정보학연구 제2권 제2호 1999년 11월(pp.145-161)
The Studies of Information Technology, Vol.2, No.2

잉여수계를 이용한 역전파 신경회로망 구현

홍봉화*/이호선**

요 약

본 논문에서는 캐리 전파가 없어 고속연산이 가능한 잉여 수 체계를 이용하여 고속으로 동작할 수 있는 역전파 신경회로망을 설계방법을 제안하였다. 설계된 신경회로망은 잉여수계를 이용한 MAC 연산기와 혼합계수 변환을 이용한 시그모이드 함수 연산 부로 구성되며, 설계된 회로는 VHDL로 기술하였고 Compass 툴로 합성하였다. 실험결과, 가장 나쁜 경로일 경우, 약 19nsec의 지연속도를 보였고, 기존의 실수 연산기에 비하여 약 40%정도 하드웨어 크기를 줄일 수 있었다. 본 논문에서 설계한 신경회로망은 실시간 처리를 요하는 병렬분산처리 시스템에 적용될 수 있을 것으로 기대된다.

Ⅰ. 서론

신경회로망 이론은 기존의 디지털 컴퓨터가 프로그래밍 되어진 순서에 의해 순차적으로 las 처리되어지는 것과는 달리 다수의 신경 세포들이 병렬적으로 상호 결합하여 동시에 한 문제를 처리하는 생물의 신경을 모방하여, 처리 구조를 전자회로로 실현하기 위해 연구되어 왔다.[(1)-(6)]

신경회로망 모델은 1940년대에 이미 제안되었지만 반도체 기술이 발전한 근래에 이르러 많은 발전을 하게 되었다. 현재까지 연구되고 있는 신경회로망의 응용분야는 인식(패턴 인식, 음성 인식), 제어이론, 영상처리분야 등이 있다.

최근 영상신호처리 및 패턴인식 분야에서 대량의 데이터를 실시간으로 처리 해야하는 필요성이 증가하고 있다. 컴퓨터와 사용자간의 인터페이스 문제에 있어서 실시간 처리는 중요한 해결 수단이다. 이와 같은 응용분야에 신경회로망을 이용하기 위해서는 대량의 데이터를 실시간으로 처리할 수 있는 고속의 MAC(Multiplier and Accumulator) 연산기와 판별함수를 고속으로 처리할 수 있는 연산회로가 요구된다.

신경회로망을 고속으로 수행시키려면 일반적으로, 비선형 전달 함수, 결합 강도의 계산, 학습을 위한 오류 계산 등의 고속 연산회로가 필요하다. 특히, 신경회로망의 각 노드는 결합강도와 입력의 승산 및 누적을 처리하는 반복적인 연산과정이 필요하다. 그러므로 디지털 VLSI 기술을 이용한 신경회로망을 설계할 경우, 기존의 2진 연산 방식에서는 자리올림(Carry)이 발생하여 연산속도를 저하시키고 연산기의 크기가 증대하므로 승산회로의 실현이 쉽지 않다.

잉여수계의 연산방식은 기존의 2진수 체계와는 달리 모듈리 간에 자리올림의 연산과정이 없고 가산과 승산이 동일한 속도로 처리되므로 신경회로망의 반복연산을 고속으로 처리하는데 적

* 세명대학교 컴퓨터과학과
** 주식회사 두일전자

합하다.[1][3][6]

본 본문에서는 잉여수계를 적용하여 고속의 MAC연산기와 함수 처리부를 설계하고 이 설계된 연산회로를 이용하여 고속의 디지털 신경회로망을 구현하고자 한다.

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

II. 역전파 알고리즘

신경회로망은 노드라 불리는 처리 단위(Processing element)로 구성된 병렬분산처리 구조이다. 이러한 신경회로망 이론은 병렬분산처리 특징을 갖는 많은 비 선형적 산술소자로 구성되어 있으며, 각 산술소자에 연결된 가중치를 갱신함으로써 주어진 임의의 목표치에 접근하는 동작을 한다. 그림 1은 대표적 신경회로망 모델인 다층 퍼셉트론 모델로서 은닉층을 가운데 두고 입·출력 층을 형성한다.

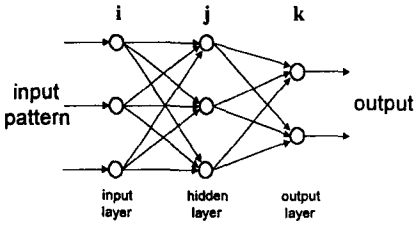


그림 1. 다층 퍼셉트론
Fig 1. The multi-layer perceptron

퍼셉트론 모델의 학습 알고리즘인 역전파 학습 알고리즘은 예제집단 패턴의 형태를 학습 식별하기 위한 효과적인 준 선형 전방향 회로망이다. 그림 1에서 단일 노드(j Layer의)를 자세히 표현하면 그림 2와 같다.[7][8][9]

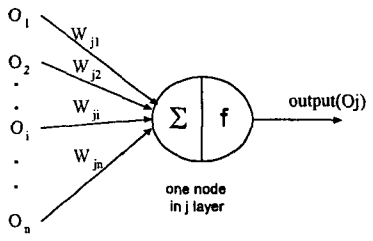


그림 2. 단일 노드의 입력과 출력
Fig 2. The input and output of single node

그림 1에서 j층의 한 노드에 대한 net_j입력은 식 (1)과 같으며 전단의 가중치(W_{ji})와 전단의 노드출력(O_i)과의 승산 및 누적 과정이 된다.

$$net_j = \sum_{i=1}^{n} W_{ji} O_i \qquad (1)$$

식 (1)에서의 net입력은 식 (2)의 시그모이드 활성함수를 통하여 출력된다.

$$O_j = f(net_j) = \frac{1}{1+\exp^{-(net_j+\theta_j)/\theta_o}} \qquad (2)$$

(θ_j : 문턱값, θ_o : 기울기)

시그모이드 활성함수는 출력값을 0과 1사이의 값으로 제한하며, 다음단 노드의 입력값이 된다.

θ_j는 문턱값(threshold)을 변화시키고, θ_o는 시그모이드 함수의 기울기를 변화시킨다. 이를 그림 3에 나타내었다.

