論文93-30B-10-8

잉여수 체계를 이용한 디지탈 뉴론 프로세서의 설계

(Design of a Digital Neuron Processor Using the Residue Number System)

尹 賢 埴*, 趙 源 敬**

(Hyun Shik Yoon and Won Kyung Cho)

要 約

디지탈 신경 회로망의 각 층사이의 전방향 연산과정은 행렬 벡터의 연산으로 표현할 수 있다. 뉴론에서 처리하는 연산을 고속으로 실행하도록 하기 위하여 본 논문에서는 연산시 모듈간 캐리(carry) 정보가 없는 잉여수체계(Residue Number System)를 적용한 뉴론 프로세서(Neuron Processor)의 설계방법을 제안 하였다. 또한, 시그모이드 함수를 3구간으로 분할하여 활성 영역의 값만을 연산표(Look-up Table)를 이용하여 구함으로써 시그모이드 함수를 고속으로 구하는 방법을 제안하였다. 본 논문에서 제안한 방법은 현재의 디지탈 VLSI기술에 의해 용이하게 실현될 수 있으므로 디지탈 신경 회로망의 실용화에 공헌할 수 있다.

Abstract

In this paper we propose a design of a digital neuron processor using the residue number system for efficient matrix · vector multiplication involved in neural processing. Since the residue number system needs no carry propagation for modulus operations, the neuron processor can perform multiplication considerably fast. we also propose a high speed algorithm for computing the sigmoid function using the specially designed look-up table. Our method can be implemented area-effectively using the current technology of digital VLSI and simulation results positively demonstrate the feasibility of our method. The proposed method would expected to adopt for application field of digital neural network, because it could be realized to currently developed digital VLSI technology.

I. 서론

디지탈 컴퓨터분야의 기술은 20세기 후반, 획기적인 발전을 거듭하여 현시점에 이른 반면에, 인공 신경회로망에 대한 연구는 1943년 맥걸럭(McCulloch)과 피쯔(Pitts)가 제안한 형식 뉴론 모델이후 거의 50년이상 명맥을 유지해 온 정도 였다.[1] [9] 그러나 인간의 끝없는 기술적 욕망은 디지탈 컴퓨터의 구조적 한계에 만족할 수 없었고, 따라서 인간의 사고처리 능력을 대신할 수 있으며 고도의 병렬성을 실현할 수 있는 신경회로망에 대한 연구에 깊은 관심을 기울이게 되었다. 신경 회로망에 대한 연구는 아직 초보적인 단계에 머물러 있으나, 최근 연구가 크게 진전

*正會員, 忠州産業大學校 電子計算學科
(Dept. ofComputer Science., Chungju Tech. Nat'l Univ.)
**正會員, 慶熙大學校 電子工學科
(Dept. of Elec. Eng., Kyunghee Univ.)
接受日字 : 1993年 4月 27日

되어 패턴및 음성인식, 자동제어, 영상처리 분야등에서 부분적으로 응용되고 있으며 앞으로 널리 확산될 것으로 기대된다.

기존의 디지탈 방식[5]에서는 반복적인 MAC (Multiplier with ACcumulator)연산시 캐리전파로 인한 속도 지연과 시그모이드 함수의 연산회로의 규모가 커지는 문제점이 있어서 이를 개선하는 연구가 필요하다. 그러므로 본 논문에서는 잉여수체계를 이용하여 고속의 MAC 연산과 시그모이드 연산을 수행하는 연산회로를 설계하고 이를 이용하여 인공 신경 회로망의 노드연산을 고속으로 실현할 수 있는 뉴론 프로세서를 설계하였다. 이 연구에서는 다수의 입력과 결합계수를 갖는 단일 노드 프로세서를 하드웨어로 설계하고 이를 반복 사용하는 것에 의하여 다층 신경 회로망을 실현하도록 하였다. MAC연산시에 잉여수계를 사용함으로써 고속연산이 가능하도록 하였으며 시그모이드 함수의 영역을 3개의 구간으로 분할하고, 활성구간의 값만을 다시 최적하게 표본화(16등분)하여 연산표에 저장함으로써 연산표의 크기가 작아 지도록 설계 하였다.[8] [10]

Ⅱ. 신경 회로망의 모형

그림 1.은 1개의 은닉층을 갖는 다층 신경 회로망의 모형을 나타내고 있다.

