論文 95-6-6-19

이산여현변환을 위한 병렬 전치메모리

正會員 金 起 徹*

A Parallel Transposition Memory for Discrete Cosine Transform

Kichul Kim* Regular Members

要 約

본 논문에서는 입출력을 병렬로 수행하는 병렬 전치메모리를 소개한다. 본 논문에서 소개되는 병렬 전치메모리는 기존의 순차 전치메모리와 같이 기억 소자로 RAM을 사용하며 입출력 네트워크에 오메가 네트워크와 역오메가 네트워크를 사용한다. 기존의 전치 네트워크와는 달리 쉬프트 레지스터는 사용하지 않는다. 소개되는 병렬 전치메모리는 기존의 순차 전치메모리와 비슷한 비용을 사용하면서 1차원 이산여현변환기와의 사이에 직병렬 데이터 변환기를 사용하지 않게 함으로써 전체 2차원 이산여현변환기의 비용을 크게 절감시킨다.

ABSTRACT

This paper proposes a new parallel transposition memory based on RAM's and switching networks. It uses the same amount of RAM's and the same amount of control circuitry as existing transposition memories. It eliminates the need for serial-to-parallel converter and parallel-to-serial converter between transposition memories and 1-dimensional discrete cosine transform units. As a result, one can reduce the hardware cost considerably.

*서울시립대학교 반도체공학과
Department of Semiconductor Seoul City University
論文番號 : 95123-0328
接受日字 : 1995年 3月 28日

Ⅰ. 서 론

전치메모리(transposition memory)는 2차원 이산여현변환(discrete cosine transform)의 구현에 널리 쓰이고 있다.[1][2][6-8][10][13-16] 이산여현변환은 대표적인 분리가능 변환(separable transform)이다.[9] 2차원 분리가능 변환을 수행할 때 먼저 각 행에 대해서 1차원 변환을 수행하고 그 후에 각 열에 대해서 1차원 변환을 수행함으로써 2차원 변환을 직접 수행하는 것보다 계산 량을 크게 줄일 수 있다. 이러한 방법을 행열분리방법(row-column separation method)이라고 한다. 전치메모리는 2차원 이산여현변환을 행열분리방법으로 구현할 때 첫 번째 1차원 이산여현변환기의 결과를 행 우선 순서(row major order)로 받아서 두 번째 1차원 이산여현변환기에 열 우선 순서(column major order)로 공급하는 중요한 역할을 한다(구현 방법과 이산여현변환의 방향에 따라서 행 우선 순서와 열 우선 순서가 바뀌기도 함). 그림 1은 2차원 이산여현변환을 두 개의 1차원 이산여현변환기와 전치메모리를 사용하여 행열분리방법으로 구현하는 전형적인 예를 보여주고 있다.

현재까지 알려진 전치메모리의 대부분은 순차 전치메모리(serial transposition memory)이다.[2][7][8][13-16] 순차 전치메모리에서는 첫 번째 1차원 이산여현변환기의 결과가 한번에 하나씩 행 우선 순서로 전치메모리에 저장되며 또한 한번에 하나씩 열 우선 순서로 전치메모리에서 읽혀져 두 번째 1차원 이산여현변환기에 공급된다. 그림 2에 8x8 2차원 이산여현변환을 위한 순차 전치메모리의 한 예가 나타나 있다. 순차 전치메모리의 가장 큰 장점은 메모리의 한 주소에서 데이터를 읽고 같은 주소에 새로운 데이터를 저장하는 'read-then-write at the same address' 방식으로 동작할 수 있다는 것이다. 이는 순차적으로 입력되는 NxN 행렬의 요소들을 NxN 크기의 메모리에 행 우선 순서로 저장한 후에 열 우선 순서로 읽는 방법과 열 우선 순서로 저장한 후에 행 우선 순서로 읽는 방법을 교대로 사용할 때, 한 방법에서 행렬 요소가 저장되는 메모리주소와 다른 방법에서 행렬요소가 읽혀지는 메모리주소가 동일하기 때문이다. 이러한 성질은 이중 메모리 버퍼(double memory buffer)를 사용하지 않고 하나의 RAM과 간단한 주소발생기(address generator)를 사용하여 파이프라인 방식(pipelined mode)으로 동작하는 순차 전치메모리의 구현을 가능하게 한다. 하지만 순차 전치메모리의 특성인 순차 입출력은 2차원 이산여현변환기에 새로운 복잡성을 도입하게 된다. 이는 대부분의 1차원 이산여현변환기가 요구되는 성능을 효율적으로 만족시키기 위하여 병렬구조를 채택하고 있기 때문이다. 병렬 이산여현변환기에서는 한번에 여러 개의 데이터가 동시에 입출력되는 병렬 입출력이 사용된다. 이러한 순차 전치메모리와 병렬 이산여현변환기에서의 입출력의 차이는 병렬 데이터를 순차 데이터로(parallel-to-serial converter) 또는 순차 데이터를 병렬 데이터로(serial-to-parallel converter) 변환하는 변환기를 도입하게 한다. 그림 3에 병렬 1차원 이산여현변환기와 순차 전치메모리를 사용하는 2차원 이산여현변환기의 구조를 자세히 나타내었다. 그림 3에서 모듈 (C)와 (E)는 병렬 이산여현변환기와 순차 전치메모리 사이의 데이터 변환을 위한 직병렬 변환기이다. 내부적으로 16 비트(bit)의 정밀성을 갖는 16x16 이산여현변환기의 경우에 모듈 (C)와 (E)

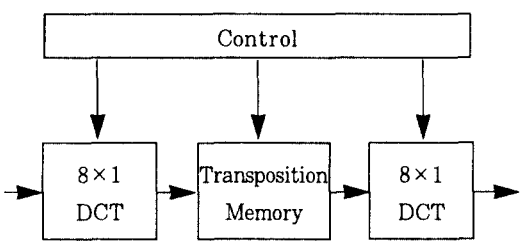

그림 1. 8×8 2차원 이산여현변환기
Fig. 1. 8×8 2-dimensional discrete cosine transform unit

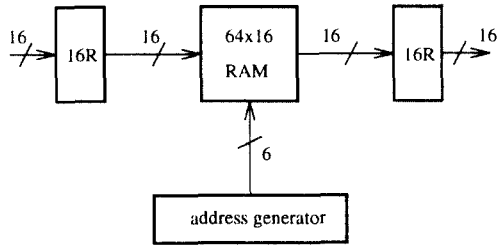

그림 2. 8×8 2차원 이산여현변환을 위한 순차 전치메모리
Fig. 2. Serial transposition memory for 8x8 discrete cosine transform unit

