特輯 技術解說

병렬형 컴퓨터 구조

孟 承 烈
韓國科學技術院 電算學科

Ⅰ. 서 론

컴퓨터 제조 기술의 눈부신 발전과 고성능을 요구하는 응용 분야의 증대로 인하여 병렬 처리를 위한 컴퓨터에 대한 관심이 높아지고 있다. 한 순간에 하나씩 명령어를 처리하는 von Neumann형의 컴퓨터 구조로는 더 높은 성능을 갖는 컴퓨터를 만들기가 매우 힘들고, 또한 최근에는 종래의 컴퓨터로는 원하는 시간내에 풀 수 없는 대규모의 문제들을 해결해야 하는 경우가 많이 생겼으며, 컴퓨터 설계자들은 반도체 제조 기술의 발달로 더 많은 논리 소자를 값싸게 사용할 수 있게 되었기 때문에 병렬 처리의 필요와 함께 병렬 처리를 위한 컴퓨터 구조에 대한 연구가 많이 진행되고 있다.

"병렬 처리"라는 것을 여러 가지로 서로 다르게 정의하는 경우도 있으나 본 고에서는 병렬형 컴퓨터를 규모가 큰 하나의 문제를 빠른 시간내에 풀기 위해서 여러 개의 프로세서가 서로 자료를 교환하며 협동하여 처리하는 시스템으로 정의하고 분산 시스템이나 throughput을 높이기 위한 멀티 프로세싱 시스템과는 구분하기로 한다.

병렬형 컴퓨터를 위한 많은 연구들이 지금까지 진행되어 왔으며 많은 연구 결과들이 제시되었다. 그리고 여러 제품들이 상용화되어 사용되고 있으나 효율적이고 사용하기 쉬운 시스템이 되기 위해서는 더욱 많은 연구가 있어야 한다. 본 고에서는 병렬 처리 컴퓨터의 여러 종류와 이 시스템들의 구성 목적 및 문제점들을 분석해 보고 앞으로의 연구 방향에 대하여 설명하기로 한다.

본 고의 구성은 다음과 같다. 제2장에서는 병렬 컴퓨터의 분류 방식과 각 부류에 속하는 시스템의 예를 살펴보고, 제3장과 제4장에서는 각각 공유메모리 다중 처리 컴퓨터와 메시지 전송 다중 처리 컴퓨터의 구조에 대하여 살펴본 후 제5장에서는 결론을 맺는다.

Ⅱ. 병렬 컴퓨터의 분류

1. Flynn의 분류

1972년에 J. Flynn은 인스트럭션과 데이타의 순차적인 수행 형태를 스트림(stream)으로 나타내고 인스트럭션 스트림의 수와 데이타 스트림의 수에 따라 여러 가지 컴퓨터 구성 형태를 다음과 같이 4가지로 분류하였다.

- Single instruction stream-Single data stream(SISD)
- Multiple instruction streams-Single data stream (MISD)
- Single instruction stream-Multiple data streams (SIMD)
- Multiple instruction streams-Multiple data streams (MIMD)

일반적으로 SIMD와 MIMD 컴퓨터를 병렬 컴퓨터라

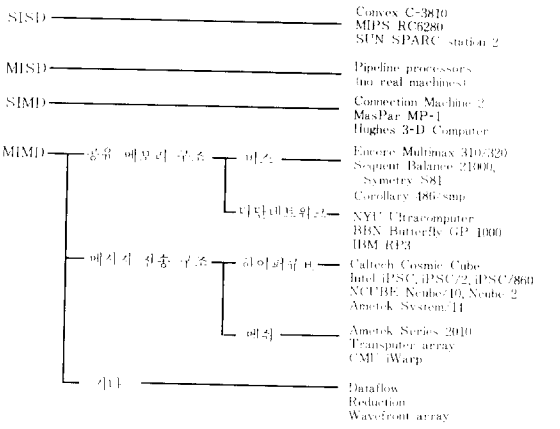


그림 1. 컴퓨터 구성 형태의 분류

고 하며 이 각각은 다시 여러 범주로 세분될 수 있다. 그림 1에 각 범주에 속하는 시스템들을 정리하였다.

2. SIMD 컴퓨터

SIMD 컴퓨터는 다수의 처리 소자들이 배열 형태로 연결되어 하나의 제어 장치에 의하여 제어되어 동기적으로 병렬 처리를 수행하는 컴퓨터를 말하며 일반적으로 그림 2와 같은 구조를 갖는다. 이러한 방식의 컴퓨터는 각각의 데이타를 인출하기 위한 명령어를 각각의 프로세서가 따로 인출할 필요가 없기 때문에 시간 낭비를 줄일 수가 있고, 모든 명령을 데이타에 병행해서 수행할 수가 있으므로 von Neumann형 컴퓨터인 SISD 컴퓨터보다 빠르게 수행할 수 있다. SIMD 컴퓨터는 주로 데이타 행렬 혹은 배열의 벡터 연산을 빠르게 수행할 필요가 있는 응용 분야를 위하여 설계되며 현재 MIMD 컴퓨터에 비해 그 응용 분야가 한정되어 있다. 따라서 본 고에서는 주로 다음에 소개되는 MIMD형 병렬 처리 컴퓨터의 구조에 대해서 소개하기로 한다.

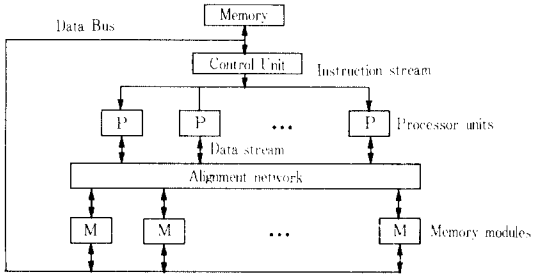


그림 2. SIMD 컴퓨터 구조

3. MIMD 컴퓨터

한 개의 명령어에 의하여 여러 개의 프로세스가 보조를 맞추어 수행되는 SIMD와는 다르게, 각각의 명령어와 데이타를 가진 여러 개의 프로세서가 각각 자신의 제어 아래 어떤 하나의 문제를 해결하는 컴퓨터를 MIMD 컴퓨터라 한다. 이런 MIMD 컴퓨터도 여러 가지 제어 방식과 데이타 전달 방식이 있을 수 있으며 이에 따라 여러 종류로 나뉠 수 있다.

