디지털 홉필드 신경망 스케쥴러를 이용한 ATM 스위치 설계에 관한 연구

정 석 진*, 이 영 주, 변 재 영, 김 영 철
전남대학교 전자공학과
광주광역시 북구 용봉동 300, 500-757
e-mail : sjjung@neuron.chonnam.ac.kr *

Study on the Design of a ATM Switch Using a Digital Hopfield Neural Network Scheduler

Suk-jin Jeong *, Youn-ju Yee, Jae-young Pyun, Young-chul Kim
Dept. of Eletronic Eng., Chonnam National University
e-mail : sjjung@neuron.chonnam.ac.kr *

Abstract

A input buffer typed ATM switch and an appropriate cell-scheduling algorithm are necessary for avoiding output blocking and internal blocking respectively. The algorithm determining a set of non-blocking data cells from the queues can greatly affect on the switch's throughput as well as the behavior of the queues.

In this paper bit pattern optimization combined with the Token method in presented in order to improve the performance of ATM switch. The digital Hopfield neural cell scheduler is designed and used for the maximum numbers of cells in real-time

1. 서 론

ATM 교환 시스템의 성능은 처리율, 지연 및 지터, 셀 손실율, 셀 삽입 오류율, 연결 블럭킹 확률, 비트오류 확률로 평가되고 일반적으로 처리율은 입력 링크에 대하여 한 셀 주기에 처리되는 평균 셀 수로 정의된다. 이때 최대 처리율은 최대 부하 상황에서의 처리율로 교환기의 성능을 평가하는 가장 중요한 요소이다.

ATM 교환 시스템은 크게 입출력 처리기와 교환망 사이의 다중화, 역다중화 장치 그리고 제어부분으로 구성된다. 교환시스템은 버퍼링에 따라 입력 버퍼형, 출력 버퍼형, 입력 스무딩, 공유 버퍼형이 있는데 입력 버퍼형인 경우 출력 블리킹에 의한 출력처리율을 높일 필요가 없고 확장성의 장점이 있으나 동일한 출력 포트를 가지고 있는 셀이 한 셀 주기에서 처리되지 못하고 버퍼에 남게되는 경우 지연이 발생하게 되는데 이러한 HOL(Header of Line)블리킹으로 인하여 최대 Throughput이 약 0.58정도 밖에 되지 않는다. 이를 개선시키는 방법으로 windowing, destination-queueing등이 있는데 이 방법들은 입력의 셀처리를 효율적으로 사용하기 위해서 입력버퍼의 셀 개수를 많이 선택하거나 셀 Tag값에 따라 각각 다른 입력버퍼로 저장되는 방식으로 모두 셀 스케쥴링 알고리즘이 필요하다.

본 논문은 스위치의 최대 Throughput을 높여 성능을 크게 개선하기 위해서 입력 버퍼형이 가지는 장점을 최대한 수용하고 입력 버퍼의 크기를 줄일 수 있는 windowing 방식[1,3]을 채택하였으며 구현이 복잡한 셀 스케쥴러를 디지털 홉필드 신경망으로 설계함으로서 단순하면서도 실시간 처리가 가능하게 하였다. Token 방식[4]을 전처리 부분에서 채택하여 비트처리가 가능하고 기존의 windowing 신경망 셀처리 알고리즘보다 간단하게 최적화 기능을 수행하여 최대의 셀 선택이 이루어지도록 하였으며 스위치 구성을 Banyan 스위치를 사용하여 하드웨어 크기를 최소화 하였다.

2. Banyan 스위치 아키텍쳐

Banyan 스위치[1,3]는 ATM 교환시스템에 사용되는 교환망으로 self-routing이 되는 다단계 상호 연결망으로 되어 있다.

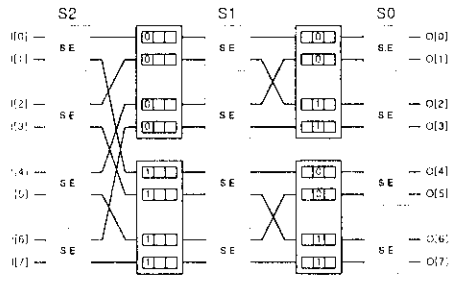

그림 1. 8 × 8 Banyan스위치

그림1 은 8×8 Banyan 스위치의 블럭 다이어그램을 보여주며 각 스위치 element(S.E)의 동작은 그림 2와 같다.

