論 文
53D-12-3

범용 신경망 연산기(ERNIE)를 위한 학습 모듈 설계

Design of Learning Module for ERNIE (ERNIE : Expansible & Reconfigurable Neuro Informatics Engine)

鄭 露 敎† · 魏 載 玗* · 董 聖 秀** · 李 鍾 浩***
(Je Kyo Jung · Jae Woo Wee · Sung Soo Dong · Chong Ho Lee)

Abstract - There are two important things for the general purpose neural network processor. The first is a capability to build various structures of neural network, and the second is to be able to support suitable learning method for that neural network. Some way to process various learning algorithms is required for on-chip learning, because the more neural network types are to be handled, the more learning methods need to be built into. In this paper, an improved hardware structure is proposed to compute various kinds of learning algorithms flexibly. The hardware structure is based on the existing modular neural network structure. It doesn't need to add a new circuit or a new program for the learning process. It is shown that rearrangements of the existing processing elements can produce several neural network learning modules. The performance and utilization of this module are analyzed by comparing with other neural network chips.

Key Words : Neural Network, Learning Module, FPGA, Verilog HDL

1. 서 론

신경망은 인간의 뇌신경 세포의 자극에 대한 반응이라는 정보 전달 과정에서 아이디어를 얻어 수학적으로 모델링 한 것으로, 1943년 W.McCulloch 와 W.Pitts가 처음으로 신경망이라는 개념을 제시하였다. 1949년 D.Hebb이 신경망의 학습법을 제안함으로써 학습이라는 개념이 처음으로 등장하였으며, 이후로 ADALINE, Perceptron, SOM 등 다양한 신경망 구조와 학습 알고리즘이 연구되었다. 주로 신경망이 응용되는 분야는 연상, 인식, 추론, 비선형 제어 등 기존 알고리즘으로는 해결하기 어려운 곳에서 적용되며, 실제 응용 예를 보면 필기체 인식[1], 전문가 시스템의 일종인 질병 진단[2], 인공지능[3] 등 분명한 해법이 증명된 곳이 아니라 통계적인 방법으로 근사 값을 찾는 곳에 신경망이 주로 응용된다.

1958년 F.Rosenblatt이 발표한 MARK I 퍼셉트론을 시작으로 현재는 다양한 아날로그 및 디지털 VLSI 기술로 신경망이 구현되고 있다. 본 논문에서 사용하는 ERNIE (Expansible & Reconfigurable Neuro Informatics Engine) [5]는 디지털 회로로 구현된 신경회로망으로 다양한 신경망을 자유롭게 구성할 수 있도록 설계된 신경망 회로이다. ERNIE는 다수의 SPE(Synapse Processing Element)와 하나의 LPE (Layer Processing Element)가 하나의 신경망 레이어를 구성할 수 있는 최소 단위인 MPU(Modular Processing Unit)를 구성하게 된다. 이러한 다수의 연산기를 적절히 조합함으로서 시냅스의 확장이 가능하고 신경망의 구성이 자유롭기 때문에 다양한 구조의 신경망을 구성할 수 있으며 재구성이 가능함과 동시에 대규모 확장이 가능하다. 이러한 장점으로 여러 가지 신경망 실험에 적용할 수 있고 빠른 연산 결과를 얻기에 적합하지만, 신경망 프로세서로 사용하기 위해서는 학습 기능을 구현하여 신경망 칩 자체적으로 학습이 가능해야 한다.

범용 신경망칩에서 자유로운 구성과 확장성 등이 강조되는 이유는 다음과 같은 신경망의 특성에 있다. 신경망은 여러 분야에 폭넓게 응용될 수 있으나, 실제 적용을 위해서는 몇 가지 고려해야할 사항이 있다. 가장 먼저 고려해야할 사항은 응용분야에 적합한 신경망 모델을 선택하는 것이다. 적절한 신경망 모델을 선택하지 않으면 원하는 결과를 얻는데 실패하게 된다. 다음 사항은 신경망의 구조를 결정하는 일이다. 단층 신경망을 구성할 것인지 다층으로 구성할 것인지, 각 층에는 몇 개의 뉴런이 배치될 것인지와 신경망의 입출력을 결정하는 등의 일이다. 신경망의 성능을 위해서 최적의 구성을 찾는 것이 중요하다. 마지막으로 신경망의 학습이 있다. 신경망이 설계자가 원하는 동작을 하기 위해서는 신경망을 학습이라는 과정을 거쳐 적절한 시냅스의 연결강도를 얻어야 한다. 그 밖에도 신경망의 입력을 선택하는 학습 패턴 선정과 특징 추출, 전처리 과정 등의 사항이 있다. 따라서 범용 신경망 연산기로서 활용이 되기 위해서는 신경망 모델 구성의 범위가 넓어야 하고, 학습 모듈 구성 역시 신경망 모델에 따라 적절히 구성될 수 있어야 한다.

