論文99-36C-4-7

학습과 시험과정 일체형 신경회로망의 하드웨어 구현

(The Implementation of Digital Neural Network with identical Learning and Testing Phase)

朴 仁 政 *, 李 天 雨 **

(In Jung Park and Chun-Woo Lee)

요 약

신경회로망은 학습 시에는 입력패턴이 변하지 않고 조정된 결합계수 값을 레지스터에 저장시키며, 시험 시에는 반대로 결합계수가 고정되고, 레지스터에 입력패턴을 기억시킴으로써 학습과 시험 뉴런회로를 공유할 수 있는 특성을 가지고 있다. 본 연구에서는 신경회로망의 이러한 특성을 고찰하여, 신경회로망 구현시 게이트의 수를 줄일 수 있으며, 학습(learning) 및 시험(testing)시의 연산처리 시간을 단축시키기 위하여 곱셈연산 대신 어드레싱LUT를 사용하여 학습과 시험이 동일한 신경회로망에서 수행할 수 있는 일체형 디지털 신경회로망 구현을 제안하였다. 제안한 신경회로망의 동작을 검증하기 위하여 수정된 오차역전파 학습 알고리듬에 의한 학습과정을 소프트웨어와 VHDL로 시뮬레이션 하였다. 7-segment 인식기 학습을 비교 검토한 결과, 입력패턴에 따라 다소 학습시간 및 학습횟수의 차이는 있지만 대체로 반복회수는 1000 ~ 10000회 정도로 학습시간은 4 ~ 200㎲로 나타났다. 신경회로망의 동작이 소프트웨어 시뮬레이션 학습 진행 상황과 동일하게 학습됨을 알 수 있었고 구현한 신경회로망이 정상적으로 수행됨을 확인하였으며, 또한 초기치 변화에 대한 실험에서도 초기치의 변화에 구애받지 않고 원할하게 학습되었다. 또한 본논문에서 구현된 신경회로망은 기존의 보드형 신경회로망보다 적은 수의 소자로 구현됨을 보였다.

Abstract

In this paper, a distributed arithmetic digital neural network with learning and testing phase implemented in a body has been studied. The proposed technique is based on the two facts; one is that the weighting coefficients adjusted will be stored in registers without shift, because input values or input patterns are not changed while learning and the other is that the input patterns stored in registers are not changed while testing. The proposed digital neural network is simulated by hardware description language such as VHDL and verified the performance that the neural network was applied to the recognition of seven-segment. To verify proposed neural networks, we compared the learning process of modified perceptron learning algorithm simulated by software with VHDL for 7-segment number recognizer. The results are as follows: There was a little difference in learning time and iteration numbers according to the input pattern, but generally the iteration numbers are 1000 to 10000 and the learning time is 4 to 200㎲. So we knew that the operation of the neural network is learned in the same way with the learning of software simulation, and the proposed neural networks are properly operated. And also the implemented neural network can be built with less amounts of components compared with board system neural network.

* 正會員, 檀國大學校 電子工學科

(Department of Electronic Engineering Dankuk Univ.)

** 正會員, 又松情報大學 電算情報系列

(Woosong Information College division of information and Communication)

接受日字:1998年5月18日, 수정완료일:1999年3月23日

Ⅰ. 서 론

신경회로망의 하드웨어 구현은 크게 디지털 방식과 아날로그 방식으로 나누인다. 아날로그 방식은 집적도와 신호의 처리 속도에 장점이 있는 반면, 신호의 저장, 시냅스를 구현하는 곱셈기의 비선형성, 동작영역의 불안정성, 잡음의 발생, 이득의 변동 등 단점이 있다. 그러나 디지털 방식의 경우, 정확도의 조정 및 유지, 데이터의 저장, 신호의 연결 및 멀티플렉싱, 회로설계, 그리고 칩(chip)간의 연결 용이성 등 장점이 있는 반면에 신호의 처리능력과 칩 면적에 상당한 영향을 미치는 곱셈연산이 요구되는 단점이 있다[1].

이러한 문제 해결을 위해 다층 퍼셉트론에 의한 고속 연산, 저 가격 및 한정된 비트로 신경회로망을 구현하려는 연구가 진행되고 있다[2-7].

그러나 한정된 비트로 하드웨어를 구현하는 경우 학습시 오차의 수렴에 문제점이 있다. 또한 고속의 연산을 위해서는 곱셈기를 이용하여야 하는데 디지털 신경회로망 구현시 수행능력과 하드웨어 구조에 상당한 영향을 주게 된다.

하드웨어를 단순하게 하기 위해 Lim[8,9] 등은 디지털 필터 설계에서 승산연산 대신 2의 거듭제곱과 2의 거듭제곱의 합에 의한 처리를 제안하였고, 고속 연산 수행을 위해 Hwang[10] 등은 한정된 정밀 오차 연산에 의한 처리를 제안하고 있다.