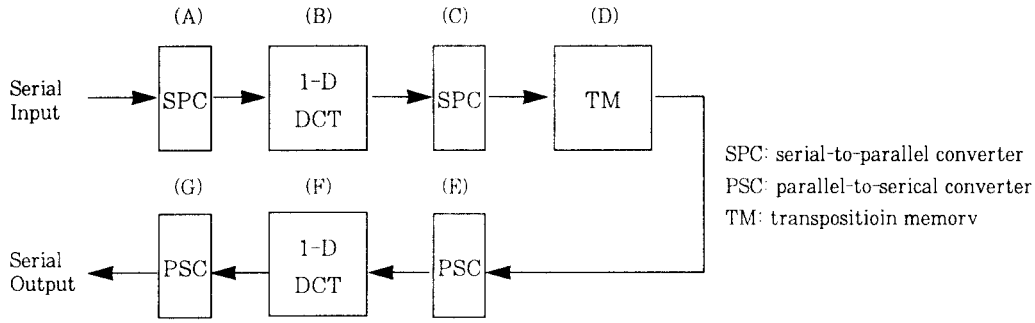

그림 3. 순차 전치메모리를 이용한 2차원 이산여현변환기
Fig. 3. 2-dimemsional discrete cosine transform unit with serical transposition memory

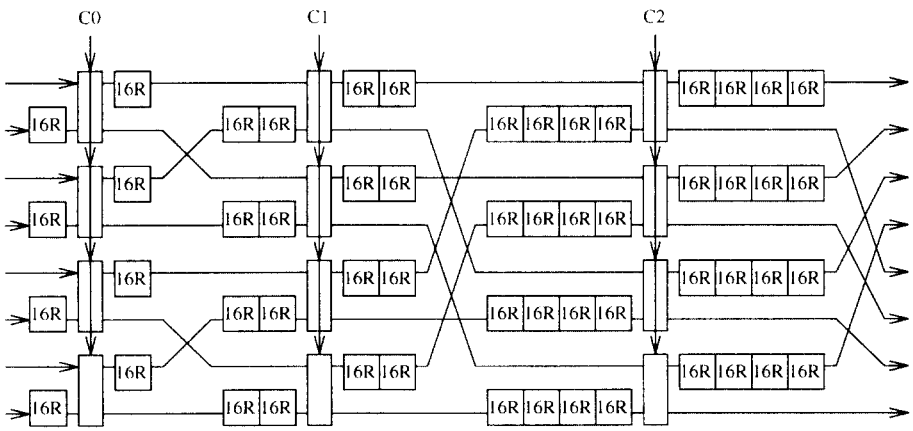

그림 4. 8×8 2차원 이산여현변환을 위한 전치 네트워크
Fig. 4. Transposition network for 8x8 discrete cosine transform

는 각각 512개의 레지스터(register)와 240개의 2x1 MUX로 구성된다. 분산산술방식(distributed arithmetic)을 이용한 1차원 이산여현변환기와 같은 경우에는 이산여현변환기가 모듈 (C)와 (G)의 일부분을 이미 포함하고 있다.[15][16] 이러한 경우에는 모듈 (C)를 약 절반 정도의 비용으로 구현할 수 있다. 하지만 모듈 (D)는 여전히 필요하게 된다.

직병렬 변환기의 사용으로 인한 복잡성의 증가를 피하기 위하여 Carlach 등은 새로운 전치기술(transposition technique)을 소개하였다[6]. 그들은 1x1, 2x2, 그리고 4x4 크기의 중간 행렬에 대하여 연속적으로 전치 행렬을 얻는 방법을 사용하여 완전한 8x8 전치 행렬을 구하였다. 이러한 연속적인 전치행렬을 구하기 위해서 그림 4에 보이는 바와 같이 쉬프트 레지스터(shift register)와 2x2 스위치를 이용한 transposition stage라 불리는 전치 네트워크(transposition network)를 구성하였다. 그림 4의 전치 네트워크에서 쉬프트 레지스터는 비트 단위로 쉬프트 연산을 수행하고 2x2 스위치는 워드(word) 단위로 전치 연산을 수행함으로써 순차 입출력이 없이 8x8 전치 행렬이 구해지고 있다. 하지만 전치 네트워크를 사용하는 경우는 직병렬 변환기의 사용을 줄이는 장점이 있는 반면에 기억 소자로 RAM 대신에 쉬프트 레지스터를 사용함으로써 전치 메모리의 비용이 현저히 증가하는 단점이 있다. 레지스

터 어레이(register array)를 전치메모리로 사용하는 방법도 소개되었으나 RAM을 사용하는 전치메모리에 비해 비용이 증가하는 단점을 극복하지 못하고 있다.[10]

본 논문에서는 쉬프트 레지스터를 사용하지 않고도 병렬 입출력을 가지는 병렬 전치메모리를 소개한다. 본 논문에서 소개되는 병렬 전치메모리는 기억 소자로 기존의 순차 전치메모리와 같이 RAM을 사용하며 인터코넥션 네트워크(interconnection network)에 2x2 스위치를 사용한다. 본 논문에서 소개되는 병렬 전치메모리는 기존의 순차 전치메모리와 비슷한 비용을 사용하면서 1차원 이산여현변환기와의 사이에 직병렬 데이터 변환기를 사용하지 않게 함으로써 전체 2차원 이산여현변환기의 비용을 크게 절감시킨다.

다음 장에서는 병렬 전치메모리의 구조와 대략적인 동작에 대한 설명을 한다. 3 장에서는 행렬 요소를 메모리 모듈에 저장하는 메모리 모듈 설정에 대한 설명을 한다. 4 장에서는 입출력 네트워크에서의 경로에 대하여 설명한다. 5 장에서는 주소 발생기와 네트워크 제어에 대해서 설명한다. 6 장에서는 본 논문의 결론을 보인다.

Ⅱ. 병렬 전치메모리의 구조

본 장에서는 병렬 전치메모리의 전체적인 구조와 대략적인 동작을 설명한다. 각 부분에 대한 동작과 이론은 다음 장 이후에 상세히 설명된다. 앞으로 본 논문에서는 가장 널리 쓰이는 크기인 8x8 2차원 이산여현변환기를 위한 병렬 전치메모리를 보기로 사용한다. 또한 보기에 대한 설명을 간단히 하기 위해서 첫 번째 이산여현변환기는 데이터를 행 우선 순서로 출력하고 두 번째 이산여현변환기는 데이터를 열 우선 순서로 입력한다고 가정한다.

그림 5에 8x8 행렬의 전치를 위한 병렬 전치메모리의 구조가 나타나 있다. 여기서 사용된 8x8 행렬의 정밀도는 16 비트이다. 병렬 전치메모리는 입력 네트워크, 출력 네트워크, 그리고 메모리 유닛(Memory Unit)으로 구성되어 있다. 입력 네트워크에는 오메가 네트워크(Omega Network)가 사용되며 출력 네트워크에는 역 오메가 네트워크(Inverse Omega Network)가 사용된다. 병렬 전치메모리가 정 방향 변환과 역 방향 변환을 모두 수행하는 이산여현변환기에 사용될 때는 변환의 방향에 따라 입출력 네트워크가 바뀔 수도 있다. 오메가 네트워크는 세 개의 단계(stage)로 이루어 졌으며, 각 단계는 셔플 연결(shuffle connection)과 2 비트 2x2 스위치로 이루어져 있다. 이를 shuffle/exchange stage 라고 한다. 오메가 네트워크에는 각각 8개의 입출력이 있으며, 각 입출력은 한번에 2 비트의 정보를 전송한다. 오메가 네트워크에는 전체 12개의 2 비트 스위치(24개의 1 비트 스위치)가 사용된다. 오메가 네트워크의 입력