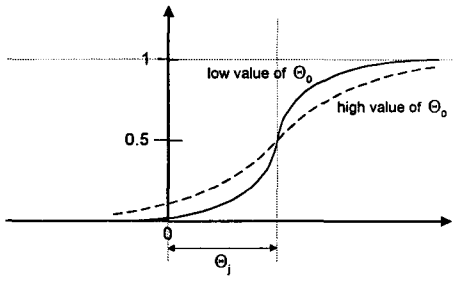


그림 3. 시그모이드 함수
Fig. 3. The sigmoid function

단일노드의 연산은 층의 순서에 의해 식 (1)과 (2)의 연산을 반복 수행하며, p번째 패턴일 경우, 최종층 k층에서의 출력 O_{pk}를 출력한다. 여기서 목표값을 t_{pk}라 하고, 실제 출력값을 O_{pk}라 하면, 오차값 E는 식 (3)의 최소자승오차(LMS: least mean square) 식으로써 구해질 수 있다.

$$E_p = \frac{1}{2}\sum_{k=1}^{n}(t_{pk}-O_{pk})^2 \quad (3)$$

실제출력과 훈련(training)패턴간의 학습과정은 가중치의 변경에 의해 처리된다. 가중치 변경은 식 (3)에서 구한 실제출력과 훈련패턴 사이의 오차 E에 의해서 조절된다. 그림 1에서 j층의 어느 한 노드의 p번째 패턴의 오차값 δ_{pj}는 식 (4)와 같다.

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

여기서 상위 층인 k층의 오차 값에 의하여 σ_{pj}는 식 (5)처럼 표현된다.

$$\sigma_{pj} = \sum_{k=1}^{n}\delta_{pk}w_{kj} \quad (5)$$

식 (4)와 (5)는 그림 1에서의 i, j층의 가중치 뿐만이 아니라 은닉층이 더욱 많을 때에도 최종층의 가중치 수정을 제외하고는 똑같이 적용한다. 최종층 k에서 p번째 패턴 오차 σ_{pk}는 다음 식에 의해서 구할 수 있다.

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

최종 층을 제외한 입력 층에서 출력 층까지의 오차 값은 식 (4)와 식 (5)에 의하여 반복적으로 계산한다. i층과 j층 사이의 t+1단계 가중치 변화량 $\triangle w_{ji}(t+1)$은 식 (7)로 구할 수 있다.

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

여기에서 η와 α는 각각 학습계수와 관성계수로서 1회의 가중치 갱신에 따른 비례 값이다. 각 층에서 계산된 오차는 역 방향으로 가중치를 순차적으로 갱신한다.

Ⅲ. 잉여 수 체계

3.1. 잉여수 표현과 연산

일반적인 2진 정수계에 의한 연산기는 캐리 정보로 인하여 가산기 및 승산회로 설계 시 문제가 된다. 특히, 대량의 입력 데이터를 처리하는 영상신호처리, 패턴인식 분야의 경우, 연산기의 크기 및 처리속도 향상에 어려움이 있다. 잉여수계는 가중치가 없는 수 체계이며, 서로 소(Relative prime number)인 모듈리(moduli)로 나눈 나머지만으로 수를 표현하여 각 모듈간에 독립성을 갖으므로 캐리 정보가 필요 없고 승산과 가산이 거의 동일한 시간에 이루어질 수 있다는 큰 장점을 갖기 때문에 대량의 데이터를 처리하는 고속의 병렬처리 연산에 유용하다.

잉여수 체계에서 모듈리를 $(m_1, m_2, \cdots, m_n)$로 취할 경우, 임의의 정수 X는 다음과 같이 표현된다.[(1)(3)(11)]

$$X = q_i m_i + r_i \ (i = 1, 2, \cdots n)$$
$$r_i = X \bmod m_i \text{ 또는 } r_i = |X|m_i \ (0 \le r_i < m_i)$$

q_i는 모듈러스 m_i에 대한 X의 몫이고 그때의 나머지는 r_i이며, X를 나머지로 표현하면 다음과 같다.

$X=\{r_1, r_2, \cdots, r_n\}$ 그리고, 정수 X의 범위는 $0 \le X < M(= m_1 \cdot m_2 \cdots m_n)$이다.

$Z = X \circ Y$의 연산을 잉여수계에서 행하면, $X=\{x_1, x_2, \cdots, x_n\}$, $Y=\{y_1, y_2, \cdots, y_n\}$로 표현되고, $Z = \{|x_1 \circ y_1|m_1, |x_2 \circ y_2|m_2, \cdots, |x_n \circ y_n|m_n\}$가 된다.

3.2. MRC 알고리즘[(1)(3)(11)]

임의의 정수 X를 혼합기수 형식으로 표현하면 다음과 같다.

$$X = a_N \prod_{i=1}^{n-1} R_i + \cdots + a_3 R_1 R_2 + a_2 R_1 + a_1 \quad (8)$$

R_i: radixes

a_i: mixed-radix digit 또는 mixed-radix 계수 $(0 \le a_i < R_i)$

$X = \langle a_n, a_{n-1}, \ldots, a_1 \rangle \ (0 \le a_i < R_i)$

잉여수로 표현된 정수 X를 혼합기수 계로 변환하기 위해 $m_i = R_i$ 라면 식 (8)은 다음과 같이 표현된다,

$$X = a_N \prod_{i=1}^{N-1} m_i + \cdots + a_3 m_1 m_2 + a_2 m_1 + a_1 \quad (9)$$

혼합기수로 변환시 혼합기수의 계수를 구하는 방법은 다음과 같다.

i) 식 (9)의 양변에 모듈로 m_1을 취하면 마지막 항을 제외하고는 모두 m_1의 곱으로 되어 있으므로, $|X|m_1 = a_1$ 즉, a_1은 첫 번째 residue digit인 r_1이 된다.

ii) 식 (8)의 양변에 a_1을 빼주고 모듈로 m_2를 취하면,

iii) i), ii)의 방식으로 나머지 혼합기수 계수를 구하게 되며, $i > 1$ 조건에서 계수 a_i는 다음과 같이 표현된다.

$$|x-a_1|_{m_2}=\left|a_N\prod_{i=1}^{N-1}m_i+\cdots+a_3m_1m_2+a_2m_1\right|_{m_2}$$

$$=|a_2m_1|_{m_2}$$

$$a_2=\left|\frac{x-a_1}{m_1}\right|_{m_2}$$

$$=\left|\left|\frac{x}{m_1}\right|\right|_{m_2}$$

$$a_i=\left|\left[\frac{x}{m_1m_2\cdots m_{i-1}}\right]\right|_{m_i} \tag{10}$$

Ⅳ. RNS를 이용한 역전파 신경 회로망 설계

역 전파 신경 화로망 모델의 전 방향 처리 과정은 임의의 입력에 대하여 목표 출력을 산출하기 위하여 각 층의 동작 값을 순차적으로 변경한다. 이 경우 각 노드의 동작 값은 식 (11)에 의해 결정되며 순환 반복적으로 수행된다.

$$\text{net}_j=\sum_{i=1}^{N}W_{ji}\cdot O_i \tag{11}$$

식 (11)을 연산하는 노드를 실수 연산기로 구성할 경우 반복적인 연산결과로 인한 속도저하와 하드웨어의 크기 증가가 문제되므로, 본 논문에서는 승산이 가산과 동일한 복잡도로 연산되는 잉여수 체계를 도입하여 그림 4와 같이 설계하였다. 입력 정보를 잉여수계로 변환하는 부분은 입력층에서만 필요하게 된다. 그림 4의 각 모듈(module)은 잉여수 연산을 수행하며, 각 모듈리에 따라 결합강도, 입출력 및 목표치 등의 잉여수 값이 연산에 사용된다.