일반적인 신경회로망 모델에서 신경회로망의 구성요소인 신경세포의 출력은 입력과 가중치를 곱하고 다시 이들을 합한 값에 활성화 함수를 가하므로서 얻어진다.

연산에서 사용되는 가중치는 내부 메모리에서 읽어오거나(read)와 내부 메모리에 쓰기(write)가 용이하다. 활성화함수는 입력과 가중치의 곱셈에서 병목(bottleneck) 현상이 일어나지 않는다면 제한된 하드웨어 합 연산과 LUT(look-up table)을 이용하여 쉽게 구현 될 수 있다.

본 논문에서는 신경회로망의 하드웨어 구현시 곱셈연산 때문에 많이 소요되는 칩 면적을 줄이기 위해 이동(shift)과 이동-덧셈(shift-and-add) 연산자를 이용하고, 또한 어드레싱LUT에 의한 분산 연산구조를 가지며, 학습과 시험과정이 동일한 하드웨어 구조상에서 수행되는 신경회로망 구조를 제안한다.

제안한 신경회로망의 학습 알고리듬은 수정된 오차 역전파 학습 알고리듬[11]을 이용하며, 신경회로망의 동작을 확인하기 위하여 소프트웨어 시뮬레이션 방법과 디지털 방식의 신경회로망을 VHDL[12-16]로 구현하고자 한다. 한 예로서 7-segment 인식기 학습을 시뮬레이션하고 게이트 에레이 소자로 구현하며 학습과 시험 결과를 통하여 구현된 신경회로망의 동작과 성능을 확인하고자 한다.

Ⅱ. 신경회로망의 어드레싱 연산

신경세포의 구성 요소인 시냅스(synapse)는 입력값에 결합계수를 곱하고 이를 합산하는 역할을 하며, 이는 신경세포의 중요한 기능이 된다. 만약, 시냅스가 고속의 곱셈 연산을 처리하기 위해 미리 계산된 데이터를 저장하고 있는 기억 장소를 사용한다면 빠르게 그 결과를 얻을 수 있을 것이다.

또한 신경회로망의 특성을 보면, 학습과정에서 입력신호는 변하지 않으므로 이동 레지스터에는 가중치를 저장한 후 이동시키며, 메모리에는 입력신호와 가중치의 곱을 저장하고, 학습이 완료된 후 이동 레지스터에는 입력 신호를 저장되게 함으로서 시냅스 연산과 동일한 계산 결과를 얻을 수 있게 된다.

이러한 관계를 다음과 같이 수식으로 전개한다.

먼저 활성화 함수인 시그모이드 함수를 거치기 전의 신경세포의 출력 g_j를 식 (1)과 같이 표현하자.

$$g_j = w_{j1}x_1 + w_{j2}x_2 + \cdots + w_{jL}x_L \tag{1}$$

여기서 w_{jl}은 l번째 결합계수, x_l은 l번째 입력 신호이며 이러한 임의의 신호를 2의보수 형식에 의한 signed B-bit 수로 표현하고, -1 ~ +1 사이로 정규화 시키면 식 (2)와 같이 된다.

$$x_l = -x_l^0 + \sum_{q=1}^{B-1} x_l^q 2^{-q}, \quad x_l^q = 0 \text{ or } 1 \tag{2}$$

다시 g_j는 식 (1)과 식 (2)을 이용하여 signed B-bit로 정규화된 형태인 식(3)으로 주어진다.

$$\begin{aligned} g_j = {} & w_{j1}\left(\sum_{q=1}^{B-1} x_1^q 2^{-q} - x_1^0\right) + w_{j2}\left(\sum_{q=1}^{B-1} x_2^q 2^{-q}\right. \\ & \left. - x_2^0\right) + \cdots + w_{jL}\left(\sum_{q=1}^{B-1} x_L^q 2^{-q} - x_L^0\right) \end{aligned} \tag{3}$$

정규화된 출력 g_j는 학습과 시험 과정에서 다음과 같이 출력 함수로 활용된다.

1. 학습(Learning) 어드레싱 연산의 경우

입력신호 함수 ϕ_L은 L개의 서로 독립적인 2진 변수들로 구성된다고 하자.

$$\phi_L\ (\ w_{j1}{}^q,\ w_{j2}{}^q, \cdots,\ w_{jL}{}^q) = w_{j1}{}^q x_1 + w_{j2}{}^q x_2 + \cdots + w_{jL}{}^q x_L \quad (4)$$

식 (3) 과 식 (4)를 이용하여 입력신호 함수를 signed B-bit 정규화 형태로 표현하면 식 (5)와 같이 된다.

$$g_j = \sum_{q=1}^{B-1} 2^{-q}\phi_L(\ w_{j1}{}^q,\ w_{j2}{}^q, \cdots,\ w_{jL}{}^q) - \phi_L(\ w_{j1}{}^0,\ w_{j2}{}^0, \cdots,\ w_{jL}{}^0) \quad (5)$$

연속적인 j-차 비트의 각 결합계수들에 의하여 새로운 벡터를 얻을 수 있으며, 이 벡터는 메모리의 특정한 어드레스를 표시한다. 이 어드레스를 학습 시에 이용되는 어드레스 벡터라 하자.

