

Expert Network의 모듈형 계층구조를 이용한 범용 연산회로 설계

General Purpose Operation Unit Using Modular Hierarchical Structure of Expert Network

*양정모, *홍광진, *조현찬, *서재용, **전홍태

* 한국기술교육대학교 정보기술공학부

** 중앙대학교 전자전기 공학부

Jung-Mo Yang*, Kwang-jin Hong*, Hyun-Chan Cho*, Jae-yong Seo*, Hong-tae Jeon**

* School of Information Technology, Korea University of Technology and Education.

** School of Electrical and Electronics, Chung-Ang Univ.

E-mail : iikimo@hanmail.net

Abstract

By advent of NNC(Neural Network Chip), it is possible that process in parallel and discern the importance of signal with learning oneself by experience in external signal. So, the design of general purpose operation unit using VHDL(VHSIC Hardware Description Language) on the existing FPGA(Field Programmable Gate Array) can replaced EN(Expert Network) and learning algorithm. Also, neural network operation unit is possible various operation using learning of NN(Neural Network).

This paper present general purpose operation unit using hierarchical structure of EN. EN of presented structure learn from logical gate which constitute a operation unit, it relocated several layer. The overall structure is hierarchical using a module, it has generality more than FPGA operation unit.

1. 서론

FPGA(Field Programmable Gate Array)는 산업 현장에서 보다 빠르게 필요한 IC(Integrated Circuit) 소자를 구현하는 방법이다. FPGA는 기본 논리 회로를 미리 만들어 놓은 후 최종적인 사용자의 회로를 전기적인 퓨즈를 사용하여 프로그래밍 할 수 있는 형태를 지니고 있다. [1] 그렇지만 프로그래밍 단계를 거치지 않고선 회로를 구성 할 수 없으므로 전문 지식이 없는 사용자가 사용하기에는 부적합하다. 이와 같은 FPGA의

단점을 본 논문에서는 신경망의 학습기능을 사용하여 개선하고자 한다.

신경망은 선택된 학습데이터를 통하여 가중치를 학습하고 그 데이터에 대한 특성을 일반화시킴으로서 학습 기능을 수행한다. 또한 학습 기능은 여러 학습 데이터에 적용할 수 있는 범용성을 지니고 있다. 따라서 신경망을 FPGA에 적용하면 학습데이터에 따라서 프로그래밍 단계 없이 학습으로 회로가 구성되는 범용적인 시스템을 만들 수 있다.

본 논문에서는 대표적인 연산 회로를 학습 데이터로 선택하고 신경망으로 사용하여 학습을 통해 시스템을 구현 하고자 EN(Expert Network)의 모듈형 계층 구조를 제안한다.

2. Expert Network

본 논문에서는 복잡한 연산회로의 전역적인 해에 접근하기 위해 EN을 도입하여 복잡한 문제를 단순한 여러 개의 부분적인 해로 나누어 해결하고자 한다. [2] 연산회로의 구성은 여러 논리 Gate로 구성된다. 본 논문에서 제안한 신경망에서 논리 Gate는 EN이 역할을 대신한다.

MLNN(Multi Layer Neural Network)으로 구성된 EN의 구조는 그림 1과 같다. 이 구조는 은닉층과 출력층에 활성화 함수와 Bias를 사용하여 구성하였다. [3]

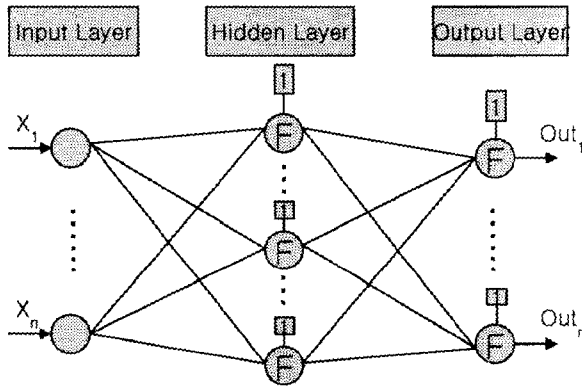


그림 1. MLNN으로 구성된 EN

2.1 EN의 전 방향 네트워크

은닉층의 1번째 뉴런의 출력은 식(1)과 같이 구할 수 있다.

$$net_k^i = \sum_{i=1}^P w_{ki}^i x_i + Bias_k^i \times w_{k0}^i \quad (1)$$

$$Bias = 1$$

$$s_k^i = f(net_k^i)$$

(S_k^i 의 활성화 함수는 Sigmoid 함수이다.)

