韓國 OA學會 論文誌
第4卷 第2號, 1999. 6.

99-4-2-1-4

디지탈 뉴런프로세서의 구현에 관한 연구

홍 봉 화* , 이 지 영**

On the Implementation of the Digital Neuron Processor

Bong-Wha Hong,* Jie-Young Lee**

요 약

본 논문에서는 캐리 전파가 없어 고속 연산이 가능한 잉여수체계(Residue Number System)를 이용하여 고속의 디지털 뉴런 프로세서를 제안하였다.

제안된 뉴런프로세서는 MAC (Multiply And Accumulator) 연산부, 몫연산부, 시그모이드(Sigmoid)함수 연산부로 구성되며, 0.8μm CMOS공정으로 설계되었다. 실험결과, 본 논문에서 구현한 디지털 뉴런프로세서는 19.2nsec의 속도를 보였으며, 실수연산기로 구현한 뉴런프로세서에 비하여 약 1/2정도 하드웨어 크기를 줄일 수 있었다.

Abstract

This paper proposes a high speed digital neuron processor which uses the residue number system, making the high speed operation possible without carry propagation,.

Consisting of the MAC(Multiplier and with Accumulator) operation unit, quotient operation unit and sigmoid function operation unit, the neuron processor is designed through 0.8 μm CMOS fabrication.

The result shows that the new implemented neuron processor can run at the speed of 19.2 nSec and the size can be reduced to 1/2 compared to the neuron processor implemented by the real number operation unit.

* 세명대학교 컴퓨터과학과 교수
** 세명대학교 컴퓨터과학과 부교수
논문접수 : 1999. 5.20. 심사완료 : 1999. 6.19

Ⅰ.서론

신경회로망 이론은 1940년대에 이미 제안되었지만 반도체 기술이 발전한 근래에 이르러 많은 발전을 하게 되었다. 현재까지 연구되고 있는 신경회로망의 응용분야는 인식(패턴 인식, 음성인식), 제어이론, 영상처리분야 등이 있다.[1][2][3][4][5]

최근 영상신호처리 및 패턴인식 분야에서 대량의 데이타를 실시간으로 처리 하여야하는 필요성이 증가하고 있다. 컴퓨터와 사용자간의 인터페이스 문제에 있어서 실시간 처리는 중요한 해결 수단이다. 이와 같은 응용분야에 신경회로망을 이용하기 위하여는 대량의 데이타를 실시간으로 처리할 수 있는 고속의 MAC연산기와 판별함수를 고속으로 처리할 수 있는 연산기가 요구된다.

신경회로망의 연산은 행렬·백터 연산을 기본으로 하고 있으므로 어레이프로세서의 구조로 구현이 가능하다. 또한, 잉여수체계는 모듈러간의 캐리 정보가 필요없으며, 승산이 가산과 거의 같은 속도로 구현될 수 있으므로 고속 MAC 연산기의 구현이 가능하다.[6][7][8]

본 본문에서는 잉여수계를 적용하여 고속의 MAC 연산기를 설계하고 이 설계된 MAC 연산기를 이용하여 고속의 디지탈 뉴런프로세서를 설계 및 구현하고자 한다.

뉴런프로세서의 구현시 문제가 되는 시그모이드 함수처리는 잉여수계를 이용한 ROM 테이블(ROM Look-Up Table)방식을 사용함으로써 효율적으로 수행할 수 있다.[9][10]

신경회로망의 기능을 디지털회로를 이용한 어레이프로세서로 구현하는 것은 아날로그 방식 및 광을 이용한 신경회로망이 아직 연구단계에 있는 것에 비하여 현재의 발전된 디지탈 VLSI 설계 기술을 이용하여 실제로 이용 가능한 수준의 신경칩 을 제작할 수 있기 때문에 중요하다.

잉여수체계를 이용한 MAC 연산기의 설계시, 가산과 승산이 동일한 동형으로 연산하기 위하여 순환군를 사용하며, 연산기에 사용되는 순환부호는 각각의 부호에 한 비트만을 "1"로 하는 코드를 사용함으로써 처리시간을 줄일 수 있고, 하드웨어을 단순하게 구현 할 수 있다.

설계된 MAC 연산기는 100MHZ 이상의 속도로 MAC연산(AC 〈-AC±A*W)을 수 행 할 수 있으며, 이 MAC 연산기를 내장한 PE를 어레이 형태로 나열함으로써 대 규모의 신경망 구현이 가능하다. 또한, 하나의 칩에 64개 정도의 PE를 내장시킴으로써 신경회로망 이론을 실제 응용하는 제품개발이나 신경회로망의 이론적인 연구를 위하여 공헌할 수 있을 것으로 기대된다.

Ⅱ. 역전파 알고리즘

역전파(Back propagation) 알고리즘은 1986년경 Rumelhart로 대표되는 PDP그룹에 의해 제한되었으며, 그림 1과 같은 구조로 되어있다. 각 층이 NL개의 노드를 가지고 L층으로 구성된 역전파 신경회로망을 고려하자.