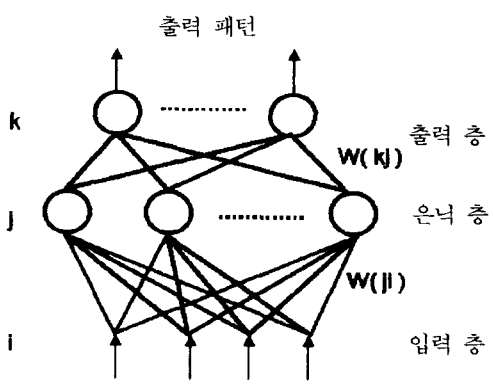


그림 1. 다층 신경망의 모형
Fig. 1. Multi-layer neural network model.

단일 노드의 입력과 결합계수사이의 연산식(net)은 식 (1)과 같다.

$$net = \sum_{i=1}^{N} w_i x_i \qquad (x_i: 입력,\ w_i: 결합계수) \qquad (1)$$

또한, 각 노드의 시그모이드 함수는 식 (2)와 같이 기술할 수 있다.

$$y = f(net) = \frac{1}{1+\exp(-(net+\theta)\theta_o)} \qquad (2)$$

(θ : 문턱값 , θ_o : 기울기)

시그모이드 함수의 출력값은 0과 1사이의 값으로 제한되며, θ로 문턱값을 변화시키고, θ_o로 시그모이드 함수의 기울기를 변화시킨다.

여기서 단일노드를 다수 연결하여 다층 신경회로를 구성하면, 전방향으로 순차적으로 활성화 되면서 식 (1)과 식 (2)의 연산을 반복하여 수행한다. 또한, 학습시에는 최종층에서 실제 출력값과 목표값을 비교하여 오차값을 계산하며, 계산된 오차를 가지고 최소평균자승(LMS) 알고리즘등을 적용하여 전층의 결합계수를 수정함으로써 학습을 수행 한다.[2] [9]

Ⅲ. 잉여수 체계(RNS: Residue Number System)

1. 잉여수 체계를 이용한 정수의 표현

잉여수 체계[8] [11]는 언웨이티드(Unweighted) 수체계로서 서로소(Relative prime number)로 이루어진 모듈리(moduli)를 이용하여 각 모듈리로 나눈 잉여수만으로 수를 표현하며, 연산시 모듈간의 캐리정보가 없어 연산속도의 고속화가 가능하며, 파이프라인 설계와 어레이 프로세서(array processor)[4] [7]로 설계하기 용이한 장점을 갖는 수체계이다.

잉여수 체계에서 모듈리를 $m_1, m_2, \cdots, m_N$로 취할 경우, 임의의 정수 X는 식 (3)과 같이 몫 Q_i와 잉여수 $|X|_{m_i}$로 표기 할 수 있다.

$$X = Q_i m_i + |X|_{m_i} \quad (단\ 0 \le |X|_{m_i} < m_i 이며\ i=1,2,\cdots,N) \qquad (3)$$

여기서 $|X|_{m_i} = X \bmod m_i$ 로 한다.

또한 식 (4)와 같이 정수 X를 N차원의 잉여수로 표시 할 수 있다.

$$X \Leftrightarrow \{|x|_{m_1}, |x|_{m_2}, \cdots, |x|_{m_N}\} \qquad (4)$$

이때 식 (4)로 표현할 수 있는 정수 X의 범위는 식 (5)와 같다.

$$0 \le X < M = m_1 m_2 \cdots m_N = \prod_{i=1}^{N} m_i \qquad (5)$$

잉여수 체계에서는 식 (5)와 같이 표현 할 수의 범위가 각 모듈러의 곱으로 제한 된다. 예를 들어 모듈러를 2, 3, 5로 취하면 표현 가능한 수의 범위는 0부터 29까지 가능하다. 그리고 제한된 범위를 넘는 수의 표현은 순환적으로 반복 된다. 다음의 표 1은 정수 -4부터 32까지의 잉여수 표현 예를 나타낸다.

표 1. 수 -4에서부터 32까지의 잉여수 표현
Table 1. The representation of residue digits from -4 to 32.

Integer	Residue digits (moduli 2 3 5)	Integer	Residue digits (moduli 2 3 5)
-4	0 2 1	15	1 0 0
-3	1 0 2	16	0 1 1
-2	0 1 3	17	1 2 2
-1	1 2 4	18	0 0 3
0	0 0 0	19	1 1 4
1	1 1 1	20	0 2 0
2	0 2 2	21	1 0 1
3	1 0 3	22	0 1 2
4	0 1 4	23	1 2 3
5	1 2 0	24	0 0 4
6	0 0 1	25	1 1 0
7	1 1 2	26	0 2 1
8	0 2 3	27	1 0 2
9	1 0 4	28	0 1 3
10	0 1 0	29	1 2 4
11	1 2 1	30	0 0 0
12	0 0 2	31	1 1 1
13	1 1 3	32	0 2 2
14	0 2 4		

2. BE(Base-Extention) 알고리즘 [8]

모듈리를 $m_1, m_2, \cdots, m_N$로 선택한 경우 정수의 표현 가능한 구간은 $[0, \prod_{i=1}^{N} m_i - 1]$ 이며 표현 가능한 구간을 $[0, \prod_{i=1}^{N+1} m_i - 1]$ 로 확장하기 위하여는 모듈러스 m_{N+1}을 추가 하여야 한다. 이 경우 확장된 모듈러스 m_{N+1}의 잉여수 $|X|_{m_{N+1}}$를 구하는 과정을 BE알고리즘이라 하며 다음과 같이 구한다.

