신경망을 이용한 Banyan 네트워크 컨트롤러의 하드웨어 구현

論文
43~5~20

Implementation of Banyan Network Controller by Using Neural Networks

延 仁 喆*·鄭 德 鎭**
(In-Chol Yeon·Duck-Jin Chung)

Abstract—By using Neural Networks, a 8×8 Banyan network controller is designed and implemented. In order to solve internal blocking and output blocking, Winner-Take-All method is used. The longer queue takes higher priority. First-in-first-out method is used among the non-blocking cells in the queue selected. The required time to select a cell is 2.7 μsec for 155Mbps. The implemented controller using Xilinx FPGA chip selects cells within 2.5μsec.

Key Words : ATM, Banyan Network, Neural Networks, Neuron, Synapse

1. 서 론

ATM[1,9](Asynchronous transfer mode) 교환은 음성서비스의 대역폭(64Kbit/s)보다 더 저속에서 수백 Gbit/s까지 대역폭이 다양하게 할당되게 될 미래의 서비스의 질적 변화에 대처해서 사용되며 고속성 및 대규모화 실현 방안이 개발의 주관심사가 되고 있다. ATM 교환이 기존의 회선교환과 두드러진 차이점은 중앙에서 통제없이 스위칭 네트워크 자체에서 자기 스스로 라우팅할 수 있다는 점과 다양한 대역폭의 여러개의 서비스를 하나의 네트워크에서 통합해서 사용할 수 있다는 점을 들 수 있다. 이러한 기능을 하는 ATM 스위치로서 Banyan 네트워크가 널리 사용되는데, 구조상 다른 스위치에서는 볼 수 없는 내부 블러킹이 생기는 단점이 있을 뿐만 아니라, 모든 스위치가 공통적으로 가지고 있는 출력단 블러킹도 해결해야 할 큰 과제이다. 기존에 제안된 것 중에는 Batch의 'Sorting 네트워크'를 이용하여 Banyan 스위치[2]를 내부적으로 넌블러킹 상태로 만들어서 ATM 스위치를 구성한 사례가 있는데, 하드웨어가 많이 소요된다. 뿐만 아니라, 출력단 블러킹 문제는 별도의 하드웨어를 추가해야 한다. 신경회로망[3]을 이용한 방법으로는 Banyan 네트워크가 셀들을 스위칭하고 있는 동안에 Banyan 네트워크 앞단에서 다음 타임 슬롯에 보낼 수 있는 셀들의 입출력 주소를 Buffer에서 찾는 알고리즘이 제시되고 있다.

본 논문에서는 이 알고리즘을 바탕으로 새로운 학습 기법과 셀의 우선순위를 주는 알고리즘을 첨가해서 8×8 Banyan 네트워크 컨트롤러를 하드웨어로 구현할 수 있는 회로를 구성해서 FPGA 칩으로 구성했다.

2. 기존의 Banyan 네트워크 컨트롤러의 문제점

Banyan 네트워크를 사용하는 스위치들은 스위치를 버퍼의 위치에 따라 분류[10]하면 입력버퍼 스위치에 속한다. 그것은 한 포트에서 동기적으로 타임슬롯에 한 개 이하의 셀이 스위치를 통과할 수 있기 때문에 망상태나 다른 포트의 영향으로 셀이 스위칭될 수 없는 경우에는 입력버퍼에 저장되어야 한다. 그래서 최악의 경우, 한 입력포트에서는 버퍼의 수만큼의 셀들이 입

*正 會 員 : 仁荷大 大學院 電子材料工學科 碩士課程
**正 會 員 : 仁荷大 工大 電子材料工學科 副敎授·工博
接受日字 : 1993年 8月 25日
1 次修正 : 1994年 1月 8日

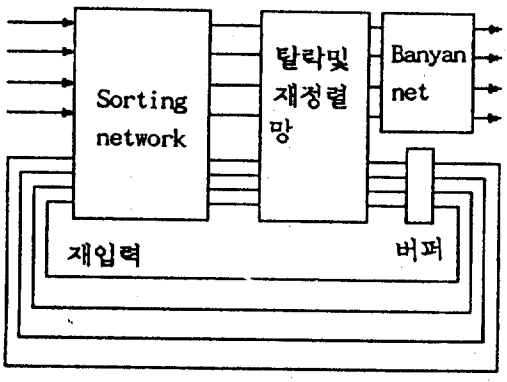

그림 1 Starlite 스위치망 구조
Fig. 1 A feature of Starlite switch

력경합을 벌이게 된다. 뿐만 아니라, 각기 다른 포트에서 한 목적지 주소를 요구할 경우 출력단 블러킹이 생기고, Banyan 네트워크 구조상의 결함 때문에 생기게 되는 내부 블러킹까지 있다.

