논문 2008-45SD-1-4

비선형 패턴 분류를 위한 FPGA를 이용한 신경회로망 시스템 구현

(Implementation of a Feed-Forward Neural Network on an FPGA Chip for Classification of Nonlinear Patterns)

이 운 규*, 김 정 섭**, 정 슬**

(Woon Kyu Lee, Jeong Seob Kim, and Seul Jung)

요 약

본 논문에서는 비선형 패턴 분류를 위해 FPGA 칩에 신경회로망을 구현하였다. 병렬처리 연산을 위해 순방향 신경회로망이 구현 되었다. 신경망의 학습을 off-line으로 한 다음에 가중치 값들을 저장하여 사용한다. 예로서, AND와 XOR 논리의 패턴 구분이 수행된다. 실험결과를 통해 FPGA에 구현된 신경회로망이 잘 작동하는 것을 검증하였다.

Abstract

In this paper, a nonlinear classifier of a feed-forward neural network is implemented on an FPGA chip. The feedforward neural network is implemented in hardware for fast parallel processing. After off line training of neural network, weight values are saved and used to perform forward propagation of neural processing. As an example, AND and XOR digital logic classification is conducted in off line, and then weight values are used in neural network. Experiments are conducted successfully and confirmed that the FPGA neural network hardware works well.

Keywords: FPGA, neural network, XOR pattern classification

Ⅰ. 서 론

최근에는 디지털 혁명에 의한 우리의 삶이 디지털화 되어 가고 있는 실정이다. Digital life라는 신조어가 생겼고, 유비쿼터스라는 환경 하에서 살게 되므로 우리 생활 중 많은 부분이 점차 지능화 되고 있다[1~2]. 이러한 디지털 시스템의 발달은 더 빠르고 값싸고, 작은 부피의 소자 및 시스템을 요구하게 되었고, 심지어는 지능적인 디지털 시스템에 대한 관심도가 높아져 가고 있다.

이처럼 디지털 시스템이 발달함에 따라 시스템 온 칩 개발에 대한 요구가 증가하고 있으며, 그 이전 단계인 Field Programmable Gate Array(FPGA) 칩에 대한 관심도 커지고 있다. FPGA는 사용자의 필요에 따라 프로그램함으로써 최적화된 프로그램을 사용할 수 있고 연산의 병렬 처리가 가능하므로 firmware에 따라 일반적인 MPU에 비교할 수 없을 정도로 고속의 연산이 가능하다. 이러한 특성으로 인해 FPGA를 고속의 처리 속도를 필요로 하는 Digital filter, 이산 제어 등에 활용하고자 하는 연구가 많이 진행되고 있다.

인간의 뇌를 모방한 신경회로망은 학습 능력과 적응 능력 등 탁월한 능력으로 인해 많은 분야에서 사용되어지고 있다. 제어시스템의 제어기로 신호처리 시스템의 필터로 사용되기도 하며 패턴 인식분야에서는 탁월한

* 정회원, 도담 시스템
(Dodam Company)
** 정회원, BK21 메카트로닉스그룹, 충남대학교
(Chungnam National University)
※ 본 연구는 산자부와 산업기술재단 지역혁신 인력양성 사업의 일부 지원에 의한 것입니다.
접수일자: 2007년7월26일, 수정완료일: 2008년1월3일

성능으로 인해 많이 사용되고 있는 실정이다[3~4]. 특히 역전파 알고리즘에 의한 학습방법은 뉴론의 수와 뉴론층의 수에 따라 연산의 양이 달라진다. 하지만 그에 따른 병렬처리에 의한 학습은 많은 연산과 시간이 필요하므로 빠른 PC나 고속 연산기인 DSP에 의존하고 있다. 이러한 경우 고비용을 요하게 되며, 그러한 시스템에 비교적 간단한 신경회로망을 적용하게 되면 충분한 역할을 수행하지 못하고 시스템 자원의 낭비를 초래하기도 한다.

최근에는 신경망의 보다 빠른 연산과 최적화된 신경망을 구현하기 위한 노력으로 FPGA로 설계하고자 하는 연구가 많이 되고 있다. 실시간으로 역전파 알고리즘을 적용하여 학습을 하기에는 아직 무리가 있으나 off-line 학습 방법으로 미리 학습된 가중치 값을 가지고 처리하고자 하는 방법들에 대한 연구가 활발하다[5~8].