학습 모듈을 구현하는 방법은 두 가지를 생각해 볼 수 있다. 첫째는 연산용 CPU를 추가해서 학습을 위한 프로그램을 직접 제작하는 방법이다. 둘째는 학습 알고리즘에 맞는 학습

† 교신저자, 學生會員 : 仁荷大 工大 情報通信工學科 碩士
E-mail : olivetti@netian.com
* 正 會 員 : 仁荷大 工大 電氣工學科 博士課程
** 正 會 員 : 龍仁松潭大 디지털電子情報科 敎授 · 工碩
*** 正 會 員 : 仁荷大 工大 情報通信工學科 敎授 · 工博
接受日字 : 2004년 6월 29일
最終完了 : 2004년 11월 8일

모듈을 별도로 설계해서 필요에 따라 선택하여 연산을 하는 방법이 있다. 전자는 프로그램이라는 특성상 구현과 디버깅이 비교적 용이하며 학습 모듈의 구현 범위가 대단히 넓고 자유롭다는 장점이 있다. 또한, 프로그램에 따라 신경망 칩과 다른 회로간의 인터페이스와 데이터 입출력을 제어해 줄 수 있다는 장점을 얻을 수 있으나, 신경망칩에 보조 연산용 CPU를 추가해야 하고 프로그램을 저장할 수 있는 저장 수단이 필요하다는 점에서 칩의 면적이 증가하는 단점이 있다. 후자는 전용 학습 모듈을 사용하므로 구성이 간단하고 프로그램 저장용 메모리가 필요하지는 않으나, 적용되는 학습 알고리즘이 증가할수록 연산 모듈의 수도 많아지게 된다. 더구나 각각 한 가지 알고리즘 연산에만 동작하기 때문에 칩의 면적 증가로 인하여 자원 활용에 있어서 효율이 떨어지게 된다. 또한, 이미 만들어진 학습 기능만 수행할 수 있다는 점에서 전자와 후자의 방법 모두 응용분야에 대한 자유도가 떨어지게 된다. 칩 면적의 증가는 전력 소모의 증가와 생산성 저하로 신경망 칩의 유용성을 떨어뜨리게 되고, 다양한 학습 모듈 구현이 어려워진다면 실제 응용분야의 폭이 좁아질 수 밖에 없게 된다. 본 논문에서는 앞서 언급한 학습 모듈 증가에 따른 면적 증가와 범용성 감소라는 문제점을 해결하기 위하여 기존의 재구성형 모듈러 구조를 응용한 새로운 학습 모듈 구현 방법을 제안하였다. 기존의 SPE와 LPE를 적절히 배치하면 새로운 연산 모듈을 설계하지 않고도 MPU가 BP (Back Propagation) 학습과 SOM(Self Organizing Map)을 위한 학습 모듈로서도 동작할 수 있음을 확인하였다.

2. 범용 신경망 회로(ERNIE)의 기본 구조

신경망의 기본 소자는 뉴런이고 이 뉴런은 외부의 자극에 대한 반응이라는 단순한 기능을 수행한다. 자극을 입력으로, 반응을 출력으로 하여 일반적인 신경망 모델의 입력과 연결강도를 다음과 같이 입력벡터 X와 연결강도 W로 표현했을 때,

$$\boldsymbol{X} = [x_1,\ x_2,\ \cdots,\ x_n]$$
$$\boldsymbol{W} = [w_1,\ w_2,\ \cdots,\ w_n]$$

입력의 가중 합 NET 과 뉴런의 출력 y의 관계는 다음과 같다.

$$NET = \sum_{i=1}^{n} x_i w_i = \boldsymbol{X}\boldsymbol{W}^T$$
$$y = f\left(\sum_{i=1}^{n} x_i w_i\right) = f(NET) \tag{1}$$

단층 또는 다층 퍼셉트론을 구성했을 경우에는 위의 식 (1)과 같으며 SOM을 구성했을 때 입력 벡터와 뉴런의 연결강도와의 오차 거리 $dist$는 다음 식 (2)와 같다.

$$dist = \sum_{i=1}^{n} (x_i - w_i) \tag{2}$$

실제 SOM에서 유클리드 거리를 구하기 위해 제곱근 연산이 필요하지만 구현이 매우 복잡하기 때문에 시티-블럭 거리를 연산하여 유사도 측정을 하였다[5].

2.1 SPE의 연산 구조

SPE에서는 Summing node 연산을 담당하여 식 (1)에서 각 연결강도와 입력 값의 가중합인 NET, 또는 식 (2)의 입력벡터와 연결강도와의 거리차를 출력해 준다. 그림 1에서와 같이 SPE는 내부에 연결강도를 저장할 수 있는 메모리 공간을 갖고 있고 가중합을 위한 산술 연산기를 내장하고 있다. 보통 하나의 SPE는 하나의 뉴런으로 동작하게 되나 입력의 수가 SPE에 포함된 연결강도를 저장하는 메모리 공간을 넘어설 경우 두 개 이상의 SPE가 하나의 뉴런으로 동작을 하게 되면서 시냅스의 확장이 이루어진다. 현재 SPE에는 256개의 연결강도를 저장할 수 있도록 설계되었으며 최대 256개의 입력을 받을 수 있는 뉴런으로 동작할 수 있다.