Treleaven[31]은 MIMD 컴퓨터를 제어 방식과 데이타 전달 방식에 따라 그림 3과 같이 더 세분하여 분류하였다. 그림3의 제어 구동형 구조들은 Ultracomputer[13], RP3[25](이상 공유 메모리 방식), Ncube[22], Transputer array(이상 메시지 전송 방식)등과 같이 von Neumann 형 구조에 가까운 것들로 현재 상용화된 병렬 처리 시스

	데이타 전달 방식	
제어 방식	공유 메모리	메시지 전송
제어 구동	공유메모리 MIMD	메시지 전송 MIMD
패턴 구동	논리	액터
요구 구동	그래프 리덕션	스트링 리덕션
데이타 구동	데이타 구동 I-구조형	데이타 흐름 토큰형

그림 3. MIMD 컴퓨터 구조의 분류

템들이 이런 부류에 속한다. 패턴 구동형의 논리 또는 액터 시스템, 요구 구동형의 리덕션 시스템, 데이타 구동형의 데이타 흐름 컴퓨터들은 현재 학교나 연구소 등에서 연구되고 있는 시스템들이며 아직은 시작품 단계로서 상용화되기 까지에는 더 많은 연구가 필요한 시스템들이다. 따라서 본 고에서는 이런 시스템들에 대한 설명은 생략하기로 하고 관심있는 독자들은 참고문헌[31]을 참조하시기 바란다. 본 고에서는 MIMD 컴퓨터라고 하는 범주를 제어 구동형 MIMD에 국한시키고 이에 대한 설명만 하기로 한다.

MIMD 컴퓨터에서 어떤 문제를 해결하기 위해서는, 이 문제를 해결하기 위한 프로그램을 여러 개의 프로그램(프로세스 또는 태스크) 들로 나누어서 각각을 하나의 프로세서에 할당하여 수행하여야 한다. MIMD는 이렇게 문제들을 여러 개의 작은 프로그램으로 나누어서 수행하여야 하기 때문에 SIMD나 파이프라인 컴퓨터에서는 발생하지 않았던 여러 문제가 발생하게 된다. MIMD 컴퓨터에서 발생하는 문제로는, 먼저 하나의 문제를 어떻게 여러 개의 조각으로 나누어 수행시킬 것인가 하는 것이며, 또 각각의 프로그램을 각 프로세스에서 수행시킬 때 각각의 수행 시간을 예측하기 어렵기 때문에, 효율적인 수행을 위해서는 각각의 프로세스의 부하를 균형있게 하기 위한 효율적이고 융통성 있는 통신 방법이 필요하게 된다.

MIMD 컴퓨터는 여러 작은 프로그램들이 어떻게 공유 데이타를 처리하는 가에 따라 공유 메모리 방식과 메시지 전송 방식으로 나눌 수 있다. 사용자(프로그래머) 측면에서 보면 하나의 공유 기억 공간을 제공하는 공유 메모리 방식이 프로그래밍하기가 편리하고 여러 가지 프로그래밍 모델을 효율적으로 지원할 수 있으므로 많은 연구가 진행되고 있으며, 또한 상용화된 제품들이 많이 있다.

반면, 하드웨어를 설계하는 입장에서는 여러 프로세

서가 기억 장치를 공유하는 것이 비용이 많이 들고 또 기억 장치를 공유하기 위해서는 복잡한 상호 연결망이 필요하게 되므로 각 프로세서는 자신의 지역 기억 장치를 사용하고 태스크 간의 통신은 메시지 전송을 통해 수행하는 방식을 선호하게 된다. 이 경우 공유 기억장치에의 접근은 감소되지만 지역 기억 장치에 저장된 데이타에 접근하려고 하는 태스크들은 그 지역 기억 장치가 있는 프로세서에 고정되게 되고, 따라서 프로세서의 사용 효율이 떨어질 수 있다. 그러나 최근에는 이러한 메시지 전송 방식을 추구하는 MIMD 컴퓨터들도 많이 연구 되고 있으며 상용화된 시스템들도 출현하고 있다.

다음 장에서는 공유 메모리 방식의 MIMD 컴퓨터와 메시지 전송 방식의 MIMD 컴퓨터를 살펴본다.

Ⅲ. 공유 메모리 MIMD 컴퓨터의 구조

공유 메모리 MIMD 컴퓨터는 모든 프로세서가 하나의 광역(global) 주소 공간을 동등하게 직접적으로 참조하는 컴퓨터로 정의된다.

본 장에서는 공유 메모리 MIMD 컴퓨터를 설계할 때 고려되어야 할 기본적인 문제에 대하여 기술하고 버스에 기초한 시스템과 다단계 연결망을 사용한 시스템의 사례를 살펴 본다.

1. 설계시 고려사항

병렬 처리를 위하여 공유 메모리 모델을 사용할 때 가장 중요한 점은 모든 프로세서에게 하나의 큰 메모리 주소 공간에 동등하게 직접 접근할 수 있는 기능을 제공하는 것이다. 이러한 특징은 메시지 전송 방식의 설계보다 프로그래밍 환경에 대한 제약을 완화시키지만, 광역 메모리 참조를 가능하게 하는 메카니즘과 공유 변수를 잘못 참조하지 않도록 하는 메카니즘을 제공하기 위하여 하드웨어 면에서 더 많은 비용이 든다.

