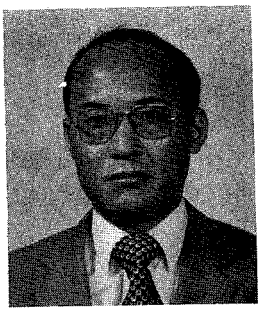
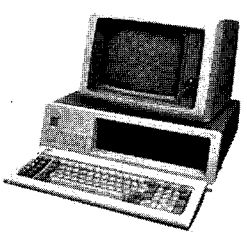


吳 吉 祿
韓國電子通信研究所
컴퓨터연구부장 / 工博

멀티프로세서 컴퓨터 시스템



Ⅰ. 序 論

1960년에 Burroughs社에서 4개의 프로세서를 사용한 최초의 멀티프로세서 컴퓨터 시스템 D-825를 발표한 이래 많은 멀티프로세서 시스템이 개발되었고, 특히 최근에 VLSI 기술의 발전으로 고수준의 처리능력을 갖춘 마이크로 프로세서와 대용량의 메모리 칩을 사용한 저가격, 고성능의 멀티프로세서 시스템을 구현할 수 있게 되었다.

멀티프로세서 컴퓨터 시스템이란 Throughput, Reliability, Flexibility, Availability를 증가시키기 위하여, 비슷한 처리능력을 갖는 복수개의 프로세서가 메모리 모듈, I/O채널, 주변장치들을 공유하면서, 단일 Integrated O. S.에 의해서 제어되는 시스템이다. 공유 메모리(Shared memory)와 I/O장치 외에 각 프로세서는 로칼 메모리(Local memory)를 가질 수 있으므로, 프로세서간 통신은 공유메모리 또는 인터럽트 네트워크를 통해서 실현될 수 있다.

본고에서는 멀티프로세서 시스템의 두가지 분류, 즉 Tightly-coupled와 Loosely-coupled 시스템에 대해서 Ⅱ절에서 살펴보고, Ⅲ절에서는 멀티프로세서 시스템의 성능을 좌우하는 프로세서·메모리간의 Interconnection network을, Ⅳ절에서는 병렬 메모리를 구성할 때의 문제점을, Ⅴ절에서는 병렬 O. S.와 관련된 Partitioning 문제, Scheduling 문제, Synchronization 문제에 대하여 기술하였다.

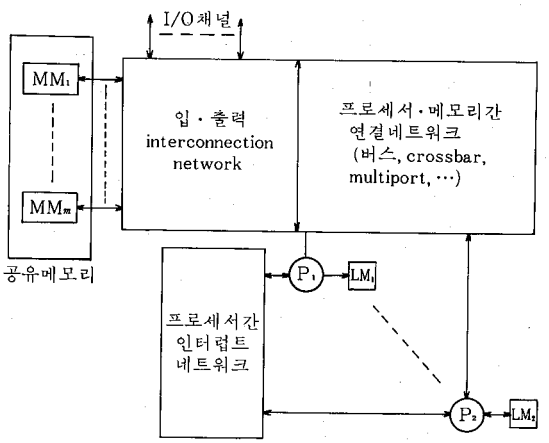
마지막으로 Ⅵ절에서는 결론을 겸하여 앞으로의 연구방향과 추세에 대해서 기술하였다.

Ⅱ. 시스템 구조

그림 1에 MIMD (multiple instruction multiple

data stream) 멀티프로세서 컴퓨터 시스템의 구성도를 나타내고 있다.

이러한 멀티프로세서 시스템은 두 가지의 구조적 모델로 분류되는데 Tightly-coupled 멀티



(MM : 메모리 모듈, LM : 로칼 메모리, P : 프로세서)

그림 1. MIMD 멀티프로세서 컴퓨터시스템의 구성도[1]

프로세서는 공유 메모리를 통하여 통신하므로 시스템의 성능이 메모리의 대역폭(Bandwidth)에 크게 영향을 받는다. 반면 Loosely-coupled 멀티프로세서는 대부분의 명령과 데이타가 로칼 메모리 內에 저장되어 있으므로 태스크간의 통신이 많지 않은 경우에 효율적이다.

