합성 결과 비교
Table 4. Synthesis result comparison of inner product using BSA and NBSA architecture.

	240	*240*	0
	4113	*5712*	38.8
	44.4MHz	*39.2MHz*	-11.7

표 5. 뉴럴 네트워크의 동작 모드에 따른 활성화되는 BSA와 NBSA의 비교
Table 5. Comparison of active BSA and NBSA depend on neural network processing.

	240	0	240	0
	240	*0*	*104*	*136*

를 나타낸다. FF의 수는 $N\cdot(T-1)$개인 240개가 필요하다. 또한, NBSA가 BSA보다 더 많은 게이트가 필요한 이유는 출력버퍼가 추가되고, NBSA의 인에이블 단자를 제어하기 위한 추가 로직이 필요하기 때문이다. 게이트 증가율(increase rate)이 38.8%로 높은 이유는 FF와 시프트 덧셈기를 제외한 순수 게이트 비교이기 때문이고, 전체를 기준으로 한 게이트 증가율은 이보다 훨씬 작아진다. NBSA에서 동작속도가 감소한 이유는 NBSA의 인에블 단자의 제어를 위한 AND와 OR 게이트 조합으로 된 추가 로직의 지연시간 때문이다.

표 5는 뉴럴 네트워크의 동작 모드에 따른 활성화 또는 비활성화 되는 BSA와 NBSA의 수를 비교한 표이다. 표 5에서 알 수 있듯이, 제안한 NBSA는 뉴럴 네트워크의 동작 모드에 따라 활성화 되는 요소가 다름을 알 수 있다. 학습모드에서 NBSA는 기존의 BSA와 같이 동작이 되지만, 인지모드에서는 훨씬 작은 NBSA만이 활성화됨을 알 수 있다. 인지모드에서 활성화 되는 NBSA의 수는 결합계수인 w에 비트 표현에 의존적이지만 대부분 50% 내외이다.

제안한 NBSA는 BSA와 비교하여 전체 면적은 증가하고, 동작속도는 줄어들지만, 뉴럴 네트워크의 인지모드에서 활성화 되는 NBSA을 50%이상 줄일 수 있다. 뉴럴 네트워크가 학습은 적절한 결합계수를 얻을 때까지 초기에만 수행되고, 인지모드에서 주로 동작한다는 점을 감안한다면, 전체 소비전력은 50%이상의 효율을 얻을 수 있다.

내적연산 프로세서는 목적 및 응용범위에 따라 매우 광범위하기 때문에 직접적인 분류는 어렵지만, 입력데이터의 형태에 따라 표 5와 같이 구분할 수 있다.

두개의 변수를 연산하는 프로세서의 경우는 하드웨어 복잡도가 비교적 높고 동작속도가 느리며, 소비전력 또한 비교적 높다. 반면에, 디지털 신호처리 분야에서 데이터의 변환(transform) 알고리즘에 사용되는 내적연산 프로세서는 두 입력데이터 중에서 하나가 고정된 상수를 가지므로, 이러한 특성을 이용하여 하드웨어 구조의 간소화, 높은 동

표 6. 입력데이터의 형태에 따른 내적연산 프로세서의 분류
Table 6. Classification of inner product process depend on input data type.

변수, 변수	가능	없음	보통	보통	높음
상수, 변수	불가능	없음	낮음	높음	보통
학습/인지과정에 따라 다름 (제안한 구조)	*가능*	*있음*	*보통*	*보통*	*보통*

작속도, 저전력 응용이 가능한 구조를 설계할 수 있다.[10-12]

본 논문에서는 뉴럴 네트워크의 재구성을 가능하도록 하기 위하여, 학습모드에서는 두 변수를 입력받아 내적연산을 수행하고, 인지모드에서는 하나의 입력 데이터가 상수라는 점을 활용하여, 활성화되는 게이트 수를 줄임으로서 소비되는 전력을 줄일 수 있다.

V. 결 론

본 논문에서는 기존의 비트-시리얼 내적연산 프로세서 구조에서 활성화되는 유닛의 수를 줄임으로서 소비전력을 줄일 수 있는 방법을 제안한다. 이를 위해 기존의 BSA의 구조를 변형하여 인에이블 단자의 제어상태에 따라서 모든 게이트 또는 XOR 게이트의 동작만으로 'Sum'을 구할 수 있는 NBSA 구조를 새롭게 제안하고, NBSA의 인에이블은 결합계수에 추가 로직을 두어 제어하도록 하였다.