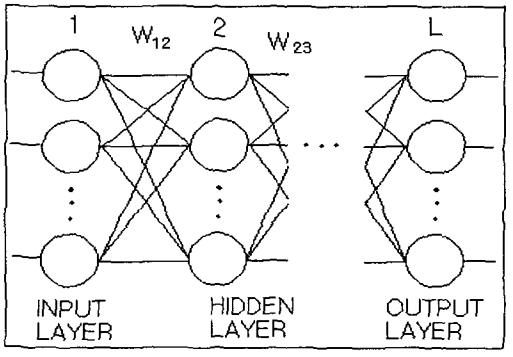


그림 1. 역전파신경망
Fig. 1. Back propagation neural network

그림 1에서 j층의 한 노드에 대한 입력(netj)과 활성함수를 통과한 후의 출력은 다음식과 같다.

$$net_j = \sum W_{ji} O_i = U_j \quad (1)$$

$$O_j = F(U_j) = Oj = \frac{1}{1+\exp(-(netj+\theta j)/\theta o)} \quad (2)$$

(θj : 문턱값, θo : 기울기)

시그모이드 활성함수는 출력값을 0과 1사이의 값으로 제한하며, 다음단 노드의 입력값이 된다. 단일노드의 연산은 층의 순서에 의해 식 (1)과 (2)의 연산을 반복 수행하며, 최종층 k층에서의 출력 Ok를 출력한다. 여기서 목표값을 Tk라 하고, 실제 출력값을 Ok라 하면, 오차값 E는 식(3)의 최소자승오차(LMS : least mean square) 식으로써 구해질 수 있다.[1][2][3][4][5]

$$E_p = \frac{1}{2}\Sigma(T_k - O_k)^2 \quad (3)$$

실제출력과 훈련(training)패턴간의 학습과정은 가중치의 변경에 의해 처리된다. 가중치 변경은 식 (3)에 의해 얻어진 오차 E에 의해서 조절된다. 그림 1에서 j층의 어느 한 노드의 p번째 패턴의 오차값 δpj는 식 (4)와 같다.

$$\delta_{pj} = O_{pj}(1 - O_{pj})\sigma_{pj} \quad (4)$$

여기서 상위층인 k층의 오차값에 의하여 σpj는 식 (5)처럼 표현된다.

$$\sigma_{pj} = \sum \delta_{pk} W_{kj} \quad (5)$$

최종층 k에서 p번째 패턴 오차 σpk는 다음식에 의해서 구할 수 있다.

$$\sigma_{pk} = T_{pk} - O_{pk} \quad (6)$$

i 층과 j층 사이의 t+1단계 가중치 변화량 △Wji(t+1)은 식 (7)로 구할 수 있다.

$$\triangle W_{ji}(t+1) = \eta(\delta_j O_i) + \alpha \triangle W_{ji}(t) = \eta(\delta_j O_i) + \alpha \triangle W_{ji}(t) \quad (7)$$

여기에서 η와 α는 각각 학습계수와 관성계수로서 1회의 가중치 갱신에 따른 비례값이다. 각 층에서 계산된 오차는 역방향으로 가중치를 순차적으로 갱신한다.

Ⅲ. 신경회로망의 기능을 어레이 프로세서 구현

Ⅱ절의 식들은 행렬식을 이용하여 다음과 같이 표현이 가능하다.

$$U = {}^t[U_1 \ldots U_n]$$
$$O = {}^t[O_1 \ldots O_n]$$
$$\delta = {}^t[\delta_1 \ldots \delta_n]$$
$$T = {}^t[T_1 \ldots T_n]$$

$$W = \begin{pmatrix} W_{11} & \cdots & W_{1n} \\ & \cdots & \\ & \cdots & \\ & \cdots & \\ W_{n1} & \cdots & W_{nn} \end{pmatrix}$$

$$U = W \cdot O$$

$O = F(U)$ 이다.

윗 식들은 그림 2와 같은 구조의 어레이프로세서 기법에 의하여 처리될 수 있다.

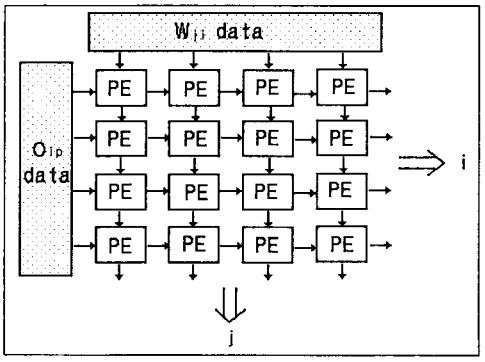


그림 2. 신경망 모델의 2차원 어레이프로세서 구현
Fig. 2. Implementation of two-dimensional Array Processor for neural network

그러나 그림 2와 같이 2차원 어레이 구조를 하드웨어로 구현시, 속도는 빠르지만 하드웨어 크기가 증대되는 문제가 발생한다.

그림 2의 구조는 어레이프로세서 설계기술에 의하여 그림 3과 같이 2차원 구조를 1차원으로 정합하는 것이 가능하며, 이러한 1차원 구조는 대량의 데이타 처리시 분할처리도 가능하기 때문에 본 연구에서는 그림 3의 구조로 설계한다.

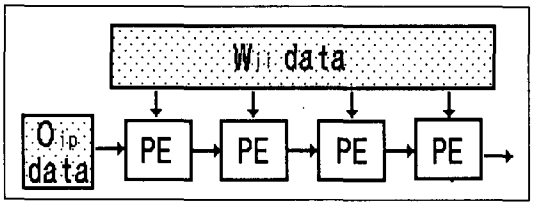


그림 3 역전파 신경회로망의 1차원 어레이프로세서로의 구현
Fig.3. Implementation of one-dimensional array processor for neural networks

그림 3에서 각 PE의 기본연산은 식 (8)과 같은 MAC연산이다.

$$AC \leftarrow AC \pm A * W \qquad (8)$$

식 (8)과 같은 연산을 수행하는 MAC의 기본 구조는 그림 4와 같다.