1. Tightly-coupled 멀티프로세서

고속 처리나 실시간 처리에 적합한 구조로, 프로세서와 메모리간에 Interconnection network을 사용하거나 Multi-port메모리를 사용한다. 또한 성능이 메모리의 대역 폭에 크게 영향을 받으므로 각 프로세서는 작은 로칼 메모리나 cache를 가지고 있다.

이러한 Tightly-coupled 시스템에서는 두개 이상의 프로세서가 동시에 하나의 메모리를 액세스하고자 할 때 발생하는 메모리 경쟁(Memory contention)문제를 해결하기 위하여 메모리 Interleaving기법을 사용한다. 또한 Tightly-coupled 시스템은 O.S.를 프로세서에 분산시키는 방법에 따라서 다음의 두가지로 분류할 수 있다.

(1) Master-slave (asymmetric)

Cyber-170, DEC system10과 같이 하나의 Master 프로세서에서 시스템내의 모든 프로세서의 상태를 관리하고 모든 Slave 프로세서에 일을 할당한다. 따라서 일이 User-mode intensive한 경우에는 효율적이지만, Slave의 수를 증가시켜 O.S.의 사용이 증가되면 Bottleneck이 발생하고, Master프로세서가 고장나면 전체 시스템이 동작하지 않는다는 문제점을 가지고 있다.

(2) Symmetric

프로세서들간에 구분이 없으며 모든 프로세서가 O.S.코드를 동작시킬 수 있다. 모든 프로세서가 시스템 서비스를 수행할 수 있으므로 시스템 Call의 경우에도 Context 스위치가 필요없으며, 인터럽트 취급이 프로세서간에 분산되어 있으므로 하나의 특정한 프로세서에 과부하가 걸리는 것을 피할 수 있다. 또한 프로세서가 고장나더라도 성능의 감소는 있지만 계속 동작할 수 있고, 여러개의 프로세서를 유지할 수 있으므로 Master-slave의 경우에 비해서 효율적이다.

2. Loosely-coupled 멀티프로세서

일반적으로 Loosely-coupled 시스템은 Tightly-coupled 시스템의 경우에 흔히 발생하는 메모리 혼잡문제가 발생하지 않는다. 프로세서와 로칼 메모리, I/O 인터페이스를 컴퓨터 모듈이라고 했을 때, 다른 컴퓨터 모듈상에서 수행되는 프로세스(Process)들은 메시지 전달시스템을 통하여 메시지를 교환함으로써 서로 통신한다.

Ⅲ. Interconnection network

멀티프로세서 컴퓨터 시스템에서 가장 중요한 요소중의 하나는 프로세서 메모리간의 Interconnection network으로, 크게 버스를 사용한 구성과 버스를 사용하지 않은 구성으로 분류할 수 있다.

1. Time-shared 버스 구조

IBM370/168과 같이 프로세서와 메모리간을 버스로 연결하면 버스 Bandwidth에서 Bottleneck이 발생하여, 4개 이상의 프로세서를 사용하면 시스템의 성능(Performance)이 급격히 떨어진다는 문제점을 갖고 있다((그림 2 (a)).