이와 같은 블러킹 문제를 해결하기 위한 기존의 방법은 Banyan 네트워크를 기본으로 하고 내부 블러킹을 극복하기 위한 Sorting 망이 앞단에 사용되고 또한 출력단 블러킹을 해결하기 위해 탈락/재정렬망과 내부 버퍼 등으로 구성된다.

이러한 방법의 단점은 첫째, 내부 블러킹을 없애기 위해서 sorting 네트워크를 앞단에 붙여야 하고, 출력단 블러킹을 줄이기 위해서도 많은 컨트롤 조직이 필요하다는 점이다. 둘째는 전송된 패킷의 재입력으로 인해 전송의 순서가 바뀔 우려가 있다는 점이다. 이러한 단점들은 Starlite 스위치[2] 뿐만 아니라, 거의 대부분의 Banyan 네트워크를 기본으로 하는 네트워크가 공통적으로 갖고 있는 문제점이라 하겠다.

3. 뉴럴네트워크를 이용한 장점

입력단 주소(Input 주소)와 목적지 주소(Destination 주소)를 (I, D)로 나타내면, 8×8 스위치인 경우, 8개의 입력단 주소와 8개의 목적지 주소를 갖을 수 있기 때문에 모든 (I, D)의 가지수는 64개가 된다. 출력단 블러킹은 한 셀 (I, D)가 선택되면 다른 포트 7개의 셀 중 목적지 주소가 같은 셀들은 함께 출력단으로 연결될 수 없다는 것을 의미한다. 그리고 내부 블러킹은 Banyan 스위치 구조상으로 생기는 블러킹으로 한 셀 (I, D)에 대해서 다른 5개의 셀 (I, D)가 서로 출력 경합을 벌이게 되는 것을 의미한다. 뿐만 아니라, 한 포트 내에는 다른 출력단 주소를 갖는 셀들, (I, D)는 입력경합을 일으킨다. 각 포트에 버퍼가 8개 있는 경우, 한 셀 (I, D)는 최대 19개의 다른 셀들과 서로 경합을 벌이는 것이다.

신경회로망을 이용하면,[4,5,7,8] 위에서 설명한 (I, D)를 뉴런(Neuron)에 대응시킬 수 있다. 즉, 64개의 뉴런 중에서 한 뉴런은 19개의 다른 뉴런들과 시냅스(Synapse)로 연결되어 있다는 뜻이 된다. 먼저 들어와서 선택된 뉴런은 승자(Winner)가 되게 하고, 그 뉴런과 연결된 다른 19개의 뉴런들에게는 억제 신호를 주게 된다. 이러한 방식으로 한 포트에서 하나씩의 적합한 셀을 선택해서 Banyan 네트워크가 블러킹이 없이 셀들을 스위칭할 수 있도록 컨트롤 할 수 있는 것이다.

Brown의 모의실험 결과[3]에 의하면, 평균 큐 사이즈는 Sorting 네트워크를 포함한 스위치들보다 조금 떨어지지만, 최대 큐 사이즈에 관계되는 버퍼의 수는 N×N 스위치일 때 N이 클수록 그리고 셀 손실률이 더 낮을수록 뉴럴네트워크를 사용한 경우가 더 효능이 좋고 N이 낮은 경우에도 크게 효능의 차이가 없는 것으로 나타났다. 따라서 뉴럴 네트워크를 이용한 방법은 적은 하드웨어를 이용해서 큰 하드웨어의 역할을 대신할 수 있는 장점을 가지고 있는 것이다.