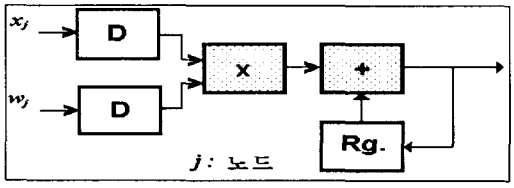


그림 4. MAC의 기본구조
Fig. 4. Structure of MAC

그러나, 식 (8)을 현재의 연산회로(정수형과 부동소수점 연산회로)를 가지고 구현할 경우, 하드웨어 크기의 증대 및 속도가 저하되는 문제점이 발생됨으로 새로운 방식의 MAC 연산회로의 설계와 이를 이용한 어레이프로세서의 설계 방식이 연구되어야 한다.

Ⅳ. 잉여수계를 이용한 MAC 연산기의설계

1. 잉여수계의 특징

일반적인 2진 정수계에 의한 연산기는 캐리정보로 인하여 가산기 및 승산기 설계시 문제가 된다. 특히, 대량의 입력 데이타를 처리하는 영상신호처리, 패 턴 인식 분 야의 경우, 연산기의 크기 및 처리속도 향상에 많은 어려움이 있다. 잉여수계는 가중치가 없는 수체계이며, 각 모듈간에 독립성을 갖으므로 캐 리 정보가 필요없고 승산과 가산이 거의 동일한 시간에 이루어질 수 있다는 큰 장점을 갖기 때문에 대량의 데이타를 처리하는 고속의 병렬처리 연산에 유용하다.[6][7][8]

2. 잉여수계의 기본연산

정수의 잉여수계 표현 방법은 다음과 같다. 서로소인 모듈리 P를 선택하고 P = { m1, m2, m3 }일때 정수 X의 표시 가능한 범위는 다음과 같다.

$$0 \le X \le M \left(M = \prod_{i=1}^{N} \right) \text{ 이다.}$$

잉여수계 정수 X는 n 테이블로 표시할 때

$$X \xrightarrow{RNS} (X_1, X_2, X_3, \ldots, X_n) \text{ 로 된다.}$$

(단, Xi = X mod mi = |X|mi)

예를 들면, P={2, 3, 5}일 경우, 정수의 잉여수 표현 예를 표1에 나타내었다.

정수 X=25는 |X|m1=1, |X|m2=1, |X|m3=0 즉(1, 1, 0)로 표현되며 기본 연산은 다음과 같다.

Z = X ∘ Y ('∘' 연산자: * + -)가 성립한다.

즉, |Z|mi = |X|mi ∘ |Y|mi = |X ∘ Y|mi
예를 들면 X = 12, Y = 5의 가산의 경우

P = (2, 3, 5)일 때

	mod 2	mod 3	mod 5	
	0	0	2	; x = 12
+	1	2	0	; y = 5
	1	2	2	; Z = 17

연산 결과는 표 1을 참조하면 17임을 알 수 있다.

표1. 모듈리{2, 3, 5}의 경우 잉여수 표현예
Table 1. Example of residue number presentation of moduli {2,3,5}

정수 X	잉여수 표현 m1	m2	m3	정수 X	잉여수 표현 m1	m2	m3
0	0	0	0	16	0	1	1
1	1	1	1	17	1	2	2
2	0	2	2	18	0	0	3
3	1	0	3	19	1	1	4
4	0	1	4	20	0	2	0
5	1	2	0	21	1	0	1
6	0	0	1	22	0	1	2
7	1	1	2	23	1	2	3
8	0	2	3	24	0	0	4
9	1	0	4	25	1	1	0
10	0	1	0	26	0	2	1
11	1	2	1	27	1	0	2
12	0	0	2	28	0	1	3
13	1	1	3	29	1	2	4
14	0	2	4	30	0	0	0
15	1	0	0				

3. 혼합기수 변환 (Fixed Radix Conversion)

임의의 정수 X를 혼합기수[6][7][8] 형식으로 표현하면 식 (9)와 같다.

$$X = a_N \prod_{i=1}^{N-1} R_i + \cdots + a_3 R_1 R_2 + a_2 R_1 + a_1 \qquad (9)$$

Ri : radides

ai : mixed-radix digit 또는 mixed- radix 계수

0 ≤ ai 〈 Ri

X = 〈an, an-1, . . . , a1 〉

혼합기수로 변환시 혼합기수의 계수를 구하는 과정은 아래와 같이 구할 수 있다.

i) 식 (9)의 양변에 모듈로 m1을 취하면 마지막 항을 제외하고는 모두 m1의 곱으로 되어 있으므로,

| X |m1 = a1

즉, a1은 첫 번째 residue digit인 r1이 된다.

ii) 식 (9)의 양변에 a1을 빼주고 모듈로 m2를 취하면 식 (10)과 같다.

$$\begin{aligned} |X - a_1|_{m2} &= \left| a_N \prod_{i=1}^{N-1} m_i + \cdots + a_3 m_1 m_2 + a_2 m_1 \right|_{m2} \\ &= |a_2 m_1|_{m2} \\ a_2 &= \left| \frac{x - a_1}{m_1} \right|_{m2} \\ &= \left| \left[\frac{X}{m_1} \right] \right|_{m2} \end{aligned} \qquad (10)$$

iii) i), ii)의 방식으로 나머지 혼합기수 계수를 구하게 되며, i 〉 1 조건에서 계수 a i는 식 (11)과 같이 표현된다.

$$a_i = |[\frac{x}{m_1 m_2 \ldots m_{i-1}}]|_{m_i} \qquad (11)$$

4. 잉여수계를 이용한 MAC의 기본 구조

신경회로망의 기본연산은 행렬연산으로 표현되며, 이와 같은 연산은 식(8)과 같이 승산과 가산이 혼합된 누적연산을 기본구조로 한다. 모듈리를 m1,m2,m3이라 할 때, 두 입력 A, W의 누적 연산을 하는 잉여수 연산기의 기본 구조를 그림 5에 나타내었다.

