신경회로망의 하드웨어 구현 및 암의 패턴 분류
Hardware Implementation of Neural Network and its Application

동성수 (디지털전자정보과)
Sung-Soo Dong (Dept. of Digital Electronics & Information)

Key Words : Neural Network(신경회로망), Hardware Implementation(하드웨어 구현).

ABSTRACT : Hardware implementation of neural network has difficulties in the extension of synapses and the programmability for relocating neurons. In this paper, the architecture of a new hardware is proposed to solve these problems. The proposed hardware based on traditional SIMD enables connections of network to be dynamically and easily reconfigured without synthesizing and mapping original design for each use. Using additional modular processing unit the number of neurons and synapses increases. To show the extensibility of the new structure, Kohonen network is formed and tested. The performance comparison with C-model simulation shows its superiority in the aspects of performance and flexibility.

1. 서론

연상, 추론, 인식 등 기존의 컴퓨터가 해결하기 어려운 응용분야와 복잡한 비선형 시스템의 제어에 유용하게 사용되고 있는 알고리즘 중 하나가 신경회로망이다. 인간 두뇌의 정보처리 방식을 모델화 한 신경회로망은 병렬구조와 분산 처리가 본질적인 특징이다. 따라서 신경망 모델의 구현에 있어서 소프트웨어 시뮬레이션만으로는 속도의 한계를 가지게 되어, 1958년 F. Rosenblatt의 MARK I Perceptron 이후로 신경회로망을 하드웨어로 구현하려는 노력이 계속되어졌다[1]. 근래에 들어서는 디지털 전자회로기술 및 ASIC 기술의 발전에 힘입어 아날로그 방식 보다는 디지털 방식 신경회로망의 구현에 더 많은 연구를 기울이고 있다.

디지털 신경회로망을 구현하기 위한 여러 방법들을 크게 두 가지로 분류할 수 있는데, 첫째는 DSP 등의 범용 프로세서를 기반으로 펌웨어로 신경망 모델을 구현하는 방법이 있고, 둘째는 시냅스, 뉴런 등의 회로 전체를 하드웨어로 병렬구성 하여 신경망 연산 전용칩(ASIC/FPGA)으로 구현하는 방법이 있다[2]. 첫 번째 방법은 두 번째 방법에 비해서 설계자유도가 높으며 구현이 쉽다는 장점이 있으나, 뉴런의 선형적인 증가에 대해서 시냅스가 지수적으로 증가하는 데에 따르는 연산 시간 및 면적의 증가를 해결하기 어려우며 범용 연산기의 사용으로 인하여 하드웨어 구현 시 상대적으로 큰 면적을 차지하게 된다.

특히 디지털 회로에서의 곱셈기는 아날로그에 비하여 큰 면적을 차지하게 되는데 이를 극복하기 위하여 곱셈기가 없는 디지털 신경망, 또는 곱셈기의 크기를 줄이기 위한 시리얼 데이터 방식의 디지털 신경망과 같은 연구가 있다[3,4]. 두 번째 방법의 경우는 설계자유도 문제에 있어서는 취약하지만 신경망 연산을 위한 전용회로를 사용하기 때문에 첫 번째 경우보다는 면적문제에 있어 많은 이점을 가진다. 그러나 이 경우에도 역시 곱셈기의 증가 문제는 회로의 집적도를 크게 떨어뜨리는 요인으로 작용한다.

하드웨어로 구현된 신경망 연산기가 처리속도의 측면에서 소프트웨어로 구현된 신경회로망 시뮬레이터에 비해 월등한 이점을 갖는다고 해도 정형화된 가중치 혹은 특정한 형태로 고정된 네트워크의 기능만을 수행한다면 다양한 공학적인 분야에서 그 이용가치가 떨어지게 되므로 특정 수준 이상의 범용성은 디지털 신경망 연산기의 구현에 있어 필요한 조건이다. 이를 위해서 다양한 연구가 있어왔으며 주로 범용 프로세서를 이용하여 신경망 연산기를 재프로그래밍 하는 방법과 마치 FPGA처럼 간단한 재구성 비트를 이용하여 기본 Processing Element(PE)의 역할과 버스구조를 변경함으로써 원하는 형태의 신경회로망을 구성하는 재구성형 하드웨어 구조 등이 제시되어 왔다[5,6].

B. Pino는 각각의 PE의 시스템 버스를 인접한 PE 사이에만 연결함으로써 PE의 직렬 확장을 통해 네트워크의 크기를 확장하는 방법을 소개하였다[7]. 이러한 구조를 통해 PE의 한계에 제한을 받지 않으면서 원하는 규모의 네트워크를 구성할 수 있지만 PE사이의 버스를 통해서만 데이터의 전송이 이루어지기 때문에 병렬성의 측면에 있어서 바람직하지 못한 결과를 가져오게 된다. 또한 B. Girau는 FPGA의 재구성 원리를 이용한 Field Programmable Neural Arrays(FPNA)라는 개념을 통해서 뉴런의 개수를 확장하는 방법을 도입하였지만 이 방법은 각각의 PE가 가지는 한계 이상의 시냅스를 처리할 수 없다는 문제점을 가지고 있다[6].