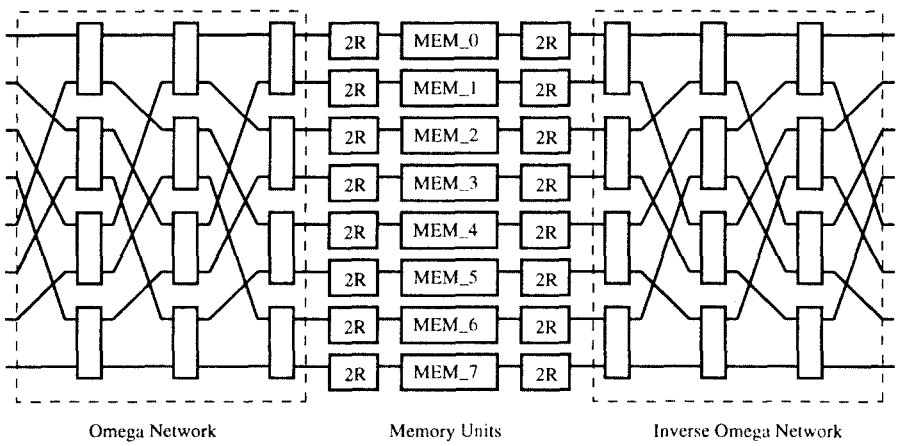

그림 5. 8×8 이산여현변환을 위한 병렬 전치메모리
Fig. 5. Parallel transposition memory for 8x8 discrete cosine transform

과 출력을 서로 바꾸면 역오메가 네트워크가 얻어진다. 메모리 유닛은 8개의 메모리 모듈로 구성되어 있다. 각 메모리 모듈은 64x2 RAM과 RAM의 입력과 출력을 위한 2 개의 2 비트 레지스터로 되어 있다.

병렬 전치메모리의 역할은 입력 네트워크에서 행 우선 순서로 병렬 입력되는 데이터를 출력 네트워크에 열 우선 순서로 병렬 출력하는 것이다. 이러한 기능을 수행하기 위해 병렬 전치메모리는 다음과 같이 동작한다. 입출력 네트워크는 워드단위(8 클락)로 병렬 전치메모리의 입출력과 메모리 모듈의 연결을 결정하는 스위치 상태를 바꾸며, 각 메모리 모듈은 매 클락마다 2 비트씩의 정보를 입출력한다. 입출력 네트워크는 병렬 전치메모리의 입출력이 메모리 모듈과 항상 일대일 대응(one-to-one mapping)이 되도록 연결 경로를 유지시킨다. 각 메모리 모듈은 'read-then-write at the same address' 방식으로 동작된다. 즉 한 메모리 모듈의 특정 주소(address)에 저장된 데이터가 출력 네트워크 측에서 읽혀지면 바로 그 주소에 입력 네트워크 측에서 공급되는 데이터를 저장하게 된다. 메모리 유닛의 각 메모리 모듈이 'read-then-write at the same address' 방식으로 동작함으로써 이중 메모리 버퍼를 사용하지 않고 최소의 RAM 용량으로 병렬 전치메모리를 구성할 수 있으며 순차 전치메모리에서와 같이 파이프라인 방식으로 동작하게 된다.

위에서 설명한 방식으로 병렬 전치메모리가 동작하려면 다음과 같은 일들이 결정되어야 한다.

(1) 메모리 모듈의 설정: 입력 네트워크에서 동시에 입력되는 한 행의 데이터(8 개)를 각각 어떠한 메모리 모듈에 저장할지를 결정하여야 한다. 이러한 메모리 모듈의 설정은 다음과 같은 조건을 만족시켜야 한다.

a) 동시에 입력되는 한 행의 데이터들이 서로 다른 메모리 모듈에 저장되어야 한다.

b) 동시에 출력되는 한 열의 데이터들이 서로 다른 메모리 모듈에 저장되어 있어야 한다.

c) 위의 방법에 의해 메모리 모듈에 데이터를 입출력할 때 입출력 네트워크에서 경로의 충돌이 없어야 한다.

(2) 네트워크 경로의 설정: 각 데이터가 저장될 메모리 모듈이 설정되면 각 네트워크의 입력과 출력사이에 8 개의 연결이 결정된다. 이러한 입출력 사이의 연결을 실현하기 위해서 각 연결에 대한 경로를 결정하고, 그 경로를 위한 스위치 상태를 구하여야 한다. 본 논문에서 제안하는 병렬 전치메모리에 사용되는 오메가 네트워크와 역오메가 네트워크에서는 하나의 입력과 하나의 출력이 주어졌을 때, 주어진 입출력을 연결하는 경로는 하나만 존재한다. 따라서 각 연결에 대한 경로는 메모리 모듈의 설정에 의하여 이미 결정되어 있으며 각 경로의 실현을 위한 스위치의 상태만을 구하면 된다.

(3) 메모리 모듈의 주소 결정: 각 메모리 모듈에 저장되는 데이터가 어떤 주소에 저장될 지가 결정되어야 한다. 특히 하나의 데이터(16 비트)는 8 클락 동안에 메모리 모듈에 저장됨으로 8 개의 주소를 필요로 한다. 이러한 메모리 모듈의 주소 결정은 8개의 메모리 모듈에 주소를 공급하는 주소 발생기(address generator)의 복잡성에 크게 영향을 준다. 따라서 간단한 구조의 주소 발생기로 발생 가능한 주소 결정 방식을 사용해야 한다.

Ⅲ. 메모리 모듈의 설정

본 장에서는 병렬 전치메모리에 입력되는 행렬 데이터를 메모리 유닛 안의 어떤 메모리 모듈에 저장할지를 결정하는 문제에 대하여 설명한다. 전 장에서 설명한 바와 같이 메모리 모듈 설정에서 가장 중요한 사항은 행렬 안에 있는 한 행의 데이터들은 각각 서로 다른 메모리 모듈에 지정되어야 한다는 것이다. 이는 같은 행에 있는 두 개의 데이터를 하나의 메모리 모듈에 지정하면 그 메모리 모듈이 동시에 두개의 데이터를 저장하여야 하는 메모리 충돌 현상이 생기기 때문이다. 마찬가지로 하나의 메모리 모듈에서 두개의 데이터를 동시에 읽으려고 하는 메모리 충돌을 피하기 위해서 행렬 안에 있는 한 열의 데이터들은 서로 다른 메모리 모듈에 지정되어야 한다. 병렬 메모리에서 메모리 충돌 현상은 시스템의 성능에 크게 영향을 미치므로 메모리 충돌을 피하기 위하여 많은 연구가 이루어졌다.[3-5][11][12] 일반적인 병렬 메모리에서 메모리 충돌을 피하는 것은 매우 어려운 문제이나 이산여현변환을 위한 병렬 메모리에서 메모리 충돌을 피하는 문제는 본 장에서 설명하는 바와 같이 매우 간단히 해결될 수 있다. 본 장에서 사용하는 메모리 모듈 설

정 방식은 STARAN의 MDA에서 사용하는 메모리 모듈 설정 방식을 비트 단위에서 행렬 요소 단위로 변형한 것이다.[4]

8x8 병렬 전치메모리에 입력되는 행렬을 $\{d_{ij},\ 0\le i,j\le 7\}$, 라하고 i를 이진법으로 표현한 것을 $i_2i_1i_0$라고 하기로 한다. 여기서 $i_2i_1i_0$는 $i_2|i_1|i_0$로 표현할 수도 있다. 연산자 '|'는 연결 연산(concatenation)을 나타낸다. 병렬 전치메모리에 입력되는 행렬 데이터를 메모리 유닛 안의 특정한 메모리 모듈에 지정하는 메모리 모듈의 설정은 다음과 같은 방법으로 한다.