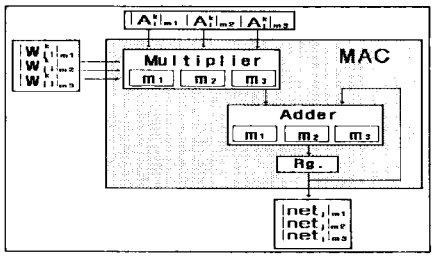


그림 5. 잉여수계를 이용한 MAC 연산기의 구성
Fig.5. The structure of MAC using residue number system

V. 순환군을 이용한 MAC 연산기의 구현

1. 순환군

집합 G가 공집합이 아니고 정수 a,b에 대하여 a,b ∈G이고, 연산 '∘'가 a∘b ∈G일 때, 이 연산이 결합법칙, 교환법칙이 성립하고 항등원 및 역원이 존재하면, 연산 '∘'에 관하여 집합 G는 군을 이루며, 〈G,∘〉로 표기한다. 또한, 〈G,∘〉에 대하여 g∈G라 할 때 생성원(generator) g에 의해서 생성된 순환 부분군 〈g〉는 다음과 같으며, G=〈g〉인 순환군이라 한다.[6][7][8]

$$\{ g^n | n \in Z \} = \{ \ldots, g^{2}, g^{1}, g^{0}, g^{-1}, g^{-2}, \ldots \}$$

(g : 생성원, g0 : 항등원, Z,g,n : 정수)

순환군 G가 위수n의 유한군이면 G={g0,g1,g2,...gn-1}이고 gn = g0이다. 모듈러 연산은 순환군을 형성하므로 모듈러 연산기는 군론을 기초로 하여 설계될 수 있다. 연산 '×'에 대한 n차 순환군 {G:×}의 연산은 표 2와 같다.

표 2. n차 순환군
Table 2. Oder n cycle group

×	g^0	g^1	g^2	. . . . .	g^{n-1}
g^0	g^0	g^1	g^2	. . . . .	g^{n-1}
g^1	g^1	g^2	g^3	. . . . .	g^0
g^2	g^2	g^3	g^4	. . . . .	g^1
.	.				.
.	.				.
.	.				.
.	.				.
g^{n-1}	g^{n-1}	g^0	g^1	. . . . .	g^{n-2}

순환군 {G:×}의 정의로 부터 mod m가산은 m차 순환군이고, 임의의 mod p승산은 (p는 소수, 0디지트는 제외) (p-1)차 순환군이다. 일반적으로 mod p 승산에 대하여, 생성원은 Fermat의 이론으로부터 다음과 같이 구 할 수 있다.[6][7][8] |gp-1|p가 가장 작은 gp-1 mod p의 잉여수일 때, g가 조건 |gp-1|p = 1을 만족한다면, 정수 g 〈 p는 mod p-1승산의 생성원이다. 예를 들어, 발생기가 2인 경우, mod 11의 승산은 10차 순환군이고, 표 3(b)와 같이 순환군을 형성하기 때문에 모듈러 연산에 이용될 수 있다. 임의의 순환군에 대해서 p가 소수인 경우 mod p-1 승산(0 디 지트 제외)에 대한 순환군은 mod (p) 가산의 순환군과 1 대 1 대응하여 사상될 수 있으므로 동형(isomorphic)이다.

표 3. mod 11 가산과 mod 10의 동형관계
Table 3. Isomorphic mod 11 addtion and mod 10 multiplication

(a) mod 11 가산(mod 11 addtion)

+	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10	0
2	2	3	4	5	6	7	8	9	10	0	1
3	3	4	5	6	7	8	9	10	0	1	2
4	4	5	6	7	8	9	10	0	1	2	3
5	5	6	7	8	9	10	0	1	2	3	4
6	6	7	8	9	10	0	1	2	3	4	5
7	7	8	9	10	0	1	2	3	4	5	6
8	8	9	10	0	1	2	3	4	5	6	7
9	9	10	0	1	2	3	4	5	6	7	8
10	10	0	1	2	3	4	5	6	7	8	9

(b) mod 10 승산(g=2)(mod 10 multiplication(g=2))

×	1	2	4	8	5	10	9	7	3	6
1	1	2	4	8	5	10	9	7	3	6
2	2	4	8	5	10	9	7	3	6	1
4	4	8	5	10	9	7	3	6	1	2
8	8	5	10	9	7	3	6	1	2	4
5	5	10	9	7	3	6	1	2	4	8
10	10	9	7	3	6	1	2	4	8	5
9	9	7	3	6	1	2	4	8	5	10
7	7	3	6	1	2	4	8	5	10	9
3	3	6	1	2	4	8	5	10	9	7
6	6	1	2	4	8	5	10	9	7	3