각 모듈리 m_i에 따라 입력값(X_j)과 결합강도(W_j)의 승산 및 누적 연산이 MAC연산부에 의해서 모듈리 별로 수행되며, 연산결과가 시그모이드 함수처리 부에 입력되어 MRC연산을 거쳐 연산표에 의하여 표본화된 시그모이드 함수의 출력인 결과 출력 값(Y_j)이 생성된다. 역전파 모델의 전방향 연산과정은 식 (11)의 연산과정인 입력과 가중치의 내적 연산 부와 시그모이드 함수 처리 부로 크게 나눌 수 있다. 잉여 수 체계를 이용하여 디지털 신경회로망으로 구성할 경우, 본 논문에서 설계한 연산회로를 반복 사용하거나 각 층의 노드 수만큼 배열하여 구성할 수 있다.

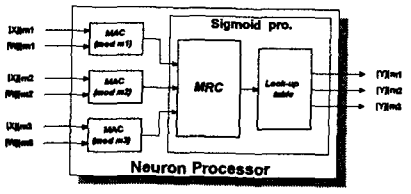
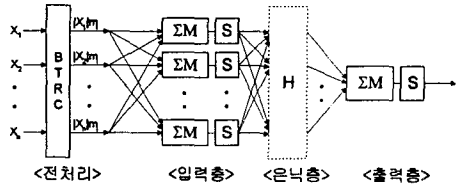


그림 4. 잉여수계를 이용한 디지털뉴런프로세서
Fig 4. The digital neuron processor using the residue number system

잉여수계를 이용할 경우 모듈별로 연산이 되기 때문에, 신경망 시스템이 모듈리 m_1, m_2,... m_i,... m_n을 사용하는 경우 i 번째 모듈리에 해당되는 처리 과정은 그림 5와 같다. 그림 6은 단일 뉴론 프로세서의 구조를 나타낸다. 신경회로망 전체의 처리과정은 그림 5의 모듈별 처리과

정의 반복적인 구조를 갖기 때문에 그림 7과 같은 구조가 된다.



$X_1 \ldots X_n$ 입력 정수

BTRC Binary to residue converter

$|X_k|m_i$ BTRC를 통해 잉여수로 변환된 입력

ΣM $|\Sigma(X \times W)|m_i$, multiply & sigma processing

S $|f(\ )|m_i$, sigmoid function processing

H 은닉층(hidden layer)

그림 5 i번째 모듈리에 대한 신경회로망의 구조
Fig 5. The structure of neural networks for moduli i th

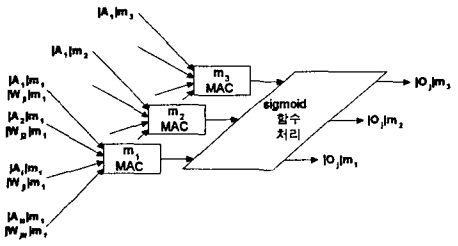


그림 6. 단일 뉴런프로세서
Fig 6. The single neuron processor

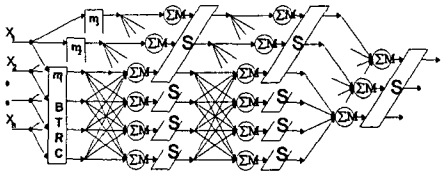


그림 7. 잉여수계를 이용한 디지털 신경회로망
Fig 7. The digital neural networks using the residue number system

4.1. MAC 연산부의 설계

단일 뉴런에서 결합강도의 순방향 MAC (Multiplier and Accumulator) 연산은 식 (12)와 같이 정의할 수 있다.

$$Rg \leftarrow Rg \pm W_i \cdot X_i \qquad (12)$$

식 (12)를 실수 연산으로 처리하기 위해서는 많은 수의 게이트가 필요하지만 본 논문에서는 잉여수계를 사용한 정수 연산만으로 회로를 간단히 하였고, 또한 잉여수계의 특징을 이용하여 승산과 가산이 동일한 속도로 가능한 고속의 MAC연산기를 그림 8과 같이 설계하였다.

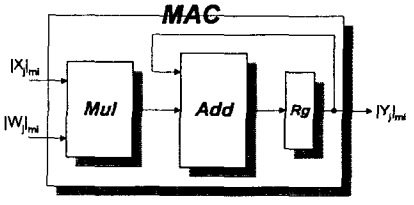


그림 8. MAC 연산 부
Fig 8. MAC operation unit

그림 8은 가산과 승산이 기본연산이 되며, 가산연산 결과를 재 입력함으로써 누산을 실행하는 MAC을 설계할 수 있다. 즉, 모듈러스 m_i인 경우 입력 데이터를 $|X_j|m_i$, 결합강도 값을 $|W_j|m_i$, 가산기의 외부 입력을 $|A_j|m_i$라할 때 기본연산 처리부의 출력 $|Y_j|m_i$는 $|Y_j|m_i = |X_j|m_i|W_j|m_i + |A_j|m_i$ 이며, 이때 출력 $|Y_j|m_i$을 다시 입력하여 MAC 연산부를 구성하면 식(13)과 같다.

$$|Y_{j+1}|m_i = |X_{j+1}|m_i \cdot |W_{j+1}|m_i + |Y_j|m_i \qquad (13)$$

그림 8에서 전체 두 입력 $|X_j|m_i$와 $|W_j|m_i$는 잉여수로 표현된 전 단(layer)의 출력과 가중치이며 전체 출력은 다시 가중치체계수로 변환되며 시그모이드 함수 처리 과정을 거치게 된다. 가산기의 다음단에 레지스터를 두어 가산기의 출력을 다시 입력으로 받아 곱셈기의 결과와 함께 가산되어지는 누산기(accumulator)의 기능을 하게 한다. 또한, 초기화(reset) 신호를 이용하여 초기값(0) 설정과 노드에서 한 패턴에 대한 연산이 끝났을 시, 누산기를 비우는(clear) 기능을 갖게 한다. 신경회로 구성시는 한 단(layer)의 노드 수만큼 누적연산을 한 후 reset 신호를 보내게 된다. 이 때, 레지스터에 있던 값은 다음단으로 보내지기 위해 시그모이드 함수 처리부로 보내지게 된다.