〔메모리 모듈의 설정〕 행렬 요소 $d_{ij}, 0\le i,j\le 7$,를 아래의 함수 $M(i,j)$에 의해 결정되는 메모리 모듈에 저장시킨다.

$M(i,j) = i_2\oplus j_2|i_1\oplus j_1|i_0\oplus j_0$

여기서 연산자 ⊕는 XOR(exclusive OR) 연산을 나타낸다.

$M(i,j)$의 몇 예가 아래에 나타나 있다. 메모리 모듈 설정 방법에 따라서 행렬 요소 d_{00}는 메모리 모듈 0에 저장되며, 행렬 요소 d_{25}와 d_{36}은 각각 메모리 모듈 7과 5에 저장된다.

$M(0,0)=0\oplus0|0\oplus0|0\oplus0=000=0$

$M(2,5)=0\oplus1|1\oplus0|0\oplus1=111=7$

$M(3,6)=0\oplus1|1\oplus1|1\oplus0=101=5$

위에서 설명한 메모리 설정 방법을 사용할 때 동일한 행 또는 동일한 열 안에 있는 두개의 행렬 요소사이에 메모리 충돌이 생기지 않음을 다음의 두 정리가 보여준다.

〔정리 1〕 제안된 메모리 설정 방법에 의해서 저장된 행렬의 동일한 행에 있는 두개의 행렬 요소는 서로 다른 메모리 모듈에 저장된다.

〔증명〕 행 i에 있는 서로 다른 두개의 행렬 요소 d_{im}, $0\le i,m\le 7$, 과 $d_{in}, 0\le i,n\le 7$이 서로 같은 메모리 모듈에 저장된다고 가정하면 메모리 모듈 설정 방법에 따라 다음 식이 성립된다.

$i_2\oplus m_2|i_1\oplus m_1|i_0\oplus m_0 = i_2\oplus n_2|i_1\oplus n_1|i_0\oplus n_0$

따라서

$m_2m_1m_0 = n_2n_1n_0$

따라서

$m = n$

위 식은 d_{im}과 d_{in}이 서로 다른 행렬 요소라는 가정에 위배된다.

〔정리 2〕 제안된 메모리 설정 방법에 의해서 저장된 행렬의 동일한 열에 있는 두개의 행렬 요소는 서로 다른 메모리 모듈에 저장된다.

정리 2는 정리 1과 같은 방법으로 증명된다. 표 1에 메모리 모듈 설정 방법에 따라 행렬 요소가 저장되는 메모리 모듈들이 나타나 있다. 행렬 요소 $d_{ij},\ 0\le i,j\le 7$, 는 표 1에서 행 i와 열 j가 교차되는 부분에 나타나 있는 메모리 모듈에 저장된다. 표 1을 자세히 관찰하면 행렬 요소 d_{ij}는 행렬 요소 d_{ji}와 같은 메모리 모듈에 저장됨을 알 수 있다. 즉 i 번째 행의 j 번째 요소는 i번째 열의 j 번째 요소와 같은 메모리 모듈에 저장된다. 따라서 출력 네트워크에 i 번째 열을 출력하고, 이 때 사용된 각 메모리 모듈의 같은 주소에 입력 네트워크에서 공급되는 그 다음 행렬의 i 번째 행을 저장하면 각 메모리 모듈이 'read-then-write at the same address' 방식으로 사용될 수 있다.

표 1. 메모리 모듈의 설정
Table 1. Memory module assignment

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

Ⅳ. 입출력 네트워크에서의 경로

입력 네트워크는 병렬 전치메모리의 입력 단들을 각각의 입력 단이 가지고 있는 행렬요소가 저장되는 메모리 모듈에 연결시켜야 한다. 마찬가지로 출력 네트워크는 출력 단들을 각각의 출력 단이 요구하는 행렬 요소를 저장하고 있는 메모리 모듈에 연결시켜야 한다. 이러한 입출력 단과 메모리 모듈사이의 연결은 입출력 단에서 요구하는 행렬 요소들이 서로 다른 메모리 모듈에 있다고 해서 항상 가능하지는 않다. 본 장에서는 병렬 전치메모

리의 입출력 네트워크로 사용되는 오메가 네트워크와 역 오메가 네트워크가 전 장에서 제안된 메모리 모듈 설정 방법에 따라 요구되는 입출력과 메모리 모듈 사이의 연결을 항상 가능하게 함을 설명한다.

병렬 전치메모리의 입력 네트워크인 오메가 네트워크의 입력 단에는 한 행의 데이터가 동시에 입력된다. 즉 i 번째 행이 입력될 때 행렬 요소 d_{ij}, $0 \le i,j \le 7$,는 j 번째 입력 단에 입력된다. 행렬 요소 d_{ij}는 메모리 모듈 설정 방법에 따라서 메모리 모듈 $M(i,j)$에 저장되어야 하므로 오메가 네트워크는 j 번째 입력 단을 메모리 모듈 $M(i,j)$에 연결시켜야 한다. 정리 1에 의해서 한 행에 있는 행렬 요소들은 서로 다른 메모리 모듈에 저장된다. 따라서 병렬 전치메모리에서 오메가 네트워크가 구현해야 하는 입출력간의 연결은 하나의 순열(permutation)로 표현될 수 있다. 8x8 병렬 전치메모리에 i 번째 행이 입력될 때 오메가 네트워크가 구현해야 하는 순열은 다음과 같이 8x2 행렬 P_i, $0 \le i \le 7$,로 표현될 수 있다.

$$P_i = \begin{bmatrix} 0 & M(i,0) \\ 1 & M(i,1) \\ 2 & M(i,2) \\ 3 & M(i,3) \\ 4 & M(i,4) \\ 5 & M(i,5) \\ 6 & M(i,6) \\ 7 & M(i,7) \end{bmatrix}$$

위의 행렬에서 좌측의 열은 입력 단의 번호를 나타내며 오른쪽의 열은 좌측의 입력 단이 연결되어야 하는 메모리 모듈의 번호를 나타낸다. 여기서 좌측 열은 0에서 7사이의 숫자이고 우측 열의 $M(i,j)$들도 0에서 7사이의 숫자이므로 이진법으로 표현하면 좌우 측 열들은 각각 3개의 열을 갖는 0/1 행렬이 된다. 따라서 행렬 P_i는 8개의 행과 6개의 열을 갖는 0/1 행렬이 된다. 이러한 8x6 0/1 행렬 P_i를 순열 행렬이라고 한다. 입력 네트워크에 2 번째 행이 입력될 때 오메가 네트워크가 구현해야 하는 순열 행렬 P_2가 아래에 나타나 있다.

$$P_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

순열 행렬 P_i로 표현되는 순열이 오메가 네트워크에 의해서 구현 가능함을 보이기 위해서 오메가 네트워크에 대하여 잘 알려져있는 다음의 정리를 이용한다.[12]

〔정리 3〕 8x6 0/1 순열 행렬 M의 열 2, 3, 4로 이루어지는 8x3 0/1 행렬 W_1(제 일 창문 행렬)과 열 3, 4, 5로 이루어지는 8x3 0/1 행렬 W_2(제 이 창문 행렬)가 각각 000에서 111까지의 모든 0/1 조합을 가지고 있음은 순열 행렬 M으로 표현된 순열이 8개의 입출력 단을 가지는 오메가 네트워크에 의해서 구현되기 위한 필요 충분 조건이다.