4. 제안된 학습방법

하나의 뉴론은 19개의 다른 뉴런과 스냅스로 연결되어 있는 상태이므로 최악의 경우, 뉴론들은 19개의 상태를 나타낼 수 있어야 그 중에서 가장 최적한 상태의 뉴론을 "Winner"로 선택할 수 있어야만 한다. 즉, 버퍼의 상태와 각 포트의 우선 순위 등을 모두 고려해서 웨이트의 차이를 20개를 만들어서 그 중에서 가장 웨이트가 높은 것을 택하는 방식을 취하면 된다.

그러나, 위의 방법은 많은 하드웨어가 필요하므로 일시에 병렬로 뉴럴네트워크를 동작시키는 것보다는, 포트를 선택할 때는 순차적으로 하고, 한 포트내에서 여러개의 셀들이 입력경합을 일으킬 때는 일시에 병렬로 동작하도록 해서 하드웨어의 양을 줄였다.

가장 많은 셀을 가지고 있고 가장 오래된 셀

을 가지고 있는 포트가 선택되고, 그 포트의 버퍼에 있는 셀들을 일시에 병렬로 Winner Take-All회로로 보내서 최적의 셀이 선택되고, 그 다음으로 우선 순위가 높은 포트가 선택되어, 그 포트내의 셀들 중 내부 혹은 출력단 블러킹이 일어나지 않는 셀이 선택되게 된다. 이러한 동작이 8번 일어나게 되면 다음 타임슬롯에 스위칭될 셀들을 모두 고를 수 있는 것이다.

그림 2의 회로에서 첫번째 F/F 레벨에서는 선택된 포트의 셀들 중에서 어떠한 셀이 먼저 들어와 있는지, 그리고 먼저 선택된 셀들과 블러킹은 없는지를 판단할 수 있다. 두번째 F/F 레벨에서는 디코더의 도움으로 선택한 셀의 (I, D)를 판단할 수 있게 한다. 그림 3에서 보는 바와 같이 선택된 셀들은 출력단 레지스터 선택됨과 동시에 메모리에 저장되어서 다른 포트의 셀들이 입력경합을 일으킬 때, 학습 작용을 돕게 된다.

예를 들어 3번 포트의 버퍼에 1번, 4번, 5번, 7번 셀이 순서대로 들어와 있다고 하자. 3번 포트가 선택되기 전에 (2, 1)이 이미 선택되었다면, 첫번째 F/F 레벨은 메모리로부터 블러킹 신호를 받기 때문에 '1'번 대신 '2'번 F/F을 ON 상태로 하고 그 나머지 F/F들은 OFF 상태로 한다. 이와 같은 첫번째 F/F 레벨의 동작은 버퍼 속에 있는 셀들 중에서 어떠한 셀이 가장 오래 기다렸는지, 다른 포트의 셀들과 블러킹은 없는지 여부를 학습하는 단계이다. 또한 두번째 F/F 레벨은 첫번째 F/F 레벨에서 학습된 내용을 실제의 출력단 주소로 바꾸어주는 디코딩의 역할을 한다.

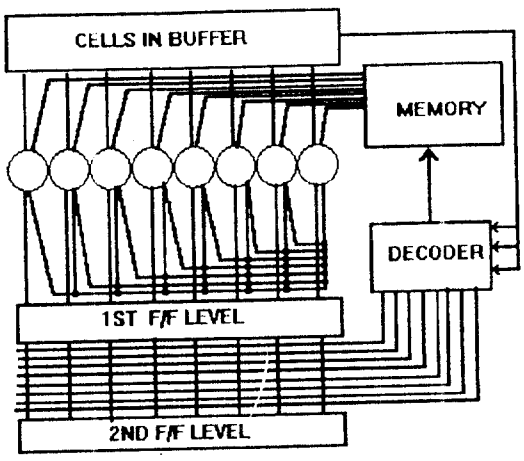

그림 2 Winner-Take-all 회로
Fig. 2 Winner-Take-all circuit

그에 따라서 '4'번 F/F를 ON으로 한다.

만약 메모리를 쓰지 않고 8×8 Banyan 네트워크를 컨트롤하기 위해서는 위에서 설명한 64개의 뉴론을 갖는 뉴럴 네트워크가 필요할 것이다. 따라서 그림 2에서처럼 메모리를 사용하면 마치 64개의 뉴론을 갖는 뉴럴네트워크를 사용하는 것과 똑같은 효과가 얻어지는 것이다.