확장구간 $[0, \prod_{i=1}^{N+1} m_i - 1]$ 내의 수를 모듈리 m_i에 대한 혼합 기수(Mixed Radix)로 표현하면 식 (6)과 같다.

$$X = a_{N+1}\prod_{i=1}^{N} m_i + a_N\prod_{i=1}^{N-1} m_i \cdots a_3 m_1 m_2 + a_2 m_1 + a_1 \qquad (6)$$

여기서 X의 잉여수는

$$X \Leftrightarrow \{|x|_{m1}, |x|_{m2}, \cdots, |x|_{mN}, |x|_{mN+1}\} \qquad (7)$$

식 (7)과 같이 표현되며 식 (6)에서 $a_1 = |X|_{m1}$ 이고 식 (7)에서 a_1을 빼고 각 잉여 디지트에 역원 $|1//m_2|_{mi}$ (i=2, 3, ⋯, N, N+1)를 곱하면 a_2를 얻는다. 다시 a_2를 빼고 각 잉여 디지트에 $|1//m_3|_{mi}$ (i=3, 4, ⋯, N, N+1)를 곱하면 a_3가 구해진다. 이와 같은 연산과정을 계속하면 $a_{N+1} = ||1//m_1 \cdot m_2 \cdots m_N|_{mN+1}|X|_{mN+1} + A|_{mN+1}$을 구할 수 있고 베이스 확장 이전의 수는 식 (6)의 정의에서 a_{N+1}=0인 경우이므로 식 (8)과 같이 쓸 수 있다.

$$a_{N+1} = \left\|1//m_1 \cdot m_2 \cdots m_N\right|_{m_{N+1}} |X|_{m_{N+1}} + A\Big|_{m_{N+1}} = 0 \left(\text{단 } A = -\sum_{i=1}^{N} a_i \prod_{j=1}^{N-1} m_j \right) \qquad (8)$$

그러므로 BE(m_{N+1})의 잉여수($|X|_{N+1}$)를 식 (8)으로 부터 구할 수 있다.

3. 스케일링(Scaling) 알고리즘 [8]

정수와 정수의 나눗셈의 결과는 유리수가 된다. 그러나 잉여수계의 연산은 그 결과로 정수만을 취할 수 있기 때문에 잉여수의 일반적인 나눗셈은 그 연산과정이 복잡하다. 이 문제를 해결하기 위하여 정수 X를 S로 나누는 경우 식 (10)과 같이 X로 부터 $|X|_S$, 즉 나머지(모듈러스를 S로 취하면 잉여수)를 미리 빼어주고 그 결과에 S의 역원(multiplicative inverse)을 곱하는 방법에 의하여 나눗셈을 대신 할 수 있다. 잉여수계에서 이러한 변형된 나눗셈 방법을 스케일 연산이라고 정의 한다. 본 논문에서 제안한 뉴론 프로세서는 MAC연산부의 크기를 작게하고 식 (1)의 연산시 발생 할 수 있는 오버플로우를 방지하기 위하여 연산 결과를 모듈리 m_1, m_2의 곱으로 나눈 몫만을 누산기에 누적시켜 이를 시그모이드 함수 처리부의 입력으로 사용한다. 이때 몫을 구하기 위한 나눗셈에 다음과 같은 스케일링 연산 방법을 이용한다. 정수 [X/S] 의 잉여수 표현은 식 (9)와 같고,

$$\left\{ \left| (X - |X|_s)/S \right|_{m1}, \left| (X - |X|_s)/S \right|_{m2}, \cdots \left| (X - |X|_s)/S \right|_{mN} \right\} \qquad (9)$$

또한, S와 m_i가 서로소이면 정의 [8]에 의하여 식 (10)과 같이 나타 낼 수 있다.

$$\left| (X - |X|_s)/S \right|_{m1} = |X/S|_{mi} = \left| (X - |X|_s)//S \right|_{mi} \quad (1/S\text{:역원}) \qquad (10)$$

s=m_1 m_2 m_{n-1}일 경우에는 m_1 m_{n-1} 각각에 대해 식 (10)을 반복 계산하여 [X/S] 를 구한다. 본 연구에서는 스케일링에 의해 S=15 * 17에 대한 Q =(X-$|X|_{255}$) /255 를 구한 후 , 이 값을 부호와 함께 별도의 누산기에 저장 하였다가 시그모이드 함수의

구간을 결정하는데 사용한다.