본 논문에서는 이처럼 신경회로망을 FPGA로 구현함으로써 높은 처리 속도와 저가의 신경회로망 시스템을 실현하고자 한다. 간단하게 실험하기 위해 비선형으로 잘 알려진 XOR 논리회로의 패턴인식을 적용하여 보았다. 우선 컴퓨터에서 비선형 논리회로의 패턴인식을 시뮬레이션으로 수행하여 결과를 얻은 다음, 각 결정된 가중치 값을 FPGA에서 사용하여 신경회로망 구조를 구성하였다. FPGA에 구현된 신경회로망에 입력을 주어 실험결과가 잘 작동하는 것을 확인할 수 있었다.

Ⅱ. 신경회로망

본 논문에 사용된 신경회로망은 가장 일반적인 형태로 입력층과 은닉층, 그리고 출력층을 가지며 은닉층과 출력층에서 비선형 함수를 사용한다. 그림 1은 본 논문에 사용된 신경회로망의 일반적인 구조를 보여준다.

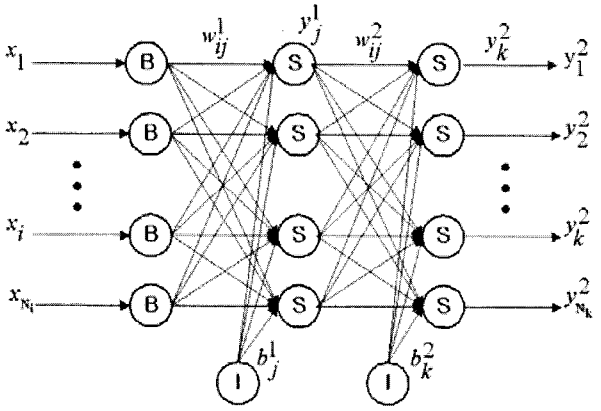


그림 1. 신경망 구조
Fig. 1. Neural network structure.

입력으로 x_i를 나타내고 은닉층과 출력층의 출력으로 각각 y_j^1와 y_k^2를 나타낸다. 입력층과 은닉층, 출력층의 수를 각각 N_I, N_J, N_K라 정하고 은닉 층과 출력층에 사용된 비선형 함수로는 시그모이드 함수를 사용한다. 입력이 입력 층을 통과해 은닉 층으로 입력되는 값을 구하면 다음과 같다.

$$s_j = \sum_{i=1}^{N_I} w_{ij} x_i + b_j^1 \tag{1}$$

위 값이 은닉층에서 비선형 함수인 시그모이드 함수를 통과한 후 출력되어 다음과 같아진다.

$$y_j = \frac{1}{1+e^{-S_j}} \tag{2}$$

출력층의 k번째 입력을 s_k라 하면 출력층의 출력은 다음과 같다.

$$s_k = \sum_{j=1}^{N_J} w_{jk} y_j + b_k^2 \tag{3}$$

$$y_k = \frac{1}{1+e^{-Sk}} \tag{4}$$

결과적으로 입력층에서 출력층까지 나타내면 다음과 같이 표현된다.

$$y_k = \frac{1}{1+e^{(-\sum_{j=1}^{N_J} wjk[\frac{1}{1+e^{-\sum_{i=1}^{N_I} w_{ij}x_i + b_j}}] + b_k)}} \tag{5}$$

위 식을 보면 신경회로망의 입력에서 출력까지 상당히 많은 연산을 요구한다는 알 수 있다. 위 식을 이용해 단순히 곱하기, 더하기 연산의 횟수만을 고려하여 각 층에서의 연산과 입력 층과 은닉층, 출력층의 수에 따른 연산을 비교해 보면 표1과 같다. 연산의 측정에서 비선형 함수의 연산은 제외했다.

출력이 1개라 가정할 때, 입력이 2개, 은닉층의 뉴론 수가 2개라면 각각 6회의 더하기, 곱하기 연산으로 최종 출력을 계산할 수 있지만 각 층의 뉴론 수가 늘어날수록 연산량은 기하급수적으로 증가 한다. 이러한 연산량 때문에 신경회로망을 실제 시스템에 실시간으로 적용하는 경우 어려움이 발생한다. 일반적인 MPU는 한번에 하나의 연산만을 수행하고 다음 연산을 수행하는

표 1. 각 층에서의 연산 횟수
Table 1. Number of calculations at each layer.

더하기 연산횟수	은닉층	Nj((Ni-1)+1)
	출력층	Nk((Nj-1)+1)
	Nji+ NkNj	
곱하기 연산횟수	은닉층	NiNj
	출력층	NjNk
	NjNi+ NkNj	

입력층 개수=Ni, 은닉층 개수=Nj,
출력층 개수 = Nk

방식을 택하고 있다. 즉, 몇 개의 연산을 병렬로 수행하는 것이 아니라 각 연산을 순위에 따라 직렬로 연산하게 된다.