위에서 제기한바와 같이 시냅스의 지수적인 증가에 따른 면적의 증가, 신경회로망의 재구성, 그리고 네트워크의 확장에 관한 문제점들을 효과적으로 극복하기 위하여 본 논문은 Single Instruction Multiple Data(SIMD)구조 및 마스터-슬레이브(Master-Slave)구조를 응용한 새로운 형태의 모듈러 신경망 구조를 제안하였다.

2. Single Instruction Multiple Data(SIMD)

하나의 명령어에 의해서 여러 개의 프로세서(또는 유니트)들이 같은 작용을 서로 다른 데이터에 대해 수행하는 SIMD 구조는 많은 양의 로컬 데이터를 병렬적으로 처리해야하는 영상 처리나 신경회로망과 같은 응용을 구현하는데 있어 매우 유용하다. 각각의 PE는 최소한의 기능만을 수행하는 간단한 구조여야 하며 자신만의 내부 메모리를 갖고 있다. 또한 모든 PE들은 상호간의 네트워크를 통해 연결되어 있으며 각각의 PE들은 배열(array)을 이루고 있다. PE는 특정한 명령을 동시에 받게 되고 자신에게 속해 있는 지역 메모리

의 데이터를 해당 명령에 의해 처리한다. 필요한 경우 서로간의 연결망을 통해서 처리된 데이터를 주고받는 것이 가능하다[8]. 이러한 SIMD 구조를 신경회로망에 적용할 경우 PE는 뉴런의 역할을 하게 되고 각각의 PE에 속하는 지역 메모리는 시냅스의 가중치를 저장하는 역할을 하게 된다. Fig. 1은 일반적인 신경회로망 모델을 나타내며 Fig. 2는 SIMD 구조를 이용한 신경회로망의 예를 보여준다.

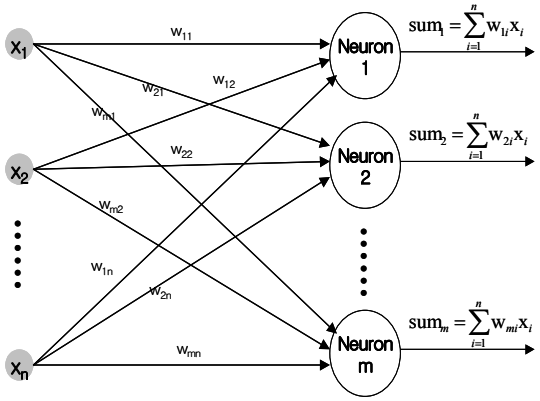

Fig. 1 General neural network architecture

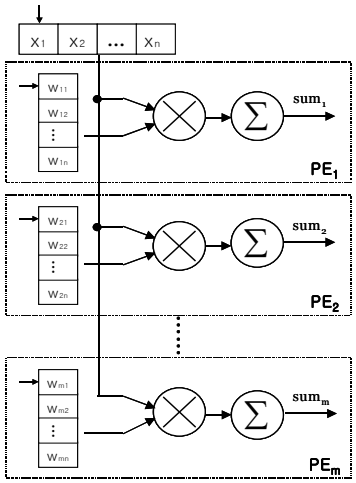

Fig. 2 Neural network using SIMD architecture

2. Expansible & Reconfigurable Neuro Informatics Engine(ERNIE)

2.1 Modular Processing Unit(MPU)

일반적인 하드웨어 신경망 연산기에서 뉴런의 시냅스 개수는 PE가 갖고 있는 내부 메모리의 크기에 의해서 결정된다. 따라서 뉴런의 역할을 하는 PE의 개수를 확장한다고 해도 내부 메모리 용량을 초과하는 입력은 받아들일 수가 없기 때문에 뉴런의 개수는 늘어나지만 뉴런 당 입력의 개수는 제한을 받게 된다. 본 논문은 기존의 확장 가능한 신경망 연산기 구조가 가지고 있는 이러한 결점을 보완하기 위하여 뉴런의 기능을 수행하는 PE를 내부 연결 버스를 통해 묶음으로써 시냅스 개수의 확장이 가능하도록 하는 형태의 하드웨어 구조를 제안하였다. 제안된 구조를 이용하면 시냅스의 확장은 물론 네트워크를 구성하는 가장 큰 기본 모듈인 프로세싱 유니트의 연결을 통하여 뉴런 및 레이어의 확장까지 가능하다. 다수의 PE로 구성되어 있으며 그 자체로서 한 층의 레이어를 구현할 수 있는 이 모듈을 Modular Processing Unit(MPU)라고 이름 지었다. MPU는 두 가지의 PE로 구성되고 설정모드에서 결정되는 상태에 따라 각기 다른 동작을 수행하며 주요 동작은 Table 1과 같다.

Table 1. Two PEs composing MPU

PE	기 능	내 용	특 징
Synapse Processing Element (SPE)	서밍 노드 (summing node)[9]	입력과 가중치의 MAC 연산	기본 신경망 연산 소자로써 상호 연결에 의하여 시냅스 확장 구현
	코호넨 노드	입력과 가중치의 거리(distance) 연산	
Layer Processing Element (LPE)	활성화 함수 LUT (Look Up Table)	SPE의 출력을 LUT의 주소값으로 이용	모든 출력을 다른 MPU의 입력으로 사용함으로써 MPU간의 모듈러 확장 구현
	승자 뉴런의 결정	가장 작은 출력을 내는 SPE의 인덱스 출력	

MPU는 퍼셉트론과 코호넨 네트워크를 구성할 수 있는 최소 단위의 모듈이며, 다수의 SPE와 LPE 1개로 구성이 가능하다. 구조의 특성상 하나의 MPU를 구성하는 SPE의 개수에 대한 제한은 없다. MPU는 Fig. 3과 Fig. 4는 퍼셉트론과 코호넨 네트워크의 동작을 MPU의 각 PE들이 어떻게 구현하는지를 개념적으로 보여주고 있다.