Ⅳ. 디지탈 신경회로를 위한 고속 연산프로세서의 설계

디지탈 신경회로를 위한 고속 연산프로세서를 설계하기 위하여 본 논문에서는 그림 2.와 같이 잉여수 체계를 이용한 뉴론프로세서를 설계 하였다. 그림 2.는 결합계수와 입력과의 승산및 누적을 수행하는 MAC연산부, MAC연산부의 크기및 오버플로우를 방지하기 위한 몫연산 처리부,그리고 시그모이드 함수 연산부로 구성된다.

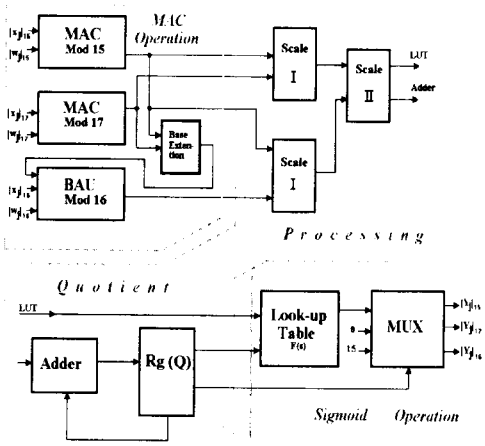


그림 2. 뉴론회로의 구성
Fig. 2. Circuit configuration of a neuron.

1. 승산을 포함한 고속 MAC의 설계

식 (1)을 디지탈 회로로 실현하기 위한 기본연산은 MAC연산으로 식 (11)과 같이 정의 할 수 있다

$$Rg \Leftarrow Rg \pm w_i \cdot x_i \qquad (11)$$

그러므로 식 (1)의 연산을 고속으로 실행하기 위하여는 식 (11)의 연산을 고속으로 실행할 수 있는 MAC을 설계하여야 한다. 식 (11)을 실수 연산으로 처리하기 위해서는 많은 수의 게이트를 필요로 하기 때문에 이 논문에서는 신경망의 노드 연산이라는 제한된 응용 분야에 적용하기 위하여 잉여수체계를 사용한 정수 연산만으로 회로를 간단히 하였고, 또한 잉여수체계의 특징을 이용하여 승산과 가산이 동일한 속도로 가능한 고속의 MAC을 그림 3.과 같이 설계하였다.

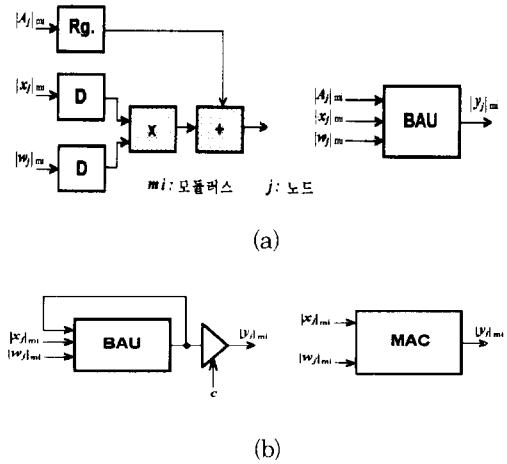


그림 3. MAC의 기본구조
(a) 기본 연산회로 (b) MAC 연산회로
Fig. 3. Basic structure of MAC.
(a) Basic Arithmetic Unit,
(b) MAC Operation Unit.

그림 3. (a)는 가산과 승산을 수행하는 기본연산 처리부(BAU:Basic Arithmatic Unit)이며, 연산 결과를 기본연산 처리부에 재입력함으로써 그림 3. (b)와 같은 MAC을 설계할 수 있다. 즉, 모듈러스 m_i인 경우 입력 데이타를 $|x_j|_{mi}$, 가중치 값을 $|w_j|_{mi}$, 가산기의 외부 입력을 $|A_j|_{mi}$라 할때 기본 연산처리부의 출력 $|y_j|_{mi}$는 $|y_j|_{mi} = |x_j|_{mi} |w_j|_{mi} + |A_j|_{mi}$이며, 이때 출력 $|y_j|_{mi}$를 $|A_j|_{mi}$로 입력하여 MAC 연산부를 구성하면 다음과 같다.