제안한 구조는 뉴럴 네트워크의 두 가지 동작 모드인, 학습과 인지모드에 따라서 각각 구분된 동작을 수행한다. 학습모드에서는 기존의 BSA와 같은 동작을 수행하여, 비트-레벨의 하드웨어 구현이 적합하고, 내부 인터페이스의 전환이 용이하며, 높은 수준의 파이프라인(pipeline) 설계가 가능하다는 장점을 가진다. 또한, 인지모드에서는 고정된 결합계수에 따라서 활성화 유닛과 비활성화 유닛을 구분하고, 활성화된 유닛만이 내적연산에 참여하도록 하여 전류 소모를 최소화 한다.

고정된 결합계수를 업데이트 하여 뉴럴 네트워크를 재구성할 필요가 있는 경우에는 다시 학습모드에서 모든 유닛을 활성화 시킨 후, 내적연산을 수행하여 필요한 연산 결과를 얻을 수 있다.

기존의 BSA 방법과 비교해보면, 구현에 필요한 게이트는 증가하고 동작속도는 다소 느려지기 때문에 뉴럴 네트워크의 학습모드에서는 오히려 비효율적이지만, 인지모드에서는 계산에 필요한 활성화 유닛을 50% 내외까지 줄일 수 있다. 뉴럴 네트워크의 동작이 학습모드보다는 인지모드에서 빈번하게 수행되기 때문에 제안한 구조는 전체 동작의 소비 전력을 효과적으로 줄일 수 있다.

참 고 문 헌

[1] James P. Coughlih, Robert H. Baran, "Neural Computation in Hopfield Networks and Boltz mann Machines", Univ of Delawarc pr, 1995.

[2] 김대수, "신경망과 이론과 응용", 하이테크정보, 1992.

[3] Manferd Glesner and Werner Pohmuller, "Neuro computers, an overvie of neural networks in VLSI", Neural Computing. Chanpman & Hall, 1994.

[4] Rodellar V., Hermida M. et al, "A VLSI arithmetic unit for a signal processing neural net work", Circuits and Systems, 1992., Proc. 35th Midwest Symposium, vol. 2, page 1044 -1047, 1992.

[5] Diaz A., Gallardo J.M. et al, "A full-custom bit -serial multiplier for neural network algorithms", Electrotechnical Conference, Proc. 7th Mediter ranean, page 258 -261, vol. 1, 1994.

[6] Amin H., Curtis K.M. et al, "Two-ring systolic array network for artificial neural networks", Cir cuits, Devices and Systems, IEE Proc., vol. 146, issue 5, page 225 -230, 1999.

[7] Girones R.G., Salcedo A.M., "Systolic implemen tation of a pipelined on-line backpropagation", Microelectronics for Neural, Fuzzy and Bio -Inspired Systems, Proc. MicroNeuro '99, page 387-394, 1999.

[8] Buddefeld J., Grosspietsch K.E., "Intelligent-memory architecture for artificial neural networks", IEEE Micro , vol. 22, issue 3, page 32 -40, 2002.

[9] Ma, G.-K., Taylor, F.J., "Multiplier policies for digital signal processing", ASSP Magazine, vol. 7, issue 1, page 6-20 1990.

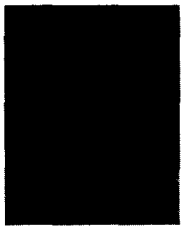
[10] 임국찬, 곽우영, 이현수, "학습된 신경망 설계를 위한 가중치의 비트-레벨 어레이 구조 표현과 최적화 방법", 대한전자공학회 논문지, 제39권, SD편, 제9호, pp819-826, 2002.

[11] Szabo T., zFeher B., Horvath G., "Neural network implementation using distributed

arithmetic", Proc. KES '98, vol. 3, pp 510-518, 1998.

[12] James-Roxby P., Blodget B.A., "Adapting constant multipliers in a neural network implementation", Field-Programmable Custom Computing Machines, IEEE Symposium, pp 335-336, 2000.

저 자 소 개



임 국 찬(정회원)

1999년 2월 경희대학교 컴퓨터공학과 학사.

2001년 2월 경희대학교 컴퓨터공학과 석사.

2000년 12월 ~ 현재 LG전자 UMTS단말연구소 주임연구원.

2003년 3월 ~ 현재 경희대학교 컴퓨터공학과 박사과정.

<주관심분야 : UMTS, VLSI 구조설계, 신경망>

이 현 수(정회원)

1979년 2월 경희대학교 전자공학과 학사.

1982년 3월 일본 게이오대학교 전자공학과 석사.

1985년 3월 일본 게이오대학교 전자공학과 박사.

1999년 9월 ~ 2000년 8월 미국 오레곤 주립대학교 전기 및 컴퓨터 공학과 방문 연구원.

미국 캘리포니아대학교(U.C.I) 전기 및 컴퓨터 공학과 방문 연구원.

1985년 ~ 현재 경희대학교 컴퓨터공학과 정교수.

<주관심분야 : VLSI 구조, 병렬 컴퓨터, 신경망, 음성 처리>