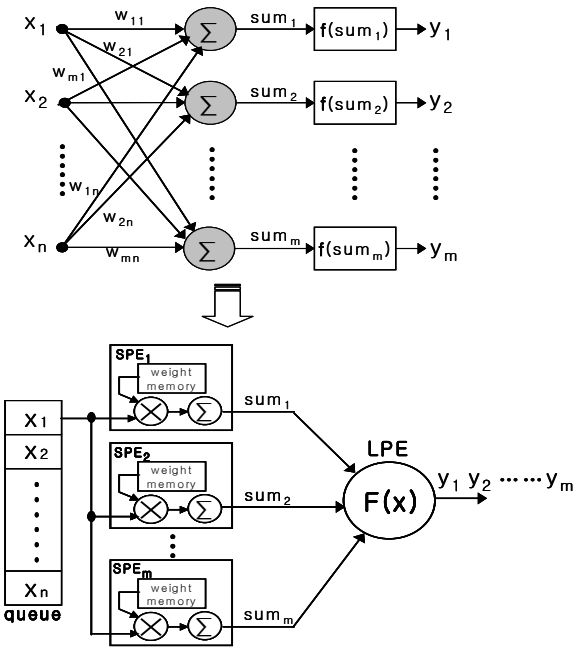

Fig. 3 Schematic diagram of MPU corresponding to perceptron

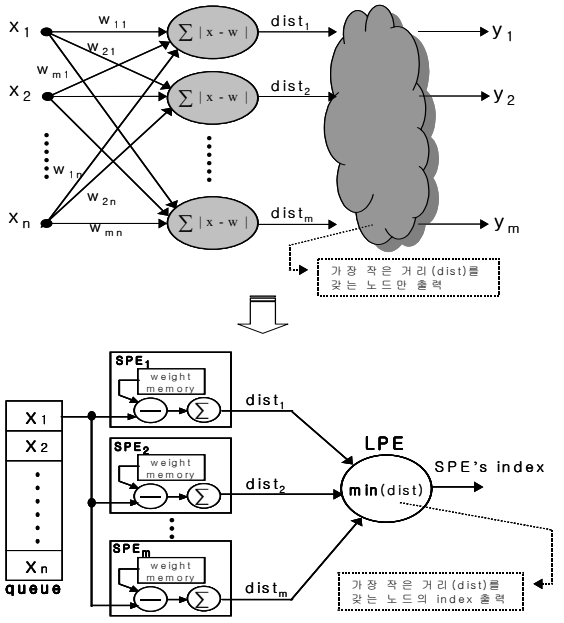

Fig. 4 Schematic diagram of MPU corresponding to Kohonen network

MPU는 모듈러 확장이 가능하도록 설계되었다. 따라서 MPU의 출력은 또 다른 MPU의 입력으로 사용되며 이러한 모듈러 확장은 MPU들로 구성되는 칩 간 확장으로까지 이어진다. MPU의 입력값은 SPE를 거쳐 병렬적으로 연산이 수행되고 이때의 결과 값과 여러 컨트롤 신호들은 LPE를 통하여 다음 MPU의 입력으로 들어간다. Fig. 5는 SPE 24개와 LPE 1개로 MPU를 구성한 블록 다이어그램이며 이들 PE간에 컨트롤 신호와 데이터가 어떻게 흘러가는지를 보여주고 있다.

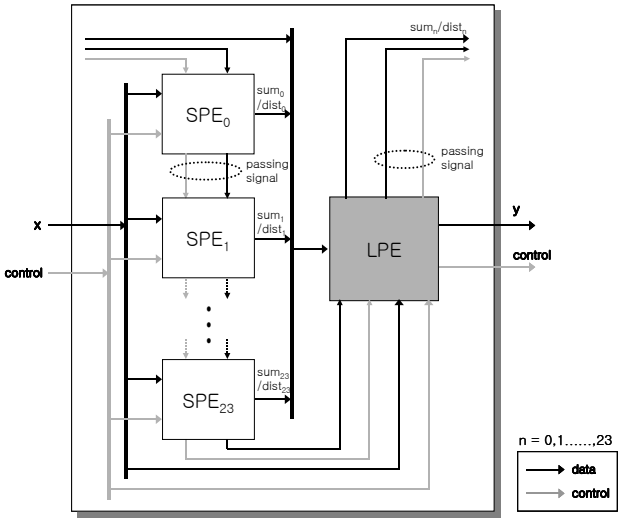

Fig. 5 Block diagram of MPU

제안된 구조는 네트워크의 확장을 위한 두 가지 특성을 갖고 있다. 첫 번째는 MPU간의 확장이며 두 번째는 SPE간의 확장이다. MPU간의 확장을 통해서 네트워크를 구성하는 뉴런의 개수 및 레이어의 개수를 확장할 수 있고, SPE간의 내부 연결 버스를 사용하여 하나의 SPE에서 누적된 시냅스 값을 인접한 SPE로 전달하는 방법으로 시냅스를 확장하는 것이 가능하다. 이와 같은 방법을 통하여 PE가 갖고 있는 지역 메모리의 용량을 초과하는 시냅스를 구현하는 문제를 해결할 수 있다. 이러한 형태의 '시냅스 확장'은 둘 혹은 그 이상의 SPE로 구현하는 것이 가능하며, MPU 내부에서만 국한되는 것이 아니라 외부의 MPU에 속해있는 SPE와도 연결이 가능하기 때문에 대용량의 시냅스를 요구하는 신경망을 매우 융통성 있게 구현할 수 있다.