(a) (b)

그림 2. 스위치 element의 동작

스위치 element들은 2개의 입력 포트의 목적지 주소에 해당하는 tag 값의 해당 비트들을 비교해서 같으면 블러킹이 발생하게 되나 다를 경우 입력 비트가 0이면 상위 출력 포트로 연결이 되고 입력 비트가 1이면 하위 출력포트로 연결이 된다. 이와같이 S.E들을 그림 1과 같이 상호연결망으로 연결하면 외부의 Routing Control이 없어도 스스로 원하는 주소를 찾아간다. 이러한 스위치의 주요한 특징은 S2 단계에서 internal 블러킹과 output 블러킹이 없는 한 S1 까지 안전하게 데이터가 전송하게 된다. 그 이유는 그림 1 에서 (S2, S1) (S1, S0) 각각 사이의 블럭을 보면 tag 값에 따라서 최상위 비트가 0(0--)과 1(1--)로 나누어지고 다음 비트에 의해서 0(00-), 1(01-), 1(10-), 1(11-) 으로 마지막으로 최하위 비트에 의해서 처리되어 출력된다. 곧 8개의 입력이 모두 블러킹만 없다면 동시에 8개 모두 전송시킬수 있게된다.

3. 제안한 입력버퍼 스위치 전체구조

그림3은 internal 블러킹과 output 블러킹을 해결하기 위하여 이 논문이 제안한 스위치의 전체 모델도이다.

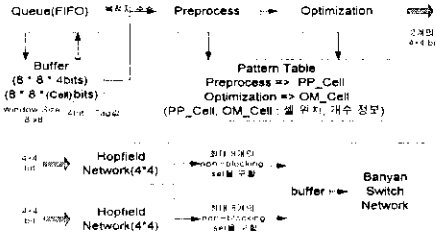

그림 3. 신경망 셀 스케쥴러를 사용한 입력버퍼형 Banyan스위치

그림 3에서 입력버퍼에 해당하는 fifo 방식의 queue와 window방식의 셀 처리를 위해 필요한 크기만큼 다른 임시버퍼에 셀 데이타들을 저장한다. 그림 4와 5는 전처리와 최적화의 동작을 나타내는 예제이다.

0	6	0	3	7	6	6	0
0	2	6	0	5	4	3	2
3	7	0	3	4	0	6	6
2	1	1	5	6	7	4	6
3	7	7	1	6	4	7	0
7	5	6	0	5	4	3	7
6	1	4	6	5	0	2	0
7	7	4	3	0	5	7	7

임시버퍼(buffer)

	0	1	2	3	4	5	6	7
0	1	0	0	1	0	0	1	1
1	1	0	1	1	1	1	1	0
2	1	0	0	1	1	0	1	1
3	0	1	1	0	1	1	1	1
4	1	1	0	1	1	0	1	1
5	1	0	0	1	1	1	1	1
6	1	1	1	0	1	1	1	0
7	1	0	0	1	1	1	0	1

전처리 2진 패턴

그림 4. 전처리 예제

그림 4 에서는 입력 포트와 출력 포트를 사상(mapping)시키는 전처리 과정을 나타낸다. 여기서 전처리 2진 패턴의 세로축은 0~7 까지의 입력포트를 그리고 가로축의 0~7은 출력포트를 가르킨다.