5. 회로구성 및 실험

그림 3에서 보는 바와 같이 Banyan 네트워크를 사용하지 않고 PC를 연결해서 Data를 얻었다. ATM 셀은 53byte이지만, 그 셀의 (I, D) 4bit만을 버퍼에 입력해서 각 타임슬롯에 스위칭될 수 있는 셀로 선택되지 못하면 저장되게 했고, 버퍼의 수를 8개로 했으며, 버퍼의 움직임은 우선순위를 주는 역할을 하기 때문에 본 회로에 첨가할 사항이다. 그러나, 이 부분은 Banyan 네트워크에서 그대로 받을 수 있는 값이므로 제안된 회로의 하드웨어량에 포함되지 않음을 밝혀 둔다.

155Mbps급 Banyan 스위치는 2.7μsec마다 9개의 포트로 목적지 주소를 가지고 도달되는 셀들을 연결해 주어야 한다. 그림 3에서 보는 바와 같이 컨트롤러는 이러한 Banyan 네트워크가 다음 셀을 스위칭하기 전까지 즉, 2.7μsec내에 다음에 연결해 줄 셀들을 골라 주어야 한다. Banyan 스위치를 직접 연결해서 실험하는 대신에 PC를 통해서 셀의 (I, D) 4bit만을 고려해서 실험했다. PC는 각 포트의 Queue에 다른 포트의 (I, D)와 블러킹 때문에 연결되지 못한 셀들과 현재 도착된 셀을 저장하게 된다.

각 Queue의 상태가 'LATCH'에 전달되어 있

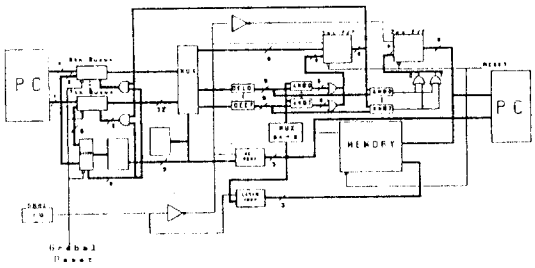

그림 3 8×8 Banyan 네트워크 컨트롤러의 블럭 다이어그램
Fig. 3 Block diagram of a 8×8 Banyan network

고, 그 값들은 'COMPARATOR'에서 서로 비교된다. Queue에서 LATCH에 전달되는 값들은 셀의 수와 셀의 분포(어떤 셀이 가장 오랫동안 스위칭되지 못하고 남아 있는가)인데 이 값들 중에서 'COMPARATOR'는 가장 큰 값을 펄스의 상승시 출력한다.

MUX에서 출력값은 1st F/F와 DECO..7로 분리되는데, 1st F/F에는 Queue에 저장된 셀의 존재여부를 나타내는 비트가 출력되고, DECO..7로는 셀의 목적지 주소가 출력된다. 1st F/F와 2nd F/F의 자세한 회로도는 그림 2에 나와 있다.

펄스의 하강시 1st F/F가 값을 받고 같은 시간에 SELPORT는 MEMORY에 저장된 값은 MUX2를 통해서 1st F/F의 B/D에 출력한다. 다음 rising edge에서는 winner가 된 값을 실제의 값으로 바꾸어서 PC에 스윗칭될 값으로 출력하고 MEMORY에 저장한다.

펄스 상승시 COMPARATOR의 출력을 DECOR-DER로 받아서 LATCH에 출력해서 Clear시킴으로써 COMPARATOR가 그 다음으로 큰 값을 출력하도록 한다. Falling edge에서 1st F/F가 출력한 값을 DECORDER로 받아서 Queue에서 값을 Clear시킨다.