모든 유니트 사이(SPE-SPE, SPE-LPE, MPU-MPU)에는 두 가지 종류의 마스터-슬레이브 형태의 버스가 존재한다. 첫 번째는 컨트롤 신호의 전달을 위한 것이고 두 번째는 여러 가지 형태의 확장을 위한 것으로서 특정한 경우에만 활성화 되며 네트워크를 구성하는 모든 기본 모듈을 유기적으로 연결해주는 역할을 한다. 이를 Table 2와 같이 정리하였다.

Table 2. Functions of master-slave buses for expansion

연결 형태	기 능	내 용
SPE-SPE	시냅스의 확장	마스터 SPE의 누적합을 슬레이브 SPE로 전달
SPE-LPE	외부 MPU에 포함된 SPE와의 시냅스 확장	마스터 SPE의 누적합을 LPE를 통해 슬레이브 SPE로 전달
MPU-MPU	뉴런 및 레이어의 확장	바이패스(bypass)

MPU를 구성하는 모든 PE는 동작 초기에 설정 모드를 통하여 그 역할이 결정된다. 이러한 과정을 거침으로써 매우 다양한 형태의 신경망 연산기를 구성하는 것이 가능하다. 따라서 이렇게 고안된 하드웨어 구조를 Expansible & Reconfigurable Neuro Informatics Engine(ERNIE)라 이름 지었다.

2.2 Synapse Processing Element(SPE)

SPE는 신경망 연산의 핵심적인 연산을 담당 하고 있으며 곱셈기, 덧셈기, 누산기의 세 가지 블록으로 구성되어 있다. 주요 역할은 퍼셉트론의 서밍 노드 연산과 코호넨 네트워크의 코호넨 노드 연산의 수행이다. 퍼셉트론에서 활성화 함수의 입력을 *sum*, 코호넨 연산에서 구해지는 입력과 가중치와의 유클리드 거리를 *dist*라 하고 $\mathbf{x}$와 $\mathbf{w}$가 각각 입력과 가중치의 n차원 벡터일 때 이들의 관계는 다음과 같이 표현 할 수 있다.

$$\sum = \sum_{i=1}^{n} x_n w_n \tag{1}$$

$$dist = \sqrt{\sum_{i=1}^{n} (x_n - w_n)^2} \tag{2}$$

여기서 식 (2)의 유클리드 거리를 구하기 위해서는 실수의 제곱 연산 및 제곱근 연산이 필요하며 이러한 연산을 하드웨어로 구현하는 것은 매우 복잡하다. 그리하여 SPE에서는 입력벡터와 가중치 벡터와의 시티-블럭 거리를 계산하여 유사도 측정의 기준으로 삼으며 이는 다음과 같이 표현된다.

$$\sum_{SPE} = \sum_{i=1}^{n} x_n w_n + passin \tag{3}$$

$$dist_{SPE} = \sum_{i=1}^{n} |x_n - w_n| \quad (4)$$

식 (4)에서 $dist_{SPE}$는 오직 정수의 덧셈 연산만을 필요로 하기 때문에 식 (2)와 비교하여 그 연산 과정이 매우 간단하다. SPE에 내장되어있는 누산기는 매 클럭마다 컨트롤 신호에 의해 덧셈기의 결과가 양인 경우 값을 더하고 음인 경우 값을 빼는 연산이 가능하여 $x_n - w_n$의 절대 값을 별도의 회로 없이 간편하게 계산할 수 있다. 여기서 *passin*은 시냅스 확장의 경우 마스터 관계에 있는 SPE로부터 전달받는 누적합 값을 의미하며 시냅스 확장을 하지 않는 경우나 해당 SPE의 마스터 SPE가 없는 경우에는 *passin* = 0 이 된다. SPE는 내부의 구성 레지스터에 의해 서밍노드 혹은 코호넨 노드로 결정된다.

2.3 Layer Processing Element(LPE)

LPE는 크게 두 가지 블록으로 나뉜다. 첫 번째 블록은 뉴런의 활성화 함수의 역할을 하는 Look Up Table(LUT)이며, 두 번째 블록은 코호넨 층을 구성하는 뉴런 중에서 승자(winner) 뉴런의 인덱스를 출력하기 위한 Kohonen Logic Block(KLB)이다. 퍼셉트론 연산을 위해 SPE의 출력값인 *sum*을 입력으로 받은 후 이를 LUT의 어드레스로 이용하며 해당 데이터를 활성화 함수의 출력으로 내보낸다. 활성화 함수는 원점을 기준으로 $\pm r$의 특정 구간만 고려하며 LUT의 출력값은 *sum*에 대한 이산함수 $F(sum)$으로 표현된다. LUT의 어드레스가 n bit 이고 x=0 인 순간에만 1의 값을 갖는 단위 임펄스 함수를 $\delta(x)$, 고려하고 있는 활성화 함수를 연속함수 $f(x)$라고 할 때 LUT의 출력 out_{percep}는 다음과 같이 표현된다.

$$\sum_{norm} = \left(\frac{\sum}{2^{n-1} - 0.5} - 1 \right) r \quad (5)$$

$$out_{percep} = F(\sum) = \sum_{k=0}^{2^n - 1} f(\sum_{norm}) \delta(\sum - k) \quad (6)$$

코호넨 연산의 경우 KLB는 Fig. 6과 같은 과정을 통하여 승자 뉴런의 인덱스를 출력하게 된다.