Output Layer의 뉴런의 j번째 출력은 다음과 같이 구할 수 있다.

$$y_k^j = f\left[\sum_{i=1}^M [w_{ki}^j \cdot f\left(\sum_{i=1}^P [w_{ki}^i x_i + Bias_k^i \times w_{k0}^i]\right) + Bias_k^j \times w_{k0}^j]\right] \quad (2)$$

$$= f(net_k^j) \quad j=1,2,\dots,Q$$

2.2 EN의 학습 알고리즘

EN의 학습 알고리즘은 역전파 알고리즘으로 감독학습(Supervised learning)에 의해 학습 목표가 주어지면 원하는 값과 실제로 출력된 값 사이의 차이를 줄여나가는 방법이다. Output Layer의 뉴런에서 나온 출력값과 학습패턴의 목표값의 오차를 계산한다. 오차 계산과 연결강도 조정은 다음과 같이 구할 수 있다.

$$\delta_{j,k} = out_j(1 - out_j)(Target - out_j) \quad (3)$$

$$\Delta w_{ij,k} = \eta \delta_{j,k} OUT_{ih} \quad (4)$$

$$w_{ij,k}(n+1) = w_{ij,k}(n) + \Delta w_{ij,k}$$

모든 학습 패턴의 출력값과 목표값 사이에서 발생하는 error가 경계영역에 도달하면 학습을 종료시킨다.

3. EN의 모듈형 계층구조

EN의 모듈형 계층구조는 그림 2와 같이 논리 Gate의 역할을 대신하는 EN이 여러 Layer에 재배치하여 계층 구조를 형성하는 구조이다.

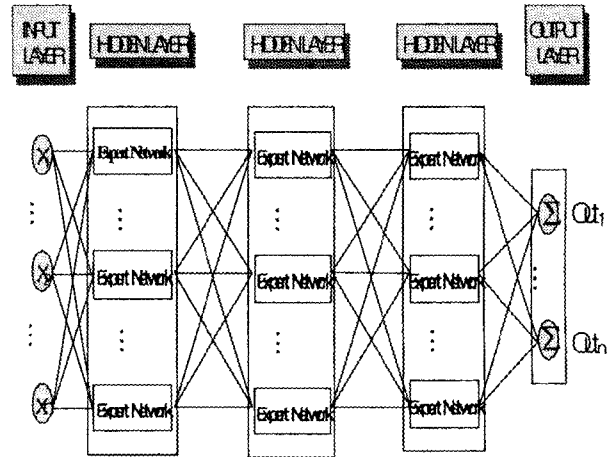


그림 2. EN의 모듈형 계층구조

층을 구성하고 있는 EN은 필요에 따라 다양한 학습 패턴으로 학습되어지는 신경망이다. 다양한 EN으로 구성된 모듈형 계층 구조 신경망은 요구되는 학습 데이터 및 학습에 따라서 여러 연산을 수행 할 수 있다.

3.1 모듈형 계층구조의 전 방향 네트워크

3.1.1 선 학습

각 EN들에게 지정된 학습 패턴에 따라 학습을 수행한다. EN의 학습방법은 Back Propagation이며 모든 출력값이 목표값에 근접하면 최종 Weight 값들을 고정시킨 후 학습을 종료한다.

3.1.2 후 학습

미리 학습된 EN을 여러 Layer에 배열한 후 EN들 사이의 연결강도를 균일하게 초기화 한다. 계층적 구조로 인해 각 EN에는 이전 layer의 EN 수만큼의 입력신호가 발생된다. EN 입장에서 어떤 입력신호가 가장 최적인가 판단하는 것은 입력신호 연결강도(Weight)의 높음으로 결정하였다. 입력신호는 EN의 입력 뉴런 수만큼 선택하였다. EN의 입력 뉴런 선택은 다음과 같다.

$$EN_input = Ascending_Sort(Weight_{ij}) \quad (5)$$

(Ascending_Sort 함수는 오름차순 정렬 함수이고 정렬된 Weight값 중 EN_input 뉴런 수만큼 출력을 반환 한다.)