8×8 Banyan 네트워크는 8개의 포트의 입력버퍼에 한 타임슬롯에 하나 이하의 셀이 도착을 하게 되는데, 이와 마찬가지로 PC에서 버퍼로 입력한다. 각 포트의 입력버퍼에 저장되어 있는 셀들의 숫자가 가장 많은 포트가 먼저 선택되어서 그 포트의 (I, D)들이 버퍼에서의 순서가 유지되면서 선택된 포트의 셀들의 (I, D)를 나타내는 데이터가 버퍼에서 뉴럴네트워크로 입력된다. 반대로 이미 뉴럴네트가 선택한 값들은 버퍼에서 지울 수 있도록 하는 데이터가 뉴럴네트워크에서 버퍼로 입력된다. 메모리는 뉴럴네트워크에 블러킹신호를 주고 뉴럴네트워크로부터 새롭게 선택된 (I, D) 값을 받는다. 출력단 래치는 버퍼에서 포트 번호를 받고, 뉴럴네트워크로부터 출력단 주소를 받아서 저장한다. 본 실험에서 이 값들을 PC로 받았지만, 이 값들은 Banyan 네트워크의 각 버퍼에 전달되어서 동작하게 된다.

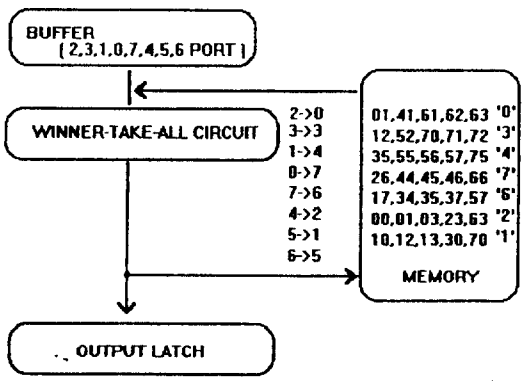

그림 4 플로우 차트
Fig. 4 Flow chart

다음은 모의실험한 예를 보인 것이다. 버퍼에 셀들이 다음과 같이 있다고 가정하자.

High priority ←

포트	Buffer에 있는 셀의 순서	우선순위
0	3763	4
1	24157	3
2	0465132	1
3	367152	2
4	724	6
5	132	7
6	45	8
7	2657	5

(우선 순위가 같은 경우는 낮은 번호의 입력 포트가 Winner가 된다.)

따라서 위의 Weight를 가지고 3번→1번→0번→7번→4번→5번→6번 포트의 순서대로 우선 순위가 결정된다. 우선순위가 가장 큰 2번 포트가 선택되어서 가장 먼저 뉴럴네트워크에 보내진다. 2번 포트의 셀들 중에서는 가장 먼저 들어와 있는 (2 0)이 가장 큰 우선 순위를 갖게 되고, 먼저 선택된 (I, D)가 없기 때문에 자동적으로 선택되게 된다. 그림 4에서 보는 바와 같이 (2 0)이 선택되면 내부 블러킹이 일어나는 (0 1) (4 1) (6 1) (6 2) (6 3)들과 출력단 주소 '0'이 Memory에 저장되게 된다. 3번 포트에서 (3 3)은 블러킹이 일어나지 않으므로 선택된다. (3 3)이 선택되면 내부블러킹이 일어나는 (1 2) (5 2) (7 0) (7 1) (7 2)들과 출력단 주소 '3'이 Memory에 저장된다.

다음으로 선택되는 1번 포트에서는 (1 2)는 (3 3)과 내부블러킹이 일어나기 때문에 (1 4)가 선택된다. (1 4)와 내부블러킹이 일어나는 (3 5) (5 5) (5 6) (5 7) (7 5)와 출력단 주소 '4'가

Memory에 저장된다.

위의 플로우 차트에서 보듯이 0번 포트에서는 (0 7)이 선택되고, 7번 포트에서는 (7 6)이 선택되고, 5번 포트에서는 (5 1)이, 마지막으로 6번 포트에서는 (6 5)가 선택되게 된다.