Ⅲ. 신경회로망의 구현

1. 비선형 분류기의 구조

본 논문에서는 그림 2처럼 2개의 입력을 가진 입력층과, 2개의 뉴론을 가진 은닉층, 그리고 1개의 뉴론을 가진 출력층의 형태인 신경회로망을 프로그램하여 그 성능을 시험하였다.

이 신경회로망을 각 연산에 대한 블록 다이어그램으로 나타내보면 그림 3과 같다.

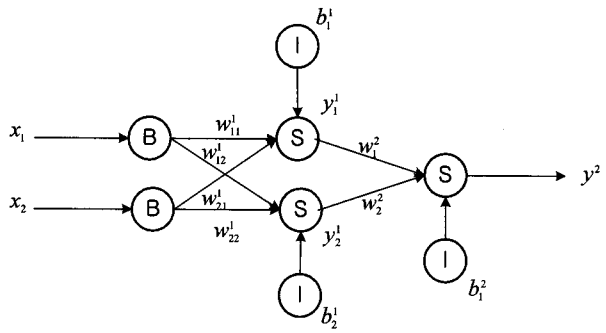


그림 2. 논리회로를 위한 신경회로망의 구조
Fig. 2. Neural network structure for Digital logic.

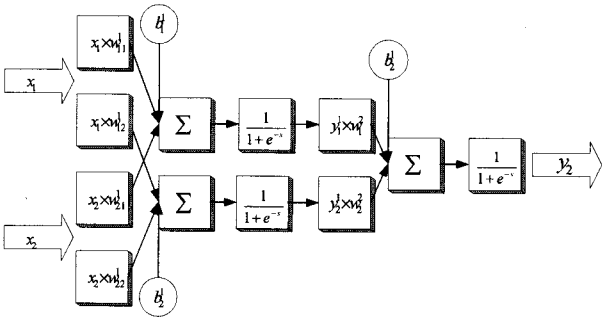


그림 3. 신경회로망 블록 다이아 그램
Fig. 3. Neural Network Block Diagram.

위 블록 다이어그램에서 입력을 2진수로 제한했다. 블록 다이어그램을 보면 6개의 곱셈이 나오지만, 입력 x_1, x_2을 weight로 곱하는 4개의 곱셈의 경우에 있어서 x_1, x_2값을 '0' 또는 '1'의 2진수로 제한하였다. 즉 입력은 2개의 디지털 입력으로 볼 수 있다. 그러면, 프로그래밍상에서는 가중치값을 출력하느냐 출력하지 않느냐를 결정하는 비교기로 대체할 수 있다. 곱셈기가 아닌 비교기로 대체하는 경우 FPGA에 요구되는 내부 용량이 상당량 줄어든다.

2. 데이터 형태 처리

VHDL에서는 소수점 연산을 제공하지 않는다. 만일 소수점 이하의 값을 모두 버리고 연산을 하면 최종오차가 지나치게 커지게 된다. 따라서 소수점 값의 연산을 위해 모든 값들은 일정한 값으로 shift시켜 소수점 이하의 값을 정수 값으로 만든 후 연산하는 방식을 택했다. 연산이 완료된 값은 상황에 따라 다시 반대방향으로 shift하여 사용했다. 이 방식을 사용해도 일정범위 이하의 소수점 이하의 값은 버림 현상이 발생하여 완전한 소수점 연산에 비해 어느 정도 오차가 발생한다.

3. 시그모이드 함수구현

은닉 층과 출력층에서 시그모이드 비선형 함수를 적용해야 하는데 VHDL에서는 이 함수를 제공하지 않는다. 그래서, 이 함수의 입력 값과 출력 값을 표로 만들어 ROM에 저장하여 사용했다.

비선형 함수의 연산을 위해 ROM에 저장되는 데이터의 실제 형식은 ROM을 이용해 (4)의 임의의 값 x에 대하여 x값이 상수 a를 곱한 ax를 ROM의 절대번지로 적용하고, y를 ROM의 ax번지가 나타내는 데이터로 정한다. ROM의 크기는 8bit X 256이므로 총 256개의 데이터를 저장한다. 그러므로 ROM의 번지 값은 0000~0xFF의 값을 가진다. 이 ROM을 이용해 x값의 범위가 -16~15.875이고 x값의 간격이 0.125단위인 경우, 각 번지에 대해 각각의 x에 대한 y값을 맵핑할 수 있게 된다. 이 경우 주소 값을 정수로 만들기 위한 상수 a=8(또는 $a=2^3$)이 된다. 즉 ROM의 번지가 0x01일 때 이 번지가 의미하는 값은 0x01/a=0.125가 된다. 0.125를 위 식에 대입하면 0.5312를 가지며 0.5312에 b=128(또는 $b=2^7$)를 곱한 값을 ROM의 데이터 값으로 사용했다.