이들 메모리 어드레스 내에는 $2^{-q}\phi_L(\ w_{j1}{}^q,\ w_{j2}{}^q, \cdots,\ w_{jL}{}^q)$와 $\phi_L(\ w_{j1}{}^0,\ w_{j2}{}^0, \cdots,\ w_{jL}{}^0)$의 값들이 저장된다.

또한 학습의 궤환(feed back)계산 결과는 점차적으로 변경된 결합계수 값 $\Delta W_{ji}(n+1)$을 필요로 한다.

학습시의 연산과정에서는, 입력신호(x_l)는 고정된 값을 갖고, 가중치(w_{ij})의 각 비트는 단지 '0' 또는 '1'의 값을 가진다. 때문에 연산시 이를 매번 계산하기보다는 저장된 메모리의 어드레스 합에 의하여 결합 계수의 변환 값을 얻는 것이 효과적이다. 이러한 방식의 연산장치를 그림 1에 표현했다.

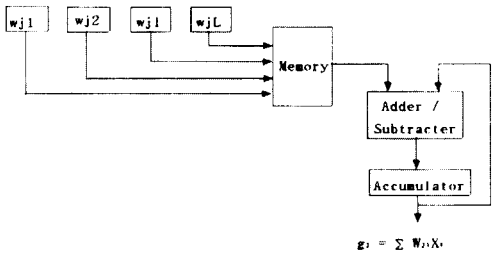


그림 1. 학습시 어드레싱에 의한 연산장치의 블록도 (입력신호 불변)

Fig. 1. Block diagram of distributed arithmetic unit in learning phase(invariant input signal).

2. 시험(Testing) 어드레싱 연산의 경우

시험시의 어드레스 벡터는 입력신호에 의해 얻어지며, 이 벡터를 $\phi_T()$라하면 식 (6)과 같이 나타낼 수 있다.

$$\phi_T() = \phi_T(\ x_1{}^q,\ x_2{}^q, \cdots,\ x_L{}^q\ ) = w_{j1}x_1{}^q + w_{j2}x_2{}^j + \cdots + w_{jL}x_L{}^q \quad (6)$$

따라서, signed B-bit로 정규화된 신경세포의 출력값은 식 (2)와 식 (6)에 의해 식 (7)로 주어진다.

$$g_j = \sum_{q=1}^{B-1} 2^{-q}\phi_T(\ x_1{}^q,\ x_2{}^q, \cdots,\ x_L{}^q) - \phi_T(\ x_1{}^q,\ x_2{}^q, \cdots,\ x_L{}^q) \quad (7)$$

시험 연산 시에는 가중치의 변화는 없으므로 입력신호와 메모리에 저장된 가중치 합에 의하여 연산 값을 얻을 수 있다. 이러한 방식의 시험 과정이 수행되기 위한 하드웨어를 그림2에 제시했다.

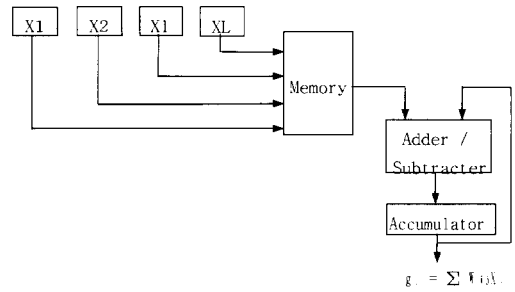


그림 2. 시험시 어드레싱에 의한 연산 장치의 블록도 (가중치 불변)

Fig. 2. Block diagram of distributed arithmetic unit in testing phase(invariant weight).

Ⅲ. 신경회로망의 구현

어드레싱LUT 연산에 의한 신경회로망의 구현은 학습 알고리듬과 하드웨어 구성 부분으로 나누어 생각할 수 있다.

1. 학습 알고리듬

신경회로망 구현에 이용된 오차 역전파 학습 알고리듬은 수정된 학습 알고리듬을 이용한다. 이는 각 패턴에 대한 개별 오차를 계산하여 이 오차가 개별 허용치에 수렴할 때까지 학습을 하게되며, 개별 오차의 값을 누적시켜 전체 허용 오차 범위에 수렴하면 학습과정을 마치게 된다.

또한 수정된 오차 역전파 알고리듬은 신경회로망의 실제 출력과 목표 출력간에 오차가 발생했을 때만 가중치가 적응된다는 것을 보여 준다. 학습률 파라미터

는 에러가 발생했을 때 빠르게 적응하기 위하여 충분히 큰 값을 가지나 적당한 값에 도달하면 가중치를 안정시키기 위하여 매우 작은 값을 취하게 된다.