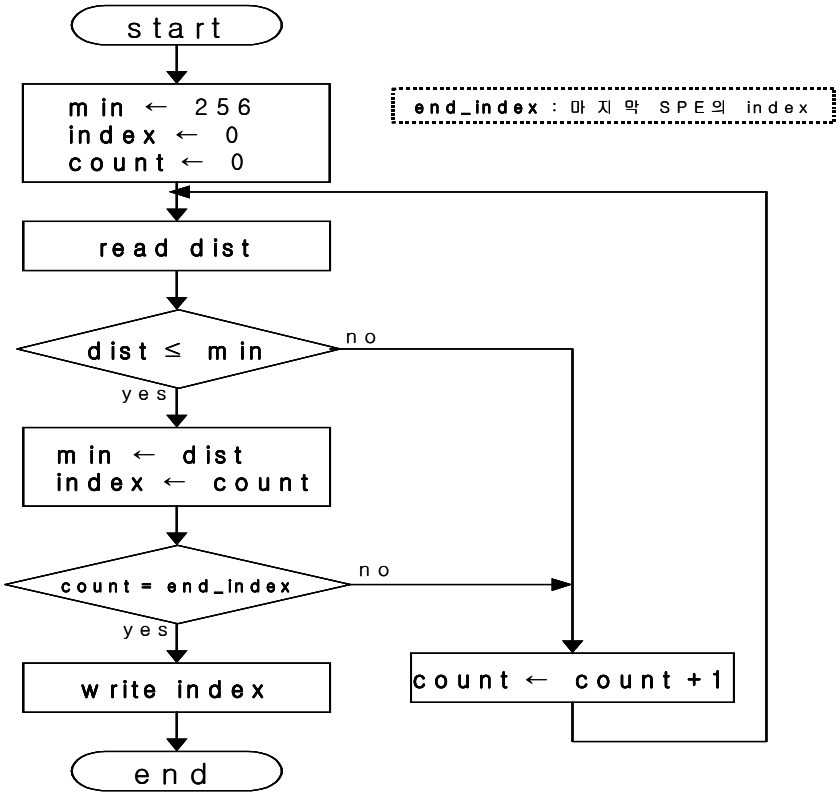

Fig. 6 Process of determining winner neuron in KLB

2.4 ERNIE를 이용한 네트워크 구현

ERNIE에서 특정한 네트워크를 구현하기 위해서는 사용된 MPU를 구성하고 있는 모든 SPE, LPE의 상태와 LUT의 참조 데이터를 결정해 주어야 한다. 따라서 SPE와 LPE의 내부에는 각 유니트의 상태를 규정하는 구성 레지스터가 존재하며 동작 초기에 설정 모드를 통하여 이러한 구성 레지스터 및 LUT의 메모리에 적절한 값을 넣어줌으로써 매우 다양한 신경회로망을 구현하는 것이 가능하다. SPE와 LPE는 각각 4bit, 2bit의 구성 레지스터를 갖고 Fig. 7과 같은 형식을 따른다.

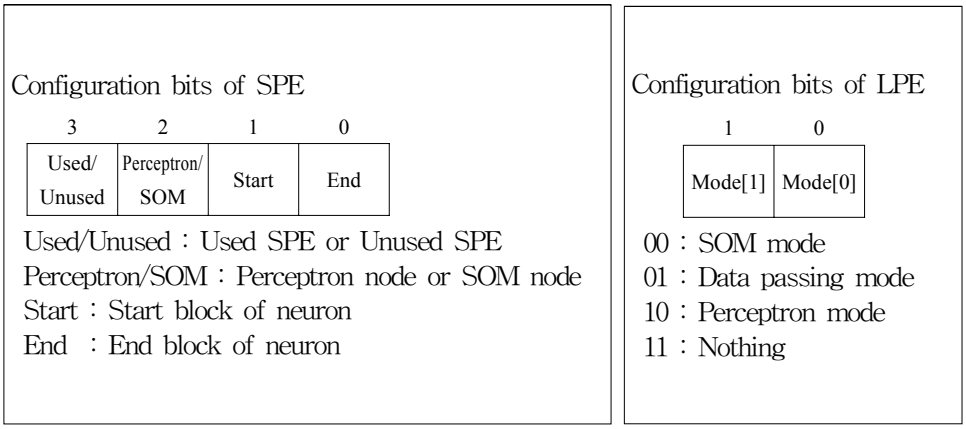

Fig. 7 Configuration register of SPE and LPE

모든 유니트들은 규정된 상태에 따라서 적절한 역할을 수행하게 되며 그 결과 단층 퍼셉트론을 비롯하여 두 개 이상의 은닉층을 가지는 다층 퍼셉트론은 물론 코호넨 층을 구성하여 Self Organization Map(SOM)을 구현할 수도 있다. 더욱이 퍼셉트론의 경우 비선형 함수의 출력값이 특정한 연산을 통해서 구해지는 것이 아니라 일종의 참조 테이블(Look up table)을 통해 얻어지는 것이기 때문에 사용 가능한 활성화 함수의 제약이 존재하지 않는다.