2. 순환군을 이용한 연산

표 3에 나타낸 순환군을 이용하여 연산기의 논리회로를 설계할 경우 순환군을 2진 부호화하여야 한다. 순환군을 2진 부호화하는 기본적인 방법은 순환군과 1 대 1 대응된 2진 부호화를 수행하여 논리회로로 설계하는 것이다. 표 2는 mod n 승산의 순환군인 경우 기본요소의 수가 n개이고 mod p의 승산을 수행하므로 p=n이며, gm은 0<gm≤(p-1)(0<m≤n-1)이므로 각 기본 요소당 소요되는 비트수 b는 b=⌈log2(n-1)⌉이다. 그러므로 순환군의 각 요소를 1 대 1 사상하여 2진 부호화 하는 경우 1개의 연산기에 n2 × ⌈log2(n-1)⌉ 이상의 논리연산이 필요하므로 연산기의 크기를 감소시키기 위하여 표 4와 같이 1-out-of-m 코드 방식을 이용하였다.

표 4. 1-out-ot-5 코드 구성
Table 4. The structure of 1-out-of-5 code

10진 수	1-out-of-10									
0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0

표4에서 알 수 있듯이 1-out-of-m코드 방식은 m개의 비트중 한개의 비트만 고유값을 나타내는 코드방식으로써, 잉여수계를 이용한 연산회로와 같이 순환군을 이루는 경우에 효율적으로 적용될 수 있다.

Ⅵ. 잉여수계를 이용한 디지털 뉴런프로세서의 구현

잉여수계 연산은 입력된 수를 선택된 모듈리에 따라 분할하고 각 모듈리별로 연산하므로, 연산기의 크기가 감소되며 모듈리간에 캐리정보가 필요없으므로 고속의 연산을 수행할 수 있다. V절의 표3과 같은 순환군과 표4의 1-out- of-m 코드방식의 부호화 기법을 이용하여 고속의 잉여수 연산기를 설계할 수 있으며, 이 설계된 연산기를 뉴런프 로세서의 핵심부분인 MAC연산부에 적용하 여 그림 6과 같은 고속의 디지탈 뉴런 프로세를 구현하고자 한다.

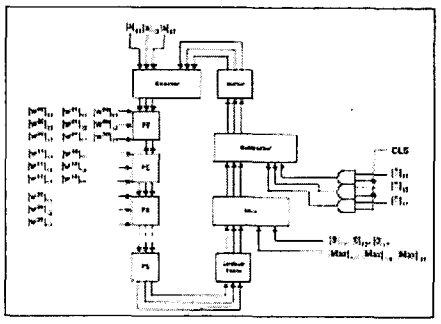


그림 6. 잉여수계를 이용한 뉴런 프로세서 의 구성
Fig.6. Configuration of neuron processor using RNS

그림 6의 뉴런프로세서는 결합강도와 입력과의 승산 및 누산을 수행하는 MAC연산부와 MAC연산부의 처리결과를 입력으로하는 시그모이드 함수연산부로 구성되며, 각 PE의 블록도는 그림 7과 같다.

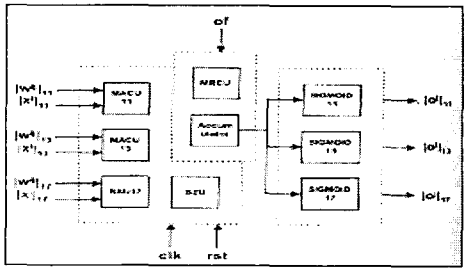


그림 7. 잉여수체계를 이용한 역전파 신경회로망의 PE블록도
Fig. 7. The block of PE for back propagation neural networks using residue number system

1. 고속 MAC 연산회로의 구현

잉여수계를 이용한 MAC연산기의 설계시 가산과 승산이 동일한 속도로 수행 된다. 잉여수계를 이용한 가

산과 승산은 표 3과 같이 순환군을 형성하기 때문에 그림 8과 같은 바렐쉬프터(barrel shifter)의 기본구조를 사용하여 고속의 연산회로를 설계할 수 있다.

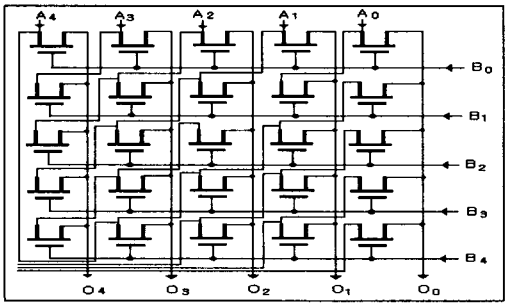


그림 8. 바렐 쉬프터의 기본형태 (5×5)
Fig. 8. Basic configuration of barrel shifter(5×5)

그림 8에서 Ai는 입력, Bi는 선택, Oi는 출력을 나타내며, 선택선 Bi에 따라, 입력된 데이터의 i 비트 순환된 결과를 출력하며, i 번 순환된 속도가 동일하므로, 배럴 쉬프터를 이용하여 순 환부호용 연산기를 구현할 수 있다. 배럴 쉬프터를 이용한 모듈러 p의 가산기와 p-1 승산기를 그림 9와 10에 나타내었다.