공유 메모리 MIMD 컴퓨터의 설계에 있어서 가장 중요한 두 가지 고려 사항은 메모리 참조에 걸리는 긴 지연 시간을 줄이는 방법과 공유 데이타를 참조할 때 동기화는 유지하면서 제약을 없애는 방법이다.

현재의 가장 뛰어난 상호 연결망 기술로도 메모리 지연 시간은 프로세서의 인스트럭션 수행 시간에 비해 무시할 수 없을 정도로 크며, 따라서 아키텍춰 설계면에서 프로세서가 메모리 요청에 대한 응답을 기다리고 있는 동안에 프로세서가 공전하고 있지 않아도 되는 방법을 강구하여야 한다. n개의 프로세서가 공유하기 위한 n-포트 메모리를 생각할 수도 있으나 현재의 기술로 비 현실적이며 가능하다 할지라도 포트 수를 늘림으로써 메모리 참조 시간이 증가되는 것은 피할 수 없다.

두번째 문제는 병행 수행되는 프로세스간에 참조 순서를 제어하는 "순서 제어(sequence control)"와, 상호 배제(mutual exclusion)와 같이 공유 자원에 대하여 경쟁하는 프로세서들의 접근을 제어하는 "접근 제어(access control)"를 어떻게 해결하는가 하는 것이다. 이러한 해결책에 있어서 불필요한 직렬화(serialization)를 피하고 가능한 한 많은 병렬성을 유지하여야 한다.

2. 캐쉬와 버스에 기초한 MIMD 시스템

일정한 성능을 갖는 프로세서들을 사용해서 높은 성능을 얻으려면 많은 프로세서를 효과적으로 사용하는 것이 중요하다. 버스는 저렴한 비용으로 높은 성능을 얻을 수 있는 효과적인 방법 중의 하나로써 현재 버스에 기초를 두고 수십 개까지의 고성능 마이크로 프로세서를 효율적으로 연결하는 연구에 관심이 집중되고 있으며 많은 상용 시스템들이 발표되었다.

1) 버스에 기초한 시스템의 구조

단일 마이크로 프로세서를 사용하는 시스템에서 효과적인 캐쉬 정책을 사용함으로써 주기억 장치에 접근하기 위한 버스의 대역폭에 여유를 남길 수 있다. 따라서 여러 개의 프로세서가 하나의 메모리와 버스를 공유하도록 할 수 있다.

이와 같이 시스템을 구성하였을 때 버스가 전체 시스템에 가장 영향을 주는 요소가 되므로 이 버스는 꼭 필요할 때에만 사용해야 한다. 전체 대역폭은 캐쉬를 효과적으로 사용하여 메모리 요청 횟수를 줄이거나 메모리 사용량을 줄임으로써 절약될 수 있다. 이러한 시스템에서는 비동기적인 프로토콜을 사용하여 메모리가 요청에 대한 응답을 준비할 동안 버스를 다른 프로세서가 사용할 수 있도록 한다.

그러나 메모리가 작업을 할 동안 버스가 사용 가능하더라도 이 때 다른 프로세서가 메모리를 사용할 수 없으면 많은 이득을 얻을 수 없다. 이에 대한 해결책으로 여러 개의 메모리 모듈을 사용하는 방법이 있다. 이렇게 시스템을 구성하면 원하는 메모리 용량을 하나의 회로 보드에 넣지 못한다거나 한번의 메모리 요청에 전송되어야 할 데이타의 용량이 보드의 I/O 용량을 초과하는

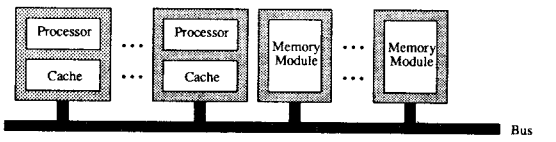


그림 4. 버스에 기초한 시스템의 구조

문제를 해결할 수 있다. 그림 4에 버스에 기초한 시스템의 구조를 나타내었다.

2) 캐쉬 메모리

버스에 기초한 MIMD 시스템에서 캐쉬를 사용함으로써 단일 프로세서 시스템에서와 같이 평균 메모리 접근 지연 시간을 줄일 수 있을 뿐만 아니라 각 프로세서에서 요구하는 버스 대역폭을 줄임으로써 더 많은 프로세서를 사용할 수 있다.

이 때 동일 메모리 블럭의 여러 사본들이 동시에 하나 이상의 캐쉬에 저장될 수 있는 데, 이와 같이 같은 자료에 대하여 여러 개의 사본들이 존재한다는 사실은 결국 캐쉬들 간의 일관성을 유지하기 위한 부가적인 작업을 요구한다.

이러한 캐쉬 일관성 문제를 해결하기 위한 방법으로 소프트웨어에 의한 방법과 하드웨어에 의한 방법이 있는 데 순수한 소프트웨어에 의한 방법은 제한이 많기 때문에 상용시스템에서는 주로 하드웨어에 의한 방법을 사용한다. 버스를 연결망으로 사용하는 시스템에서는 캐쉬 일관성 유지에 관한 명령들을 버스를 통하여 방송하며 이 때 각 캐쉬에는 버스를 통하여 지나가는 모든 일관성 유지 관련 명령들을 감시하면서 일관성 유지를 수행하는 하드웨어를 둠으로써 캐쉬 일관성 유지 문제를 해결할 수 있다. 이러한 기법을 캐쉬 스누핑(cache snooping)이라고 한다.

이러한 상황에서 캐쉬 일관성 유지를 위하여 사용하는 정책에는 한 처리기가 자신의 캐쉬에 저장된 블럭의 내용을 변경할 때 다른 캐쉬들에 있는 사본들의 내용을 모두 무효화시키는 write 무효화 정책(write invalidate policy)과, 다른 캐쉬에 저장된 사본들의 내용을 모두 같은 값으로 변경시키는 write 변경 정책(write update policy)이 있으며 이 각각에 대한 많은 프로토콜들이 제안되고 구현된 바 있다.[2]

3) 상용화된 시스템

버스에 기초한 상업용 시스템으로는 Encore Multimax 310 / 320[30], Sequent Balance 21000[11], Sequent Symetry S81[10], 그리고 Corollary 486 / smp[6]등이 있다. 이 시스템들은 기존의 마이크로 프로세서 혹은 custom CPU를 사용한 프로세서 모듈을 최대 12~30개까지 버스를 통해 연결하며[12] 각 프로세서 모듈에는 각 회사 나름의 독창적인 캐쉬 프로토콜을 사용하고 있다.