2. Crossbar 스위치 구조

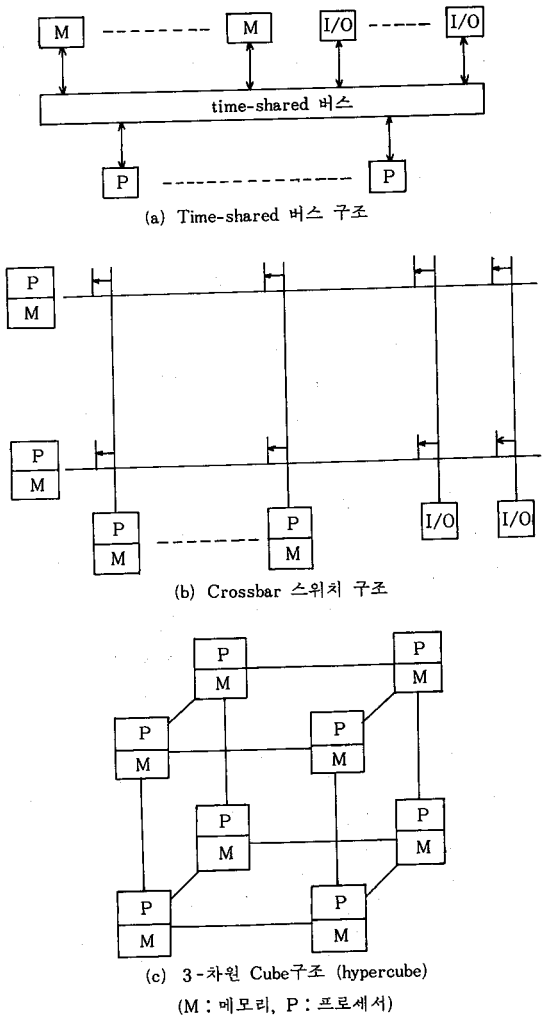


그림 2. Interconnection network topology[2]

버스의 문제점을 해결하기 위하여 C. mmp에서 처럼 Crossbar 스위치를 사용하면 모든 Communication을 가능하게 해주지만, 가격이 비싸고 효율(Efficiency)이 떨어질 뿐만 아니라 시스템의 확장성(Expandibility)이 작다는 문제점을 갖고 있다. 특히 핀수의 제한때문에 VLSI化 될 수 없다는 단점을 가지고 있다.(그림2(b)).

3. n-차원 Cube구조(Hypercube)

버스를 사용하지 않은 대표적인 구조로 2^n개의 프로세서·메모리 쌍을 연결할 수 있으며, 두 노드간의 최대통신거리는 n이 된다. 특히 노드간에 복수개의 Interconnection이 존재하여 Fault tolerant가 가능하다는 특징을 가지고 있다(그림 2 (c)). 최근에 Hypercube를 이용한 商用 시스템이 발표되었는데, Intel社의 IPSC, Ametek社의 System14, Ncube社의 Ncube/ten이 그것이다.

이외에 Baseline network, Omega network, Data manipulator 등 수많은 Interconnection topology가 대학교 차원에서 개발되었지만 아직 商用化되지는 않았다.[3] 한편 이러한 Interconnection network을 구성하는데에는, Circuit 스위치 소자 또는 Packet 스위치 소자 중 어느 것을 이용할 것인가를 결정하는 스위치 방법론 문제, 집중제어냐 분산제어냐를 결정하는 제어 전략문제, 동기 또는 비동기 동작을 결정하는 동작모드 문제 등을 시스템 설계자가 적절히 선택해야 한다.

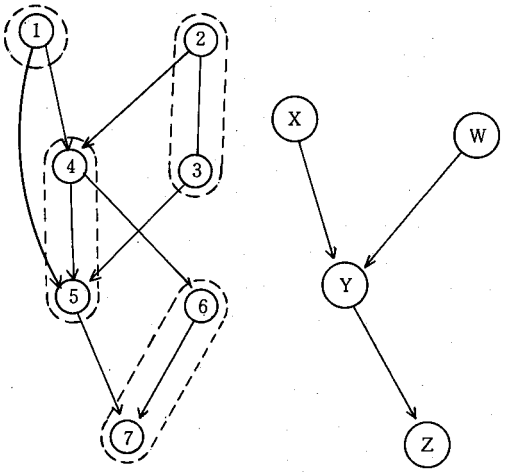
Ⅳ. 메모리 구성

프로세서의 수가 증가할수록 메모리 Conflict에 의한 유효시스템 대역폭은 급격히 감소한다. 반면 메모리와 시스템 대역폭을 증가시키기 위하여 병렬 메모리 유니트의 수를 증가시키면, 대역폭의 유효증가정도(Effective rate of growth)가 문제가 된다. 이러한 메모리 Latency 문제를 해결하기 위해서는 Cache 메모리를 사용해야 하는데, 대부분의 Tightly-coupled 시스템에서는 복수개의 Private cache가 존재하므로 Cache coherence문제(Multicache consistency 문제)를 고려해야 한다.[4]