또한 EN 사이의 연결강도들의 경쟁을 유발하기 위해서 선택된 연결강도만을 수정하였다. 모든 값이 목표값에 만족하면 학습을 종료한다.

3.2 모듈형 계층구조의 오차 역전파

출력층의 오차전파는 다음과 같다.

$$Out_k = \sum_{i=1}^n w_{im} EN_out_i \quad (6)$$

$$\delta = (Target - Out_k)$$

은닉층 Layer_3에서 j번째 EN에서의 오차전파는 다음과 같이 구할 수 있다.

$$\delta_j = \sum_{i=1}^n OutputLayer_weight_{ji} \cdot \delta_i \cdot \sum_{i=1}^n Layer[3]_EN_Out_i \quad (7)$$

EN의 weight가 서로 경쟁 하는 Layer 에서의 오차전파는 다음과 같이 구할 수 있다.

$$\delta = \sum_{i=1}^n Layer(n)_Error_j \cdot \sum_{i=1}^n Layer(n-1)_EN_Out_i \quad (8)$$

각 층에서의 새로운 연결강도들은 최종적으로 다음 식에 의해 조정된다. 단 Out_layer층의 연결강도를 제외한 나머지 연결강도들은 EN에 의해 선택되어진 연결강도만을 조정한다.

$$w_{kj}(n+1) = w_{kj}(n) + \eta \delta Layer[n-1]_EN_out_j \quad (9)$$

$$w_{ji}(n+1) = w_{ji}(n) + \eta \delta Layer[n-1]_EN_out_i$$

4. 실험 및 결과

본 논문에서 제안한 Expert Network의 모듈형 계층구조를 사용하여 전가산기 연산회로를 구성하여 보았다. EN의 학습 패턴 및 구성은 표 1과 같다.

EN 학습패턴	뉴런의 구성	학습 방법
AND	2-5-1	Back propagation
OR	2-5-1	Back propagation
XOR	2-5-1	Back propagation
YES	1-5-1	Back propagation

표 1. Expert Network의 종류와 구성

전가산기 연산회로 구성을 위해 선택된 학습패턴으로 학습된 EN으로 구성된 모듈형 계층구조는 그림 3과 같다.

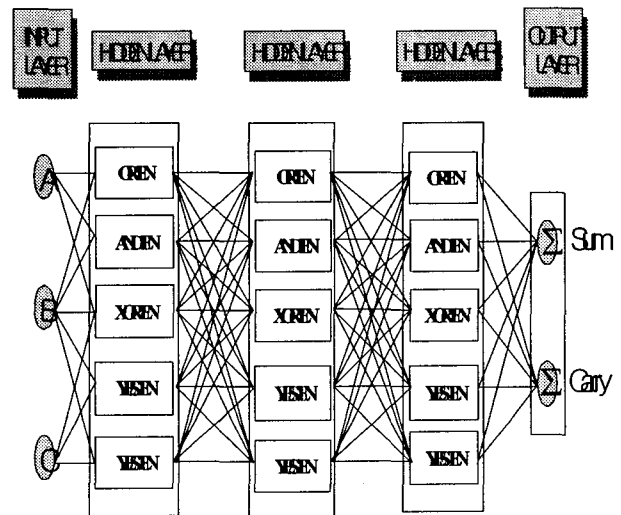


그림 3. 전가산기 연산회로 구성을 위한 EN의 모듈형 계층구조 신경망

학습을 위해서 EN의 학습률은 [0.05], 모듈형 계층구조에 사용된 학습률은 [0.3]으로 설정하였고 학습은 batch 형태로 진행 시켰다. 전가산기 연산회로 학습을 위한 학습 데이터는 표 2와 같다.[4]

표2. 전가산기 진리표

INPUT			OUTPUT	
A	B	Carry _{n-1}	SUM	Carry _n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

제안된 신경망의 최종 학습된 후 출력은 다음과 같다.