가중치를 학습하기 위하여, 일련의 학습 패턴을 가진 신경회로망을 사용하며 다음과 같이 정의하자. 즉 $\{(\mathbb{X}(i), \tau_k(i)), i=1,2,...,P\}$ 여기서 $\mathbb{X}(i)$는 i번째 입력이고 $\tau_k(i)$는 i번째 목표 값이라 하자. 또한 $y_k(i)$는 훈련 쌍 i에 따르는 가중치의 벡터 $\mathbb{W}_k(i)$와 $\mathbb{X}(i)$에 응답하는 실제 출력을 나타낸다고 하자(여기서 P는 학습 패턴의 수이다). 학습 과정 동안에 각 패턴은 신경회로망에 여러 번 적용되므로 i는 학습 쌍보다 더욱더 많이 적용될 것이다. 따라서 학습 패턴의수 i가 > P 보다 클 때는 다음과 같이 주어진다.

만일 i > P 이면

$$\begin{cases} \mathbb{X}(i) \text{ 는 } & \text{시간 } i\text{에서} \\ & \text{계산된}[\mathbb{X}(.)]_{\,\mathrm{modulo}\ i} \text{ 를 의미하고} \\ \tau_k(i) \text{ 는 } & i\text{에서} \\ & \text{계산된 } [\tau_k(.)]_{\,\mathrm{modulo}\ i} \text{ 를 의미한다.} \end{cases}$$

여기서 $[\mathbb{X}(.)]$ 와 $[\tau_k(.)]$ 는 P-차인 입력 패턴을 나타낸다.

이러한 학습 과정을 요약하면 다음과 같다.

단계 1) node의 수가 n_k인 신경세포를 구성한다.

단계 2) 초기 가중치 값은 $\mathbb{W}_k(0)$로 학습 상수는 η로 설정한다.

단계 3) i=1,2,...에 대해 다음을 수행한다.

[P개의 훈련 패턴을 통한 계속적인 반복]

출력 계산: $y_k(i) = S(\mathbb{W}_k^T(i)\mathbb{X}(i))$.

오차 계산: $\varepsilon_k(i) = \tau_k(i) - y_k(i)$.

여기서 $\tau_k(i) = \begin{cases} 0, & class\ 1 \\ 1, & class\ 2. \end{cases}$

적응된 가중치: $\mathbb{W}_k(i) = \mathbb{W}_k(i-1) + \eta\varepsilon_k(i)$

단계 4) 오차가 목표치에 수렴 할 때 멈춘다.

2. 신경회로망의 하드웨어

오차역전파 학습 알고리듬을 이용하여 구현한 신경회로망은 신경세포의 게이트(gate)수를 줄이기 위하여 학습 신경세포와 시험 신경세포 부분을 서로 공유 할 수 있게 설계하여 신경회로망의 확장 및 변형에 용이하게 하였으며, 또한 신경회로망 학습시의 학습 연산시간을 줄이기 위하여 곱셈기 대신 어드레싱LUT를 이용한다.

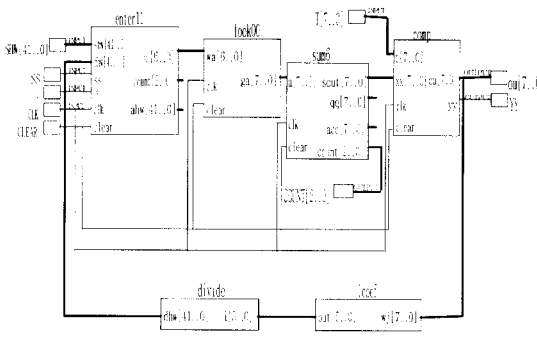


그림 3. 신경회로망의 전체 구성도
Fig. 3. The whole configuration of neural network.

이러한 점을 고려하여 구현한 디지털 신경회로망의 전체 구성도는 입력패턴과 가중치를 받아들이는 입력부, 연산된 어드레스에 의해 LUT의 값을 선택하여 변환시키는 연산부, 연산된 신경세포의 값과 목표 값의 판단에 의해 출력 뉴런을 결정하는 출력부 및 전체의 학습 및 시험 과정을 제어하는 제어부로 구성된다. 전체구성도를 그림3에 제시했다.

(1) 입력 부분

MUX에서는 입력 초기 가중치(SHW)와 누적 가중치(AHW)중 하나를 결정하는 기능을 수행하며, 이때 카운터의 값이 low이면 이전의 가중치와 변환된 가중치의 값을 누적시키며, 그렇지 않으면 이전의 가중치 값을 유지하게 된다.