이와 같이 ERNIE는 확장성 및 범용성을 고려한 설계로 인하여 매우 다양한 종류의 신경회로망을 동일한 플랫폼(platform)위에서 별도의 노력 없이 구현할 수 있으며 상황에 따라서는 퍼셉트론과 SOM이 결합되어 있는 형태의 하이브리드(hybrid) 신경회로망을 구현하는 것 역시 가능하다.

2.5 성능분석

하나의 입력패턴에 대하여 출력패턴이 나오기까지의 시간 t(clock cycle)는 식 (7)과 같이 표현된다.

$$t = \sum_{i=0}^{l-1} n_i s_i + (l-1)c + m \tag{7}$$

여기서 n_i 는 i번째 레이어의 뉴런수, l 은 레이어의 개수, c 는 SPE의 연산시간(clock cycle), m 은 MPU의 개수 그리고 s_i는 네트워크의 시냅스 확장과 관련된 파라미터로서 식 (8)과 같은 범위를 만족하는 정수이다.

$$s_i - 1 < \frac{n_i}{h} \leq s_i \tag{8}$$

여기서 h는 1개의 SPE가 처리할 수 있는 시냅스의 수를 나타낸다. 식 (7)로부터 Fig. 8, Fig. 9와 같은 그래프를 얻을 수 있다.

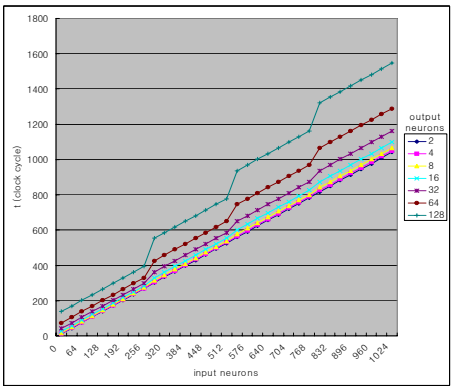

Fig. 8 Execution time of ERNIE

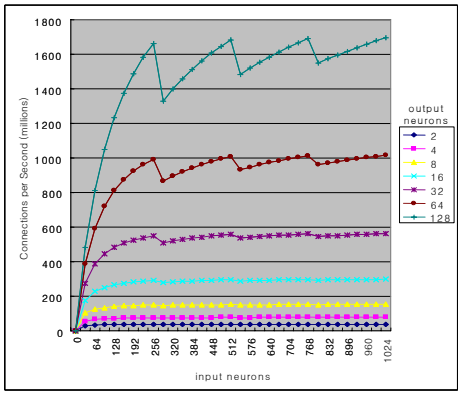

Fig. 9 Execution speed of ERNIE

Fig. 8에서 입력뉴런과 출력뉴런이 많아질수록 연산시간은 선형적으로 늘어나는 것을 확인할 수 있다. 일반적인 신경회로망은 뉴런의 증가에 따라 연산횟수가 지수적으로 증가한다. 그러나 ERNIE에서는 병렬적으로 연산이 수행되기 때문에 연산횟수의 지수증가에 대해 연산시간은 선형적으로 증가한다. Fig. 9는 20MHz로 동작시킬 때 ERNIE의 연산속도를 보여주는 그림이다. 지수적인 연산횟수에 대해 연산시간이 선형적으로 증가한다는 것은 바꿔 말하면 연산횟수가 늘어날수록 연산속도는 빨라진다는 말과 같다. 따라서 뉴런의 수가 증가할수록 연산속도는 점점 증가하게 된다. 출력뉴런의 수가 고정되어 있는 경우 입력뉴런의 수가 256의 배수인 경우에 연산속도는 극대값을 가지며 128개의 출력뉴런을 사용할 경우 1초에 최대 17억 번 정도의 시냅스 연산이 가능하다는 것을 알 수 있다.

3. 실험방법 및 결과

제안된 ERNIE를 Verilog HDL을 이용하여 설계하였고, 설계된 회로의 동작을 검증하기 위하여 HDL-모델과 C-모델에 신경망을 구성하는데 필요한 데이터를 동일하게 적용하여 그 실험결과를 서로 비교하였다. HDL 검증은 Modelsim 을 이용하여 타이밍 수준에서 수행 되었다. ERNIE의 동작을 증명하기 위하여 MPU(24 SPE/LPE) 3개를 이용하여 코호넨 네트워크를 구성하고 연산결과를 확인하였다. Fig. 10과 같이 8*8의 코호넨 층을 구성한 후 패턴 분류기의 벤치마킹용 데이터 세트로 자주 사용되는 Wisconsin

Breast Cancer Database (January 8, 1991)의 패턴 분류를 시도하였다[10].