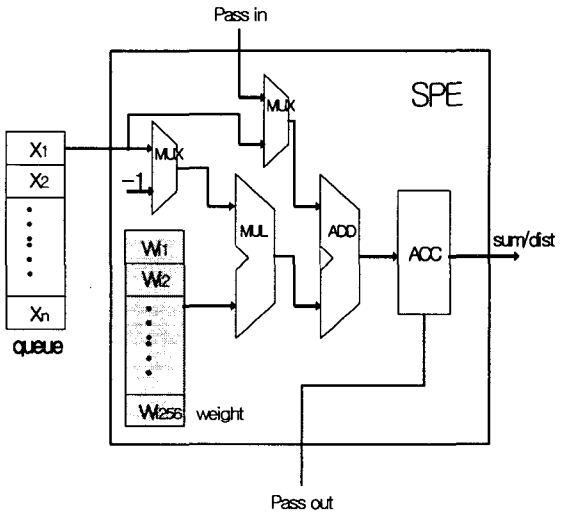

그림 1. SPE 블록 다이어그램
Fig. 1. SPE block Diagram

2.2 LPE의 연산 구조

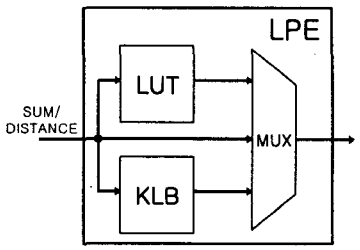

그림 2. LPE 블록 다이어그램
Fig. 2. LPE block diagram

LPE에서는 식 (1)에서 $f(NET)$에 해당하는 시그모이드 또는 기타 활성화 함수에 대한 연산이 이루어진다. LPE 내부에는 LUT(Look Up Table)이 있고 여기에는 미리 계산되어 얻어진 활성화 함수의 값이 저장된다. 그러므로 실제로 연산

이 이루어지는 것이 아니라 SPE에서 출력되는 값 NET을 주소 값으로 입력받아 $f(NET)$에 해당 하는 값을 출력해 준다. 이러한 방식은 복잡한 연산기를 설계해 줄 필요가 없으므로 그림 2와 같이 LPE의 내부 구조가 간단해지고 연산에 필요한 시간을 줄일 수가 있다. 그 밖에 SOM을 구성했을 때, KLB(Kohonen Logic Block)에서는 각각의 뉴런에서 출력해 주는 값을 입력 받아 그 중 가장 작은 값의 인덱스를 출력해 준다. 이 인덱스는 승자 뉴런을 가리킨다. 모든 뉴런의 출력 값을 정렬하는 것이 아니라 가장 작은 값을 선택하는 것이기 때문에 실제 연산에 필요한 시간은 데이터 전송에 걸리는 시간과 같다.

2.3 MPU의 구성

하나의 LPE와 다수의 SPE가 MPU를 구성하고, 하나의 MPU는 신경망에서 하나의 레이어를 구성할 수 있는 최소 단위가 된다. 이 MPU를 추가함으로써 레이어의 확장이 가능하게 된다. 그림 3은 SPE의 시냅스 확장과 MPU의 레이어 확장을 보여준다. 두 개 또는 그 이상의 SPE가 하나의 뉴런으로 동작할 때의 구성 개념과 MLP(Multi Layer Perceptron)같은 다층레이어의 신경망을 구성할 때의 블록 다이어그램이다. 입력의 수가 256개를 넘으면 두 개 이상의 SPE가 하나의 뉴런으로 동작을 하게 된다. 하나의 MPU는 하나의 레이어로서 동작할 수 있고, 뉴런의 수가 MPU0에 있는 SPE로 구성할 수 있는 개수 보다 많을 경우 MPU0와 MPU1이 하나의 레이어로서 구성 되며 이 때, MPU0의 LPE는 SPE의 데이터를 바이패스해주는 역할을 한다.

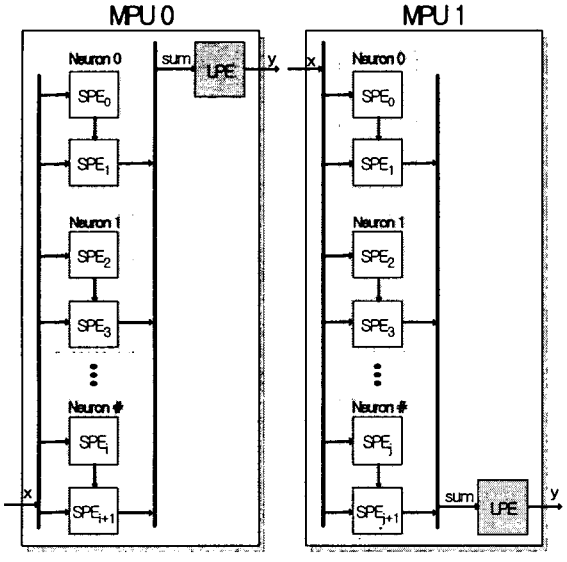

그림 3. SPE 및 MPU 확장의 블록 다이어그램
Fig. 3. SPE and MPU expansion block diagram

3. BP(Back Propagation)학습을 위한 구성

BP 학습 알고리즘을 이용한 다층 신경망의 학습 과정을 살펴보기 위해 다음과 같이 신경망을 구성한다.