3. NYU Ultracomputer

Ultracomputer[13]는 뉴욕 대학에서 설계, 제안한 공유 메모리 MIMD 컴퓨터로서 최대 4,096 개의 프로세서를

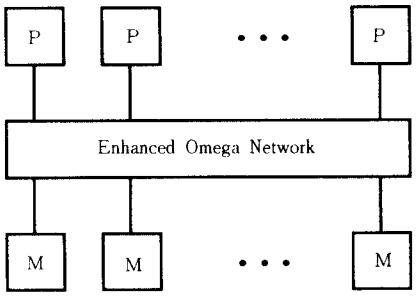


그림 5. Ultracomputer의 구조

개선된 형태의 omega 연결망으로 연결한 구조를 갖는다. 그림 5에 Ultracomputer의 구조를 나타내었다.

Ultracomputer는 "fetch-and-add" 동기화 기법의 사용과 같은 메모리 주소에 대한 둘 이상의 참조를 하나로 결합(combine)하는 상호 연결망의 사용을 그 특징으로 한다. 따라서 fetch-and-add와 결합 연결망(combining network)의 사용은 N개의 프로세서가 어떤 자료구조를 참조하는 데에 걸리는 시간을 하나의 프로세스가 그 자료 구조를 참조하는 데 걸리는 시간과 같게 함으로써 3.1절에서 언급한 "동기화는 유지하면서 제약을 없애는" 멀티프로세서의 기본적인 요구 조건을 만족시킨다.

공유 메모리 프로그램에 대하여 높은 성능을 얻기 위하여 Ultracomputer가 지니고 있는 하드웨어상의 특징으로 다음과 같은 점을 들 수 있다.

① 각 프로세서는 메모리에 다중의 엑세스 요청을 할 수 있다.

② 같은 메모리 모듈에 있는 다른 데이타를 참조할 때 충돌을 줄이기 위하여 연속된 메모리 주소는 메모리 모듈에 차례로 분산되어 있다.

③ 연결망에 의한 지연을 줄이기 위하여 각 프로세서에는 캐쉬 메모리를 둔다.

④ 각 스위칭 노드는 스위치에서의 충돌을 처리하기 위한 큐를 갖는다.

⑤ 각 스위치는 같은 메모리 주소에 대한 메모리 요청을 결합(combine)하는 장치를 갖는다.

⑥ 각 메모리 모듈은 덧셈기를 가지고 fetch-and-add 동기화 연산을 수행한다.

동기화를 위한 기본 연산인 fetch-and-add는 다음과 같이 정의된다. V는 공유 메모리에 저장된 정수 변수이고, e는 정수값 또는 수식 표현, 그리고 F&A를 fetch-and-add 명령이라 할 때 연산 Y ← F&A(V,e)는 Y ← V; V ← V+e의 두 연산을 하나의 인스트럭션으로 수행함

을 의미한다.

Ultracomputer의 오메가 연결망의 스위칭 노드는 그림 6과 같이 충돌이 일어나는 메시지들을 결합하기 위한 큐를 가진 형태로 구성되며 그림 7에 이들의 동작 예를 나타내었다. 두 개의 병행 수행되는 fetch-and-add 인스트럭션이 같은 변수를 참조하려고 할때 다음과 같은 4단계의 순서대로 수행된다.

t1 : 공유 변수 X를 참조하려는 두 개의 fetch-and-add 인스트럭션 F&A(X,e)와 F&A(X,f)가 스위치에서 만난다.

t2 : 스위치는 e+f의 합을 구해서 F&A(X,e+f)를 전송하고 e는 대개 버퍼(wait buffer)에 저장한다.

t3 : 변수 X에 기억되어 있던 값 Y가 응답된다.

t4 : Y값은 F&A(X,e)에 응답되고 Y+e가 F&A(X,f)에 응답된다.

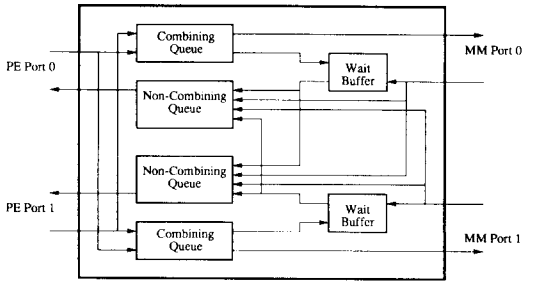


그림 6. Ultracomputer의 오메가 연결망 스위치의 구성

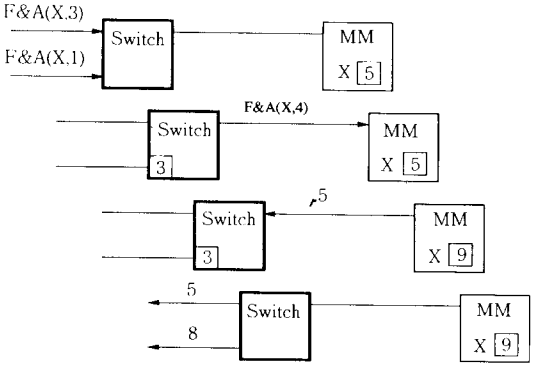


그림 7. 두 개의 병행 fetch-and-add 인스트럭션의 결합

4. BBN Butterfly

BBN Butterfly[7]는 BBN Advanced Computers 사의 상용 컴퓨터로서 Motorola 68020 / 68882를 사용한 노드 프로세서를 256개까지 연결한 공유 메모리 MIMD 컴퓨터이다. 그림 8에서 보듯이 오메가 연결망을 둥글게 연결한 점이 Ultracomputer와 다르다는 것을 알 수 있다. Ultracomputer와 BBN butterfly는 프로세서와 메모리를 연결하는 방법이 있어 유사하지만 공유 메모리 프로그램을 지원하는 방법에 차이가 있다. Ultracomputer는 메모리 참조에 따른 지연을 줄이고 공유 메모리에 대한 동기화를 유지하면서 제약을 없애는 목적을 위해 부가적인 하드웨어를 사용하지만 butterfly는 결합 연결망을 사용하지 않아 하드웨어를 간소화하고 그 대신 소프트웨어를 사용한다.