$$|y_{j+1}|_{mi} |x_{j+1}|_{mi} |w_{j+1}|_{mi} + |y_j|_{mi} \qquad (12)$$

만일, 그림 3.의 회로에서 모듈러스 m_i를 15로 선택한다면 $|x_j|_{mi}$, $|w_j|_{mi}$, $|y_j|_{mi}$가 각각 4 bits로 표현 되기 때문에 일반적인 정수 연산회로에 비해 작은 하드웨어로 고속의 연산이 가능하다. 즉 MAC 연산회로의 크기는 모듈러스 15인 경우 1024($2^8 * 4$) Bits ROM 2개와 4 Bits 누산기 3개만으로 구성될 수 있다. 또한 승산과 가산 모두 연산표를 이용하기 때문에 연산속도가 동일하므로 회로의 동기도 용이하다.

그림 2.의 MAC연산부는 모듈리를 m_1=15, m_2=17, m_3=16로 정하였을때 뉴론에 입력된 정수들의 연산을 수행하는 회로를 나타내고 있다. 그림 2.에서 MAC 연산시의 표현 가능한 수의 범위는 $-2040 \leq SM$

<2040으로 결정되며 입력 정수 X의 범위를 $0 \le X \le 15$, 결합계수 w의 범위를 $-127 \le w \le 127$ 로 제한하였을때 1회의 승산에서 발생 가능한 최대 값은 $15 \times (\pm 127) = \pm 1905$로서 1회의 MAC연산 결과는 SM의 범위내에 포함되므로 오버플로우가 발생하지 않는다.

2. MAC의 연산부를 작게하고 오버플로우를 방지하기 위한 방법

그림 2.에서 MAC연산부의 승산과 가산이 반복되면 제안된 수의 범위(SM)를 초과하여 오버플로우가 발생한다. 그러므로 몫 연산부를 그림 2.와 같이 설계하여 1회의 MAC연산마다 m_1, m_2의 곱인 255로 나눈 몫을 R_g에 저장함으로써 MAC 연산부에서의 오버플로우를 방지할 수 있고 또한 MAC연산부의 크기를 작게할 수 있다. 몫 Q = (Y- | Y | 255) / 255를 구하기 위하여 식 (10)의 스케일 연산(스케일I)에서는 $|X/17|_{mi} = |(X-|X|_{17})//17|_{mi}$를 구하고 그 결과를 다시 모듈러스 15에 대해 스케일 연산(스케일II)을 실행하여 식 (13)과 같이 몫(Q)를 구할 수 있다.

$$\left|(Y-|Y|_{255})/255\right|_{mi} = \left\{\left|(Y-|Y|_{255})/255\right|_{15}, \left|(Y-|Y|_{255})/255\right|_{17}, \left|(Y-|Y|_{255})/255\right|_{16}\right\} \quad (13)$$

이때 나머지는 r=Y-Q*255이므로 $|r|_{m1} = |Y|_{15}$, $|r|_{m2} = |Y|_{17}$, $|r|_{m3} = |Y|_{16} - |Q255|_{16}$이며 $|Y|_{15}$와 $|Y|_{17}$은 현재 MAC의 연산 결과이므로 새로운 연산이 필요 없으나 $|Y|_{16}$은 $|Y|_{15}$와 $|Y|_{17}$을 이용하여 식 (8)과 같이 BE알고리즘을 사용함으로써 $|Y|_{16}$을 구할 수 있다. 그림 2.에서 몫(Q) 처리부분의 가산기를 8bit로 설계할 경우 오버플로우를 유발하지 않고 처리할 수 있는 총 입력 노드 수는 17개이며 16bit로 설계할 경우는 4386개의 입력 노드를 처리할 수 있다.

3. 고속으로 시그모이드 함수를 구하기 위한 방법

그림 2.에서 시그모이드 함수 연산부는 시그모이드 함수를 고속으로 구하기 위한 부분으로써 그림 4.와 같이 함수의 영역을 3구간으로 구분하고 구간 가를 16등분한후 각 함수값들을 0 ~ 15사이의 정수값으로 취하여 연산표에 저장한다. 4.2절의 처리과정에서 구한 Q값을 다음과 같이 구간 결정에 사용한다. $Q \le -1$ 이면 F(s)= 0, $Q \ge 1$이면 F(s) = 15의 값을 취하고 Q =0이면 그림 2의 LUT(look-up table)값에 의하여 $0 < F(s) < 15$ 값을 연산표에서 구한다.