Ⅴ. 멀티프로세서 시스템에서의 고려사항

1. Partitioning 문제

Parallelism과 Scheduling & synchronization overhead는 시스템의 성능에 영향을 주는 주요한 두가지 요소이다. 예를 들어 그림3 (a)는 7개의 노드와 9개의 에지를 가지는 Fine-granularity 그래프이다. 이 경우에는 각 노드를 Schedule하는 시간과 각 에지를 Synchronize시키는 시간이 추가되므로 그림3 (b)의 경우보다 시간이 많이 걸리지만, 그림3 (b)의 경우보다 더 많은 Parallelism을 추출할 수 있다. 즉 노드6과 7이 새로운 노드 Z로 합쳐졌을때 노드 Z는 노



(a) Fine-granularity (b) Crude-granularity

그림 3. 제어그래프[5]

드 Y 다음에 수행되므로, 노드5와 6은 순차적으로 수행될 수밖에 없다.

즉, Partitioning 문제란 프로그램을 태스크들로 분할하는 것으로, 수행속도를 최대로 하기 위하여 프로그램內의 모든 가능한 Parallelism을 추출하는 Parallelism detection과 머신의 Throughput을 향상시키기 위하여 여러 오퍼레이션을 하나의 태스크로 결합하던가, 하나의 프로그램을 여러 개의 태스크로 분할하는 Clustering문제로 세분화된다.

Parallelism detection 문제는 크게 3가지 경우로 나눌 수 있는데, 첫째로 Parallel 태스크와 그들간의 데이타 통신을 정의할 수 있는 Occam[7]과 같은 언어를 사용하여 알고리즘을 설계할 때 사용자에 의해서 Parallelism이 추출될 수 있고, 두번째로 Bulldog[8]과 같은 컴파일러에 의해서 순차프로그램을 멀티프로세서에 적합한 병렬형식으로 변환할 수도 있고, 마지막으로 CDC6600과 같은 머신에 의해서 Parallelism이 추출될 수도 있다.

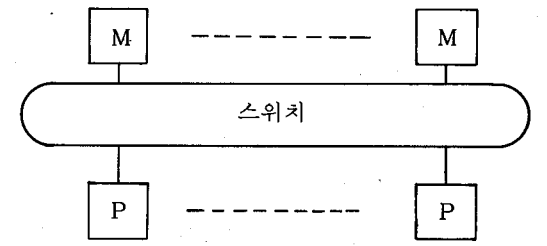
2. Scheduling문제

단일 프로그램의 높은 Performance나 멀티프로그래밍 환경에서의 높은 프로세서 Utilization을 얻기 위하여 각 태스크를 하나 또는 다수의 프로세서에 할당하는 것으로, Static scheduling과 Dynamic scheduling이 있다.

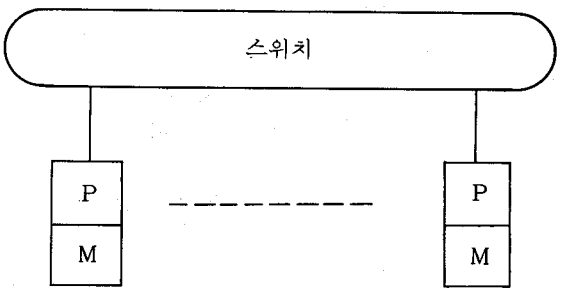
Static scheduling이란 태스크가 알고리즘을 설계할 때나 컴파일시에 프로세서에 할당되는 것이고, Dynamic scheduling은 Run-time에서 할당되는 것이므로, Dynanamic scheduling이 높은 프로세서 Utilization을 제공하지만, 추가적인 Scheduling시간을 필요로 한다는 문제점이 있다.

3. Synchronization문제

공유된 자원(Resource)이 한 순간에 하나의 프로세스(Process)에 의해서만 액세스되는 경우에 자원관리(Resource management)에 관계되는 문제로써, Shared variable(그림 4 (a))이나 Message passing(그림 4 (b))과 같은 프로세스간 통신방식 각각에 대해서 Mutual exclusion과 Condition synchronization 방법이 있다.