	0	1	2	3	4	5	6	7
0	1	0	0	1	0	0	1	1
1	1	0	1	1	1	1	1	0
2	1	0	0	1	1	0	1	1
3	0	1	1	0	1	1	1	1
4	1	1	0	1	1	0	1	1
5	1	0	0	1	1	1	1	1
6	1	1	1	0	1	1	1	0
7	1	0	0	1	1	1	0	1

전처리 2진 패턴

	0	1	2	3	4	5	6	7
0	0	0	0	0	1	1	1	1
1	0	0	0	0	1	1	1	1
2	0	0	0	0	1	1	0	1
3	0	0	0	0	1	1	1	1
4	1	1	1	1	0	0	0	0
5	1	0	0	1	0	0	0	0
6	1	1	1	1	0	0	0	0
7	1	1	1	1	0	0	0	0

최적화 2진 패턴

그림 5. 최적화 예제

그림5는 전처리 최적화 처리가 되어 나온 결과를 나타낸다. 스위치 element는 2개의 입력을 가지므로 입력 포트 부분을 4부분으로 나누고 출력 포트를 최상위 비트가 0과 1로 2부분으로 나눈다. 0과 1의 입력 포트에서 각각의 0 출력 포트와 4 출력 포트는 비트 패턴이 1이므로 (0, 4)로 하나의 목적지 주소쌍을 만들수 있다. 그러면 (0, 4), (0, 5), (0, 6), (3, 4), (3, 5), (3, 6)과 같은 internal 블러킹이 없는 목적지 주소쌍을 만들수 있다. 이와같이 최적화를 하는 이유는 Banyan 스위치는 첫번째 단계에서 internal 블러킹을 피할수 있으면 output 블러킹이 없는 경우 셀을 목적지 주소까지 전송할수 있기 때문이다. 하지만 위와 같은 패턴 처리는 처리되는 셀 데이터의 위치나 개수를 전혀 알수 없으므로 따로 패턴 테이블에 window크기 만큼 전처리 할때마다 비트패턴으로 기록하고 최적화를 하는 경우에도 패턴 테이블의 전처리 비트 패턴도 같이 묶어져

야 한다. 그림 6은 패턴 테이블의 블럭다이어그램이다.

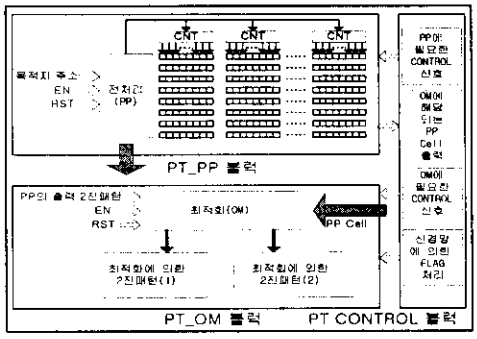

그림 6. 패턴 테이블의 전체 구성도

4. 홉필드 신경망 셀 스켜줄링

ATM 스위치에서의 셀 스케줄링을 어떻게 비선형 최적화 문제로 공식화 할수 있는지 알아본다. 셀 스케줄링[2]은 TSP 문제[5]를 해결하는 방법과 같은 원리로 동작하지만 차이점은 거리를 변수로 사용하여 최소거리를 구하는 것이 아니라 최적화 비트 패턴에서 출력 블러킹을 피할 수 있는 셀의 최대개수를 구하는 것이다. 여기에서 사용되는 신경망은 16개의 뉴런들로 구성되는 홉필드 네트워크이다.

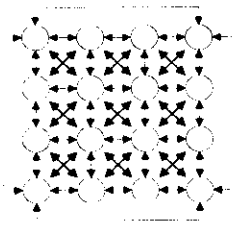

그림 7. 홉필드 신경망

그림 7에서 원은 뉴런을 나타내고 실제로 뉴런들간의 연결은 full connnetion을 이루며 최적해 비트 패턴 1비트가 뉴런 하나에 해당한다.

홉필드 신경망 셀 스케줄러의 최적해를 위한 목적함수는 $\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}N_{total}$이며, 최적화 패턴행렬이 (n × n)이 될때 셀이 행렬(i, j)에서 선택이 되면 X_{ij}가 1이 되고 그렇지 않으면 0이된다. 그리고 N_{total} 은 선택된 Cell들의 최대 개수이다.

이때 $\sum_{i=1,\,i\neq j}^{n} X_{ij}=1 \quad j=1, \ldots n$ (선형제한)

$\sum_{j=1,\,i\neq j}^{n} X_{ij}=1 \quad i=1, \ldots n$ (선형제한)

이 선형제한은 같은 열또는 행에서의 셀 선택은 1개 뿐이라는 것을 나타낸다.