정리 3을 이용하여 다음의 정리를 증명할 수 있다.

〔정리 4〕 오메가 네트워크는 순열 행렬 P_i, $0 \le i \le 7$, 로 표현되는 입력 단과 메모리 모듈사이의 모든 순열을 구현할 수 있다.

〔증명〕 오메가 네트워크에 i, $0 \le i \le 7$, 번째 행이 입력될 때 오메가 네트워크가 구현해야 하는 순열 행렬 P_i는 다음과 같다.

$$P_i = \begin{bmatrix} 0 & 0 & 0 & i_2 \oplus 0 & i_1 \oplus 0 & i_0 \oplus 0 \\ 0 & 0 & 1 & i_2 \oplus 0 & i_1 \oplus 0 & i_0 \oplus 1 \\ 0 & 1 & 0 & i_2 \oplus 0 & i_1 \oplus 1 & i_0 \oplus 0 \\ 0 & 1 & 1 & i_2 \oplus 0 & i_1 \oplus 1 & i_0 \oplus 1 \\ 1 & 0 & 0 & i_2 \oplus 1 & i_1 \oplus 0 & i_0 \oplus 0 \\ 1 & 0 & 1 & i_2 \oplus 1 & i_1 \oplus 0 & i_0 \oplus 1 \\ 1 & 1 & 0 & i_2 \oplus 1 & i_1 \oplus 1 & i_0 \oplus 0 \\ 1 & 1 & 1 & i_2 \oplus 1 & i_1 \oplus 1 & i_0 \oplus 1 \end{bmatrix}$$

순열 행렬 P_i의 제 일 창문 행렬 W_1은 다음과 같다.

$$W_1 = \begin{bmatrix} 0 & 0 & i_2 \oplus 0 \\ 0 & 1 & i_2 \oplus 0 \\ 1 & 0 & i_2 \oplus 0 \\ 1 & 1 & i_2 \oplus 0 \\ 0 & 0 & i_2 \oplus 1 \\ 0 & 1 & i_2 \oplus 1 \\ 1 & 0 & i_2 \oplus 1 \\ 1 & 1 & i_2 \oplus 1 \end{bmatrix}$$

위의 식에서 i_2의 값이 1 또는 0이 됨에 상관없이 W_1은 000에서 111까지 모든 값을 가짐을 알 수 있다. 또한 순열 행렬 P_i의 제 이 창문 행렬 W_2는 다음과 같다.

$$W_2 = \begin{bmatrix} 0 & i_2 \oplus 0 & i_1 \oplus 0 \\ 1 & i_2 \oplus 0 & i_1 \oplus 0 \\ 0 & i_2 \oplus 0 & i_1 \oplus 1 \\ 1 & i_2 \oplus 0 & i_1 \oplus 1 \\ 0 & i_2 \oplus 1 & i_1 \oplus 0 \\ 1 & i_2 \oplus 1 & i_1 \oplus 0 \\ 0 & i_2 \oplus 1 & i_1 \oplus 1 \\ 1 & i_2 \oplus 1 & i_1 \oplus 1 \end{bmatrix}$$

W_1에서와 마찬가지로 i_2의 값과 i_1의 값에 상관없이 W_2는 000에서 111까지 모든 값을 가짐을 알 수 있다. 따라서 정리 3에 의해서 순열 행렬 P_i는 오메가 네트워크에 의해서 구현될 수 있다.

8x8 병렬 전치메모리에서 i 번째 열이 출력될 때 역 오메가 네트워크가 구현해야 하는 순열은 다음과 같이 8x2 행렬 Q_i, $0 \le i \le 7$, 로 표현될 수 있다.

$$Q_i = \begin{bmatrix} M(0,i)\ 0 \\ M(1,i)\ 1 \\ M(2,i)\ 2 \\ M(3,i)\ 3 \\ M(4,i)\ 4 \\ M(5,i)\ 5 \\ M(6,i)\ 6 \\ M(7,i)\ 7 \end{bmatrix}$$

$M(m,n) = M(n,m)$이므로

$$Q_i = \begin{bmatrix} M(i,0)\ 0 \\ M(i,1)\ 1 \\ M(i,2)\ 2 \\ M(i,3)\ 3 \\ M(i,4)\ 4 \\ M(i,5)\ 5 \\ M(i,6)\ 6 \\ M(i,7)\ 7 \end{bmatrix}$$

위에서 Q_i는 P_i의 입력과 출력을 서로 바꿈으로써 구해짐을 알 수 있다. 역오메가 네트워크는 오메가 네트워크의 입력과 출력을 서로 바꿈으로써 얻어지는 네트워크이므로 순열 행렬 P_i가 오메가 네트워크에 의해서 구현이 가능하면 순열 행렬 Q_i는 역오메가 네트워크에 의해서 구현이 가능하다. 따라서 병렬 전치메모리의 출력 네트워크인 역오메가 네트워크는 메모리 모듈과 출력 단 사이의 모든 순열을 구현할 수 있다. 이를 요약하여 다음의 정리로 나타낸다.

〔정리 5〕 역오메가 네트워크는 순열 행렬 $Q_i, 0 \le i \le 7$,로 표현되는 출력 단과 메모리 모듈사이의 모든 순열을 구현할 수 있다.

Ⅴ. 주소 발생과 입출력 네트워크 제어

본 장에서는 메모리 유닛 안의 각 메모리 모듈에 주소를 공급하는 주소 발생기와 오메가 네트워크와 역오메가 네트워크의 제어 방법에 대해서 설명한다.

병렬 전치메모리의 메모리 모듈은 64x2 RAM으로 되어 있다. 따라서 각 메모리 모듈에는 $\log_2 64=6$ 비트의 주소 정보가 필요하다. 병렬 전치메모리에서는 6 비트의 주소를 발생하기 위해서 6 비트 이진 계수기(6-bit binary counter)를 사용한다. 하나의 메모리 모듈에는 8개의 데이터가 저장되며 하나의 데이터(16 비트)는 8개의 주소를 사용하므로 6 비트 이진계수기의 하위 3 비트는 하나의 데이터를 연속한 주소에 저장하는데 사용된다. 나머지 상위 3 비트는 현재 입력되는 행을 나타내며 아래에서 설명하는 각 메모리 모듈의 주소 발생에 사용된다.