만일, 여기서 모듈러스 m_i를 15로 선택한다면 $|X_j|m_i$, $|W_j|m_i$, $|Y_j|m_i$가 각각 4 비트로 표현 되기 때문에 일반적인 정수 연산회로에 비해 작은 하드웨어로 고속의 연산이 가능하다. MAC 연산회로의 크기는 모듈러스 15인 경우 곱셈기와 가산기를 위한 900(15×15×4) 비트의 연산표 두개와 4비트 누산용 레지스터만으로 구성될 수 있다. 또한 승산과 가산 모두 연산표를 이용하기 때문에 연산속도가 동일하므로 회로의 동기도 용이하다.

여기서, 곱셈기의 경우는 두 입력 중 하나 이상이 0이면 출력이 0이 되어야 하므로 설계시 이 조건을 이용하게 되므로 모듈리 15의 경우, 곱셈기는 196(14×14)개의 어드레스를 갖게 되며 가산기는 225(15×15)개의 어드레스를 갖게 된다.

모듈리를 m_1=11, m_2=13, m_3=15로 정하였을 때 뉴론에 입력된 정수들의MAC 연산시의 표현 가능한 수의 범위는 $-1072 \leq M < 1072$가 된다. 입력 정수 X의 범위를 $0 \leq X \leq 14$, 결합계수 W의 범위를 $-8 \leq W \leq 8$로 제한하였을 때, 1회의 승산에서 발생 가능한 최대 값은 $14 \times (\pm 8) = \pm 112$가 된다. MAC 연산부에서 누적되는 횟수는 노드의 개수와 같으므로 최대표현범위 1072를 넘지 않는 범위내에서 노드의 최대 개수를 설정해야 할 것이다. 위의 입력 정수와 결합계수의 범위를 갖는 경우는 9개의 노드 수를 갖게 되면 1008(112×9)이 되어 1072를 넘지 않는 최대 노드 수가 된다.

4.2. 시그모이드 함수 연산부의 설계

시그모이드 함수는 무한영역의 입력에 대해 일정범위로 출력을 결정하는 판별하는 함수의 역할을 한다. 일반적인 시그모이드 함수는 식(2)의 기본식으로써 식 (14)와 같이 표현되며 식(14)의 포화상태는 0 과 1이다.[(1)(3)(7)(8)(9)]

$$S(X) = (1 + e^{-x})^{-1} \qquad (14)$$

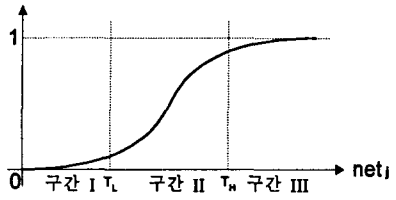


그림 9. 시그모이드 함수의 구간설정

Fig 9. The decision of a boundary for the sigmoid function

구간Ⅱ의 크기와 연산표의 값은 신경회로망 시스템의 수렴오차에 따라 고려되어야 할 것이다. 또한, 시그모이드 함수의 세 구간의 경계 값인 threshold 값을 기준으로 문턱값 이하($<T_L$)인 경우 0 (구간Ⅰ)을 출력하고, 이상($>T_H$)인 경우 1(최대값 ; 구간Ⅲ)을 출력한다. 그리고 구간Ⅱ에 해당되는 문턱 사이 값은 미리 저장된 연산 표를 통해 출력된다.

본 논문에서는 시그모이드 함수가 갖고 있는 판별특성을 살리고, 신경회로망의 디지털 구현 시 회로의 크기 감소와 동작 속도를 향상시키기 위해, 그림 10과 같이 시그모이드 함수를 표본화한 형태를 이용하였다.

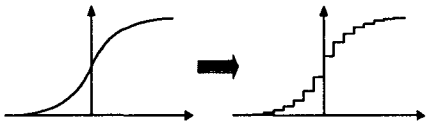


그림 10. 시그모이드 함수의 표본화

Fig. 10. The digitalization of sigmoid function

앞 절에서 기술된 바와 같이 동일한 연산기를 잉여수 체계를 이용하여 구성할 경우, 같은 수 표현 범위에서 기존의 가중치 체계를 이용하는 경우에 비하여 구조가 간단하고, 연산의 고속화가 가능하다. 또한 연산이 각 모듈리 별로 수행되고 각 모듈리 간의 캐리 정보가 없기 때문에 회로설계가 간편해지며 병렬적인 구조를 갖는다. 그러나, 잉여 수는 비 가중치 수 체계이므로 수의 크기 비교가 어려운 단점을 갖는다.

본 논문에서는 잉여수계의 단점을 보완하기 위하여 혼합기수변환 알고리즘을 이용한 시그모이드 함수 처리 부를 설계하였다. 잉여수계를 이용한 신경회로망 시스템에서, j층의 시그모이드 함수부 입력 net$_j$ 는 잉여 수 상태로 입력되고 입력된 잉여 수는 혼합기수 변환을 거쳐 연산표(Look-up table)에 저장된 시그모이드 함수 값을 출력하게 된다. 전체적인 처리과정을 간단히 나타내면 그림 11과 같다. 3개의 모듈러스 m_1, m_2, m_3를 사용하는 경우, 표본화된 시그모이드 함수를 구하는 회로는 그림 12이며, MRC 연산부, 표본화 시그모이드 함수의 연산 표로 구성된다.

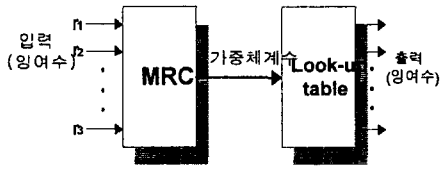


그림 11. 잉여수계를 이용한 시그모이드 함수 처리 부

Fig 11. The processing unit of sigmoid function using the residue number system

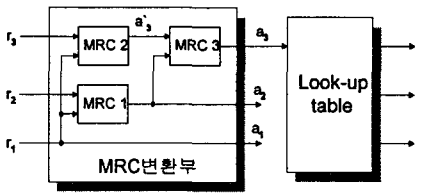


그림 12. 세 개의 모듈리를 이용한 시그모이드 함부 부

Fig 12. The sigmoid function unit using the three moduli

잉여수인 입력을 받아서 MRC(혼합기수변환)를 거쳐 가중치 체계의 수로 바꾼 다음 이 값을 어드레스로 이용하여 연산표에서 해당하는 잉여수를 찾아 출력하게 된다. 그림 13에 MRC 변환부의 내부연산을 나타내었다.