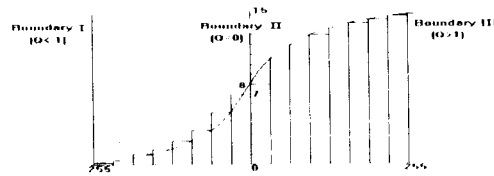
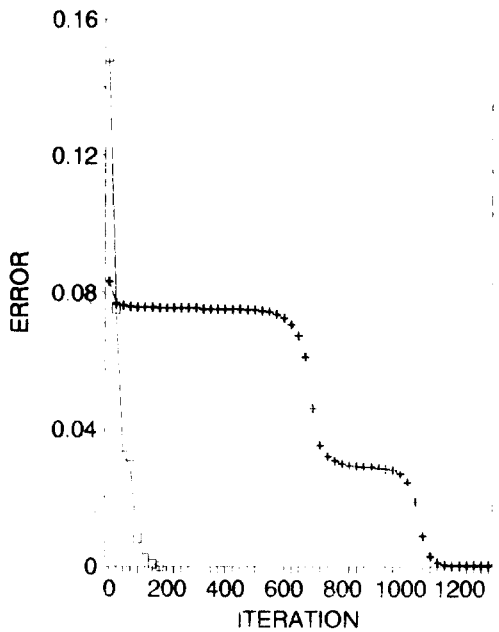


그림 4. 시그모이드 함수의 구간 설정
Fig. 4. Decision of boundary in sigmoid function.

V. 모의실험 및 고찰

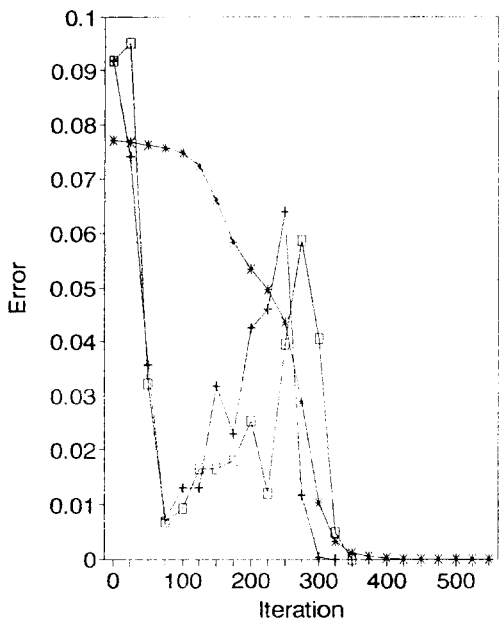
앞절에서 설계한 뉴론 프로세서를 평가하기 위하여 기존의 실수연산기를 이용한 경우와 본 논문에서 설계한 프로세서를 사용한 경우의 오차수렴 과정, 시그모이드 함수에서의 파라메터 변동에 따른 오차수렴 과정 등을 컴퓨터 모의실험을 통하여 비교검증 하였다.

모의 실험의 실행 조건은 노드 수가 각각 3, 3, 1인



□ : 잉여수체계 연산 + : 실수 연산

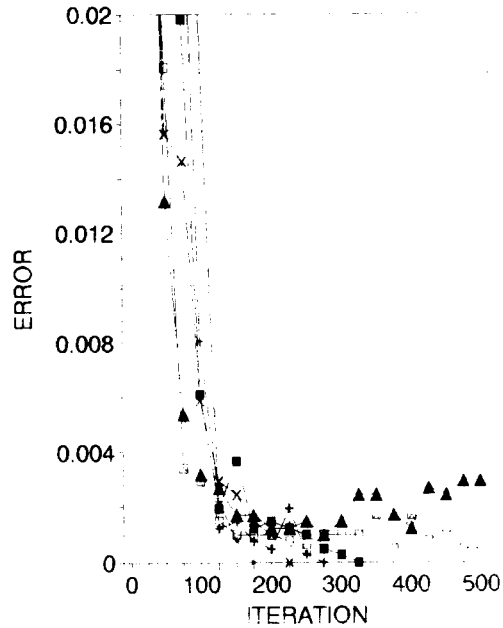
그림 5. 패리티 문제에 대한 뉴론의 잉여수체계 연산과 실수연산의 수렴속도 비교
Fig. 5. Comparing iteration to approaching target value of floating point processing vs RNS processing for parity problem.



*: 실수 연산 +: 표본화 16 □: 표본화 32

그림 6. XOR문제에 대한 뉴론의 잉여수체계 연산과 실수연산의 수렴속도 비교

Fig. 6. Comparing iteration to approaching target value of floating point processing vs RNS processing for XOR problem.



■: 80 +: 90 *: 100 □: 110 ×: 120 △: 130
($-1 \leq Q < 1$일때 기울기의 변화량)

그림 7. 시그모이드 함수의 기울기 변화에 대한 수렴속도 비교

Fig. 7. Comparing iteration to approaching target value withchanging slope of sigmoid function.