그림 5는 Viewlogic sumulator를 이용해서 모의실험한 결과이다. 여기서 port 2, port 1, port 0는 포트 번호를 2진수로 나타낸 값이고, TT0~TT7은 목적지 주소이다. Rising edge에서 보면, (2 0), (3 3), (1 4), (0 7), (7 6), (4 4), (5 1), (6 5)이 다음 Time Slot에 Banyan 네트워크를 통해서 스위칭될 셀들의 (I D)들이다. 하드웨어는 Xilinx FPGA를 이용하여 구현하였다. 버퍼부분을 제외한 나머지의 하드웨어량이 2,978TR 정도(Viewlogic에서 modul 단위)가 소요되었는데, 이것은 Sorting 네트워크를 구성하는데 소요되는 하드웨어량[6] 2,712TR(element당 TR 수가 113개가 소요되는데, 8×8 network에서는 24개 element가 필요하므로)과 비교될 만하다. 기존의 스위치에서는 이 Sorting 네트워크 외에도 더 많은 하드웨어를 써서 출력단 블러킹을 없애야 하기 때문에, 제안된 회로의 하드웨어량면에서 우수함을 알 수 있다. 버퍼의 움직임을 고려해서 그 순서를 우선 순위로 나타냈기 때문에 가능한 한 FIFO(First-In-First-Out)를 실현할 수 있었다.

Clock을 10Mhz를 쓸 경우, 제안된 네트워크는 Rising edge에서 학습이 수행되어서 Falling edge에서 리콜을 수행한다. 그러므로 학습과 리콜 시간이 각각 50ns임을 알 수 있다. 이러한 학습과 리콜이 각각 8번이 필요하고 포트의 우선순위를 줄 때 한 clock이 지연되고, 출력단 래치부분에서 한 clock이 지연되어서 전체 1μsec의 시간이면, 8개의 넌블러킹한 셀의 (I D)를 찾을 수 있다. 이러한 속도를 갖는다면, 155Mbps보다 더 빠른 스위치도 컨트롤이 가능하다. 그러나 bread board를 이용하여 FPGA chip을 연결함으로 인하여 실지 측정결과는 2.5μsec로 나타났다. 만약 one-chip으로 디자인하면, 위에서 설명한 바와 같이 1μsec 이내로 가능할 것으로 기대된다. 그림 6에는 제안된 회로를 하드웨어로 구현했을 때, 걸린 시간이 나타나 있다. 1번의 신호는 starting 신호이고, 2번의 신호는 가장 마지막으로 래치가 되는 (6 5)의 high bit이다. 그림 7에는 구현된 하드웨어 보드를 보여준다.