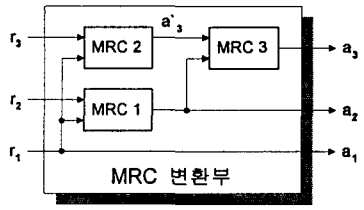


그림 13. MRC 연산부

Fig 13. The operation unit of MRC

$$\text{MRC1} : a_2 = \left| (r_2 - r_1) \left| \frac{1}{m_1} \right| m_2 \right| m_2$$

$$\text{MRC2} : a_3 = \left| (r_3 - r_1) \left| \frac{1}{m_1} \right| m_3 \right| m_3$$

$$\text{MRC3} : a_3 = \left| (a_3' - a_2) \left| \frac{1}{m_2} \right| m_3 \right| m_3$$

(15)

식 (15)의 연산식에서와 같이 MRC1, MRC2, MRC3 는 감산과 승산의 두 가지 연산을 한다. 식 (9)과 식 (10)에 의해서 MRC된 계수 a_1, a_2, a_3 를 구하게 되는데, r_1은 그 상태로 a_1이 되며, MRC1의 출력이 a_2, MRC3의 출력이 a_3가 된다. MRC 결과의 상위 계수인 a_3를 사용하여 시그모이드 함수의 구간(상위 임계값: T_H, 하위 임계값: H_L)에 따라 시그모이드 함수 부의 최종출력을 수행한다. 즉 $a_3 < T_L$ (구간 Ⅰ)이면 출력이 '0'이고, $a_3 > T_H$ (구간 Ⅲ)이면 출력은 '최대값'이며, $T_L \le a_3 \le T_H$ (구간 Ⅱ) 이면 연산표의 값이 출력된다. 연산 표는 입력값(net_j)에 해당하는 시그모이드 함수 값을 연산 표에 저장하기 위한 것으로, 저장될 함수 값과 연산표의 크기는 신경회로망 특성에 따라 사전에 구성된다. 즉, 그림 9와 같이 시그모이드 함수의 최소값(구간 Ⅰ), 최대 값(구간 Ⅲ)을 설정하고 상위 모듈리를 이용하여 전체구간을 등분하고 구간 Ⅱ에 해당하는 부분을 몇 개로 구성할 것인가를 결정한다.

본 논문에서는 사전에 저장된 시그모이드 함수 출력 값을 15등분했으므로 모듈리 15를 갖는 a_3 값만을 이용하여 연산표의 입력 값으로 사용하게 된다. 즉, MAC 연산을 거친 수를 11, 13, 15로 표현되는 가중치 체계수로 변환하고 최상위 계수인 a3은 MAC 연산후의 값을 0~14인 15단계로 표현하게 된다. 표 1은 11, 13, 15 세 개의 모듈리를 이용할 경우, 3개의 구간Ⅰ(a_3 = 0, 1, 2)과 3개의 구간Ⅲ(a_3 = 12, 13, 14)은 0과 1을 출력하고, 9개의 구간Ⅱ(a_3 = 3, 4, 5, 6, 7, 8, 9, 10, 11) 는 연산 표에 저장된 시그모이드 함수 값 출력하게 되는 것을 나타낸다.

표 1. 시그모이드 함수의 연산 표

Table 1. The look up table of sigmoid function

0~14 분포	a_3	11 r_1	13 r_2	15 r_3
14	7	3	1	14 (14)
13	6	3	1	14 (14)
12	5	3	1	14 (14)
11	4	1	12	12 (12)
10	3	0	11	11 (11)
9	2	10	10	10 (10)
8	1	9	9	9 (9)
7	0	7	7	7 (7)
6	14	5	5	5 (5)
5	13	4	4	4 (4)
4	12	3	3	3 (3)
3	11	2	2	2 (2)
2	10	0	0	0 (0)
1	9	0	0	0 (0)
0	8	0	0	0 (0)

MAC의 연산범위는 음수가 포함된 -1072 ~ 1072이므로 a_3는 0을 기준으로 위, 아래로 각각 일곱 단계가 된다. 최소 값을 0으로 봤을 때, a_3가 0, 1, 2 일 때는 0, a_3이 12, 13, 14 일 때는 14을 갖는다. 표 1에서 음영 된 부분은 잉여 수 값이 실제 연산표에 저장된 값이고 괄호 안의 값은 그에 대응되는 십진수 값이다.

4.3. 적용 알고리즘의 검증 및 설계변수의 설정

신경회로망의 동작 및 학습과정에 대한 변수는 모델의 종류와 노드의 연결 구성에 따라 다소 차이가 있으나, 보통 입력 값의 수와 그 크기, 목표 값과 목표 값에 대한 오차의 범위, 그리고 시그모이드 함수의 특성에 따라 학습 효율이 달라지게 되며, 델타룰의 경우 가중치 갱신을 조절하는 두 개의 변수(학습계수, 전달계수) 또한 학습의 효율에 영향을 미치게 된다. 또한 수렴속도는 목표 값과의 오차의 크기와 연산표의 크기에 따라 달라지므로 신경회로망의 응용분야가 얼마만큼의 정확도를 필요로 하는가에 따라 모듈리의 선택이 결정되며, 그때마다 연산표에 구성될 시그모이드 함수 값도 입출력의 범위와 오차정도에 따라 달라지게 될 것이다.[(1)(2)(3)]

본 논문에서 사용한 잉여수계를 이용한 디지탈 뉴론 프로세서의 동작 알고리즘과 변수의 설정 및 타당성을 정리하면 다음과 같다.

1) 제안된 알고리즘의 동작을 참고문헌 [5]의 GDR(Generalized-Delta Rule)과 비교하였을 때, X-OR는 1.5~6배정도 빠른 오차수렴을 보였으며, 3-패리티 문제는 4~7.5배 정도 빠른 오차 수렴 결과를 나타내었다.
2) 11, 13, 15의 세 개의 모듈리를 사용 할 경우 연산표의 크기는 2145, 195, 165, 15개의 4가지로 구분된다. 제안된 알고리즘은 목표값과의 오차가 0.005이하인 경우에 수렴을 하지 않았으며 오차가 0.005 이상인 경우에는 실수연산을 이용하는 경우보다 학습속도가 8배 정도 빠르게 수렴함을 보이며, 연산표의 크기가 15개로 사용할 경우에도 2145개를 이용한 경우와 비슷한 결과를 나타내었다.