표 3. ROM 데이타
Table 3. ROM Data Table.

번지값	DATA	번지값	DATA
0x00	0x40;	0x09	0x60
0x01	0x43;	0x0A	0x63
0x02	0x47;	0x0B	0x66
0x03	0x4B	0x0C	0x68;
0x04	0x4F	0x0D	0x6A
0x05	0x53;	0x0E	0x6D
0x06	0x56;	0x0F	0x6E
0x07	0x5A;	…	…
0x08	0x5D		

그리고, 음수의 표현은 2의 보수를 사용해 0x80~0xFF는 음수로 -1~-128을 나타낸다. 각 주소 값에 대한 ROM 데이터를 표 3에 나타냈다.

연산에 사용되는 가중치값과 bias값의 경우 2^3를 곱하여 소수점 이하의 값이 연산에 사용될 수 있도록 했다.

4. 신경회로망의 설계

그림 3의 신경회로망에서 은닉층으로 입력되는 값, 즉 비선형 함수로 통과하기 전의 값은 다음과 같다.

$$s_j' = \sum_{i=1}^{N_I} w_{ij}' x_i + b_j^{1\prime} \tag{6}$$

$w_{ij}' = 2^3 \times w_{ij_j}^1$, $b_j^{1\prime} = 2^3 \times b$, $s_j' = 2^3 \times s_j$이다.

비선형 함수를 사용하기 위해 ROM에 번지로서 입력되는 값은 실제 값에 상수 a=8를 곱한 값이므로 s_j'을 ROM의 주소 값으로 그대로 사용한다.

비선형 함수를 통과하기 전의 일반식을 보면 다음과 같다.

$$s_k = \sum_{j=1}^{N_I} w_{jk} y_j + b_k^2 \tag{7}$$

$w_{jk}' = 2^3 \times w_{jk}$, $y_j^{1\prime} = 2^7 \times y_j^1$,
$w_{jk}' y_j^{1\prime} = 2^{10} \times w_{jk} y_j^1$, $b_k^{2\prime} = 2^3 \times b_k^2$

위 식에서 앞에서 정의한 규칙으로 가중치 값과 은닉층의 출력 값을 곱하면 bias값과 배율이 맞지 않는다. 따라서 $w_{jk}' y_j^{1\prime} \times 2^{-7}$에 bias값 $b_k^{2\prime}$를 더한 다음 비선형 함수를 통과시킨 후 최종결과가 출력된다.

최종결과는 마지막 ROM 데이터의 출력이므로 $y_k^{2\prime} = 2^3 \times y_k^7$이 된다. 그러므로 실제 결과와 배율을 맞추기 위해서는 $y_k^{2\prime}$에 2^{-7}을 곱하면 된다. 위에서 설명한 방법들을 이용해 프로그래밍하기 위해 다시 블록 다이어그램으로 나타내 보면 그림 4와 같다.

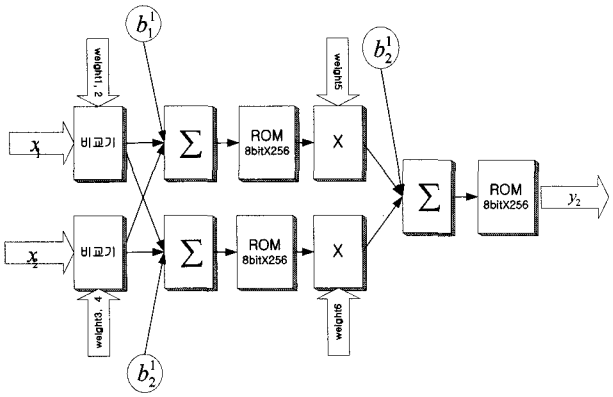


그림 4. 프로그래밍 블록 다이아 그램
Fig. 4. Programming Block Diagram.

5. FPGA 프로그래밍

그림 4를 보면 2개의 비교기와 3개의 비선형 함수의 연산을 위한 ROM, 그리고 2개의 곱셈 연산기, 더하기 연산 등이 사용된다. 이런 연산 과정을 프로그래밍하기 위해 A~F의 5개의 구역으로 구분하였다. 이 5개의 영역을 모두 더함으로써 그림 9의 전체 프로그램을 완성할 수 있다.