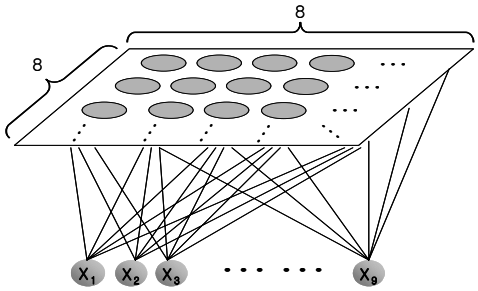

Fig. 10 2-dimensional Kohonen layer(8*8) with 9 input

사용된 데이타 세트는 유방암 진단을 위한 것으로서 9개의 필터링 된 유전자 데이터로 이루어진 샘플들이며 이에 대한 정보는 Table 3과 같다.

Table 3. Data set for breast cancer diagnosis

• 샘플수 - 699 (as of 15 July 1992) • 속성 ① Clump Thickness ② Uniformity of Cell Size ③ Uniformity of Cell Shape ④ Marginal Adhesion ⑤ Single Epithelial Cell Size ⑥ Bare Nuclei ⑦ Bland Chromatin ⑧ Normal Nucleoli ⑨ Mitoses • 패턴 분포 - 정상 : 458 (65.5%) - 비정상 : 241 (34.5%)

이들 중 정상 패턴 229개와 비정상 패턴 120개를 학습패턴으로 사용하고 나머지 350개의 패턴은 테스트 패턴으로 사용하였다. 학습은 PC에서 수행하였으며 학습율은 학습 횟수 10000회 동안 0.05에서 0.001까지 점진적으로 작은 값을 적용하였다. 학습의 결과로 얻은 가중치를 바탕으로 테스트 패턴을 C 모델링과 ERNIE를 이용한 신경회로망에 동일하게 적용시킨 결과는 Fig. 11과 같았다.

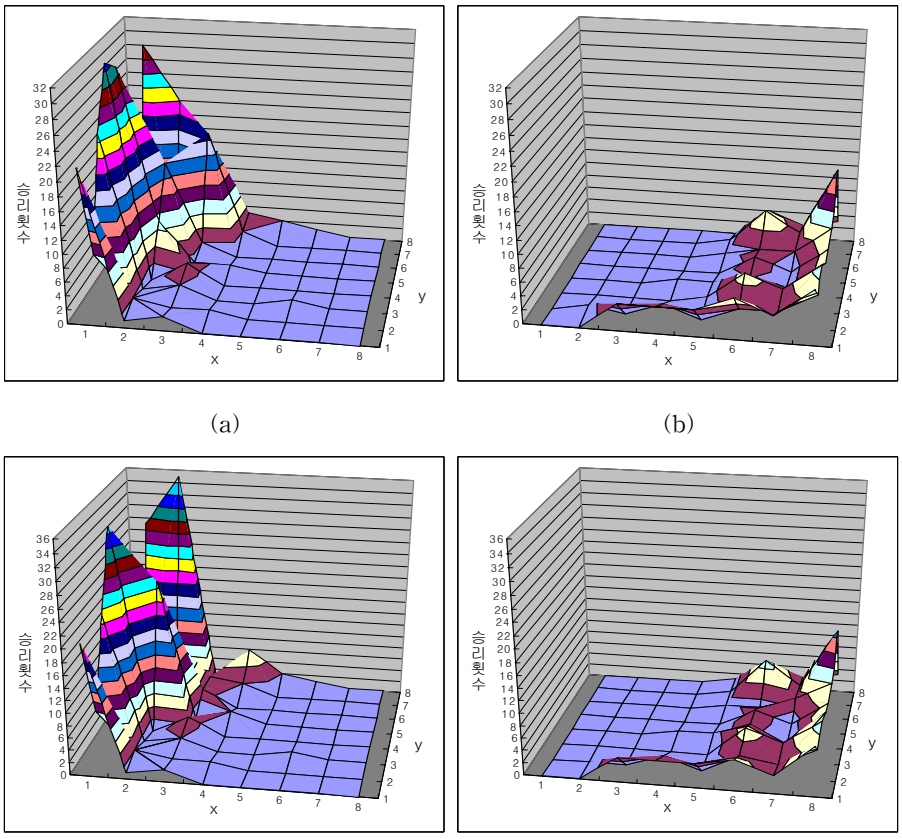

Fig. 11 Distribution chart of each neuron's winning number in Kohonen layer
(a) ERNIE benign (b) ERNIE malignant
(c) C modeling benign (d) C modeling malignant

Fig. 11에서 x-y 평면은 각 격자가 하나의 뉴런인 코호넨 층을 의미하며 z축은 입력 패턴에 대해 각 뉴런이 승자가 된 횟수를 나타낸다. 정상 패턴이 입력되었을 때 ERNIE와 C 모델링 모두 코호넨 층의 좌측 영역의 뉴런을 활성화 시켰으며 비정상 패턴이 입력

되었을 때에는 우측 영역의 뉴런을 활성 시켰다는 것을 확인할 수 있다. 이 실험을 위한 코호넨 연산에서는 하나의 패턴이 입력될 때 마다 63번의 비교 연산을 통해 승자 뉴런이 결정되는데 350개의 테스트 패턴에 대하여 ERNIE와 C 모델링과의 비교 연산 결과가 다르게 되는 경우는 최대 166번이었다. 따라서 코호넨 층의 뉴런수를 N, 테스트 패턴의 수를 I라고 하면 비교 연산 오차율 err_{comp}는 다음과 같다.

$$err_{comp} = \frac{166}{(N-1)I} \times 100 = \frac{166}{(64-1)\times 350} \times 100 \approx 0.75\,\% \tag{9}$$

또한 이 경우 서로 다른 비교 연산 결과로 인하여 승자 뉴런이 달라질 경우는 최대 47번이었으며 이 때의 오차율 err_{win}는 다음과 같이 표현된다.

$$err_{win} = \frac{47}{I} \times 100 = \frac{47}{350} \times 100 \approx 13.43\,\% \tag{10}$$

식 (10)은 C 모델링과 비교하여 그 결과가 최대 13.43%만큼 틀려진다는 것을 의미하지만 이러한 경우 대부분 서로 인접해 있는 뉴런이 승자가 되므로 코호넨 층에서의 비주얼한 결과를 확인하는 데에는 큰 문제가 되지 않는다.