(a) Shared variable을 사용한 경우



(b) Message passing 방식을 사용한 경우

그림 4. 프로세스간 통신방식[5]

Ⅵ. 結 論

VLSI 집적기술이 발전함에 따라서 병렬 알고리즘을 직접 하드웨어로 실현하는 VLSI Architecture[9] 신뢰도와 성능을 개선시키고 단가를 절감하기 위하여 Full-slice기법을 이용한 WSI (Wafer Scale Integration) 시스템[10], IC 층을 종(縱) 방향으로도 겹쳐쌓음으로써 구조적으로

병렬처리가 간단히 될 수 있는 3차원 소자[11] 등의 Technology에 관한 연구와 머신 명령이 간단하고 대용량의 고속 레지스터를 이용하여 메모리 액세스를 감소시킴으로써 고속 동작이 가능한 RISC프로세서[12]를 멀티프로세서 컴퓨터에 이용하는 방법과 명령의 오퍼란드가 다 갖추어지면 다른 명령과는 무관하게 실행되는 Dataflow 컴퓨터[13]와 어떤 명령을 실행하려고 할 때 그것에 필요한 오퍼란드를 생성하는 다른 명령에 요구(Demand)가 전달되는 Reduction머신[14] 등과 같은 새로운 Architecture에 관한 연구가 진행되고 있다. 또한 Connection machine과 같이 수치 처리뿐만 아니라 기호 처리까지도 할 수 있는 멀티프로세서 시스템이 현재 연구중에 있다.

우리 나라에서도 이러한 멀티프로세서 시스템에 관한 필요성이 인식되면서 몇개의 대학과 연구소에서 연구가 시작되고 있으며, 대학, 연구소, 기업이 합심하여 이 분야에 전념한다면 좋은 성과가 나오리라고 기대된다.

參 考 文 獻

[1] K. Hwang and F. A. Briggs, "Computer Architecture and Parallel Processing," McGraw-Hill, 1984.

[2] J. Miklosko and V. E. Kotov, "Algorithms, Software and Hardware of Parallel Computers," Springer-Verlag, 1984.

[3] H. J. Siegel, "Interconnection networks for large-scale Parallel Processing," Lexington Books, 1985.

[4] M. Dubois and F. A. Briggs, "Effects of cache coherency in Multiprocessors," IEEE Trans. Computers, Nov., 1982, pp. 1083-1099.

[5] D. D. Gajski and J. K. Peir, "Essential issues in Multiprocessor Systems," IEEE Computer, June, 1985, pp. 9-27.

[6] P. C. Patton, "Multiprocessors : Architecture and Applications," IEEE Computer, June, 1985, pp. 29-40.

[7] P. Wilson, "OCCAM Architecture eases system design-Part I," Computer Design, Nov. 1983, pp. 107-115.

[8] J. A. Fisher, "Very Long Instruction Word Architectures and the ELI 512," Proc. 10th Symp. Computer Architecture, 1983, pp. 140-150.

[9] C. L. Seitz, "Concurrent VLSI Architectures," IEEE Trans. Computers, Dec., 1984, pp. 1247-1265.

[10] R. L. Petritz, "Current Status of Large Scale Integration technology," IEEE J. of Solid-State Circuits, Dec. 1967, pp. 130-147.

[11] J. Grinberg, et. al., "A cellular VLSI Architecture," IEEE Computer, Jan., 1984, pp. 69-81.

[12] D. A. Patterson and C. H. Séquin, "A VLSI RISC," IEEE Computer, Sept., 1982, pp. 8-21.

[13] J. B. Dennis, "Dataflow Super Computer," IEEE Computer, Nov., 1980, pp. 48-56.

[14] R. M. Keller and F. C. H. Lin, "Simulated Performance of a Reduction-Based Multiprocessor," IEEE Computer, July, 1984, pp. 70-82. *

생활속에 과학있고 과학속에 번영있다