A. 입력 값에 따라 w_{ij}'의 출력을 결정하는 비교기

그림 5의 모듈을 살펴보면 가중치값 w_{11}', w_{12}'이 상수 형식으로 입력이 되고 입력값 A(또는 B)가 입력된다. 입력 A(또는 B)에 따라 w_{11}',w_{12}'(또는 w_{21}',w_{22}')이 출력되거나 0이 출력된다.

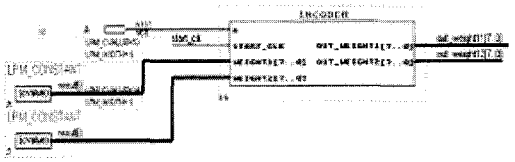


그림 5. 입력 값의 비교기
Fig. 5. Comparator of inputs.

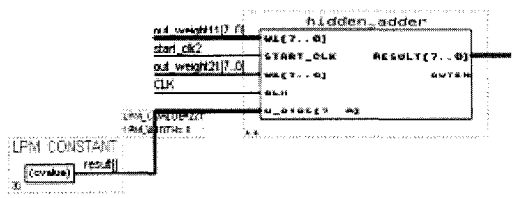


그림 6. 은닉층 더하기 블록
Fig. 6. Summing block at the hidden layer.

B. 은닉층에서의 $s_j' = \sum_{i=1}^{N_I} w_{ij}' x_i + b_j^{1}'$ 연산

이 구역은 $w_{ij}' x_i$의 합에 bias값 b_j^1를 더하는 역할을 하는 모듈로 구성되어 있다.

C. 은닉 층의 ROM 영역

이 영역은 8bit x 256 크기의 ROM으로 구성되어 있는 은닉층에서의 출력을 위한 비선형 함수를 연산하기 위한 영역이다. ROM의 주소 값으로 s_j'이 입력되면 거기에 대응되는 ROM DATA인 y_j'가 출력된다. y_j'는 은닉 층에서의 출력이다.

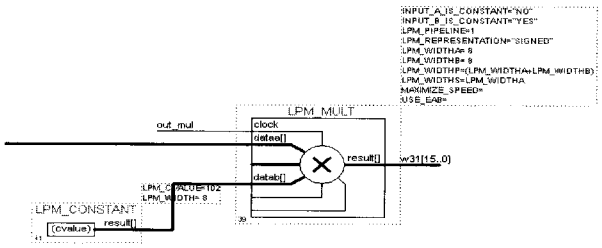


그림 7. 곱셈기 블록

Fig. 7. Multiplication block.