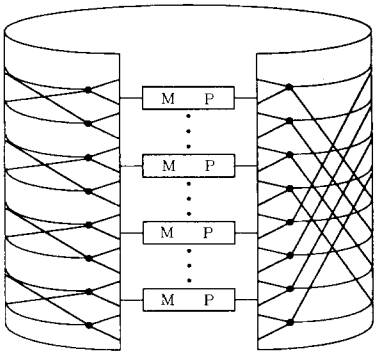


그림 8. BBN Butterfly의 구조

Butterfly의 각 노드는 2.5 MIPS의 성능을 가지며 MC68020 마이크로 프로세서와 마이크로 코드로 제어되는 프로세서 노드 제어기(processor node controller, PNC), 그리고 4MB의 메모리로 구성된다. 최대 256개까지의 노드를 확장할 수 있으므로 최대 성능은 약 600 MIPS이며 여러 benchmark에 대하여 거의 선형적으로 (256개의 프로세서 사용시 230배의 성능 향상) 수행 속도가 증가한다는 연구 결과가 있다.[26] 각 메모리는 자신의 노드 프로세서에 대해 지역성을 가지며 모든 프로세서는 모든 메모리를 참조할 수 있다. PNC는 메모리 관리 장치를 사용하여 가상 주소를 실제 주소로 변환하며 따라서 모든 노드의 메모리는 하나의 광역 주소 공간을 이룬다. 오메가 연결망은 다른 노드에 있는 메모리를 참조할 때 충돌 없이 일정한 시간이 걸리도록 한다. 그러나 각 스위치에는 버퍼가 없으므로 두 개의 메시지가 충돌하였을 때 그 중 하나는 소멸되고 나중에 다시 메모리 참조 요청을 하게 된다.

Ⅳ. 메시지 전송 MIMD 컴퓨터의 구조

메시지 전송 MIMD 컴퓨터는 모든 프로세서가 각각 자신의 지역(local) 메모리를 가지고 명령을 수행하며 프로세서간의 통신은 상호 연결망을 통한 메시지 전송에 의해서 수행되는 컴퓨터로 정의된다.

주로 사용되는 연결망은 하이퍼큐브나 메쉬 형태이다. 따라서 본 고에서는 이 두 가지 종류의 컴퓨터에 대하여 살펴 보기로 한다.

1. 하이퍼큐브 컴퓨터의 구조

1) 하이퍼큐브 구조

일반적으로 이진 n-큐브 연결망 구조를 갖는 컴퓨터를 n-차원 하이퍼큐브 컴퓨터라고 한다. 이러한 컴퓨터는 n-차원 이진 큐브 형태로 연결된 $N=2^n$ 개의 노드로 구성된다. 각 노드는 2^n 개의 n-비트 주소를 가지며 이웃한 n개의 노드와는 한 비트만 다르다.

어떤 차원의 하이퍼큐브를 구성하여 각 노드에 주소를 할당하는 방법은 다음과 같다. 우선 두 개의 노드를 갖는 1-차원 하이퍼큐브에서 시작하여 각 노드를 0과 1로 지정한다. n-차원 큐브는 두 개의(n-1)-차원 큐브로 구성할 수 있으며 이 때 하나는 각 노드의 주소 앞에 0을 붙여서 0-큐브를 구성하고 다른 하나는 각 노드의 주소 앞에 1을 붙여서1-큐브를 구성한 후 0-큐브의 각 노드를 이에 대응되는 1-큐브의 각 노드와 연결하면 n-차원 큐브가 구성된다.

하이퍼큐브 구조는 많은 유용한 성질을 가지고 있다. 예를 들면 메쉬나 트리 같은 구조는 하이퍼큐브에 포함될 수 있으며 이웃한 노드들은 하이퍼큐브의 이웃한 노드로 대응될 수 있다. 많은 과학용 응용 분야에서 메쉬와 트리 같은 계산 구조를 사용하기 때문에 하이퍼큐브 구조는 범용 컴퓨터 구조 설계에 유용하게 사용될 수 있다.

2) 하이퍼큐브 컴퓨터

1983년 캘리포니아 공과 대학에서는 submicron 시스템 아키텍춰 프로젝트[29]를 통하여 Cosmic Cube[28] 라는 프로토타입 메시지 전송 병행 컴퓨터를 개발하였다. Cosmic Cube는 상용 마이크로 프로세서를 사용한 64개의 노드 프로세서로 구성되었으며 각 노드 프로세서는 다른 6개의 노드 프로세서와 포인트-포인트 연결 방식에 의해 연결되어 6-차원 하이퍼큐브 연결망을 구성하였다. 그림 9에 3-차원 하이퍼큐브 컴퓨터의 구조를 나타내었다.