입력층	$X=[x_1,\ x_2,\ \cdots,\ x_n]$
은닉층	$Z=[z_1,\ z_2,\ \cdots,\ z_p]$
출력층	$Y=[y_1 y_2 \cdots y_m]$
연결강도(은닉층)	$V(p\times n)$
연결강도(출력층)	$W(m\times p)$

이 때, 출력층의 i번째 뉴런의 오차(에러) E와 오차 신호 δ_y는 다음과 같이 정의한다[5].

$$\text{오차} \quad E\equiv\frac{1}{2}\sum_{i=1}^{m}(d_i-y_i)^2 \tag{3}$$

$$\text{오차신호} \quad \delta_y\equiv-\frac{\partial E}{\partial(NET_i)} \tag{4}$$

식 (3), (4)와 같이 정의하였을 때 출력층과 은닉층의 오차 신호 δ_y, δ_z는 다음과 같이 얻을 수 있다.

$$\delta_y=\begin{cases}(d-y)y(1-y) & (unipolar)\\ \frac{1}{2}(d-y)(1-y^2) & (bipolar)\end{cases} \tag{5}$$

$$\delta_z=\begin{cases}z(1-z)\sum_{i=1}^{m}\delta_y\vec{w} & (unipolar)\\ \frac{1}{2}(1-z^2)\sum_{i=1}^{m}\delta_y\vec{w} & (bipolar)\end{cases} \tag{6}$$

이 때, 입력 값의 범위에 따라 활성화 함수가 단극성(unipolar)과 양극성(bipolar)으로 나뉘며, 여기서 출력층의 가중합을 NET_y라 했을 때, 시그모이드 활성화 함수를 거친 출력값은 식 (7)과 같다.

$$y=\begin{cases}\frac{1}{1+e^{-NET_y}} & (unipolar)\\ \frac{1-e^{-NET_y}}{1+e^{-NET_y}} & (bipolar)\end{cases} \tag{7}$$

이때, y를 미분하면 다음과 같은 결과를 얻을 수 있다.

$$\begin{aligned}y'=f'(NET_y) &= \frac{e^{-NET_y}}{(1+e^{-NET_y})^2}\\ &=\left(\frac{1}{1+e^{NET_y}}\right)\left(\frac{1+e^{-NET_y}-1}{1+e^{NET_y}}\right)\\ &=f(NET_y)[1-f(NET_y)]\\ &=y(1-y)\end{aligned} \tag{8}$$

$$\begin{aligned}y'=f'(NET_y) &= \frac{2e^{-NET_y}}{(1+e^{-NET_y})^2}\\ &=\frac{1}{2}\left[1-\left(\frac{1-e^{-NET_y}}{1+e^{-NET_y}}\right)^2\right]\\ &=\frac{1}{2}(1-y^2)\end{aligned} \tag{9}$$

식 (8), (9)에 의해서 식(5), (6)을 식 (10), (11)과 같이 얻을 수 있다.

$$\delta_y = (d-y)f'(NET_y) \tag{10}$$

$$\delta_z = \left(\sum_{i=1}^{m} \delta_y w\right) f(NET_z) \tag{11}$$

ERNIE에서는 활성화 함수인 $f(NET)$을 LUT로 대치해서 사용했던 것처럼 $f'(NET)$에 대한 값 역시 LUT로 대치해서 사용 한다면 복잡한 연산 과정을 거치지 않고도 오차신호 δ_y와 δ_z를 구할 수 있음을 알 수 있다.

가장 먼저 식 (3)에 따라 그림 4와 같이 SPE를 구성하면 출력 오차를 구할 수 있다. 목표 값 d를 SPE의 연결강도를 저장하는 곳에 저장해두고 신경망의 가중 합을 구하는 것과 같이 신경망의 출력벡터를 입력해 주면 ACC(Accumulator)에서 오차 누적 값을 얻을 수 있게 된다. 이렇게 얻은 오차 값은 LUT에서 오차 한계 E_{max}와 비교하게 되는데, LPE에서는 학습을 진행할 것인지 멈출 것인지를 판단해서 결과를 출력해 준다. LPE에서 나온 오차 판별 신호는 MPU 내부적으로 학습의 반복을 멈추고 순방향 신경망 연산만을 할 것인지, 학습을 반복할 것인지를 결정하고 신경망 칩 외부로는 지금이 학습 모드인지 실행 모드인지를 구별해 주는 상태 레지스터로 사용된다.

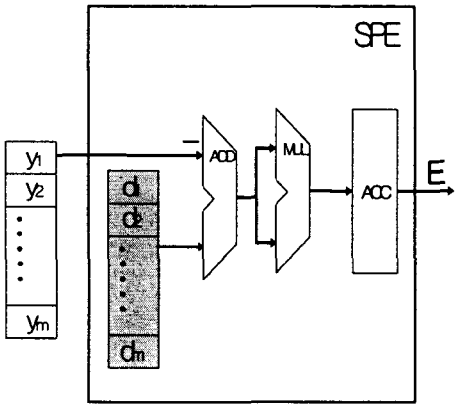

그림 4. 오차(E)를 구하기 위한 SPE 구성
Fig. 4. Configuration for error calculation

학습을 진행하기 위해 출력층의 오차 신호 δ_y를 먼저 구해야 한다. 식 (10)의 값을 구하기 위한 SPE의 구성은 그림 5와 같다. 단 여기서 ACC(Accumulator)는 입력받은 값을 바이패스 해준다. 여기서 얻어진 오차 신호는 다시 다른 SPE의 내부 메모리에 저장되어 다음 연산에 사용된다.