D. 은닉 층의 출력 값과 가중치의 곱셈

이 영역에서 가중치값과 은닉 층의 출력 값의 곱인

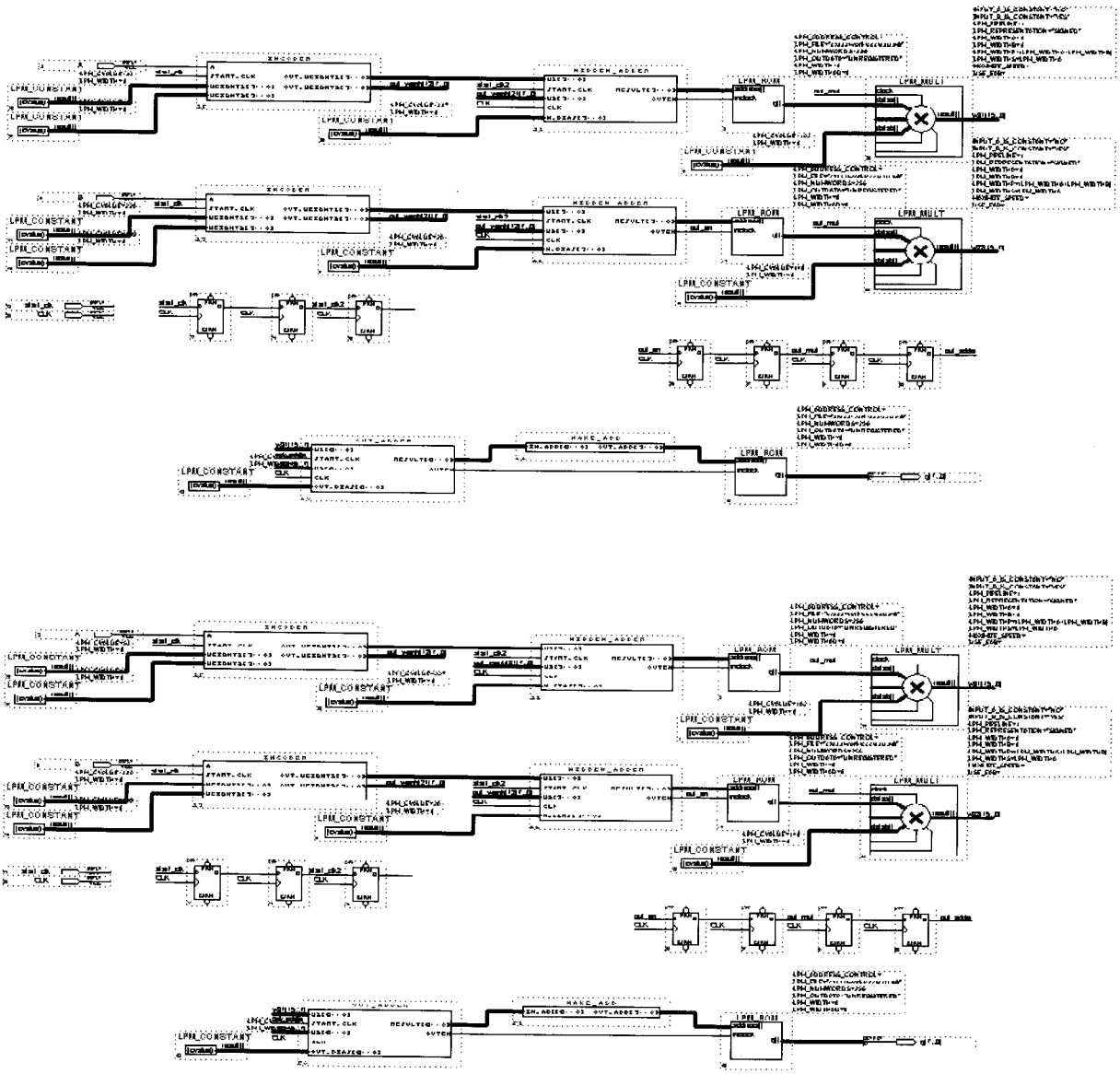


그림 9. 전체 신경회로망 설계

Fig. 9. Overall design of neural network.

$w_{jk}{}'y_j{}'$ 연산이 실행된다. 이 연산 결과의 출력은 16bit의 Data로 출력된다. 그림 7에 곱셈기 블록이 나타나 있다.

E. 출력 층에서의 $s_k = \sum_{j=1}^{N_j} w_{jk} y_j + b_k^2$ 연산

이 영역에서는 출력의 비선형 함수로 입력되는 값인 s_k 값이 연산된다. 이 함수로 입력되는 $w_{jk}y_j^1$는 $w_{jk}{}'y_j^{1\prime} \times 2^{-7}$값이다. $w_{jk}{}'y_j^{1\prime} \times 2^{-7}$는 $w_{jk}{}'y_j^{1\prime}$값의 하위 7bit를 버리고 상위 8bit만을 사용함으로써 간단하게 연산할 수 있다. 즉, $w_{jk}{}'y_j^{1\prime} \times 2^{-7}$값은 $w_{jk}{}'y_j^{1\prime}$값을 왼쪽으로 7bit shift 연산한 값이다. 그리고 출력층의 bias 값인 $b_k^{2\prime}$가 입력된다. 이 값들을 모두 합한 값이 이 영역의 출력이 된다. 그림 8에 연산 블록이 나타나 있다.

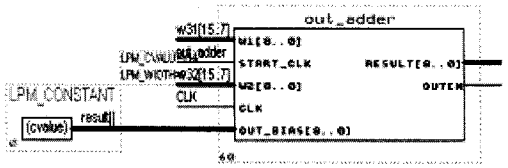


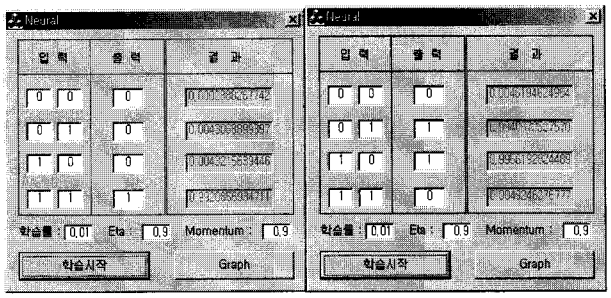
그림 8. 출력 층 더하기 블록
Fig. 8. Summing block at the output layer.