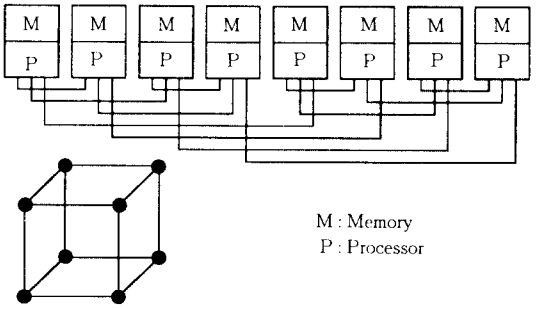


그림 9. 3차원 하이퍼큐브 컴퓨터의 구조

1985년에는 Intel iPSC / 1[17], Ametek System / 14[1], NCUBE / 10[15], [22], FPS T Series[14]등의 Cosmic Cube를 상용화한 시스템들이 개발되었는데 이러한 시스템들은 모두 하이퍼큐브 연결망으로 연결되었기 때문에 하이퍼큐브 컴퓨터라는 이름으로 널리 알려지게 되었으며 이들은 1세대 멀티컴퓨터라고도 한다.[3] 그림 10에 Intel iPSC / 1의 구조를 나타내었다.

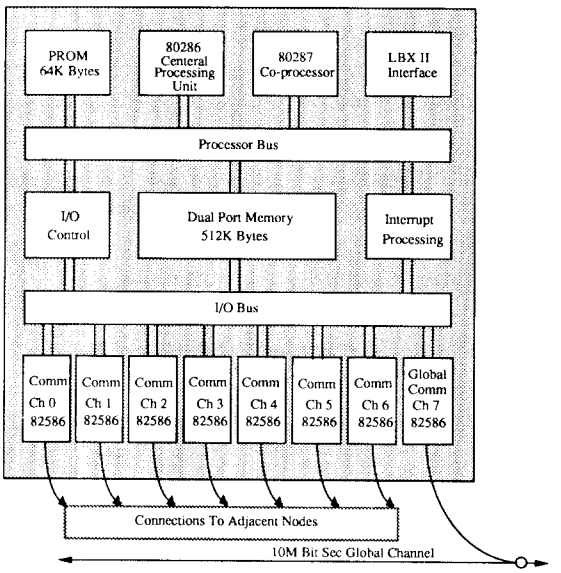


그림 10. Intel iPSC / 1의 구조

이후 메시지가 전송될 때 하드웨어에 의해 라우팅(routing)을 수행하는 방법이 제안되었는데 이 방법에서는 각 패킷은 flit(flow control digit)이라는 작은 단위로 나누어지고 각 flit은 한 노드에 도착 즉시 다른 노드로 전송됨으로써 메시지 전송의 파이프라인 수행이 가능하게 되었으며 라우팅 시간은 메시지 길이와 거쳐가는 노드 수의 곱이 아닌 합에 비례하도록 함으로써 메시지 지

연 시간을 크게 단축하였다. 뒤 따라가는 flit들은 헤드 flit을 줄줄이 따라가게 되어 마치 벌레가 꿈틀거리며 기어가는 모습을 연상시키기 때문에 이러한 라우팅을 "웜홀(wormhole)" 라우팅이라고 한다. 또한 하나의 실제 채널을 두 개의 가상 채널[9]로 나누거나 전체 네트워크를 여러 개의 가상 네트워크로 나눔으로써 이러한 라우팅 방법이 교착 상태(deadlock)에 빠지는 일이 없도록 하는 방법도 개발되었다. 그리고 같은 VLSI 집적 기술을 사용하였을 때 하이퍼큐브 보다는 토러스(torus)나 메쉬(mesh)같은 구조가 더 높은 성능을 나타낸다는 연구 결과도 있었다.[9]

고성능 마이크로 프로세서의 개발과 함께 1세대 멀티컴퓨터들의 단점을 보완, 개선한 2세대 멀티컴퓨터가 등장하게 되었는 데 Intel에서는 32-비트 마이크로 프로세서인 80386/80387을 사용한 iPSC/2[18]와 RISC 마이크로 프로세서인 i860을 사용한 iPSC/860[19]을 개발하였으며, Ametek에서는 모토롤라의 68020 32-비트 마이크로 프로세서를 사용한 시리즈 2010[29]을 발표하였다. 또한 NCUBE에서는 개선된 custom IC를 사용한 Ncube-2[23]를 발표하였다. iPSC/2, iPSC/860 및 Ncube-2는 웜홀 메시지 라우팅 하드웨어와 하이퍼큐브 구조를 사용하며 Ametek 시리즈 2010은 2차원 웜홀 라우팅 메쉬 구조를 채택하였다.

이러한 멀티컴퓨터는 발전되는 기술을 쉽게 적용시킬 수 있다는 장점이 있으며 1990년대에 VLSI 기술이 더욱 발전하면 노드 프로세서와 통신 채널의 성능이 더욱 높아져서 전체 성능이 더욱 높아지리라 예상된다.

2. 기타 메시지 전송 MIMD 컴퓨터의 구조

1) Transputer array

Transputer는 Hoare의 CSP(communicating sequential processes)에 기초를 둔 언어인 OCCAM을 수행 모델로 하여 영국의 Inmos사에서 개발된 단일 칩 프로세서로서 16-bit 프로세서인 T212, 32-bit 프로세서인 T414, 부동 소수점 연산 기능을 갖춘 T800등 다양한 모델을 지원하며, T800[16]의 경우 2.25 MFLOPS와 15 MIPS의 계산 능력과 4Kbytes의 on-chip SRAM, 그리고 20 Mbps의 4개의 시리얼 채널(입출력 도합 8개)을 단일 칩에 내장하고 있다.

Transputer는 현재 가장 낮은 가격으로 손쉽게 높은 성능을 얻을 수 있는 방법 중의 하나이다. Transputer는 다중 컴퓨터를 구축하기 위한 building block으로 사용되며 이 경우 1에서 4 MBytes 정도의 지역 메모리와 함께 transputer module(TRAM)을 구성하고 이 TRAM을 transputer의 채널을 통하여 point-to-point 방식으로 연결함으로써 다양한 형태의 연결망을 구축할 수 있다.