따라서 본 논문에서 설정한 11, 13, 15의 세 개의 모듈리를 이용하고 연산표의 크기를 15개로 사용하는 잉여수계를 이용하는 뉴런 모델의 동작 알고리즘은 타당성이 있음을 확인 할 수 있다.

V. 설계 및 실험

설계한 디지탈 뉴론 프로세서는 일반적인 2진 연산이 아닌 11, 13, 15의 세 개의 모듈리를 갖는 잉여수계의 연산을 하게 되므로 내부의 모든 데이터의 흐름은 4bit가 된다. 뉴론 프로세서 모듈은 크게 MAC 연산부와 시그모이드 함수 처리부로 나뉘게 되며, 모듈리 11, 13 ,15 MAC 연산부가 각각 독립적으로 연산을 수행한다. 세 개의 MAC 연산기는 시그모이드 함수 처리 모듈(sig_top)로 연결된다. Top-block인 뉴론 프로세서 모듈(npro)은, 모듈리 11 MAC연산기(mac_11), 모듈리 13 MAC연산기(mac_13), 모듈리 15 MAC연산기(mac_15), 시그모이드 함수 처리부(sig_top) 의 네 개의 큰 Unit으로 구성된다. 본 절에서는 4개 Unit 각각의 기능을 검증하고, 동작되는 결과를 고찰한다. Compass tool 환경하(0.8μm CMOS 공정)에서 VHDL로 기술 및 합성, 시뮬레이션 하였다.

5.1. MAC 연산부

MAC 연산부는 mac_11, mac_13, mac_15 세 개의 Entity 명을 가진 세 개의 cell들로 구성된다. 잉여수 승산과 잉여수 가산을 하는 부분으로써, 승산기와 가산기 모두 Look-up table 방식을 이용한다. 세 모듈이 table값만 다를 뿐, 같은 연산을 하게 된다.

mac_11의 경우, 승산기의 두 개의 입력은 모듈리 11의 잉여 수이므로 십진수 0에서 10사이의 값으로 4bit($0000_{(2)}$~$1010_{(2)}$)가 된다. 또한, 두 개의 입력 중 하나 이상이 0일 경우는 출력 값이 0이 되어야하므로 연산표를 거치지 않고 0으로 출력하고 그 외의 100개(10×10)의 경우는 연산표에 저장하여 출력 값을 설정한다. 가산기도 역시 입력 값에 따라 121(11×11)개의 출력 값을 출력하게 된다. 가산기의 출력이 4bit이므로 레지스터도 4bit가 되며, 레지스터(reg)는 리셋신호(rst)가 low일 때에는 출력을 가산기의 입력으로 보내게 되며, rst가 high일 때 누적되어 갖고 있던 값을 모듈리 11 MAC 연산기(mac_11)의 전체 출력으로 내보내게 된다. 동시에 가산기의 초기값 설정을 위해 가산기 입력 값으로 0(0000)을 내보낸다.

그림 14는 mac_11의 합성 결과 블록으로써 가산기와 레지스터로 구성된 누적 연산부(acc_11)와 그 앞의 곱셈기(mul_11)로 구성됨을 알 수 있다. 버스로 표현된 입·출력 선은 4bit이며, 누적 연산부의 레지스터 동작을 위한 리셋신호(rst)와 출력의 동기를 위한 클럭(ck)이 있음을 볼 수 있다. 모듈리 13 MAC연산기(mac_13)와 모듈리 15 MAC연산기(mac_15)도 같은 구조를 나타낸다.

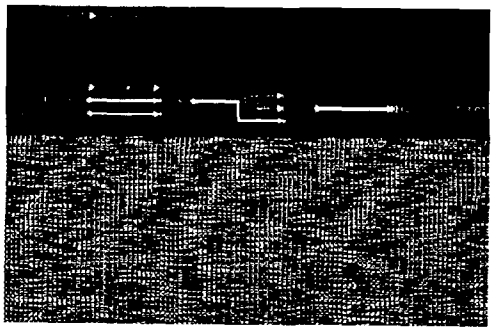


그림 14. 모듈리 11의 MAC 연산 부에 대한 합성결과

Fig 14. The result of synthesis for MAC operation unit of the moduli 11

5.2. 시그모이드 함수 처리부

식 (15)의 연산을 하는 MRC 연산부는 세 개의 cell (mrc1, mrc2, mrc3)로 구성되며, 감산과 승산의 연산을 하게 된다. 그림 15는 MRC 연산부의 합성결과 블록이다. 세 개의 MRC 연산기는 모두 연산표 방식으로 기술을 하였는데, 승산되는 값은 상수(constant)이므로 승산 후 감산까지 수행한 값들을 연산표에 두는 방식으로 VHDL 기술하였다. 예를 들어 mrc1의 경우 모듈리 13의 잉여 수와 모듈리 11의 감산이 감산결과 값은 -11에서 13사이의 값이 된다. 이 감산결과값에 상수를 곱한 결과를 연산 표에 저장하게 된다. 따라서 MRC연산도 역시 연산표에서 값을 읽어들이는 시간이 곧 연산속도가 된다.

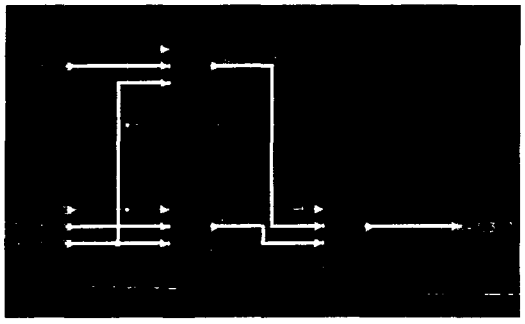


그림 15. MRC 연산 부의 합성결과

Fig 15. The result of synthesis for MRC operation unit

그림 16은 시그모이드 함수 처리부의 합성 결과 블록으로, MRC 연산 부와 시그모이드 함수 연산 표으로 구성됨을 알 수 있다. MRC 연산부의 출력은 시그모이드 함수 연산표의 입력이 되어 표 2의 값에 따라 시그모이드 함수 처리부의 출력, 즉 최종적인 뉴론 프로세서의 출력 값이 나오게 된다. 세 개의 입력신호는 MAC 연산부의 출력 즉, 잉여수계이므로 각각 4bit로 되며, 출력 역시 잉여수계의 세 모듈리 값인 4bit가 된다. 이 출력 값은 다음단 뉴론 프로세서의 세 입력이 되어 다음 단의 MAC연산부로 연결된다.