4. 결 론

본 논문에서는 신경망 연산기를 디지털 하드웨어로 구현함에 있어서 발생하는 면적 증가 문제 및 네트워크의 확장을 비롯한 재구성 문제를 효과적으로 개선하기 위해서 제안된 ERNIE를 이용하여 암의 분류 문제에 대해 실험을 수행 하였고, 이상의 실험에서 C 모델링 검증의 결과와 HDL 검증 결과와의 오차가 신경회로망의 성능에 거의 영향을 미치지 않는 매우 근소한 범위 내에서만 발생한다는 것을 확인 하였다. ERNIE는 활성화 함수의 종류에 대한 제약이 없고 모듈러 구조를 통하여 얼마든지 확장이 가능하며 각 유니트의 상태에 따라 두 가지 종류의 신경회로망 연산이 가능하기 때문에 다양한 종류의 퍼셉트론 네트워크 및 코호넨 네트워크를 구성할 수가 있다. 이러한 재구성 과정은 단지 설정 모드에서 적절한 구성비트를 입력해주는 것만으로 가능하다. 또한 SIMD 구조를 적용하여 하드웨어의 병렬성을 최대로 높였으며 모든 연산을 파이프 라인의 형태로 수행하게 함으로써 연산 속도의 측면에서도 그 성능을 극대화 시켰다. 이와 같은 구조적 특성으로 인하여 ERNIE는 다목적 신경회로망 하드웨어 구조로서 매우 다양한 분야에서의 응용이 가능하다. 또한 ERNIE가 가지는 가장 중요한 특징 중의 하나는 동작 중에 재구성이 가능하다는 것이며 이를 이용하면 실시간 재구성이 필요한 진화 하드웨어 분야에서 그 효용성이

매우 클 것이라 기대된다.

ERNIE에는 퍼셉트론 네트워크 및 코호넨 네트워크의 학습을 위한 기능이 구현되어 있지 않으며, 온 보드(On-board) 학습을 가능케 하기 위해서는 별도의 학습 모듈이 필요하다. 학습 모듈은 ERNIE의 출력을 입력으로 받아서 저장된 목적 값과 비교한 후 적절한 가중치의 변화량을 산출하는 기능을 수행해야 하고 전체적인 학습 루프를 통제하는 컨트롤러의 역할을 해야 할 것이다. 이러한 학습 모듈과 진화를 위한 부가적인 부분이 보강된다면 ERNIE는 수많은 공학적 분야에서 훨씬 더 유용하게 사용될 수 있을 것이다.

참 고 문 헌

(1) Robert J. Schalkoff, 1997, *Artificial neural networks*, McGraw-Hill.

(2) Yingang Wang, Zuocheng Ma, Huaxiang Lu, Shoujue Wang, 2001, “Discussion on the methodology of neural network hardware design and implementation”, *Solid-State and Integrated-Circuit Technology*. vol. 1 pp. 113 -116.

(3) Miroslav Skrbek, 1999, *Fast neural network implementation*, Neural Network World, pp. 357-391.

(4) Tams Szab, Lrinc Antoni, Gbor Horvth, Bla Fehr, "A full-parallel digital implementation for pre-trained NNs", *Neural Networks, Proc. 2000 IJCNN*, vol. 2, pp. 49-54.

(5) Masahiro Murakawa, Shuji Yoshizawa, Isamu Kajitani, Xin Yao, Nobuki Kajihara, Masaya Iwata and Tetsuya Higuchi, 1999, "The GRD Chip : Genetic Reconfiguration of DSPs for Neural Network Processing", *IEEE Transaction on computers*, vol. 48, no. 6.

(6) Bernard Girau, "Digital hardware implementation of 2D complatible neural networks", *Neural Networks, Proc. 2000 IJCNN*, vol. 3, pp. 506-511.

(7) B. Pino, F.J.Pelayo, J. Ortega and A. Prieto, 1999, "Design and Evaluation of a Reconfigurable Digital Architecture for Self-Organizing Maps", *Microelectronics for Neural, Fuzzy and Bio-Inspired Systems*, pp. 395-402.

(8) S. Vitabile, A. Gentile, G.B Dammone, F. Sorbello, 2002, "Multi-layer perceptron mapping on a SIMD architecture", *Neural Networks for Signal Processing*, pp. 667-675.

(9) Simon Haykin, 1999, *Neural networks a comprehensive foundation*, Prentice hall, pp. 135, pp.448.

(10) O. L. Mangasarian and W. H. Wolberg, 1990, "Cancer diagnosis via linear programming", *SIAM News*, vol. 3, No. 5.