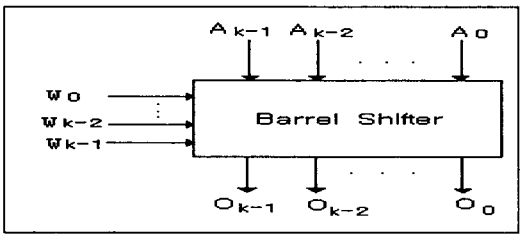


그림 9. 배럴 쉬프터를 이용한 가산기
Fig. 9. The Structure of Adder using barrel shifter

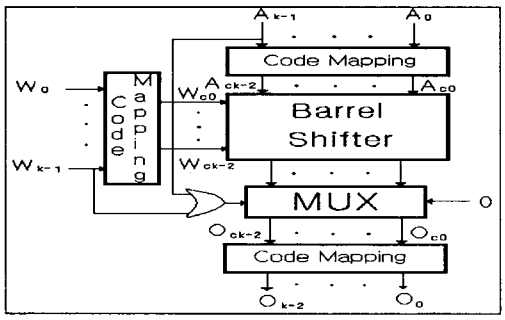


그림 10. 바렐쉬프터를 이용한 승산기의 구조
Fig. 10. The Structure of multiplier using barrel shifter

그러나, 모듈러 (p-1)의 승산기는 모듈러 p 의 가산기를 이용하여 설계하므로 승산기의 입력 및 출력시, 표 3의 동형관계에 따라 부호 정합용 해독기가 필요하며, 승산기의 입력 '0' (m0=0)인 경우는 출력이 '0'이 되도록 한다.

본 논문에서는 부호정합용 해독기를 사용하지 않기 위하여 표 4의 1-out-of-m 코드 방식을 이용하고자 한다. 이 방식은 m 비트중 한 비트만 고유값을 가지므로 부호정합시 각 비트가 1 대 1 대응되므로 입출력선을 재배열함으로써 부호정합이 가능하며, 부호를 재생성하기 위한 부가 회로가 필요하지 않다.

그림 11은 위에서 기술된 내용을 기본하여 구현된 MAC연산기를 나타낸다.

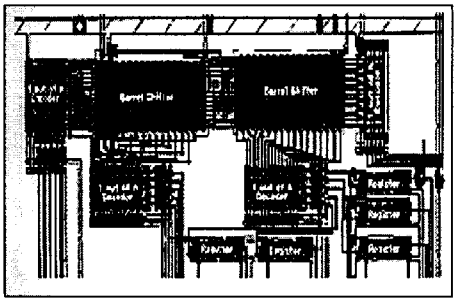


그림 11. MAC 연산부의 레이아웃
Fig. 11. The layout of MAC operation unit

그림 11의 MAC 연산부는 각 모듈 리가 11, 13, 17인 입력벡터와 연결강도의 행렬의 곱 연산을 수행한 후 누적되며, 승산과 가산 모두 1 clock으로 연산을 수행할 수 있다.

2. 잉여수계를 이용한 시그모이드 함수처리

시그모이드 함수처리는 II절의 식(2)와 같이 표현되며 이 부분은 신경망 의 디지탈회로 구현시 문제가 되는 부분이다. 본 연구에서는 이 부분을 잉여수계의 MRC(Mixed Radix Conversion)을 사용하여 구간을 분할하여 처리하고자 한다. 식(2) 에서 θj,θ0에 따라 시그모이드 함수는 그림 12와 같은 특성을 갖는다.

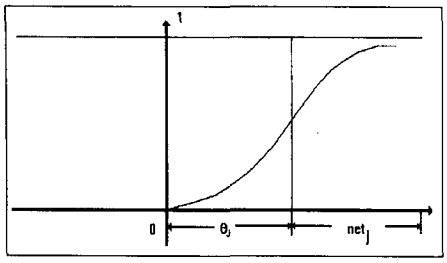


그림 12. θj 와 θ0에 의한 시그모이드 함수의 이동과 기울기
Fig. 12. Shifting and Slope of sigmoid function by θj and θ0

그림 12를 그림 13과 같이 3구간으로 분할하여 처리하고 구간 분할시 MRC를 사용하였다.

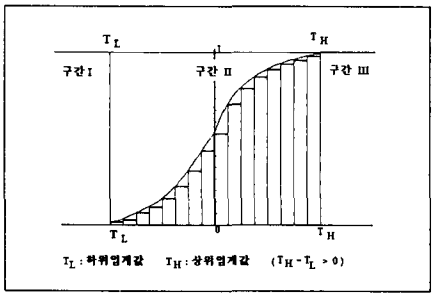


그림 13. 시그모이드 함수의 구간설정
Fig. 13. Decision of boundary in sigmoid function

시그모이드함수 처리부분을 그림 14와 같이 구성하였다.

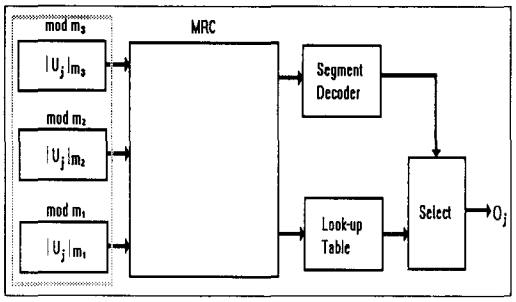
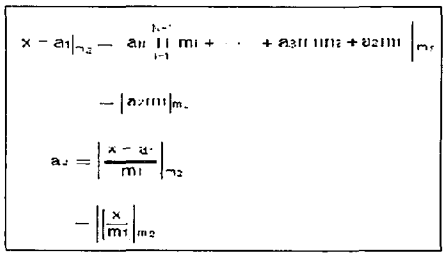


그림 14. 시그모이드 함수의 연산처리과정
Fig.14. Processing of sigmoid function

그림 14에서 MRC처리부분을 나타내면 그림 15와 같으며, 내부연산과정은 식 (12)와 같다.