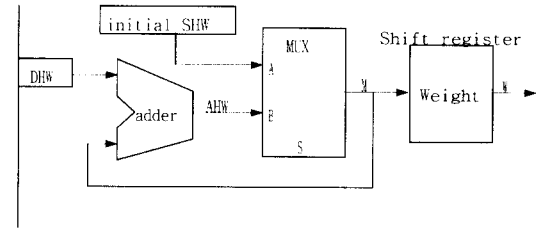


그림 4. 신경회로망의 입력부 블록도
Fig. 4. Block diagram of neural network input unit.

또한 선택신호의 값이 low이면 초기 가중치를 누적 가중치 레지스터로 이동시키며, high이면 변환 누적된 가중치를 누적 레지스터로 이동시킨다. 그림 4에 신경회로망의 입력부를 블록도로 표시했다.

(2) 계산 부분

입력패턴의 값은 어드레싱LUT의 shift and add 연산에 의해 확장 변환된 비트의 데이터를 저장하고 있는데, 입력부분 회로에서 계산된 데이터는 LUT 어드레싱의 어드레스 선택 정보가 된다.

이 정보를 이용하여 LUT의 변환 값에 접근하며. LUT에서 변환된 값은 각 가중치 비트 모두가 이동될 때까지 누적시킨다. 그림 5에 신경회로망의 연산부를 블록도로 제시했다.

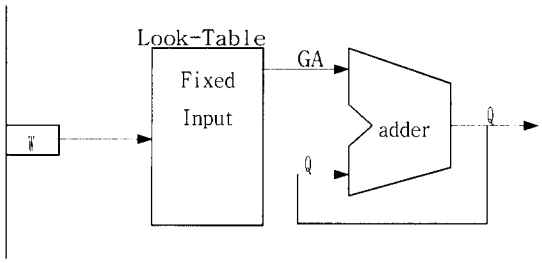


그림 5. 신경회로망의 연산부 블록도
Fig. 5. Block diagram of neural network arithmetic unit.

(3) 출력 부분

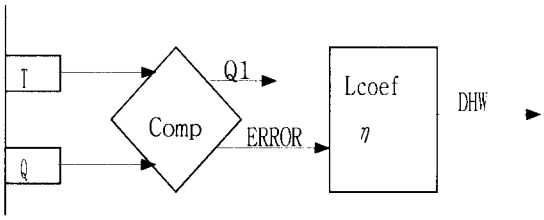


그림 6. 신경회로망의 출력부 블록도
Fig. 6. Block diagram of neural network output unit.

연산처리 과정을 거친 신경회로망의 학습판정은 신경회로망의 출력 값과 목표 값을 검토하여 학습한계 범위 내에 수렴하게 되면 학습을 끝내게 된다. 그렇지 않으면 오차값에 학습계수(η)와 가중치 연산에 의해 변환된 값을 입력부분으로 궤환시켜 학습을 계속하게 한다.

(4) 제어부분

학습 및 시험과정에서 출력된 오차값과 학습계수(η)를 누적시킨 미소 변환 값(DHW)을 각 입력패턴의 각 가중치 입력부분으로 궤환시켜 허용 한계오차에 수렴 될 때까지 학습 및 시험의 반복 과정을 제어한다.

Ⅳ. 실험 및 검토

제안한 디지털 신경회로망의 동작을 검증하기 위하여 구현한 다층 퍼셉트론은 입력층 7개, 은닉층 1개 및 출력층 10개로 구성했다. 각 가중치와 입력은 6 비트로 연산처리되며 하드웨어를 VHDL로 기술하고 게이트 어레이로 구현하여 7-segment 인식기 학습을 수행하였다.

실험에 적용한 입력 패턴의 수는 7개로, clock은 50 ns로 각 초기 치는 $000001_{(2)}$, $000010_{(2)}$과 $001000_{(2)}$으로 오차의 한계는 $00000001_{(2)}$로 하였다.

또한 사용된 입력패턴의 값은 최고치 0.935로, 최소치는 0.0625로 나타내는 실수 데이터를 표 1과 같이 8 비트 2진 값으로 변환하여 사용하였다.

표 1. 8비트 어드레스 표시방법
Table 1. Convert method for 8bits addressing.

sign (0 또는 1)	2^2(4)	2^1(2)	2^0(1)	2^{-1}(0.5)

sign (0 또는 1)	2^{-2}(0.25)	2^{-3}(0.125)	2^{-4}(0.0625)

7-segment의 0(zero) ~ 9(nine)의 입력 패턴 a ~ f는 0.0625 ~ 0.9375로 최소 값과 최고 값을 갖게 하였다. 각 segment마다 10개의 입력패턴을 가지며, 이를 각 초기 치에 적용하여 모두 350회 수행하였다.

1. 학습 시간 실험

7-segment 인시기의 입력패턴을 low와 high값으로 구성하여 학습을 시킨 결과, low 입력패턴의 경우 대체로 100μs내에서 학습이 이루어 졌으며, high 입력 패턴의 경우는 패턴의 성격에 따라 학습시간이 약 3 ~ 200μs정도로 나타났다.