표3. 모듈형 계층구조 신경망으로 구현한 전가산기 연산회로 결과표

INPUT			OUTPUT	
A	B	Carry _{n-1}	SUM	Carry _n
0	0	0	0.012	0.037
0	0	1	0.997	-0.040
0	1	0	0.998	0.002
0	1	1	-0.014	1.019
1	0	0	0.998	0.002
1	0	1	-0.014	1.019
1	1	0	0.038	0.944
1	1	1	0.999	0.002

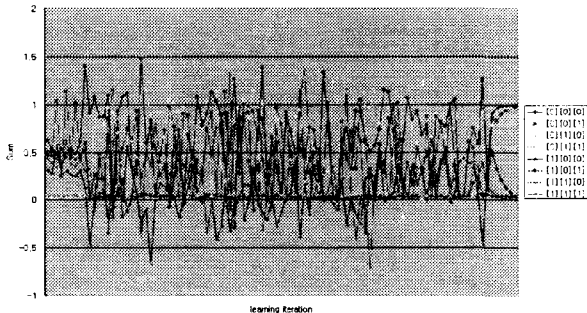


그림 4. 전가산기 Sum 의 학습결과

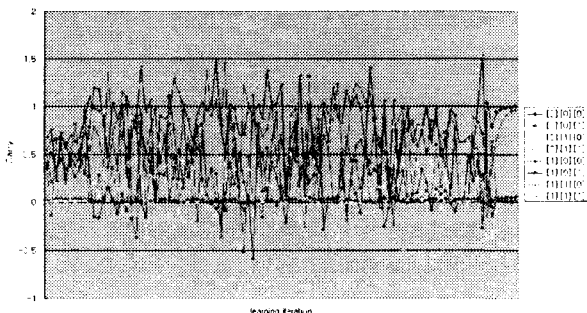


그림 5. 전가산기 Carry 의 학습결과

최종 결정된 EN의 연결강도는 그림 6과 같다.

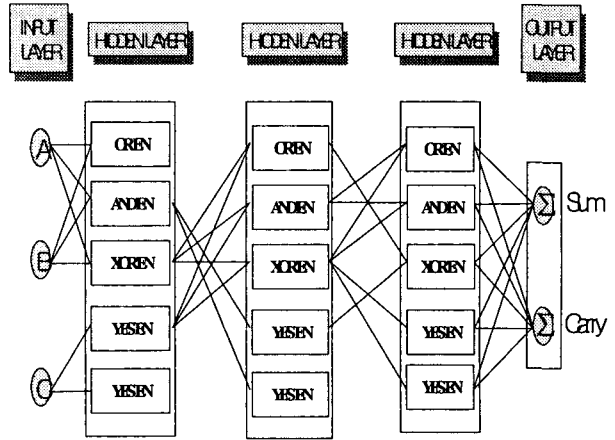


그림6. EN의 모듈형 계층구조를 이용한 전가산기 최종 연결강도

5. 결론

본 논문에서 EN의 모듈형 계층구조가 전가산기 연산회로를 성공적으로 구성함과 프로그래밍 없이 신경망의 학습을 통하여 전가산기 회로 설계가 가능함을 보였다. 모듈형 계층구조에서 사용되는 EN의 선택 및 배열을 최적화 한다면 보다 간단하면서 범용 연산이 가능한 신경망 설계를 할 수 있다.

앞으로 과제는 적절한 EN과 EN의 배열을 결정하여 보다 범용 연산이 가능한 최적화된 모듈 계층 구조 설계가 필요하다.

참고문헌

- [1] Ashok K. Goel, "VLSI Layout and Design"
- [2] Simon Haykin, "Neural Networks, A Comprehensive Foundation"
- [3] Philip D. Wasserman, "Neural Computing, Theory and Practice"
- [4] M. Morris Mano, "Digital Design"
- [5] Richard O. Duda, Peter E. Hart, David G. Stork "Pattern Classification"