(12)



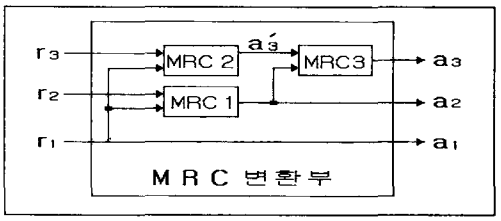


그림 15. MRC 처리부분
Fig. 15. Processing unit of Mixed radix conversion(MRC)

그림 15에서 잉여수계 입력(r1,r2,r3)를 받아서 MRC(혼합기수 변환)를 거쳐 가중 치 체계의 수로 변형된 출력(a1,a2,a3) 은 연산표(Look up table)에 저장된 잉여 수를 찾아 출력한다. 가중치 체계로 변환된 출력 a3는 연산표의 입력값으로 구 간설정에 쓰이며, a1, a2를 이용하여 그림 11에 있는 구간 II에 해당되는 값을 사전에 저장된 연산표의 값으로 출력한다.

VII. 모의 실험 및 고찰

본 논문에서 설계된 디지털 뉴런프로세서는 Full Custom방식으로 진행되었으며, LG 0.8 μm CMOS 공정을 사용하여 구현하였다.

C 언어에 의한 알고리즘의 검증이 끝난 후, Magic 4.1.3 Layout Tool로 레이아웃을 진행하였고 Compass V8r9 라이브러리를 이용하여 DRC를 수행하였다. 또한, 스파이스3(SPICE3)를 이용하여 각 연산부에 대한 논리적인 검증을 수행한후 Irsim으로 모

의실험을 수행하였다.

모의 실험의 실행조건으로 입력값과 연결강도을 각각 0 ~ 10, -32 ~ 32의 정수값으로 하였으며, 그림 13의 시그모이드 함수처리를 위하여 구간 II를 9, 11, 15등분하여 실험 하였 다. 또한, 표 3과 같이 순환군이 존재하는 잉여수중, 본 논문에서 사용한 잉여수는 모듈리 11, 13, 17로하여 실험하였다. 설계된 PE는 약 6300개의 Tr로 구성되며, Pad를 포함하여 5m × 5m크기의 die size를 가지며, 100pin QFP Pakage로 제작되었다.

본 논문에서 구현한 뉴런프로세서의 전체 Layout는 그림 16과 같다.

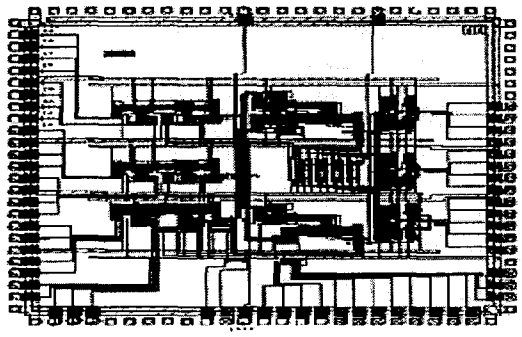


그림 16. 잉여수 체계를 이용한 뉴런프로세서의 레이아웃
Fig. 16. The layout of neuron processor using residue number system

그림 17은 본 논문에서 구현한 뉴런프로세서의 출력 파형을 나타낸다.

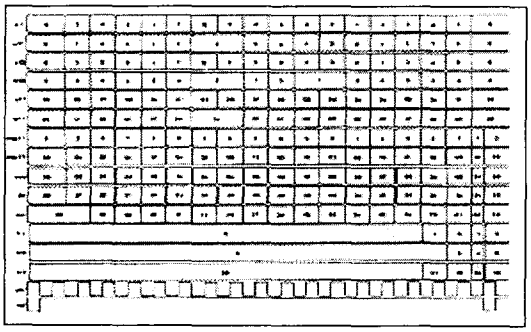


그림 17. 뉴런프로세서의 출력 파형
Fig. 17. The wave of output for neuron processor

그림 16을 이용하여 연산기를 구성할 경우, 쉬프트하는데 소요되는 시간은 트랜지스터 하나를 통과하는데 걸리는 시간과 같다.

따라서, 바렐쉬프터를 이용하여 연산기를 구성할 경우, 고속연산이 가능하며, 부호정합은 표 3의 순환부호를 사용하여 입력선의 재배열로 수행되므로 논리소자가 소요되지 않는다.

그림16에서 모듈러가 11인 경우, $|outj+1|_{11}=|Aj+1|_{11}|Bj+1|_{11} + |outj|_{11}$ 연산을 수행하며, $|Aj+1|_{11}$, $|Bj+1|_{11}$, $|outj+1|_{11}$가 각각 (112+102+44)게이트로 표현되기 때문에 일반적인 정수 연산회로에 비해 작은 하드웨어로 고속의 연산이 가능하다.

그림 16의 뉴런프로세서를 SPICE3로 실험한 결과, 0.6ns만에 연산 결과가 출력 되었다. 이 결과를 각층의 노드가 4개인 뉴런프로세서에 적용할 경우, 처음 연 산 결과를 얻는데 걸린 시간은 19.2(0.6×2×4×4)ns가 된다.

표 5는 실수 연산기를 이용한 뉴런프로세서와 모듈리 11, 13, 17를 이용하여 뉴런프로세서를 구현할 경우, 하드웨어의 크기 비교를 나타낸다.