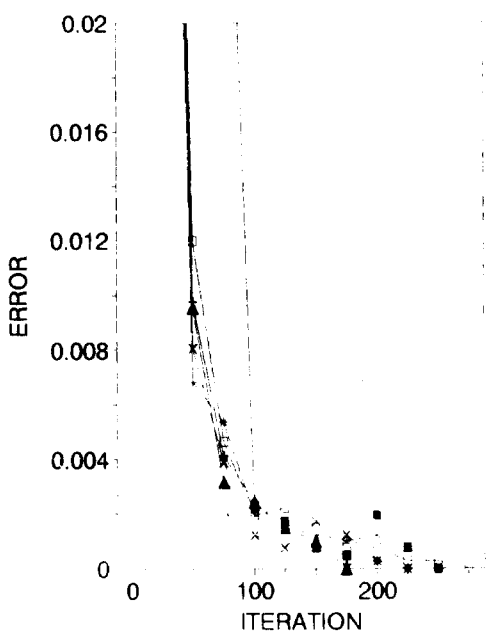
3층 BP신경망에서 패리티 검사와 노드수가 각각 2, 2, 1인 XOR 문제를 대상으로 하였으며 관성 계수 α =0.7, 학습 계수 η =0.9 로 설정 하였다. 그림 5는 패리티 문제에 대한 실수연산기를 사용한 시그모이드 함수 연산과 4.3절에서 제안한 시그모이드 함수연산부를 뉴론 프로세서에 적용 하였을때의 수렴속도를 비교한 그래프이며 제안한 뉴론 프로세서는 수렴 오차값을 10^{-3}로 설정 하였을때 200회 정도의 학습으로 수렴 하였으나 실수연산의 경우는 1200회정도의 학습횟수를 보이고 있다. 그림 6은 XOR문제에 대한 실수 연산과 제안한 알고리즘의 신경망 연산과의 오차에 따른 반복횟수를 나타낸 것이다. 수렴오차값을 10^{-3}로 설정 하였을 경우, 실수 연산은 534회의 반복 연산을 나타 내었으나, 제안된 알고리즘에서는 시그모이드 함수의 구간 표본화수가 16개인 경우 301회를 보였으며, 표본화수가 32개인 경우는 373정도의 학습횟수를 나타내었다. 기존의 시그모이드 함수의 연산 결과(출력)는 미세하고 정확한 값을 갖는다. 그러나 제안된 알고리즘은 활성 영역을 2^N(N은 4이상) 개로 표본화 하였으며 이때의 함수값은 계단함수의 형태를 취하므로 원래의 값보다 큰 값으로 가중치를 갱신하기 때문에 오차 수렴이 빠를 것으로 예상 되었고 모의 실험 결과도 빠른 오차 수렴을 나타내고 있다. 그러나 이 경우 실제값에 의한 가중치 갱신이 아니므로 오차 수렴시 실제값을 적용하는 경우보다 다소 진동하고 있음을 알 수 있다.

그림 7.은 잉여수체계 신경 회로망의 시그모이드 함수의 기울기 e_0(Slope)를 80, 90, 100, 110, 120, 130으로 증가하는 경우 그에 따른 학습 속도의 변동을 나타내고 있다. 수렴 오차 0.002에서는 기울기 80, 90, 100 , 110의 경우가 비슷한 횟수로 수렴 하였고 오차 0.001에서도 기울기 100의 경우가 150회에서 가장 빠르게 수렴 하였으며 오차 10^{-4}일때 역시 기울기 100의 경우가 175회로서 가장 빨리 수렴하였다. 그림 7.로 부터 시그모이드 함수의 기울기 e_0가 100정도로 설정되어서 비교적 완만한 특성곡선을 이

룰때 가장 신속하게 수렴함을 알 수 있으며 이는 기존의 실수 연산 신경 회로망의 시그모이드 함수 특성과 동일하다.

그림 8.은 시그모이드 함수의 구간(시그모이드 함수를 구하기 위한 연산표의 크기에 관계됨)을 2배(32)로 표본화 했을 경우, 기울기를 100, 140, 180, 200, 220, 240으로 변경 시키면서 오차에 따른 수렴속도의 결과를 나타낸다. 목표값과의 오차가 10^{-3}이하인 경우, 기울기가 240일때 175회로 가장 빠르게 수렴하였으며 구간 표본화값을 증가시켜도 수렴속도는 그림 7.의 기울기 100인 경우와 동일 하였다.