행렬 요소 $d_{ij}, 0 \le i,j \le 7$,가 저장되는 주소의 상위 3 비트는 아래에 보이는 주소 함수 A_o와 A_e에 의해서 결정된다.

$A_o(i,j) = i$

$A_e(i,j) = j$

병렬 전치메모리에 입력되는 행렬 중 홀수 번째에 입력되는 행렬에는 주소 함수 A_o가 쓰이며 짝수 번째에 입력되는 행렬에는 주소 함수 A_e가 쓰인다. 즉 홀수 번째의 행렬을 메모리 모듈에 입출력할 때는 각 행렬 요소는 자신의 행 번호와 동일한 주소에서 입출력되며, 짝수 번째의 행렬을 메모리 모듈에 입출력할 때는 각 행렬 요소는 자신의 열 번호와 동일한 주소에서 입출력된다. 주소 함수 A_o와 A_e사이에는 다음 관계가 성립한다.

$A_o(i,j) = A_e(j,i)$

즉, 홀수 번째 행렬의 요소 d_{ij}와 짝수 번째 행렬의 요소 d_{ji}는 같은 주소에 저장된다는 것이다. 또한 메모리 설정 방법에 의하면 d_{ij}와 d_{ji}는 같은 메모리 모듈 $M(i,j)$에 저장된다. 따라서 병렬 전치메모리에서 짝수 번째 행렬을 출력하며 동시에 홀수 번째 행렬을 입력할 경우에 메모리 모듈 $M(i,j)$는 출력 네트워크에 짝수 번째 행렬의 i 번째 열의 j 번째 요소 d_{ji}를 주소 i 에서 출력하고 입력 네트워크에서 입력되는 홀수 번째 행렬의 i 번째 행의 j 번째 요소 d_{ij}를 동일한 주소에 입력하게 된다. 또한 병렬 전치메모리에서 홀수 번째 행렬을 출력하고 짝수 번째 행렬을 입력할 경우에는 메모리 모듈 $M(i,j)$는 출력 네트워크에 홀수 번째 행렬의 j 번째 열의 i 번째 요소 d_{ij}를 주소 i에서 출력하고 입력 네트워크에서 입력되는 짝수 번째 행렬의 j 번째 행의 i 번째 요소 d_{ji}를 동일한 주소에 입력하게 된다. 이러한 방법으로 각 메모리 모듈은 'read-then-write at the same address' 방식으로 동작할 수 있다.

위에서 설명한 주소 발생 방법에 의하면 각 메모리 모듈에 입력되는 행렬 요소의 행 또는 열이 각 메모리 모듈의 주소로 사용된다. 각 메모리 모듈에 입력되는 행렬 요소의 행은 6 비트 이진 계수기의 상위 3 비트에서 얻어지나 각 행렬 요소의 열은 6 비트 이진 계수기에서 직접 얻을 수가 없다. 따라서 각 메모리 모듈에 입력되는 행렬 요소 d_{ij}의 열 j를 얻기 위하여 다음의 관계를 이용한다.

$j = j_2 i_1 j_0$

$= i_2 \oplus i_2 \oplus j_2 \mid i_1 \oplus i_1 \oplus j_1 \mid i_0 \oplus i_0 \oplus j_0$

위에서 $i_k \oplus j_k, k \in \{0,1,2\}$,는 $M(i,j)$의 k번째 비

트이다. 따라서 메모리 모듈 $M(i,j)$에 저장되는 행렬 요소 d_{ij}의 열 j는 $M(i,j)$와 현재 입력되는 행렬 요소의 행인 i의 각 비트를 XOR 하여서 얻을 수 있다. 메모리 모듈 0에 공급되는 주소의 경우 $M(i,j)$=0이므로 열 번호 j는 다음과 같다.

$$j = i_2 \oplus 0 \mid i_1 \oplus 0 \mid i_0 \oplus 0$$
$$= i_2 i_1 i_0$$
$$= i$$

따라서 홀수 번째 행렬이 입력될 때와 짝수 번째 행렬이 입력될 때 모두 $i_2 i_1 i_0$를 상위 주소 3 비트로 쓰면 된다. 메모리 모듈 1에 공급되는 주소의 경우 $M(i,j)$=1이므로 열 번호 j는 다음과 같다.

$$j = i_2 \oplus 0 \mid i_1 \oplus 0 \mid i_0 \oplus 1$$
$$= i_2 i_1 \overline{i}_0$$

따라서 홀수 번째 행렬이 입력될 때는 $i_2 i_1 i_0$를 상위 주소 3 비트로 쓰고 짝수 번째 행렬이 입력될 때는 $i_2 i_1 \overline{i}_0$를 상위 주소 3 비트로 쓰면 된다. 마지막 예로 메모리 모듈 7에 공급되는 주소의 경우 $M(i,j)$=7이므로 열 번호 j는 다음과 같다.

$$j = i_2 \oplus 1 \mid i_1 \oplus 1 \mid i_0 \oplus 1$$
$$= \overline{i_2 i_1 i_0}$$

따라서 홀수 번째 행렬이 입력될 때는 $i_2 i_1 i_0$를 상위 주소 3 비트로 쓰고 짝수 번째 행렬이 입력될 때는 $\overline{i_2\, i_1 i_0}$를 상위 주소 3 비트로 쓰면 된다. 그림 6은 각 메모리 모듈에 주소를 공급하는 주소 발생기의 구조를 보여주고 있다. 그림 6에서 각 메모리 모듈에 공급되는 주소를 보면 하위 3 비트는 6 비트 이진 계수기의 하위 3 비트를 그대로 사용하고 상위 3 비트는 위에서 설명한 주소 발생 방법에 따라 6 비트 이진 계수기의 상위 3 비트에서 얻어지고 있다. 주소 발생기에서 odd/even 신호는 현재 입력되는 행렬이 홀수 번째 행렬인지 짝수 번째 행렬인지를 나타내는 신호이다. 그림 6의 주소 발생기를 기존의 순차 전치메모리의 주소발생기와 비교하면 2x1 MUX의 수가 6 개에서 3 개로 줄어들고, 대신에 3 개