F. 출력 층의 ROM 영역

이 영역은 출력 층에서의 출력을 위한 비선형 함수를 연산하기 위한 영역이다. 최종 출력결과는 실제 결과와 비교하여 $y_k^{2\prime} = 2^7 \times y_k^2$로 표현할 수 있다. 위의 A~F 영역을 모두 더하여 전체 프로그램을 완성하면 그림 9와 같다. 그림 9를 보면 부가적인 모듈이 더 있는데 이것들은 각 연산의 제어 clock 생성과 정확한 연산을 위해 각 data의 bit수를 같게 맞춰주는 기능을 수행한다. 각 연산은 A~F의 순으로 제어 clock에 의해 순차적으로 진행되면 계속해서 반복된다.

Ⅳ. 실험 및 결과

위 프로그래밍 결과를 검증하기 위해 AND연산과 대표적인 비선형 논리 함수인 XOR를 Off-line으로 학습한 시킨 후 그 출력 결과를 확인해 보았다[9]. 각 논리 연산의 학습 결과는 그림 10에 나타난다. 학습 완료 후 출력되는 결과에서 약 0.008 이하의 오차를 두고 수렴



(a) AND (b) XOR

그림 10. AND와 XOR의 학습 결과
Fig. 10. Result of AND and XOR.

표 4. weight값과 bias값
Table 4. Values of weights and biases.

	AND	XOR
w_{11}^1	-3.008843	6.694559
w_{12}^1	-4.887427	4.300140
w_{21}^1	-2.988632	-7.142780
w_{22}^1	-4.910026	-4.475965
w_1^2	-5.649880	12.735007
w_2^2	-8.432431	-13.479579
b_1^1	3.783285	-3.593142
b_2^1	6.825010	2.458523
b_1^2	5.798102	6.703323

하는 것을 볼 수 있었다. 학습이 완료된 후의 가중치 값과 bias 값은 표 4에 나타냈다. 이 값들을 앞에서 작성한 그림 9의 프로그램에 입력한 후 시뮬레이션을 했다. 입력이 "00", "01", "10", "11"일 때의 출력 결과를 각 논리 연산의 진리표와 비교했다.

표 5. 각 논리 연산의 진리표와 신경회로망 출력
Table 5. Truth table and neural network output.

AND의 진리표			AND의 출력결과		
A	B	Y	A	B	Y
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	1	1	126
XOR의 진리표			XOR의 출력결과		
A	B	Y	A	B	Y
0	0	0	0	0	0
0	1	1	0	1	127
1	0	1	1	0	127
1	1	0	1	1	0

표 5의 진리표와 출력결과의 비교를 보면 진리표가 1일 때 이 결과를 실제 논리 연산의 진리표와 비교한 결과는 다음의 표 5와 같다.

표 5의 진리표와 출력결과의 비교를 보면 진리표가 1일 때 출력은 126~127이 된다. 실제 출력 결과는 $y_k^2 = 2^{-7} \times y_k^{2\prime}$이므로 126~127을 2^{-7}로 곱하면 0.9844375~0.9921875가 된다. 약 0.008의 오차를 두고 수렴하는 것을 알 수 있다. 하지만, 결과가 '0', '1'로 표현된다고 할 때 '0', '127'('126')도 그 의미가 크게 다르지 않으므로 0.9844375~0.9921875의 오차는 전체 시스템에 크게 영향을 끼치지 않는다고 볼 수 있다.

입력에 따른 출력의 연산 시간을 측정해 보면, 모든 연산은 CLK 신호 즉 동작 CLOCK에 동기를 맞춰서 실행된다. 여기에 사용된 CLOCK은 25MHz이다. start_clk이 입력된 후 결과 q[7..0]이 출력되는 시간을 한 번의 연산 시간으로 보아야 할 것이다. 시뮬레이션 결과를 확인 하면 그 시간이 약 500ns 임을 알 수 있다. 은닉 층, 출력 층의 수가 증가하면 각 값의 덧셈 연산을 하는 횟수가 증가 하여 어느 정도 연산 시간이 어느 정도 증가되겠지만, 각 층이 증가하는 수에 정비례하여 증가하지는 않는다.

시뮬레이션 결과는 그림 11과 그림 12에 나타나 있다.

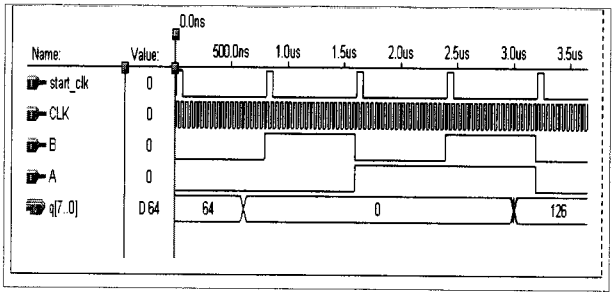


그림 11. AND 연산의 결과
Fig. 11. Results of AND.