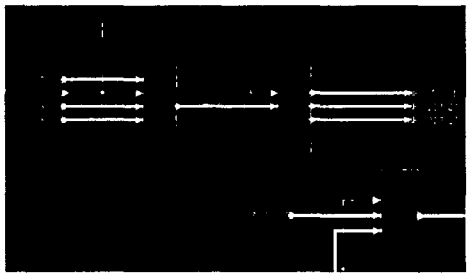


그림 17. 시그모이드 함수 처리부에 대한 합성결과

Fig 16. Result of synthesis for processing unit of sigmoid function

5.3. 뉴런 프로세서 모듈

뉴론 프로세서의 모듈은 그림 17과 같다. 세 개의 모듈리 모두 각각 4bit로 표현될 수 있으므로 입력선은 여섯 개 모두 4bit가 되며, 세 개의 출력선도 마찬가지이다. 또한, 동기를 위한 클럭신호(ck)와 MAC 연산부의 누산연산시 레지스터를 초기화하는 리셋(rst)신호가 입력된다. 세 개의 MAC연산부 Cell에서 누적 반복연산 후 리셋신호에 의해 출력되는 각각의 세 개 출력은 시그모이드 함수 처리부의 입력포트로 연결되며 MRC 연산부의 입력이 된다. 세 개의 MAC연산 모듈은 같은 구조를 갖게 되는데, 모듈리 11 MAC연산 모듈의 경우 곱셈기와 누적연산 모듈로 나뉘게 되며 누적연산은 다시 가산기와 레지스터로 나뉘게 된다.

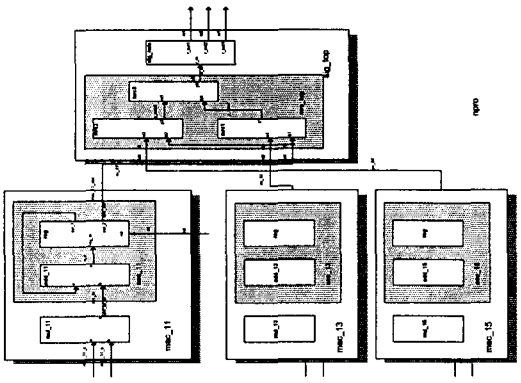


그림 17. 뉴런프로세서의 구조

Fig. 17. The structure of neuron processor

시그모이드 함수처리 부는 내부에 MRC 연산부와 연산 표로 구성되며 MRC 연산 부는 mrc1, mrc2, mrc3로 나뉘게 된다. 즉, 그림 18은 VHDL로 기술한 계층적인 cell들의 구성을 보여주고 있으며, 각 cell의 이름(entity name)과 입, 출력 포트명도 나타내고 있다. 포트연결의 이탤릭 문자로 표현된 신호선은 'signal'선언을 사용한 신호선이다.

그림 18에서 세 개의 MAC 연산부에는 각각 4bit의 두 입력이 들어가며 리셋과 클럭 신호선이 입력된다. 역시 4bit인 세 개의 MAC 연산부 출력은 시그모이드 함수 처리부로 입력되고 있다. 시그모이드 함수 처리부에서는 4bit씩 세 개의 출력이 동기 클럭에 맞추어 나오게 된다. 모든 연산 모듈은 클럭이 high일 때 출력하도록 동기를 설정하였으며, 리셋(rst) 신호가 high 일 때 레지스터에 누적되어 있던 값이 출력된다. 최상위 블록은 물론 각 모듈별로 Compass tool의 'sim' 회로 시뮬레이션을 실행한 결과 이론값과 상응하는 입, 출력 결과를 확인할 수 있었다.

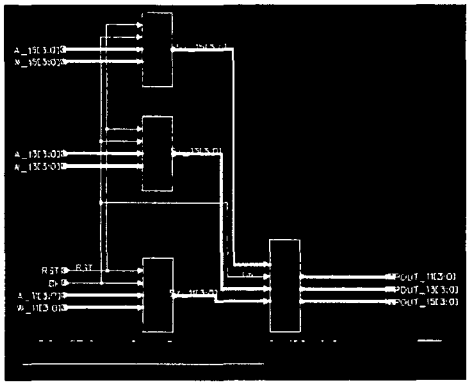


그림 18. 뉴런 프로세서에 대한 합성 최상위 블록

Fig 18. Top-block of synthesis for the neuron processor

그림 19는 뉴론 프로세서 모듈의 입·출력파형을 나타낸다. 6개의 4bit 입력 선과 3개의 4bit 출력 파형을 보이고 있으며, 리셋(rst) 신호가 high일 때, 출력이 변하는 것을 확인 할 수 있다. 초기의 가산기 입력의 부재로 인하여 출력의 delay가 생기고 있으나, 두 번째 이후부터는 차례로 출력이 나오는 것을 확인 할 수 있다. Compass의 합성 결과 약 2400개의 게이트 수를 보였으며, 19.24nsec의 critical path delay time을 나타냈다. Cell name 별로 MAC 연산기 4개와 시그모이드 함수 처리부의 게이트 수도 확인 할 수 있다. 한 개의 MAC 연산기 즉, 승산기와 누산기는 약 600 정도의 게이트 수로 표현된다.

5.4. 연산기의 크기 비교

설계한 뉴론 프로세서의 하드웨어 크기 비교를 위해 같은 조건의 연산을 수행하는 2진 연산

기와 게이트수의 비교표를 표2에 나타내었다. 같은 연산범위에서의 비교가 되어야 하므로 잉여수계 뉴론 연산기의 최대 표현 값인 2145와 비슷한 2048(2^{11})를 표현범위로 갖는 2진 수계와 비교하였다. 2진 연산기는 승산기에 많은 게이트가 필요함을 알 수 있으며, 반면 승산과 가산이 동일한 구조인 잉여수계를 이용한 연산기는 별 차이가 없음을 알 수 있다. 2진 연산기는 MRC연산과정이 필요 없으므로 시그모이드 처리 부는 잉여수계를 이용한 연산기에 비해 연산표만 필요하므로 크지 않은 게이트 수를 나타낸다. 전체 게이트 수를 비교해보면 승산기에서 유리함을 보인 본 논문에서 설계한 잉여수계를 이용한 뉴론 프로세서가 적은 게이트 수를 보임을 알 수 있다.

연산속도에서는 2진 연산기의 경우 MAC 연산부에서 불리하나 MRC연산과정이 없는 관계로 잉여수계 연산기와 비슷한 시간 지연을 나타내었다. 그러나 위의 결과는 한번의 연산과정이 기준이므로 실제 신경회로망 구성하여 반복동작을 적용시킬 시에 2진 연산기의 경,우 캐리 전파로 인한 지연이 누적되며, 캐리 정보가 전혀 없는 잉여수계를 이용한 연산기에 비해 연산속도가 저하됨을 예상할 수 있다.