위에서 구한 출력층의 오차 신호로 은닉층의 오차 신호를 얻는다. 식(11)에 의해 그림 6과 같은 SPE 구성이 가능하다. 레이어가 3개 이상인 다층 신경망일 경우에는 은닉층의 오차 신호 δ_z를 구하는 것과 동일한 방법을 반복하면 모든 레이어의 오차 신호를 구할 수 있다. 현재 하나의 MPU에는 24개의 SPE와 1개의 LPE로 구성되어 있다. 신경망에서와 마찬가지로 SPE의 내부 메모리가 부족하게 되면 두 개 이상의 SPE가 서로 맞물려서 연산을 하게 된다. 이러한 설정을 위해서는 신경망 모듈과 학습 모듈을 설정할 때 사전에 설정 비트와 함께 구성되도록 한다.

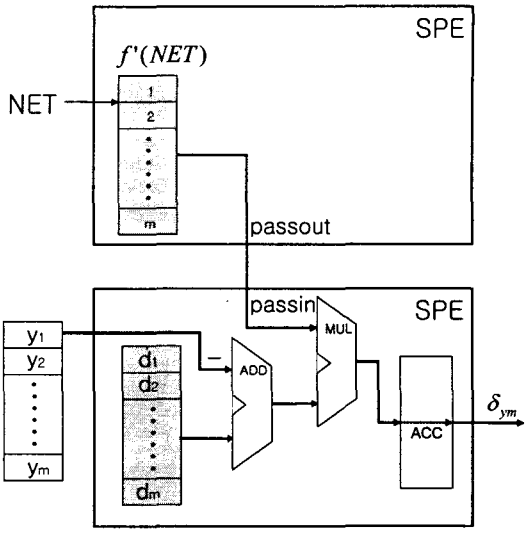

그림 5. 출력층 오차신호를 구하기 위한 SPE 구성
Fig. 5. Configuration for output layer error signal

지금까지의 연산으로 출력층과 은닉층의 연결강도를 수정하기 위한 연결강도 변화량 ΔW 와 ΔV를 얻었을 수 있었다. 연결강도는 수정하는 방법은 오차 신호를 구하는 방법과 같다. 그림 7과 같이 학습 모듈을 구성하고 있는 SPE의 내부 메모리에 각각의 ΔW와 ΔV를 저장해 놓고 신경망을 구성하고 있는 MPU의 SPE에 저장된 연결강도를 내부 버스를 거쳐서 전달해 준다. 이렇게 전달받은 연결강도의 값들은 학습 모듈의 SPE를 거치며 수정되고 수정된 값은 다시 신경망의 입력으로 전달된다. 이렇게 만들어진 은닉층과 출력층의 연결강도 변화량 ΔW, ΔV를 SPE의 내부에 저장한 다음 버스를 통해 각 뉴런들의 연결강도가 들어오면 바로 수정되고 다시 신경망으로 입력이 된다.

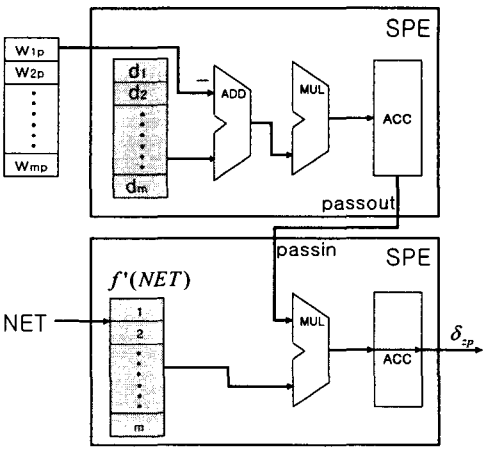

그림 6 은닉층 오차 신호를 구하기 위한 SPE 구성
Fig. 6. Configuration for hidden layer error signal

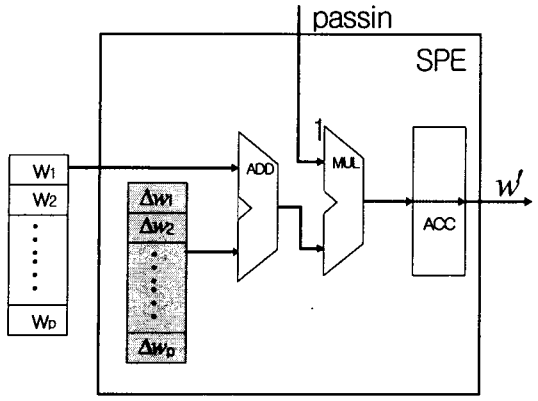

그림 7. 연결강도 수정을 위한 구성
Fig. 7. Configuration for weight update

4. SOM 학습 모델을 위한 구성

SOM 학습에서 고려해야할 점은 승자 뉴런의 인덱스와 그 주변 뉴런의 인덱스를 얻는 것이다. 신경망을 구성하는 LPE에서는 모든 뉴런의 distance 값을 입력받아 그 중 최소값의 인덱스를 출력한다. 인덱스는 승자 뉴런을 가리키는 번호가 된다. SOM 신경망을 다음과 같이 표현 했을 때,

입력층	$X=[x_1\ x_2\ \cdots\ x_n]$
출력층(Kohonen Layer)	$Y=[y_1\ y_2\ \cdots\ y_m]$
연결강도	$W(m\times n)$

ERNIE에서 입력 벡터와 j 번째 뉴런간의 거리차이를 구하는 식은 다음과 같다.