또한 입력패턴의 값을 임의로 low와 high값으로 구성한 실험에서는 low, high로 입력된 패턴 보다 학습시간이 적게 소요되었다.

이를 소프트웨어 시뮬레이션 학습과 비교하여 보면, 구현한 신경회로망 학습 진행이 거의 일치하였고, 입력패턴의 값에도 매우 동일한 학습의 결과를 보였다. 따라서 제안한 신경회로망의 학습이 정상적으로 수행됨을 확인하였다. 표 2와 표 3에 이러한 결과를 제시

했다.

표 2. Low 입력패턴 데이터 학습결과
Table 2. Learning result for low value Input data.

입력패턴	입력신호(L) a	b	c	d	e	f	g	교사신호
0	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	01100000
1	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.4375 (000111)	0.4375 (000111)	0.4375 (000111)	00100000
2	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.5625 (001001)	01010000
3	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.4375 (000111)	0.5625 (001001)	01010000
4	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	01000000
5	0.5625 (001001)	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	01010000
6	0.5625 (001001)	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	01100000
7	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.4375 (000111)	0.4375 (000111)	0.4375 (000111)	00110000
8	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	01110000
9	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.5625 (001001)	0.4375 (000111)	0.5625 (001001)	0.5625 (001001)	01100000

표 3. High 입력패턴 데이터에 의한 학습결과
Table 3. Learning result for high value input data.

입력패턴	입력신호(H) a	b	c	d	e	f	g	교사신호
0	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	01100000
1	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.0625 (000001)	0.0625 (000001)	0.0625 (000001)	00100000
2	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.9375 (001111)	01010000
3	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.0625 (000001)	0.9375 (001111)	01010000
4	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	01000000
5	0.9375 (001111)	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	01010000
6	0.9375 (001111)	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	01100000
7	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.0625 (000001)	0.0625 (000001)	0.0625 (000001)	00110000
8	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	01110000
9	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.9375 (001111)	0.0625 (000001)	0.9375 (001111)	0.9375 (001111)	01100000

2. 초기치 변화 실험

소프트웨어 시뮬레이션에 있어서 초기 가중치의 설정이 신경회로망의 학습에 중요한 변수가 되므로 구현한 신경회로망의 초기치 환경을 검증하기 위해 다양한 초기치에 따른 실험을 하였다.

표 4. 7-segment 인식기의 평균 학습시간 및 반복회수
Table 4. Average learning time and number of iterations for 7-segment recognizer.

입력패턴	VHDL 구현 실험 (단위: μsec) low	high	mixed	반복회수 S/W	VHDL
0	173.37	59.17	46.87	1025	1000
1	15.17	13.97	13.25	980	300
2	9.57	7.97	6.57	990	200
3	34.72	17.57	15.89	967	600
4	48.57	42.37	34.65	1117	800
5	47.17	40.56	39.82	1190	800
6	27.83	8.87	7.88	1164	200
7	3.97	3.97	3.97	978	80
8	11.17	3.17	3.17	997	200
9	27.34	8.77	7.88	1080	200

초기값을 0.00625 ~ 0.5로 변화시키면서 시뮬레이션 학습과 VHDL 학습을 수행한 결과를 표 4에 제시했다. 표4에 나타난 바와 같이 소프트웨어 시뮬레이션에서는 초기치의 변화에 따라 학습의 차가 크나, VHDL 시뮬레이션에서는 초기 값 변화에 크게 적용을 받지 않음을 알 수 있었다.

초기치가 0.00625로 주었을 때가 가장 수렴이 빨랐으며, 0.5로 주었을 시에는 입력 패턴에 따라 다소 차이는 있지만 약 5μs정도 시간이 더 소요되었다.

이는 설계된 신경회로망이 초기 치에 큰 영향을 받지 않음을 알 수 있었다. 구현된 신경회로망의 초기치는 0.00625가 적당하다고 보이며, 시뮬레이션 학습에서는 초기치 설정에 상당한 어려움이 있으나 구현된 신경회로망은 이러한 점을 해소 할 수 있었다.

표 5. 소프트웨어 시뮬레이션 결과
Table 5. Software simulation results.

	simulation 결과 (단위: 회수)		
초기값	low	high	mixed
0.9	5498	1117	2003
0.5	8636	2684	3858
0.25	16381	4454	7783
0.125	33527	9412	15578
0.0625	73741	19347	20514

표 6. 초기값 변화에 따른 VHDL 학습시간
Table 6. VHDL learning time depending on initial value.