Transputer는 특히 유럽 각국의 집중적인 지원을 바탕으로 그 응용 분야를 크게 넓히고 있으며 많은 회사에서 transputer 관련 제품을 생산하고 있다. 프로그래밍 언어도 Fortran, C, Pascal, Prolog, Smalltalk, Ada, Modular-2 등 다양하게 지원되고 운영 체제도 Helios, Express 등 여러 가지가 개발되어 있다. 현재 상용화된 transputer system으로는 PC, VMEbus, Sun workstation 등을 위한 add-on board 등과 시스템으로 Parsytec의 Super Cluster(crossbar switch 사용)[24]등이 있다. 그림 11에 64-노드를 갖는 Super Cluster의 구조를 나타내었다.

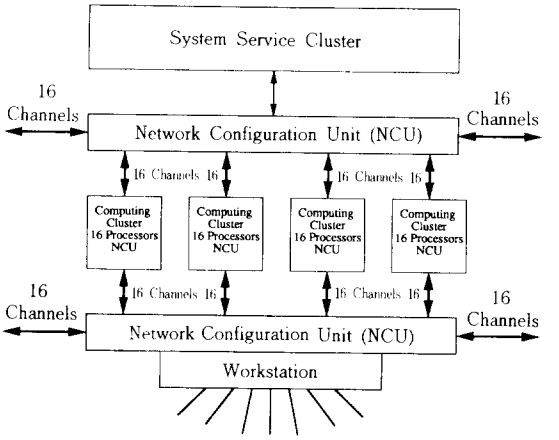


그림 11. 64-노드 SuperCluster의 구조

2) CMU iWarp

iWarp[4,5,21]는 인텔사의 지원을 받아 카네기 멜론 대학에서 개발한 단일 칩 프로세서로서 20 MFLOPS의 계산 능력과 40 MBytes/sec의 통신 채널을 8개(입력 4개, 출력 4개) 갖추고, transputer와 마찬가지로 다중 프로세서 building block으로 사용된다. 그림 12에 나타낸 것과 같이 iWarp 프로세서는 지역 메모리와 함께 iWarp 셀(cell)을 구성하며 이 iWarp cell은 다시 iWarp 시스템을 구성한다.

iWarp는 통신 단위에 따라 하나의 메시지를 기본 단위로 하는 메시지 전송 방식과 워드 단위의 시스톨릭 통신의 두 가지 통신 모델을 지원하며 메시지 전송에는 웜홀 라우팅 방식을 사용한다. iWarp는 고성능의 계산 능력과 높은 I/O 대역폭을 요구하는 특수 목적의 배열처리기와 다양한 프로그래밍을 지원하기 위한 범용 배열처리기를 모두 지원할 수 있으며 iWarp를 사용하여 다

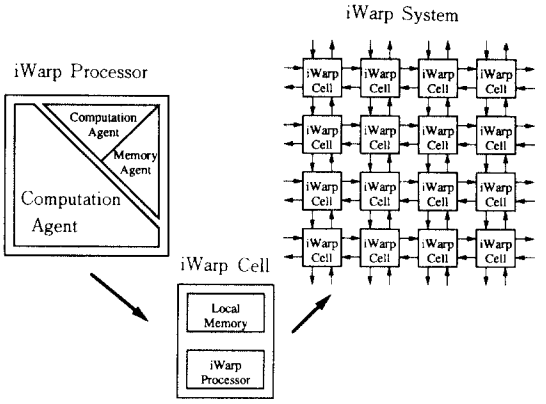


그림 12. CMU iWarp의 구조

양한 형태의 연결망을 구축할 수 있다.

iWarp는 과학 계산에서 신호 처리까지 다양한 응용 분야를 지원하기 위한 fine-grain 및 coarse-grain의 통신 기능, 단일 프로세서로서도 손색이 없는 계산 능력, 그리고 현재 개발중인 병렬화 컴파일러[20]의 구비 등으로 미래의 수퍼컴퓨터를 위한 building block으로 기대를 모으고 있다.

V. 결 론

병렬 처리 컴퓨터는 싼 값에 높은 성능을 얻거나, 단일 프로세서나 현재의 벡터 수퍼컴퓨터 등으로는 도달할 수 없는 고성능을 달성하기 위해서 필수적으로 연구되어야 할 분야이다. 아직 해결해야 될 문제는 많이 있으나 현재의 상태로도 연 2억 달러의 세계시장이 있다고 보고되고 있으며 이는 해마다 증가하고 있는 추세이다. 또한 앞으로의 수퍼컴퓨터로 teraflops(10^{12} FLOPS)의 성능을 갖는 시스템을 개발하기 위하여 여러 회사들이 대규모 병렬 컴퓨터(massively parallel computer)를 생각하고 연구 중에 있는 것으로 보고되고 있으며 Cray사도 이러한 계획을 발표하였다.[27]

그러나 이러한 시스템들이 널리 사용되기 위해서는 해결되어야 할 문제들이 아직도 많다. 현재의 단일 프로세서 시스템이나, 벡터 수퍼컴퓨터, 네트워크, 그리고 고성능 워크스테이션의 조합은 여러 가지 목적으로 사용될 수 있으며, 또 사용하기 편리한 프로그래밍 환경을 제공하는데 비해 SIMD형 컴퓨터는 그 응용 범위가 제한된다는 문제점이 있으며, MIMD 컴퓨터는 아직 여러 가지 소프트웨어의 미비로 사용하기가 어렵고, 발표되는 최고치 성능을 유지하기가 매우 어렵다는 문제가 있다. 또한 널리 사용되어 거의 표준화 되다 시피한 보편화된 구조나 소프트웨어들이 없고 각 회사나 시스템마다 독특한 구조와 소프트웨어를 사용한다는 것이 매우 큰 문제가 되고 있다. 따라서 앞으로 어느 것이 보편화 될 것인지를 모르는 환경에 처한 사용자들에게는 병렬 처리 컴퓨터를 구입, 사용하기가 그리 쉬운 일이 아니다. 이러한 문제를 해결하기 위한 시도의 일환으로 하부의 병렬 처리 컴퓨터 구조와 상관 없이 병렬 프로그램을 개발하는 방법들도 연구되고 있으며, 이런 연구에 큰 성과가 있을 경우 병렬 처리 시스템을 사용하기가 훨씬 쉬워질 것으로 기대된다.