이러한 0과 1의 선형문제를 신경망을 사용하여 비선형적인 목적함수에 반영될수 있는지 알아보자. 위의 2가지 선형제한과 선택될수 있는 셀 수는 n 개, 그리고 비트 패턴에서 0을 선택하면 안되므로 모두 4 가지로 Energy함수를 만들었다.

$$E=A_1\sum_i\sum_k\sum_{l\neq k}X_{ik}X_{il}+A_2\sum_l\sum_k\sum_{j\neq k}X_{kj}X_{jl}+A_3[(\sum_i\sum_k X_{ik})-n]^2+A_4\sum_i\sum_k\sum_l X_{ik}X_{kl}$$

같은 행(열)에 있는 어떠한 두 뉴런도 네트워크의 동일한 동작 사이클 동안에 점화되어서는 안된다. 결국 측면 연결은 여기가 아닌 억제성 결합을 의미한다. 이러한 측면 연결은 Kronecker delta function을 사용하였다. 그러면 신경망에 사용되는 가중치값은 다음과 같이 설정된다.

$$W_{ik,\,lj}=-A_1\delta_{il}(1-\delta_{kj})-A_2\delta_{kj}(1-\delta_{il})-A_3-A_4(MAP_{ij}Map_{kl})$$

$$\delta_{ik}=\begin{cases}1 & i=k\\ 0 & i\neq k\end{cases}, \quad \text{MAP : 최적화 비트 패턴}$$

그러나 이와같은 Weight로 부터 출력을 구하면 지역적 최소값에 빠질 문제가 있다. 그러므로 최대의 셀 선택이 되려면 아래의 식과 같이 활성화 값을 갱신시켜야 되며 2개의 항을 사용한다.

$$\Delta a=\Delta t(term1+term2)$$

$$term1=-A_1\sum_{k\neq j}X_{ik}, \quad term2=-A_2\sum_{k\neq i}X_{kj}$$

위의 결과식으로 부터 변경된 활성화값을 이용하여 i 번째 행과 j 번째 열에 있는 뉴런의 활성화 값은 다음 식과 같이 갱신 되어져야 한다.

$$A_{ij\neq w}=A_{ijold}+\Delta A_{ij}$$

출력값은 winner-take-all방식에 의해 최대의 활성화값을 가지는 뉴런이 선택된다.

5. 시뮬레이션 결과

제안한 스위치의 가장 중요한 부분인 전처리, 최적화, 패턴 테이블, 그리고 신경망 셀 스케줄링의 파형을 그림 7, 8에 그리고 합성결과를 그림 9와 10에 나타냈다. (사용툴 : PeakVHDL, Synopsys v3.4b)

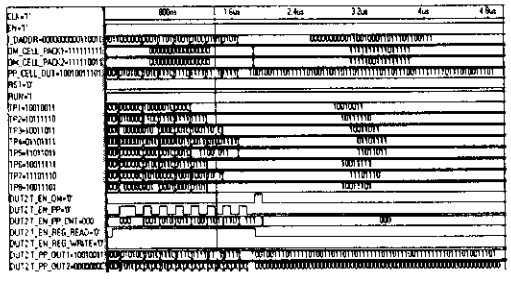

그림 7. 전처리(PP), 최적화(OM), 패턴 테이블(PT)

여기서 입력Tag값은 그림 4의 임시 버퍼값을 그대로 I_DADDR로 입력하고 전처리 2진 패턴의 출력이 PP_CELL_OUT에서 최적화 블럭의 출력은 각각 OM_CELL_PACK1과 OM_CELL_PACK2에서 나오게 된다.