입력 패턴	VHDL 초기 값 (단위: μsec)											
	0.5			0.25			0.125			0.0625		
	L	H	M	L	H	M	L	H	M	L	H	M
0	94.37	59.17	46.87	93.55	58.65	52.14	96.12	55.21	54.12	89.46	56.56	45.89
1	15.17	13.17	13.25	12.37	165.18	15.23	14.96	161.35	16.15	16.12	13.25	12.56
2	97.57	7.97	6.57	88.54	8.65	7.89	96.13	9.24	8.02	95.65	95.45	7.21
3	34.72	17.57	15.89	39.25	15.26	16.34	35.12	16.25	15.24	33.54	33.45	14.65
4	48.57	62.37	34.65	38.12	60.45	38.64	48.33	59.32	37.86	46.55	45.32	33.25
5	47.17	181.56	39.82	39.12	175.23	37.84	45.16	175.32	35.12	48.12	16.32	38.55
6	27.83	8.87	7.88	30.12	9.14	9.21	26.35	9.35	10.25	29.25	25.46	8.02
7	3.97	3.97	3.97	4.09	4.96	3.28	4.36	4.02	4.36	4.12	4.25	4.05
8	11.17	3.17	3.17	13.28	4.25	4.21	15.11	4.68	4.06	12.11	12.65	4.01
9	27.34	8.77	7.88	25.46	9.28	8.24	26.38	9.04	8.05	23.68	26.55	8.04

그림 7은 segment값 "0"의 low 입력패턴 학습의 경우 학습이 완료된 상태의 타이밍도이다. 여기서 SHW [0..42] 는 초기 가중치이고, 목표변수 T는 "1"을 표 1에 의해 변환된 값 $00100000_{(2)}$으로 주었다. 학습완료 오차변수 OU가 $00000000_{(2)}$으로 나타난 것은 학습 시작한지 약 173.55μs 후이며, 학습이 완료됨을을 의미한다.

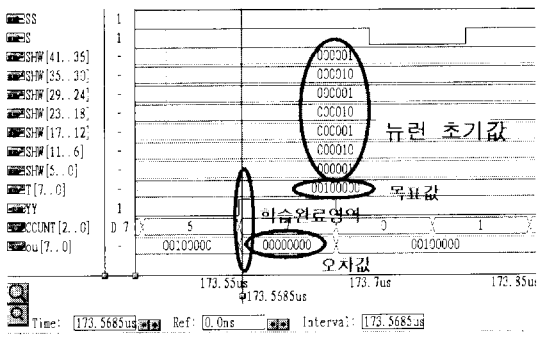


그림 7. '0' segment의 타이밍도
Fig. 7. Timing diagram of segment '0'.

3. 하드웨어의 소자수 비교

일체형 연산 방식에 의해 구현된 신경회로망에서 플립플롭의 수는 150개, Logic Cell(LC)은 520개가 필요하였으며 표 7에 그 결과를 제시했다.

보드시스템 [7] 으로 구현된 신경회로망(6비트 분해능)은 8개 신경세포를 Xilinx사의 XC4005-5PG156 4개 사용하여 구현하였다. 본 연구에서는 신경회로망의 분해능을 7비트로 하였고 ALTERA사의 EPF8820GC192으로 구현하였으며 각 항목별 비교 갯수를 표 8에 나타내었다.

표 8. 구현된 신경회로망의 플립플롭 및 LC 갯수
Table 8. Number of flip-flops and that of LC for the implemented neural network.

Entity	플립플롭의 수	LC 수	cell 분해능 (비트)
Enter	115	284	6
Look_up	7	117	7
Sum	19	46	8
Compare	9	25	8
Coefficient	0	6	8
Divide	0	42	6

표 8에서 보듯이 본논문에서 구현한 신경회로망과 보드시스템으로 구현한 신경회로망을 비교해 보면 본 신경회로망이 게이트 수에서 2500개 적고 플립플롭 수는 약 450개 적다. 그러나 Logic Cell의 수는 많은 것으로 나타났는데 이것은 다수 기능으로 분리되어 구성되어 있음을 의미한다.

표 8. 구현된 신경회로망과 보드 시스템 신경회로망의 비교
Table 8. Comparison between implemented neural network and board system neural network.

	게이트 수	플립플롭 수	Logic Cell 수
구현된 신경회로망	2500	160	520
보드 시스템 신경회로망	5000	616	196

V. 결 론

본 연구에서는 회로 구현시의 게이트 수를 줄이기 위하여 학습과 시험과정이 동일한 하드웨어에서 이루어 질 수 있으며, 연산시간 단축을 위해 곱셈기 대신 add-and-shift연산에 의한 어드레싱LUT를 사용하는 신경회로망을 제안하였다.

제안한 신경회로망은 시그모이드 연산 대신 add-and shift에 의한 어드레싱으로 뉴런의 값을 처리하였다. 7-segment 인식기 학습결과 입력패턴의 값

이 비슷한 패턴으로 구성된 경우 보다 조금씩 다르게 구성된 패턴의 학습이 보다 효과적이었으며, 또한 초기치에는 거의 영향을 받지 않고 학습되었다.

이는 프로그램에 의한 시뮬레이션 학습방법이 초기값 설정에 따라 신경회로망 전체의 성능이 크게 변하지만 제안한 신경회로망이 매우 안정적인 동작을 함이 확인되었다.