또한 기존의 프로그램을 변환 수행 시킬 수 있는 병렬화 컴파일러의 개발도 중요한 문제이며, 시분할이 아닌 다중 사용자 환경 제공, 편리한 프로그래밍 환경 제공 등 소프트웨어의 개발이 중요한 문제로 부각된다.

參 考 文 獻

[1] Ametek Inc., System / 14 brochures.

[2] J. Archibald and J.-L. Baer, "Cache coherence protocols : evaluation using a multiprocessor simulation model," *ACM Transacions on Computer Systems*, vol. 4, no. 4, pp. 273-298, November 1986.

[3] W. C. Athas and C. L. Seitz "Multicomputers : message-passing concurrent computers," *IEEE Computer*, vol. 21, no. 8, pp. 9-24, August 1988.

[4] S. Borkar, et. al., "iWarp : An Integrated Solution to High-Speed Parallel Computing," *Proc. of Supercomputing '88*, pp. 330-339, 1988.

[5] S. Borkar, et. al., "Supporting Systolic and Memory Communication in iWarp," *Proc. of 17th Annual International Symposium on Computer Architecture*, IEEE Computer Society, pp. 70-81, June 1990.

[6] Corollary, 486 / smp brochures.

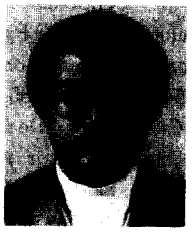
[7] W. Crowther, et. al., "Performance Measurements on a 128-Node Butterfly Parallel Processor," *Proc. of 1985 International Conference on Parallel Processing*, pp. 531-540, August 1985.

[8] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547-553, May 1987.

[9] W. J. Dally, *A VLSI Architecture for Concurrent Data Structures*, Kluwer Acadmic Publishers, 1987.

[10] J.J. Dongarra and I.S. Duff, "Advanced Architecture Computers," Technical Report CS-89-90, Computer Science Department, University of Tennessee, November 1989.
[11] G. Fielland and D. Rodgers, "32-Bit computer system shares load equally among up to 12 processors," *Electionic Design*, pp. 153-168, September 6, 1984.
[12] E. F. Gehringer, J. Abullarade, and M. H. Gulyn, "A survey of commercial parallel processors," *Computer Architecture News*, vol. 16, no. 4, pp. 75-107, September 1988.
[13] A. Gottlieb, et. al., "The NYU Ultracomputer-Designing an MIMD shared memory parallel computer," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 175-189, February 1983.
[14] J. L. Gustafson, et. al., "The Architecture of a Homogeneous Vector Super-computer," *Proc. of the 1986 International Conference on Parallel Processing*, IEEE Computer Society, pp. 653-660, August 1986.
[15] J. P. Hayes, T. Mudge, and Q. F. Stout, "A microprocessor-based hyper-cube supercomputer," *IEEE Micro.*, vol. 6, no. 5, pp. 6-17, October 1986.
[16] Inmos Ltd., "IMS T800 Architecture," Technical Note 6, 1987.
[17] Intel Scientific Computers, *iPSC User's Guide*, Order Number. 175455-004, April 1986.
[18] Intel Scientific Computers, iPSC / 2 brochures.
[19] Intel Scientific Computers, iPSC / 860 brochures.
[20] C. Lengauer, "On the Projection Problem in Systolic Design," Technical Report CMU-CS-88-102, Computer Science Dept., Carnegie-Mellon University, February 1988.
[21] H.G. Mayer and B. Baxter, "Software and Hardware Parallelism on the iWarp Multi-Computer," *Proc. of 1991 International Conference on Supercomputing*, Cologne, Germany, pp. 224-233, June 1991.
[22] NCUBE Corporation, NCUBE / ten brochures.
[23] NCUBE Corporation, NCUBE 2 brochures.
[24] Paracom, Transputer products brouchures.
[25] G.F. Pfister, et. al., "The IBM Research Parallel Processor Prototype(RP3) : Introduction and Architecture," *Proc. of 1985 International Conference on Parallel Processing*, pp. 764-771, August 1985.
[26] R. Rettberg and R. Thomas, "Contention is No Obstacle to Shared-Memory Multiprocessing," *Communications of the ACM*, vol. 29, no. 12, pp. 1202-1212, December 1986.
[27] J. Sanders and A. Mitchel, "Dataline 1995!," *Parallelogram International*, pp. 8-9, November 1990.
[28] C. L. Seitz, "The cosmic cube," *Communincations on ACM*, vol. 28, no. 1, pp. 22-33, January 1985.
[29] C. L. Seitz, et al., *Submicron Systems Architecture Project Semiannual Technical Report*, Computer Science Technical Report CS-TR-88-5, California Institute of Technology, April 1988.
[30] S. Thakkar, et. al., "The Balance Multiprocessor System," *IEEE Micro.*, pp. 57-69, February 1988.
[31] P.C. Treleaven, "Control-driven, data-driven and demand-driven computer architecture(Abstract)," *Parallel Computing*, vol. 2, 1985.

筆 者 紹 介



孟 承 烈

1951年 1月 15日生
1977年 2月 서울대학교 공과대학 전자공학과(학사)
1979年 2月 한국과학기술원 전산학과(석사)
1984年 2月 한국과학기술원 전산학과(박사)
1984年 3月～1989年 9月 한국과학기술원 전산학과 조교수
1988年 3月～1989年 2月 Univ. of Pennsylvania 교환교수
1989年 9月～현재 한국과학기술원 전산학과 부교수
주관심 분야 : 병렬처리 구조, 데이타 흐름 컴퓨터, 병렬처리 알고리즘, 멀티미디어