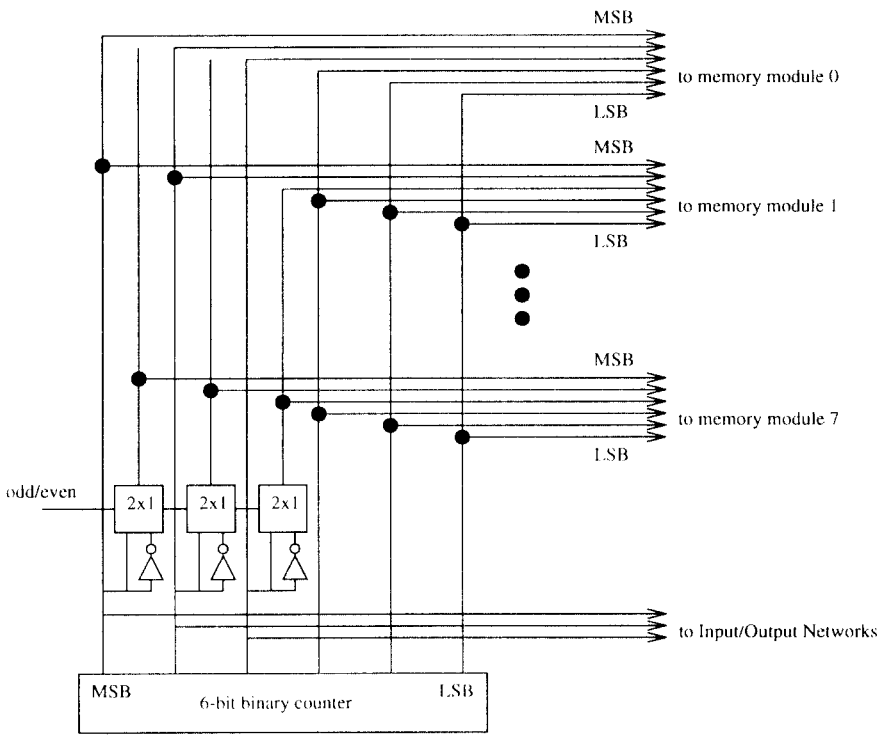

그림 6. 8×8 병렬 전치메모리의 주소 발생기
Fig. 6. Address generator for 8x8 parallel transposition memory

의 인버터(inverter)가 더 사용되었음을 알 수 있다. 따라서 병렬 전치메모리의 주소 발생기가 순차 전치메모리의 주소발생기보다 더 간단하지만 그 차이는 매우 적다. 병렬 전치메모리의 주소발생기는 다음에 설명하는 바와 같이 네트워크의 제어에 필요한 제어신호도 공급하여 준다. 순차 전치메모리의 주소 발생기가 메모리의 주소만을 공급함에 반하여 병렬 전치메모리의 주소 발생기는 메모리의 주소 공급과 함께 네트워크의 제어신호도 공급하면서 순차 전치메모리의 주소발생기와 같은 간단한 구조를 가진다는 것은 특기할만한 점이다.

그림 6의 주소 발생기는 오메가 네트워크와 역오메가 네트워크의 각 단에 있는 스위치 상태를 결정하는 네트워크 제어 신호도 제공하고 있다. 아래에 네트워크 제어 신호에 대한 상세한 설명을 한다.

오메가 네트워크의 구성요소인 스위치의 상태에는 '0'과 '1'이 있다. 스위치의 상태가 '0'이라 함은 스위치가 병렬 연결을 함을 나타낸다. 이 때 스위치의 제어 신호에는 '0'이 공급된다. 또한 스위치의 상태가 '1'이라 함은 스위치가 교차 연결을 함을 나타낸다. 이 때 스위치의 제어 신호에는 '1'이 공급된다. 그림 7의 아래 부분에 두 개의 스위치의 상태가 나타나 있다. 8개의 입출력을 가지는 오메가 네트워크를 구성하는 shuffle/exchange stage 한 단에는 4개의 스위치가 있다. 각 스위치에 위로부터 00, 01, 10, 11의 번호를 부여한다. Shuffle/exchange stage의 기능은 다음과 같이 설명할 수 있다.

〔Shuffle/exchange stage의 기능〕 모든 스위치 xy, $x, y \in \{0,1\}$, 에 대해서,

(1) 스위치 xy의 상태가 '0'일 때는 입력 $0xy$를 출력 $xy0$에 그리고 입력 $1xy$를 출력 $xy1$에 연결시킨다.

(2) 스위치 xy의 상태가 '1'일 때는 입력 $0xy$를 출력 $xy1$에 그리고 입력 $1xy$를 출력 $xy0$에 연결시킨다.

오메가 네트워크의 각 단(shuffle/exchange stage)은 전 단의 출력을 입력으로 받아 위에 설명한 기능을 수행한다. 오메가 네트워크에 $i, 0 \le i \le 7$, 번째 행이 입력될 때 오메가 네트워크가 구현해야 하는 순열을 표현하는 순열 행렬 P_i를 자세히 보면 위에서 설명한 shuffle/exchange 기능이 세 번 수행되었으며 0 번째 단에 있는 스위치들의 상태는 i_2에 의해서, 1 번째 단에 있는 스위치들의 상태는 i_1에 의해서, 그리고 2 번째 단

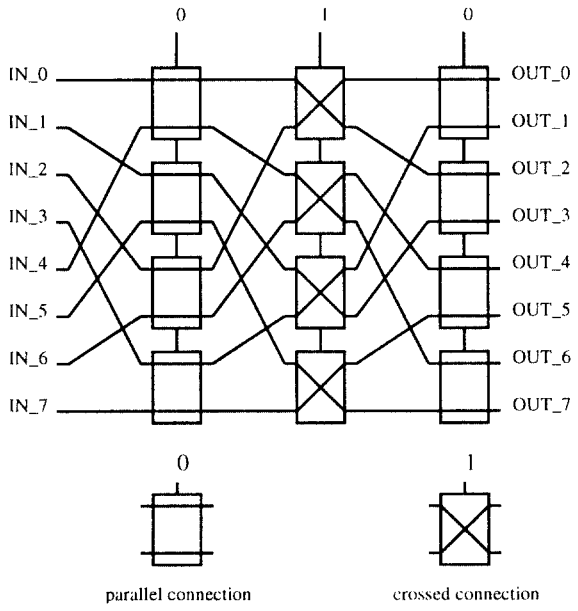

그림 7. 오메가 네트워크에서 순열 행렬 P_2의 구현
Fig. 7. Realization of permutation matrix P_2 in Omega network

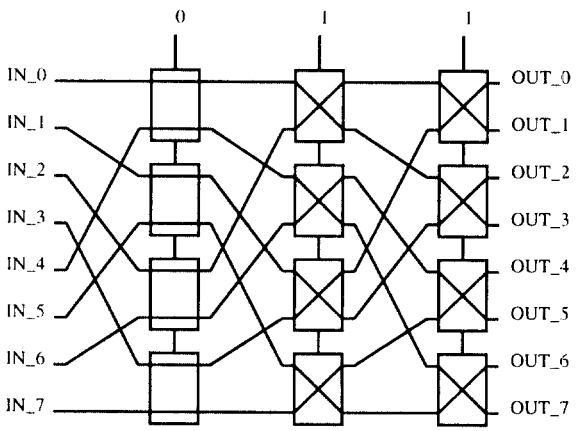

그림 8. 오메가 네트워크에서 순열 행렬 P_3의 구현
Fig. 8. Realization of permutation matrix P_3 in Omega network

에 있는 스위치들의 상태는 i_0에 의해서 결정됨을 알 수 있다. 예를 들면 i_2=0일 때는 0 번째 단에 있는 스위치들의 상태는 모두 '0'이 되며 i_2=1일 때는 0 번째 단에 있는 스위치들의 상태는 모두 '1'이 된다. 따라서 i_2를 그대로 0 번째 단에 있는 스위치들의 제어 신호로 사용하면 된다. 나머지 단에 있는 스위치들의 제어 신호들도 i_1과 i_0를 그대로 사용하면 된다. 그림 7은 2 번째 행이 입력될 때의 순열 행렬 P_2를 오메가 네트워크가 구현한 것을 보이고 있으며 그림 8은 3 번째 행이 입력될 때의 순열 행렬 P_3를 오메가 네트워크가 구현한 것을 보이고 있다. 병렬 전치메모리의 출력 네트워크인 역오메가 네트워크는 오메가 네트워크가 구현하는 순열 행렬의 입력과 출력이 서로 바뀐 순열 행렬을 구현해야 한다. 그런데 역오메가 네트워크는 오메가 네트워크의 입력과 출력을 서로 바꿈으로써 얻어짐으로 오메가 네트워크와 같은 방법으로 제어를 하면 된다. 한가지 다른 점은 i_2가 2 번째 단에, i_1가 1 번째 단에, 그리고 i_0가 0 번째 단에 있는 스위치들의 제어 신호로 사용되어야 한다는 것이다.