하드웨어의 소자수를 본논문에서 구현된 신경회로망과 보드 시스템 형태의 신경회로망을 비교해 보면 플립플롭과 게이트 갯수에서 본논문에서 구현된 신경회로망이 단연 적었다.

지금의 제안은 단일 신경망에 의한 방법이므로 추후 확장이 용이한 일체형으로 구현을 한다면 소프트웨어 시뮬레이션처리에서 어려운 실시간 처리가 가능할 것으로 사료된다.

참 고 문 헌

[1] D. E. Rumelhart and J. L. McCleland, Eddisons, Parallel Distributed Processing Vol. I and II, New York, 1986.

[2] B. A. White and M. I. Elmasry, "The digi-neocognition :a digital neocognition neural network model for VLSI," IEEE Trans.Neural Networks, Vol. 3, pp. 73-85, Jan. 1992.

[3] H. P. Gart, L. D. Jacckel and W. E. Hubbard, "VLSI Implementation of Neural Network Model," IEEE Trans. on Computer, pp.41-49, March 1988.

[4] J. J Vidal, "Implementing neural Nets with Programmable Logic", IEEE Trans. ASSP, Vol. 36, Np7, pp.1180-1190, July 1988.

[5] B. Linares-Barrance, E. Sanchez-Sinencio, A. Rodriguez-Vazquez, and J. L. Huertas, "A CMOS analog adaptive BAM with on-Chip learning and weight refreshing, "IEEE Trans. Neural Networks, vol. 4, pp. 445-455, May 1993.

[6] B. K. Dolenko and H. C. Card, "The effects of analog hardware properties on back-prorpgation networks with on chip learning, "Proc. ICNN'93 San Francisco, vol. I, pp.110-115, Mar. 1993.

[7] Edurd Sackinger and Hans-Peter Graf, "A Board System for High-Speed Anaysis and Neural networks, "IEEE tran. neural network., Vol. 7. No. 1., pp. 214-221, 1996.

[8] Y. C. Lim and B. Liu, "Design of cascade from FIR filters with discrete valued coefficients, "IEEE trans. Acoust., Speech, Signal Proc., vol. ASSP-36, pp. 1735-1739, 1988.

[9] Z. Jiang, "FIR filter design and implementation with powers-of-two coefficients, "in Proc. IEEE ICASSP'89, Glasgow, U.K., May 1989, pp. 1239-1242.

[10] S. Kung and J. N. Hwang, "parallel architecturs for artificial neural nets," in proc. IEEE Int. Conf. Neural Networks, San Diego, CA, 1988, vol. II, pp. 165-172.

[11] I. J. Park, C. W. Lee and H. S. Jang,"A Fast algorithm for Training Multilayer Perceptron Model of Neural Network", Proc. NNASP'93 Singapore, Vol. I, pp. 311-317, Aug. 1993.

[12] "MAX+PLUS II User Guide", ALTERA, 1992.

[13] "MAX+PLUS II Graphic & symbol Editor", ALTERA, 1992.

[14] "MAX+PLUS II Compiler", ALTERA, 1992.

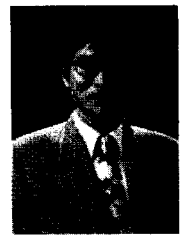
[15] "MAX+PLUS II Programmer", ALTERA, 1992.

[16] "MAX+PLUS II Simulator, Timing Analyzer & Waveform Editor", ALTERA, 1992.

저 자 소 개

朴 仁 政(正會員)

1948년 1월 28일생. 1974년 2월 고려대학교 전자공학과 졸업(공학사). 1980년 9월 고려대학교 대학원 전자공학과 졸업(공학석사). 1986년 2월 고려대학교 대학원 전자공학과 졸업(공학박사). 1977년 6월 ~ 1979년 2월 대한통신공업주식회사. 1981년 3월 ~ 현재 단국대학교 전자공학과 교수. 1988년 10월 ~ 1989년 7월 미국 Bowling Green 주립대학 객원교수. 1996년 1월 ~ 현재 대한전자공학회 교육이사. 1999년 1월 ~ 현재 대한전자공학회 멀티미디어 연구회 위원장. 주관심 분야는 멀티미디어 통합 신호처리 소자, 멀티미디어 통신, 신경망 등임



李 天 雨(正會員)

1957년 12월 29일생. 1980년 2월 영남대학교 전자공학과 졸업(공학사). 1988년 8월 영남대학교 대학원 전자공학과 졸업(공학석사). 1995년 2월 단국대학교 대학원 전자공학과 박사과정 수료. 1983년 3월 ~ 19985년 2월 한국전자통신연구소 근무. 1985년 3월 ~ 우송정보대학 전산정보계열 교수. 주관심분야는 신경망, 멀티미디어, 음성인식 등임