따라서, 본 논문에서 설계한 잉여수계를 이용한 뉴론 프로세서는 2진 연산과정을 이용한 연산기에 비해 하드웨어 크기와 속도 면에서 유리함을 확인할 수 있다.

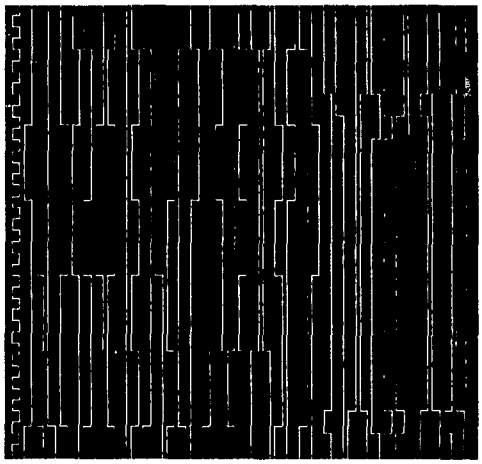


그림 19. 뉴런프로세서의 입·출력 타이밍 도
Fig 19. The timing map of input and output for the neuron processor

표 2. 연산 부의 크기 비교
Table 2. The comparing of the size for operation unit

연산기의 종류	연산기	연산기의 크기(gate)		표현수의 범위	gate 수
2진 연산기를 이용한 뉴런 프로세서	MAC	승산기 (6bit)	2187	2^{11}=2048	2829
		가산기 (12bit)	432		
		레지스터(12bit)	210		
	시그모이드	152(15x10)			
잉여수계를 이용한 뉴런 프로세서	MAC	승산기	1000	11x13x15 =2145	2403
		가산기	643		
		레지스터	219		
	시그모이드	540			

Ⅵ. 결론

영상신호처리 및 패턴인식 분야에서 대량의 데이터를 실시간으로 처리하여야 하는 필요성이

증가하고 있다. 이와 같은 응용 분야에 신경회로망을 이용하기 위해서는 대량의 데이터를 실시간으로 처리할 수 있는 고속 MAC 연산기와 시그모이드 함수처리를 위한 연산기가 요구된다. 잉여 수 체계 연산은 가중치 없이 연산을 수행하며, 수를 각각의 모듈리 단위로 연산함으로 모듈리 간에 캐리 정보가 필요치 않다. 따라서, 대량의 연산을 고속으로 요구하는 경우에 있어서 연산기의 크기가 감소하며, 고속 연산기의 설계가 가능하다. 본 논문에서는 뉴론 프로세서의 MAC 연산부를 잉여수계 연산방식을 이용하여 모듈리 단위로 설계함으로써 승산과 가산이 동일한 속도로 연산을 할 수 있도록 설계하였다. 또한, 신경회로망 구현 시 문제가 되는 시그모이드 함수 처리는 혼합기수 변환 알고리즘을 이용하여 가중치 수로 일단 표현하고 이를 이용하여 연산표의 주소 값으로 사용하였다. 이 경우, 연산 표는 세 구간으로 분할하여 제Ⅰ구간과 제Ⅲ구간은 각각 0과 최대 값을, 제Ⅱ구간에는 상위계수를 이용하여 등분한 표본 값을 저장하여두고 이용함으로써 연산표의 크기 감소 및 연산속도의 향상을 실현하였다. 즉, 디지털 신경회로망의 고속화 및 하드웨어 크기를 줄이기 위하여 역전파 신경회로망의 전방향 연산을 수행하는 단일 뉴론 프로세서를 잉여수계를 적용하여 설계하였으며, 알고리즘의 타당성 및 동작 검증을 위하여 VHDL을 이용하여 회로를 기술하고 Compass tool로 합성 및 회로 시뮬레이션을 하였다.

모의 실험결과 목표 값과의 오차가 0.005 이상인 경우 실수 연산기에 비해 빠른 수렴 결과를 보였으며, 실제 회로 합성 결과에서는 약 19ns의 클럭 속도와 약 2400개의 게이트 수를 보임으로써 약 15% 정도 하드웨어 크기를 줄일 수 있었다. 또한, 2진 연산과정의 캐리 전달을 고려할 경우, 본 논문에서 설계한 연산기가 기존의 실수 연산기에 비하여 유리함을 확인 할 수 있었다. 따라서 본 논문에서 설계한 디지털 신경회로망은 고속의 처리를 요하는 신경회로망의 응용분야에 적용될 수 있을 것으로 기대된다.

참고문헌

[1] 윤현식, 1994, 잉여수계를 이용한 고속 디지털 신경망의 설계, 경희대학교 대학원 박사 논문

[2] 정윤돈, 1992, 디지탈 신경회로망의 시그모이드 함수 연산회로 설계에 관한 연구, 경희대학교 대학원 석사 논문

[3] 조 원경 외 1, 1993, 잉여수계를 이용한 디지털 신경회로의 실현, 전자공학회 논문집 제30권, B편, 2호

[4] Arlas, L. E. and Suzuki, Y. 1990, Digital Systems for Artificial Neural Networks, IEEE Circuits and Device Magazine. : 20-24

[5] Robert. L. Harvey, 1994, Neural Network Principles, Prentice Hall International Editions. : 146-164

[6] Fornadiari, W. and Salice, F. 1995, New Architecture for the Automatic Design of Custom Digital Neural Network, IEEE trans. VLSI sys., VOL. 3, NO. 4

[7] Pao, Y. H. 1989, Adaptive Pattern Recognition and Neural Networks, Addison Wesley Publishing Company Inc. :

113-139

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

[9] Fukushima, K. 1988, A Neural Network for Visual Pattern Recognition, IEEE Computers, Vol. 21, no. 3. : 65-75

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

The Implementation of Back Propagation Neural Network using the Residue Number System

Bong-Wha, Hong/Ho-Sun, Lee*

Abstract

This paper proposes a high speed back propagation neural networks which uses the residue number system. making the high speed operation possible without carry propagation

Consisting of MAC(Multiplication and Accumulation) operator unit using Residue number system and sigmoid function operator unit using Mixed Residue Conversion is designed, The Designed circuits are descripted by VHDL and synthesized by Compass tools.

Result of simulations shows that critical path delay time is about 19nsec and the size can be reduced to 40% compared to the neural networks implemented by the real number operation unit. The proposed design circuits can be implemented in parallel distributed processing system with desired real time processing.

* Dept. of Computer Science, Semyung Univ.
** Dooil Electronis Co.