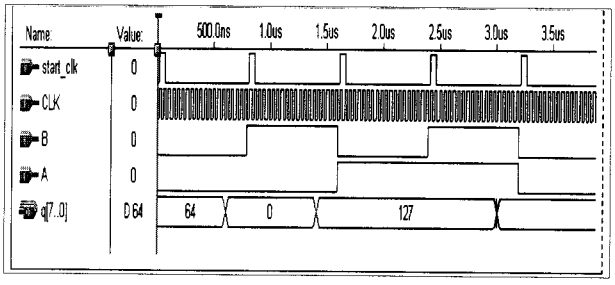


그림 12. XOR 연산의 결과
Fig. 12. Result of XOR.

V. 결 론

본 논문에서는 많은 계산을 요하는 신경회로망 시스템을 병렬 연산이 가능하도록 프로그래밍된 FPGA에 적용함으로써 실시간으로 시스템에 적용이 가능하도록 구현하고자 했다. 실험 결과로 충분히 만족할 정도의 빠른 연산 속도를 보여주었으며, 그 결과 또한 정확했다. 비록 본 논문에서는 간단한 논리 회로를 실험하였지만, 좀 더 복잡한 시스템에도 이 신경회로망시스템이 적용하는 것이 가능하다. 물론 곱하기 연산 등은 FPGA의 용량을 증가시키므로 대용량의 Gate 수가 많은 FPGA를 사용해야 한다.

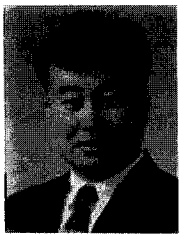
앞으로의 연구로는 복잡한 구조의 역전파 학습 알고리즘을 실시간으로 구현할 수 있도록 신경회로망을 FPGA 칩에 구현하는 것이다.

참 고 문 헌

[1] J. H. Lee and H. Hashimoto, "Intelligent Space, its past and future", pp. 126-131, *IECON*, 1999.
[2] 유비쿼터스 시스템(특집), 제어 자동화 시스템 공학회 논문지, 11권 12호, 2005.
[3] 김 성수, 정 슬, "FPGA를 이용한 범용모션 컨트롤러 개발", 제어 자동화 시스템 공학회 논문지, 10권 1호, pp. 73-80, 2005.
[4] P. Kollig, B. M. Al-Hashimi, and K. M. Abbott, "FPGA Implementation of High performance FIR Filters", *IEEE Symposium on Circuits and Systems*, pp. 2240-2243, 1997.
[5] Xinsheng Yu and Don Dent, "Implementing neural network in FPGAs", *Proceedings of IEE*, pp. 1-5, 1994.
[6] Acosta Gerardo and Tosini Marcelo, "A Firmware Digital Neural Network for Climate Prediction Applications", Proc. *IEEE intelligent Control*, pp. 127-131, 2001.
[7] Marco Krips, Thomas Lemmert, and Anton Kummert, "FPGA Impementation of a Neural Network for a Real-Time Hand Tracking System" *Proc. IEEE Workshop of Electronic Design, Test and Applications*, pp. 1453-1457, 2002.
[8] Y. Taright and M. Hubin, "FPGA Implemneta-tion of a Multilayer Percepttron Neural network using VHDL", *Proceedings of ICSP*, pp. 1311-1314, 1998.
[9] 정 슬, "인공 지능 시스템 I,II : 신경회로망 구조 및 사용법", 충남대학교 출판부, 2003.

저 자 소 개

이 운 규(정회원)
1996년 충남대학교 메카트로닉스' 공학과 학사 졸업.
2000년 충남대학교 메카트로닉스 공학과 석사 졸업.
2008년 현재 도담시스템 연구원
<주관심분야 : 차량 로봇 시스템, 원격제어>



정 슬(정회원)
1988년 미국 웨인 주립대 전기 및 컴퓨터 공학과 학사졸업.
1991년 미국 캘리포니아대 전기 및 컴퓨터 공학과 석사졸업.
1996년 미국 캘리포니아대 전기 및 컴퓨터공학과 박사졸업
2008년 현재 충남대학교 메카트로닉스공학전공 교수
<주관심분야 : 지능시스템 및 로봇 시스템>

김 정 섭(정회원)
1996년 충남대학교 메카트로닉스 공학과 학사졸업.
2008년 현재 충남대학교 메카트로닉스공학과 석사과정.
<주관심분야 : 신경회로망 칩 설계, 지능 로봇 시스템>