$$D(j)=\sum(w_{ji}-x_i) \tag{12}$$

연결 강도의 수정은 식(13)에 따라 이루어진다.

$$w_{ij}(t+1)=w_{ij}(t)+\alpha(x_i(t)-w_{ji}(t)) \tag{13}$$

MPU의 관점에서 봤을 때, SOM의 학습 모듈을 구현하는 것은 앞서 구현한 BP 학습 모듈에 비해서는 비교적 간단하다. 그러나 LPE의 내부의 구조는 매우 복잡해지는데 이는 LPE에 있는 카운터로 MPU내에 있는 SPE의 동작 모드를 결정하기 때문이다. 앞에서 언급했듯이 신경망을 구성한 MPU의 LPE는 승자 뉴런의 인덱스를 출력해 준다. 학습모듈에서는 인덱스를 입력 받아서 내부의 카운터 세팅을 마친다. 신경망을 담당한 MPU에서는 인덱스를 출력하고 바로 각 SPE에 저장된 연결강도를 차례로 출력해준다. 이 때 학습 모듈에서는 카운터에 맞추어 입력 받은 연결강도를 연산 없이 그대로 출력하거나 반경 안에 있는 뉴런일 경우 거리에 따라 결정된 학습률이 적용된 연결강도로 수정을 해주게 된다.

승자뉴런이 선택 되면 그 일정 반경 안에 있는 이웃 뉴런들도 연결강도의 수정이 이루어지며, 학습이 반복되면서 반경은 점점 줄어들게 된다. 이처럼 학습 횟수와 학습 반경에 따라 달라지는 학습률 α 역시 LUT와 같이 저장 될 수 있다. 이 때 학습 모듈로 사용되는 SPE에는 신경망의 입력 값 (x_i)이 임시로 저장된다. 신경망 연산을 위한 입력 패턴이 학습모듈에 임시로 저장이 되고 여기에 연결강도 값을 입력해 줌으로 수정된 연결강도 값이 출력된다. SOM에서는 모든 연결강도를 수정해 주는 것이 아니라 승자 뉴런을 찾아서 선택적으로 연결강도를 수정해 준다. 학습모듈의 각 SPE는 학습 반경에 다른 α 값을 갖는다. 최대 학습 반경이 3 일 때 4개의 SPE가 각각에 해당하는 학습률 α 값을 갖고 학습이 반복됨에 따라 승자뉴런과의 거리에 따라 차례로 halt mode가 되며 최종적으로 학습이 완료되면 학습 모듈의 기능은 정지되고 신경망 연산 결과만 출력된다. 그림 8에서 SPE에 있는 r은 학습 반경이다. 학습 반경이 줄어들면서 반경에서 벗어난 SPE는 halt 모드로 변경되어 정지한다. 최종적으로 winner에 해당하는 SPE 만이 남게 되고 학습이 종료되면 학습 모듈은 입력된 값, 즉 SOM 신경망의 출력(승자뉴런의 인덱스)을 바이패스 해주게 된다.

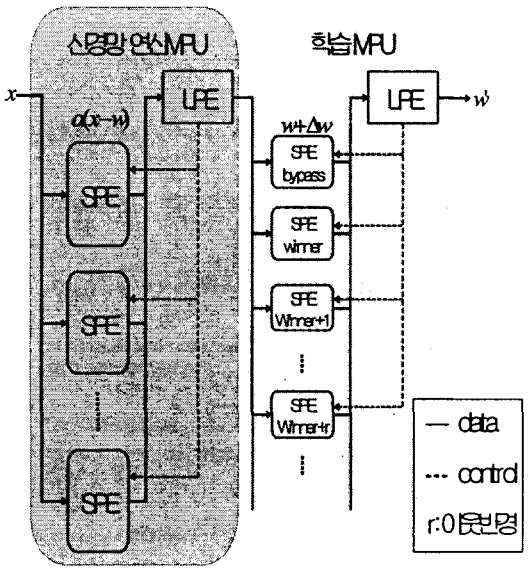

그림 8. SOM 연산 모듈 및 학습 모듈 구성
Fig. 8. Learning module construction of SOM

기존의 SPE와 LPE에 신경망 연산 외에도 학습과 학습을 위한 MPU의 제어를 위해서 몇 가지 기능과 제어 신호가 추가 되었다. 이에 따라 MPU는 하나의 신경망 레이어를 구성하는 단위가 될 뿐만 아니라 하나의 학습 모듈로 동작할 수 있다. 따라서, 내부 구성과 설정에 관한 부분이 좀 더 복잡해지게 되어 SPE와 LPE의 동작의 상태를 정의해주는 설정 비트가 추가 되었고, 동작과 내부 연산 구조를 설정해 주기 위한 설정 비트가 추가 되었고, 각 동작 모드는 표 1에 정리되었다.