표 5. 잉여수를 이용한 뉴런프로세서와 실수 연산기를 이용한 뉴런프로세서의 크기비교
Table 5. The comparison of size of neuron processor

연산기의 종류	연산기	연산기의크기 (Tr)		표현수의 범위	Tr수
실수연산기를 이용한 뉴런프로세서	MAC	승산기(10×10)	8748	210 = 1024	20652
		가산기(20bit)	928		
		레지스터(20bit)	828		
	시그모이드	10148(10×17)			
잉여수계를 이용한뉴런프로세서	MAC(모듈리11,13,17일 경우)	승산기 (10×10)	2316	11×13×17 =2431	12308
		가산기 (20bit)	2000		
		MUX (2:1)	288		
	시그모이드	7704			

표 5에서 알 수 있듯이 수의 범위를 1024로 할 경우, 실수 연산기를 이용한 뉴런프로세서는 총 20652 여개의 Tr로 구성되며, 모듈리 11, 13, 17를 이용한 뉴런프로세서는 12308 여개의 Tr로 구성된다. 따라서 잉여수계를 이용한 방법이 일반적인 실수 연산기를 이용하여 구현한 뉴런프로세서에 비하여 하드웨어 크기가 1/2이상 감소됨을 고찰하였다. 또한, 승산기, 가산기및 시그 모이드 함수 처리부의 속도를 각각 tm, ta, ts라 할 경우 뉴런프로세서의 속도(tN)는 tN = max(tm, ta, ts)가된다.

Ⅷ. 결론

영상신호처리 및 패턴인식 분야에서 대량의 데이타를 실시간으로 처리하여야 하는 필요성이 증가하고 있다. 컴퓨터와 사용자간의 인터페이스 문제에 있어서 실시간 처리는 중요한 해결 수단이다. 이와 같은 응용분야에 신경망을 이용하 기 위하여는 대량의 데이타를 실시간으로 처리할 수 있는 고속의 MAC 연산기와 시그모이드 함수처리를 위한 연산기가 요구된다.

잉여수 연산은 정수연산을 수행하며, 수를 각각의 모듈리로 분리하여 연산함으로 모듈리간에 캐리정보가 필요치 않다. 그러므로 연산기 크기가 감소하며, 고속의 연산기 설계가 가능하다. 순환군은 mod p-1(p:소수) 승산이 mod p의 가산과 동형이므로 부호정합에 의하여 승산기 설계가 가능하다. 그러므로 본논문에서는 순환 부호를 이용한 잉여수 연산시, 소수만을 모듈리로 사용함으로써 수범 위가 확장되는 단점을 감소시켰다. 표 3의 순환부호를 사용하여 부호정합을 입출력선의 재배열만으로 수행함으로써 연산기의 크기를 감소시켰으며, 연산속도를 향상시킬 수 있었다. 또한, 신경망 구현시 문제가 되는 시그모이드 함수 처리는 MRC를 이용하여 활성영역을 세 구간으로 분할하고 제 Ⅱ구간을 최대 모듈리를 이용하여 등분한 표본 값을 연산표(Look Up Table)에 저장하여 두고 이용함으로써 연산표의 크기및 연 산속도의 향상을 기대할 수 있다.

일반적인 실수 연산기를 이용한 뉴런프로세서에 비하여 본 논문에서 구현한 뉴런프로세서가 1/2 이상 연산기 크기가 축소하며, 연산속도가 3배 이상 향상됨을 알 수 있다.

앞으로의 연구방향은 본 논문에서 구현한뉴런프로세서를 실제 응용화하기 위한 연구가 계속되어야 할 것이다.

참고문헌

[1] D.E Rumelhart, J. L. McClelland, etc, Parallel Distributed processing ,Vol. 1., the MIT Press 1986.

[2] DARPA Neural Network study, AFCEA International press, 1987.

[3] You-Han Pao, Adaptive Pattern Recognition and Neural Networks, Addison Wesley Publishing Company Inc.1989.

[4] G. A. Carpenter, "Neural Network Models for Pattern Recognition and Associative Memory", Neural Networks, Vol.2, pp.243-257, 1989.

[5] K. Fukushima, "A Neural Network for Visual Pattern Recognition", IEEE Computers, Vol. 21, no.3, pp. 65-75, March 1988.

[6] Nicholas S. Szabo, M.S. & Richard I. Tanaka, Ph.D. Residue Arithme -tic And Its Applications to computer Technology, McGRAW Hill Book Company, 1987.

[7] K.H.O, Keefe, "A Note on Fast Base Extension for Residue Number Systems with Three Moduli", IEEE Transaction on Computers, Vol. C-24, pp. 1132-1133, November 1975.

[8] D. K. Banerji and J. A. Brzozowski, "On Translation Algorithm in Residue Number Systems", IEEE Transaction on Computers, Vol. C-21, pp. 1281-1285, December 1972.
[9] 정윤돈 "디지탈 신경회로망의 시그모이드 함수 연산회로 설계에 관한 연구", 경희대학교 대학원 전자공학과 석 사학위 논문, 1992.
[10] 조원경 외 1 "잉여수계를 이용한 디지털 신경회로의 실현", 전자공학회 논문집 제30권,B 편,2, 1993.

저 자 소 개

홍봉화
현재 세명대학교 컴퓨터과학과 교수
이지영
한국 OA학회 논문지 제3권 제2호 참조
현재 세명대학교 컴퓨터 과학과 부교수