VI. 결 론

본 논문에서는 입출력을 병렬로 수행하는 병렬 전치메모리를 소개하였다. 본 논문에서 소개되는 병렬 전치메모리는 기존의 순차 전치메모리와 같이 기억 소자로 RAM을 사용하며 입출력 네트워크에 오메가 네트워크와 역오메가 네트워크를 사용한다. 기존의 전치 네트워크와는 달리 쉬프트 레지스터는 사용하지 않는다. 병렬 전치메모리에서 사용되는 주소발생기는 네트워크 제어기의 역할을 동시에 수행하면서도 기존의 순차 전치메모리의 주소발생기와 같은 간단한 구조를 가진다.

병렬 전치메모리에 사용되는 메모리 모듈의 크기가 순차 전치메모리에 사용되는 메모리 모듈의 크기보다 작으므로 병렬 전치메모리는 적어도 순차 전치메모리의 동작 속도이상으로 동작한다. 또한 1차원 이산여현변환기와의 사이에 직병렬 데이터 변환기를 사용하지 않음으로서 전체 2차원 이산여현변환기의 지연시간(pipe fill-up time)은 크게 줄어들게 된다. 이러한 특징은 이산여현변환기가 정방향 변환과 역방향 변환을 교대로 수행하는 경우에 시스템의 성능 향상에 도움이 된다.

본 논문에서 소개되는 병렬 전치메모리는 기존의 순차 전치메모리와 비슷한 비용을 사용하면서 1차원 이산여현변환기와의 사이에 직병렬 데이터 변환기를 사용하지 않게 함으로써 전체 2차원 이산여현변환기의 비용을 크게 절감시킨다. 표 2에 기존의 순차 전치메모리와 본 논문에서 제안된 병렬 전치메모리을 사용하였을 때 두개의 1차원 이산여현변환기 사이의 비용을 비교하였다. 1차원 이산여현변환기에는 fast algorithm을 바탕으로 한 병렬 구조가 사용되었다고 가정하였다. 주소 발생기는 두 방식의 전치메모리에서 그 비용이 거의 같고 다른 비용에 비해 매우 작으므로 (6 또는 8 비트 이진 계수기) 비교에서 생략하였다. 표 2를 보면 병렬 전치메모리의 사용 시 기존의 순차 전치메모리를 사용하는 것에 비해 현저한 비용의 감소가 있음을 볼 수 있다.

표 2. 순차 전치메모리와 병렬 전치 메모리의 비용 비교
Table 2. Cost comparison between serial and parallel transposition memories

	Existing Transposition Memories	Proposed Parallel Transposition Memory
16×16 2-D DCT (16-bit precision)	4096-bit RAM 1024 register 480 2×1 MUX	4096-bit RAM 32 register 64 2×2 SW
8×8 2-D DCT (16-bit precision)	1024-bit RAM 512 register 224 2×1 MUX	1024-bit RAM 32 register 48 2×2 SW

참고문헌

1. N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Trans. Computers*, vol. C-23, pp.90-93, Jan. 1974.
2. A. Artieri, S. Kritter, F. Jutand, and N. Demassieux, "A One Chip VLSI for Real Time Two-Dimensional Discrete Cosine Transform," *Proc. Intl. Symposium on Circuits and Systems*, pp.701-704, 1988.
3. M. Balakrishnan, R. Jain, and C. S. Raghavendra, "On Array Storage for Conflict-

free Memory Access for Parallel Processors," *Proc. Intl. Conf. on Parallel Processing*, vol. 1, pp.103-107, 1988.

4. K. Batcher, "The Multidimensional Access Memory in STARAN," *IEEE Trans. Computers*, vol. C-26, pp.174-177, Feb. 1977.
5. P. Budnik and D. J. Kuck, "The Organization and Use of Parallel Memories," *IEEE Trans. Computers*, vol. C-20, no. 12, pp.1566-1569, 1971.
6. JC. Carlach, P. Penard, and JL. Sicre, "TCAD: a 27MHZ 8x8 Discrete Cosine Transform Chip," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp.2429-2432, 1989.
7. N. Demassieux, G. Concordel, J-P. Durandeau, and F. Jutand, "An Optimized VLSI Architecture for a Multiformat Discrete Cosine Transform," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp.547-550, 1987.
8. L. J. D'Luna, W. A. Cook, R. M. Gudash, G. W. Brown, T. J. Tredwell, J. R. Fisher, T. Tarn, "An 8 X 8 Discrete Cosine Transform Chip with Pixel Rate Clocks," *Proc. The 3rd Annual IEEE ASIC Seminar and Exhibit*, pp.P7-5.1~P7-5.4, 1990.
9. R. C. Gonzalez and P. Wintz, *Digital Image Processing*, 2nd Ed., Addison-Wesley Publishing Company, 1987.
10. F. Jutand, Z. J. Mou, N. Demassieux, "DCT Architectures for HDTV," *Proc. Intl. Symposium on Circuits and Systems*, pp.196-199, 1991.
11. K. Kim and V. K. Prasanna, "Latin Squares for Parallel Array Access," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 4, pp.361-370, 1993.
12. D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Computers*, vol. C-24, no. 12, pp.1145-1155, 1975.
13. W. Li and D. Slawecki, "A High Speed 2-D DCT/IDCT Processor," *Proc. Intl. Symposium on Circuits and Systems*, pp.192-195, 1991.
14. L. Matterne, D. Chong, B. Mc Sweeney, and R. Woudsma, "A Flexible High Performance 2-D Discrete Cosine Transform IC," *Proc. Intl. Symposium on Circuits and Systems*, pp.618-621, 1989.
15. M-T. Sun, T-C. Chen, and A. M. Gottlieb, "VLSI Implementation of a 16x16 Discrete Cosine Transform," *IEEE Trans. Circuits and Systems*, vol. 36, no. 4, pp.610-617, 1989.
16. S. Uramoto, Y. Inoue, A. Takabatake, Y. Yamashita, H. Terane, and M. Yoshimoto, "A 100-MHz 2-D Discrete Cosine Transform Core Processor," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp.492-499, 1992.

金起徹(Kichul Kim) 정회원

1982년 2월 : 서울대학교 전기공학과 졸업(공학사)
1984년 2월 : 서울대학교 대학원 전기공학과 졸업(공학석사)
1991년 8월 : University of Southern California, 전기공학과 졸업(Ph. D)
1984년 3월~1994년 2월 : 한국전자통신연구소 선임연구원
1994년 3월~현재 : 서울시립대학교 반도체공학과 전임강사