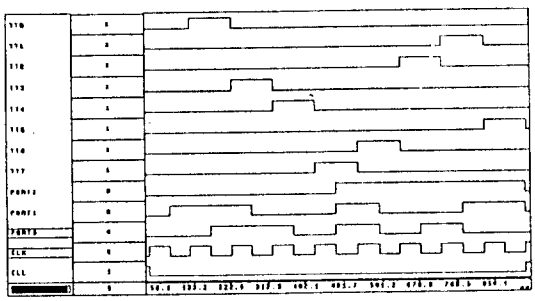

그림 5 모의실험 결과
Fig. 5 Simulation result

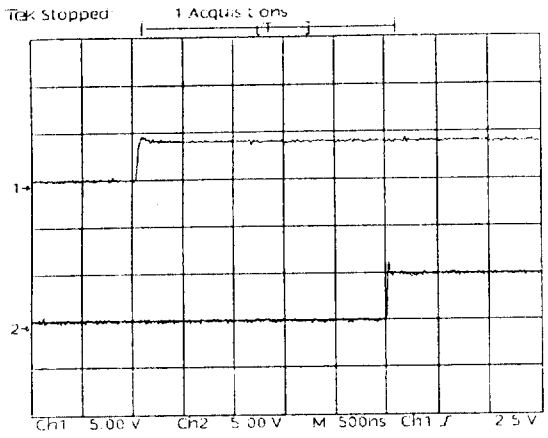

그림 6 칩의 동작 속도
Fig. 6 A speed of designed chip

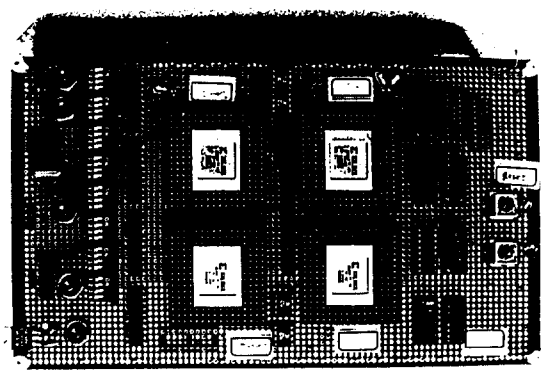

그림 7 하드웨어 보드
Fig. 7 Hardware Board

6. 결 론

ATM 스위칭 네트워크에서 가장 널리 사용되

어지는 Banyan 네트워크는 출력단 블러킹 뿐만 아니라 내부 블러킹이 발생되어서 스위치의 성능을 저하시키는 단점을 가지고 있다. 이러한 블러킹 문제를 해결하려는 기존의 방법은 Batcher의 sorting망을 Banyan 네트워크 앞단에 두어 내부 블러킹 문제를 해결했고, 출력단 블러킹 문제를 해결하기 위해서도 스위치의 종류에 따라 각기 다르지만, 많은 하드웨어가 필요했던 것이 사실이다.

뉴럴 네트워크의 병렬처리 방법을 이용해서 이와 같은 출력단 블러킹과 내부 블러킹을 한꺼번에 처리하고자 하는 노력이 전개되고 있는데, 고속의 학습과 최적 그리고 정확한 해를 보장하는 제한적인 조건이 있기 때문에 어려움이 있는데, 본 논문에서는 local minima에 빠지지 않으면서 고속으로 학습하고 리콜할 수 있는 8×8 Banyan 네트워크 컨트롤러를 제안했으며, Viewlogic simulater를 이용하여 모의실험을 했다.

4Mhz 수정발진기로 칩이 동작하므로 8×8 Banyan 네트워크를 동작시키는 데에는 2.5μsec가 걸리는 셈이다. 이 값은 한 타임슬롯에 2.7μsec가 걸리는 155Mbps급 8×8 Banyan 네트워크를 동작시키기에는 충분한 값이다. 그렇지만, 만약 one-chip으로 회로를 구성하고 보다 빠른 chip을 사용한다면, 155Mbps보다 더 빠른 Banyan 네트워크도 컨트롤이 가능할 것으로 기대된다.

이 연구는 1992년도 인하대학교 연구비 지원에 의하여 수행되었음.

참 고 문 헌

[1] 방윤학, "ATM 교환기술 개요", 전자공학회지, 제19권 8호, 1992. 8.

[2] Joseph Y. Hui and Edward Arthurs, "A Broadband Packet Switch for Integrated Transfort", IEEE J. on Sel. Areas in Com. vol. SAC-5, No. 8, 1987. 10.

[3] Timothy X. Brown, "Neural Network Design of a Banyan Network Controller", IEEE J. on Sel. Areas in Communications. vol. 8, No. 8, 1990. 11.

[4] Timothy X. Brown, "Neural network for switching", IEEE Communication Mag., vol. 27, pp. 72~81, Nov. 1989.

[5] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," Biol. Cybern., vol. 52, pp. 141 152, 1985.

[6] 광대역종합정보통신망핵심기술연구, 한국전자통신연구소, 1991. 12.

[7] Partick K. Simpson, "A Survey of Artificial Neural Systems", International Workshop on VLSI for Artifitial intelligence, University of Oxford, pp. 1~21, 1987.

[8] Richard P. Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4~22, 1987. 4.

[9] Harold S. Stone, "Parallel Processing with the Perfect Shuffle", IEEE Tran. on Computers, vol. C-20, No. 2, pp. 153~161, 1971. 2.

[10] Michael G. Mluchyj and Mark J. Karol, "Queueing in High-Performance Packet Switching", IEEE J. on Sel. Areas in Com. vol. 6, no. 9, 1988. 12.

저 자 소 개

연인철(延仁喆)

1967년 8월 17일생. 1991년 인하대 공대 응용물리학과 졸업. 1993년 8월 동 대학원 전자재료공학과 졸업(석사). 현재 제일중공업 전자기술부 근무

정덕진(鄭德鎭)

1948년 2월 8일생. 1970년 서울대학교 공대 전기공학과 졸업. 1984년 미국 유타주립대학교 대학원 전기공학과 졸업(석사). 1988년 미국 유타대학교 대학원 전기공학과 졸업(공박). 현재 인하대학교 전자재료공학과 부교수.