표 1. SPE와 LPE의 동작 모드
Table 1. SPE and LPE configuration mode

Processing Element	State Mode	Function Mode	Function
SPE	Halt		SPE 동작 정지
	Running	Kohonen	거리(distance) 연산
		Perceptron	Summing node 연산
	Learning	Kohonen	SOM 학습 연산
		BP	BP 학습 연산
LPE	Bypass		데이터 전달
	Running	Kohonen	승자뉴런 선택
		Perceptron	활성화 함수
	Learning	Kohonen	학습 반경 설정
		BP	데이터 피드백

SPE와 LPE의 내부 구조는 그림 9, 10과 같이 변경되었다. SPE는 신경망 연산과 학습을 위한 다양한 함수 구현을 위해 내부 데이터 패스 구조가 복잡해지고, LPE는 SPE제어를 위한 디코더 부분이 추가되었다. 전체적인 동작의 흐름은 이전과 같으나 각 연산소자를 설정하는 부분에 학습모듈의 설정이 추가되었다.

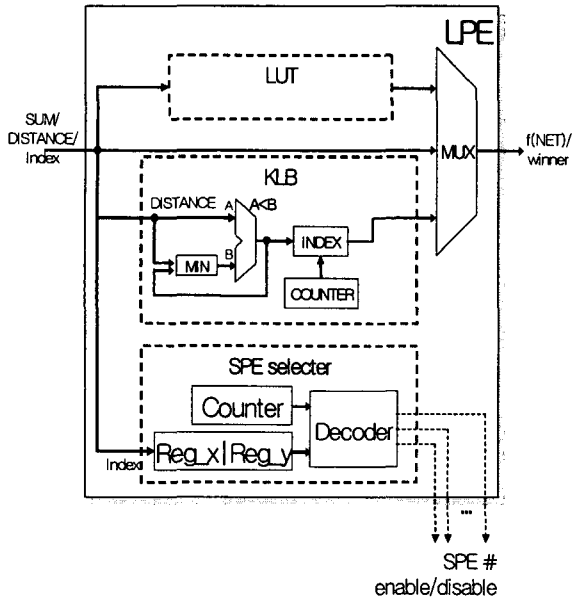

그림 9. 수정된 LPE 블록 다이어그램
Fig. 9. Modified LPE block diagram

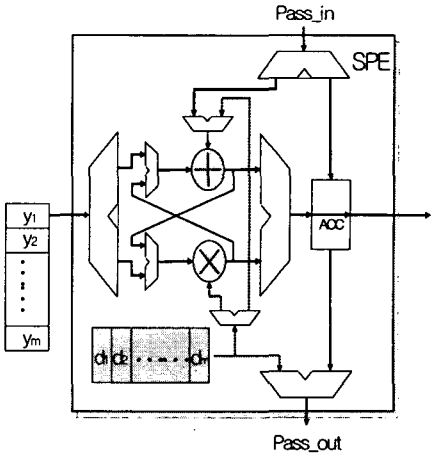

그림 10. 수정된 SPE 블록 다이어그램
Fig. 10. Modified SPE block diagram

5. 실험 결과 및 분석

회로의 구현은 Virtex-II(xc2v6000-4f1152) 기반으로 구현되었고, XST(Xilinx Synthesis Tool)을 사용하여 합성한 후 ModelSim v5.7d로 타이밍 시뮬레이션으로 검증하였다. 각 실험 결과는 ERNIE를 20MHz로 동작시켰을 때의 결과이다.

5.1 성능분석

ERNIE의 시냅스 연산에 걸리는 시간은 레이어의 수 l 과 각 레이어에 있는 뉴런의 수 n_i 에 따라 달라지며 그 밖에 SPE의 연산시간 c, 전체 MPU의 개수 M, 시냅스 확장 파라미터 s_i 에 따라 달라지며 연산시간 t_s(clock)와 위의 다섯 가지 변수와의 관계는 식 (14)와 같다.[5]

$$t_s = \sum_{i=0}^{l-1} n_i s_i + (l-1)c + M \tag{14}$$

식 (14)는 신경망 연산에만 소요되는 시간으로 입력된 패턴에 대한 결과가 출력되기 까지 걸리는 시간이다. 식 (3)에 해당하는 신경망의 BP 학습에 필요한 시간은 식(15)와 같다.

$$t_l = m + mp \tag{15}$$

m : 출력층의 뉴런 수

p : 은닉층의 뉴런 수

식 (15)는 식 (10)과 (11)에 해당하는 오차신호를 구하는데 필요한 시간으로 출력오차를 구하는 시간과 연결강도를 수정하는 시간은 제외하였다. 은닉층이 여러 개인 다층 신경망일 경우에는 식 (16)과 같다.

$$t_l = m + mp_1 + mp_2 + \cdots = m \sum_{i=0}^{l-1} p_i \tag{16}$$

p_i : i 번째 은닉층의 뉴런 수

BP학습을 위해 구성했던 신경망의 학습 연산을 순차방식의 CPU로 연산한다면 연산 회수만 비교했을 때 식 (17)과 같이 연산 시간을 얻을 수 있다.

$$t_l = m^2 + (m^2+1)p \tag{17}$$

학습모듈을 포함한 ERNIE와 다른 신경망 연산기와의 성능 비교를 표 2에 정리하였다. 비교 대상에 비하여 성능상의 우위를 볼 수 있다.