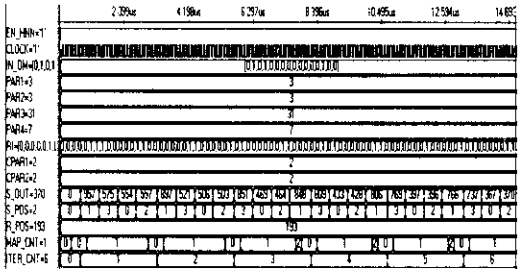

그림 8. 신경망 셀 스케줄링

신경망의 입력은 최대의 셀 선택이 이루어 지는 과정을 보여주기 위해 테스트 벡터를 {0101 0000 0000 0100}으로 주었고 그때의 신경망 파라미터 값들로 초기 가중치에 PAR1 ~ PAR4로 활성화값 변경에 CPAR1, CPAR2에 각각 적당한 값을 입력하였다. 출력에 크게 영향을 주지 않는 random한 입력값이 RI로 들어가고 그때의 활성화값이 S_OUT으로 그때의 셀 선택 위치를 S_POS, 그리고 실제로 선택이 되어 지는 셀의 최대 개수를 MAP_CNT에 Iteration을 ITER_CNT에 출력이 되게 하였다. 위의 파형에서 처음 Iteration 1에서는 S_POS가 (1,3,0,2)가 되는데 테스트 벡터에서 보면 0101 에서 1이 0000 에서 0이 0000 에서 0이 0100 에서 0이 선택이 되어 MAP_CNT가 1개 뿐이므로 1이 된다. Iteration 3에서는 (3,0,2,1)이 되어 MAP_CNT가 2개 되어 원하는 최대의 셀 선택을 할 수 있다.
이 최대의 셀 선택이 이루어지는 패턴 테이블의 셀 위치에 해당하는 임시버퍼의 셀 데이타를 Banayn스위치의 입력 포트에 보낸다.

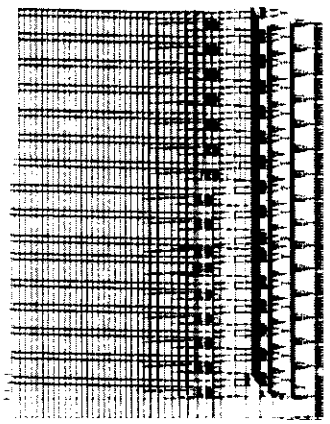

그림 9. 신경망 셀 스케줄러 합성도

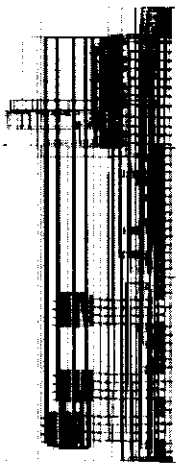

그림 10. 패턴 테이블 합성도

6. 결 론

본 논문에서는 Banyan스위치의 internal 블러킹과 output 블러킹 문제를 해결하기 위해 입력 버퍼와 고속 처리 셀 스케줄러를 설계하여 교환지연을 throughput 향상으로 개선하고자 하였다. 특히 목적지 출력 주소 값을 가지고 있는 tag 부분을 전처리와 최적화를 통해 비트패턴으로 처리하는 경우 하드웨어의 최소화와 확장성 그리고 스케줄러가 가지는 처리 부담을 줄일수 있고 입력버퍼 스위치가 가지는 최대의 성능을 얻을 수 있다. 하지만 홉필드 신경망 셀 스케줄러는 기능 자체는 간단하지만 많은 덧셈과 곱셈 연산을 하므로 지연시간을 해결하기 위해 고속 연산 처리기가 필요하다.

참고문헌

[1] Timothy X. Brown, Kuo-Hui Liu, "Neural Network Design of a Banyan Network Controller", IEEE. Communications. Vol.8 No.8 October 1990.

[2] Young-Keun Park, Vladimir Cherkassky, "Omega Network-Based ATM Switch with Neural Network-Controlled Bypass Queueing and multiplexing", IEEE. Communications. Vol 12. No 9. 1994.

[3] Shigeo Urushidani, "A High-Performance Self Routing Switch for B-ISDN", IEEE. Vol.9 No.8 October 1991.

[4] ATM Networks(권택근 저, 홍릉 출판사, pp.125 ~ pp.253).

[5] C++ Power Programming C++ Fuzzy and Neural Network(Valluru B. Rao. Hayagriva V. Rao 저, pp.454 ~ pp.493, 삼각형 출판사)