실험 결과 제안된 알고리즘이 동일한 조건 하에서 실수연산을 이용하는 경우에 비하여 9배 정도 빠르게 수렴 하였다. 또한, 목표값과의 오차를 10^{-3}이하로 줄이기 위하여 시그모이드 함수의 기울기를 변경하면서 실험한 결과 기울기 100일때 가장 빠르게 수렴 하였다. 기울기 계수 θ_0만을 크게하면 수렴속도는 개선되나 발산할 염려가 있고 구간 분할의 폭을 크게하면



■: 80 +: 140 *: 180 □: 200 ×: 220 △: 240
($-2 \le Q < 1$일때 기울기의 변화량)

그림 8. 시그모이드 함수의 구간값 변경에 의한 수렴속도

Fig. 8. Comparing iteration to approaching target value with changing boundary of sigmoid function.

연산표가 증가하여 하드웨어가 커지는 점을 고려하여 활성 영역의 구간을 16 등분하고 기울기 계수 e_0를 100으로 설정 하는것이 가장 바람직 하였다.

Ⅵ. 결론

오늘날 신경 회로망 이론은 다양한 응용분야에서 적합성을 인정받고 있으나, 신경 회로망이 대량 분산 병렬처리(Massive distributed parallelism)방식으로 되어 있기 때문에 하드웨어로 실현할 경우 회로의 크기와 속도에 대한 대응책이 요구된다. 본 논문에서는 잉여수 체계를 이용하여 신경 회로망의 반복적인 MAC연산을 고속으로 수행할 수 있는 알고리즘과 시그모이드 함수처리를 위한 연산표의 크기를 줄이기 위한 방법을 제안하고 이를 이용한 뉴론 프로세서를 설계 하였다.

기존의 실수 연산 방법[1]과 제안한 방법의 비교 실험에서 본 논문에서 제안한 뉴론 프로세서가 보다 고속 연산이 가능 하다. 20nsec의 연산표를 사용하여 MAC을 설계할 경우 40nsec의 연산 속도로 MAC연산이 가능하며 256개의 노드 연산에 10.24μsec의 연산 속도를 기대할 수 있다. 또한 시그모이드 활성영역[10]을 16구간으로 표본화하여 연산표를 만들어 출력을 구함으로써 시그모이드 함수의 고속처리와 경제적인 VLSI칩 설계가 가능 하다.

앞으로의 연구 방향은 제안된 알고리즘을 이용한 신경 회로망 시스템이 적용될 분야에 따른 최적의 모듈리 설정 및 임의의 입력에 대한 최적의 기울기와 적절한 활성 영역의 구간분할 방법이 이론적으로 전개 되어야 할 것으로 생각 된다.

參 考 文 獻

[1] Yoh-Han Pao, *Adaptive Pattern Recognition and Neural Networks*, AddisonWesley Publishing Company, Inc., 1989.

[2] STEPHAN, I. GALLANT, "Perceptron-Based Learning Algorithms", *IEEE Trans-action Neural Networks* vol. 1, No. 2, 1990.

[3] D. E. Rumelhart, J. L. McClelland, et al, *Parallel Distributed Processing*, the MIT Press, Vol. 1, 1986.

[4] Takeshi Oohashi, "An Implementation of Multi Layered PDP Models on

theLoosely Coupled Multi-processor", 日本電子工學會 論文誌 D-2 vol.j73-D- 2, No. 8, pp. 1354-1359, Aug. 1990.

[5] JENQ-NENG HWANG, "A Systolic Neural Network Architecture for Hidden Markov Models ", *IEEE Transactions coustics Speech, and Signal Pro-cessing*, vol. 37, No. 12, December 1989.

[6] H. T. Kung, "Why Systolic Architectures?", *IEEE Computer*, pp. 37-46, Jan-uary 1982.

[7] S. Y. Kung, *VLSI Array Processing*, Prentice Hall international edition. 1988.

[8] Nicholas S. Szabo, M. S. & Richard I. Tanaka, Ph. D., *Residue Arithmetic andIts Applications to Computer Technology*, McGRAW-HILL Book Company, 1967.

[9] 조 원경, "RNS를 이용한 연산 프로세서의 설계에 관한 연구", 한양 대학교박사학위 논문 1986.

[10] 정 윤돈 외 3인, "디지탈 신경망 모델의 시그모이드 함수 연산회로 설계에관한연구" 전자공학회 하계학술대회 논문집, pp. 184-187, 1991, 6.

[11] 윤 현식 외 3인, "디지탈 신경 회로망 실현을 위한 어레이 프로세서의 설계" , 인공지능/신경망및 퍼지시스템 워크샤, 1991, 11.

著 者 紹 介

尹 賢 埴(正會員) 第 30卷 B編 第 2號 參照
현재 충주산업대학 전자계산과 교수

趙 源 敬(正會員) 第 30卷 B編 第 1號 參照
현재 경희대학교 전자공학과 교수