표 2. ERNIE와 다른 신경망 연산기의 성능[7]
Table 2. Performance comparison of neuro processors and ERNIE[7]

Neuro Processors	Number of Processing Units	Performance (MCUPS)
ERNIE	120	18.9
CM-1	16000	2.8
Warp	10	17.0
Fujitsu AP100	64	15.8
Alex AVX-2	64	17.7

MCUPS = Mega Connections Update per Second

다만, LUT의 사용과 병렬 연산으로 인하여 ERNIE의 연산 능력은 최고 6.3GCPS(Giga Connections per Second)[6]에 달하는데 반해 학습 속도가 크게 뒤쳐진다. 이는 ERNIE의 학습 모듈의 설정 구조에 그 원인이 있다. ERNIE에서는 학습 데이터 연산을 모듈 내부에 기록해 두었다가 연산이 끝나면 학습 데이터를 갱신하게 되는데 이 때 데이터 전송 속도에 한계가 있기 때문에 학습 속도가 연산 속도에 뒤지게 되는 것이다. 추후 학습데이터를 따로 갱신하는 절차를 생략하고 학습과 동시에 연결 강도의 수정이 이 가능하도록 구조 개선을 한다면 학습 성능이 크게 향상될 것이다.

5.2 문자 인식 실험

BP 학습 모듈의 성능을 평가하기 위해 문자 인식 실험을 실시하였다. 다섯 가지 알파벳 폰트를 8bit color/pixel 인 8×8 그레이 스케일의 영상으로 만든 다음 0~255 사이의 값을 -0.5 ~ 0.5 의 크기로 정규화 하여 신경망의 입력 패턴으로 사용하였다. 입력 패턴은 알파벳 대문자 5가지 폰트로 정상 폰트로 학습을 한 후 손상된 폰트를 입력하여 인식률을 분석하였다. 64개의 입력 뉴런과 10개의 은닉층 뉴런, 5개의 출력 뉴런으로 구성하여 각 알파벳에 해당하는 코드를 출력하도록 하였다. 신경망의 파라미터로 에러 임계값 E_{max}는 0.01을 주었으며 학습률은 0.1로 주었다. 이 실험 결과로 얻은 결과와 C 코드로 구현된 신경망과의 평균 오차는 근소한 차이를 보였으며, 각 실험의 결과를 표 3에 정리하였다.

표 3. 평균 오차 비교
Table 3. Mean errors

네트워크 종류	미리 생성된 연결강도 사용	학습 모듈 사용
5X20	0.0147	0.0151
5X40	0.0150	0.0149
25X10X2	0.0181	0.0184
64X10X5	0.0284	0.0326
256X96X26	0.0435	0.0485

표 3의 설정에서 미리 생성된 연결강도는 C로 구현된 소프트웨어로 미리 학습된 연결강도를 사용하였고 학습 모듈 사용은 ERNIE에 구현된 학습모듈을 사용했을 때의 결과이다. 학습 모듈을 사용했을 때의 에러 값이 조금씩 커지는 이유는 C에서는 double 형의 데이터로 계산하였고 ERNIE는 18bit 연산을 했기 때문으로 해상도의 차이로 생긴 결과이다. 미리 생성된 연결강도에 비해서 오차의 증가는 무시할 수 있을 만큼 미미하였다. 오차의 증가가 일정하지 않은 부분이 보이지만 이는 LUT를 사용했기 때문 보다는 네트워크 종류와 초기값에 따라 달라 질 수 있는 부분이라고 본다. 다만 네트워크의 크기가 커질수록 평균 오차 역시 커지는 것으로 보아 LUT 사용으로 인하여 누적 오차가 증가되는 것을 알 수 있다. 5X40의 네트워크를 구성 했을 때는 오차가 오히려 감소하는 것을 볼 수 있는데, 이는 특정한 경우로 일반적인 경우라고 볼 수는 없다.

5.3 암 관련 유전자 발현 정보의 분류 실험

두 번째 실험으로 신경망을 이용한 질병 진단 실험을 시도하였다. 이 실험은 유전자 정보를 분석하여 얻어낸 정보로 백혈병 환자를 분류하는 실험[8]으로 72명의 백혈병 환자의 골수 샘플로 제작된 DNA Microarray 데이터를 사용하였다. 환자 한 사람당 7129개의 유전자가 있고, 이 중에서 질병 분류에 관련되었다고 생각되는 유전자를 선택해서, 급성 림프구성 백혈병(ALL, Acute Lymphoblastic Leukemia)와 급성 골수성 백혈병(AML, Acute Myeloid Leukemia)의 두 클래스로 분류 한다[9]. 유전자 발현 정보 분류는 크게 두 단계로 나눌 수 있다. 하나는 7129개의 유전자 정보로부터 특징 유전자를 추출해 내는 부분이고 다른 하나는 이렇게 추출된 유전자 정보를 패턴 분류기법으로 분류하는 부분이다. 여기서는 피어슨 상관 관계법으로 특징 추출을 하여 얻은 유전자 정보로 BP와 SOM을 사용한 패턴 분류 방법을 시도하였다. 72명의 환자 중 34명 유전자 데이터로 학습을 하고 나머지 34명의 데이터를 테스트 하였다. BP는 25X10X2의 구성으로 하여 실험을 하였고 SOM은 8X8 크기의 코호넨 레이어를 구성하여 패턴 분류를 시도하였다. 환자의 특징 추출 방법에 따라서도 인식률에 상당한 차이가 있고 패턴 분류 모델에 따라서도 서로 다른 인식률을 보였으므로[8], 다양한 신경망을 구성하여 실험한다면 좀 더 많은 분석 결과를 얻을 수