密結合 멀티프로세서 시스템의 具現 및 性能評價

(Implementation and Performance Evaluation of a Tightly-Coupled Multiprocessor System)

金 悳 鎭*, 金 永 川**, 朴 石 千*

(Duck Jin Kim, Young Chon Kim and Seok Cheon Park)

要 約

本 論文에서는 16비트 MC68000 CPU를 갖는 4대의 PE(processing element)와 共通 메모리 128KB, 그리고 이들을 상호연결하기 위한 한 개의의 時分割 버스를 利用하여 密結合 멀티프로세서 시스템을 具現하였다.

시스템의 運營體制는 MTOS를 修正 移植하여 멀티태스킹 및 멀티프로세싱이 가능하도록 하므로써 並列處理가 가능하도록 하였다.

시스템의 하드웨어 性能을 評價하기 위하여 stochastic petri net(SPN)을 이용한 시스템의 모델링과 시스템 負荷率에 따른 processing power(P) 및 각 프로세서의 效率을 최대 16대의 PE를 사용하는 경우까지 시뮬레이션 하였다.

마지막으로 並列處理가 가능한 quicksort, FFT, 行列 곱셈등의 benchmark 프로그램을 적용하여本 시스템에서의 處理速度 및 速度改善을 比較檢討하였다.

Abstract

In this paper, a tightly-coupled multiprocessor system is implemented with four processing elements based on MC68000 CPU, a common memory (128KB), and a single time-shared bus.

The multi-tasking operating system, MTOS, is modified so that the multiprocessor system can support multitasking and multiprocessing.

The performance of the proposed system is evaluated by stochastic Petri Net system modeling. The efficiency and the processing power are simulated for various load factors and up to 16 PEs.

By running benchmark programs, such as quicksort, FFT, and matrix-multiplication, the speed of parallel processing is compared with that of a single processor.

*正會員, 高麗大學校 電子工學科
(Dept. of Elec. Eng., Korea Univ.)

**正會員, 全北大學校 電子計算機工學科
(Dept. of Comp. Eng., Chonbuk Nat'l Univ.)

接受日字 : 1987年 4月 14日

(※ 本 研究는 1986年度 高麗大學校 校內 研究費에 依하여 遂行된 것임.)

I. 序 論

最近 VLSI技術의 발달로 인하여 컴퓨터 하드웨어 특히 프로세서 및 메모리 등의 가격이 저렴하게 되었으나, 現在 많이 사용되고 있는 단일 프로세서를 이용한 Von Neumann 구조의 컴퓨터 시스템으로는 速度改善에 한계가 있어 高速 實時間處理를 要求할 경우에는 多重處理 시스템의 構成이 不可避하다.[1]

멀티프로세서 시스템은 速度向上과 價格대 性能比에 있어서 單一 프로세서 시스템보다 好評을 받고 있다.[1,2,3,4]

멀티프로세서 시스템은 지금까지 汎用 컴퓨터보다는 특수한 목적으로 많이 구성되고 있으며 응용 분야로는 영상신호처리등 매우 다양하다.

현재 개발된 멀티프로세서 시스템의 예로는 카네기 멜론대학에서 개발한 C.mmp, C.m*를 비롯하여 ILLIAC IV, S-1, HEP, IBM/168MP 및 Cray X-MP등이 있으나 實時間 處理를 요하고 업무가 한정된 산업응용 분야에 이와 같은 대규모 시스템을 사용하는 것은 경제성이 맞지 않는다.

따라서 소수의 프로세서를 이용한 실험적인 멀티프로세서 시스템들이 많은 연구소 및 대학에서 연구 개발중에 있으며, 특히 Torino의 Politecnico에서 개발된 TOMP(torino multiprocessor)[5]는 좋은 例이다. TOMP는 3개의 Z-8001 프로세서, 한 개의 時分割 버스와 공통 메모리로 구성되어 있다.

국내에서는 1985년에 高麗大學校에서 VME/10 개발 시스템과 MC68000 CPU를 갖는 모노보드 컴퓨터 3대를 이용하여 마스터-슬레이브 구조의 멀티프로세서 시스템이 구성된 바 있다.[6] 이런 마스터-슬레이브 구조의 시스템은 구현이 쉽고 기존의 다중 프로그래밍 시스템을 확장하여 만들 수 있는 장점은 있으나 슬레이브 프로세서에서 실행되던 태스크가 운영체제의 介入을 필요로 할 때는 마스터에게 인터럽트를 요구해야 하므로 프로세서간의 통신이 증가하게 되며, 슬레이브 프로세서의 수가 많아질수록 시스템 프로그램의 서비스를 요구하는 태스크가 증가하므로 시스템 버스에 병목현상을 일으켜 시스템의 性能을 低下시킨다. 또 마스터 프로세서의 故障은 전체 시스템에 致命的인 영향을 미친다. 또한 마스터 프로세서가 슬레이브 프로세서의 요구에 신속히 응답하지 못하면 슬레이브 프로세서의 활용도가 낮아져 프로세서 전체의 效率이 떨어진다.

本 論文에서는 高速 實時間 處理를 요구하는 小規模 産業用 시스템에 적합하고 프로세서의 效率性 및 經濟性등을 고려하여 時分割 버스 구조를 갖는 密結合 멀티프로세서 시스템을 具現하여 性能評價와 Benchmark 프로그램을 적용하여 速度改善(speedup) 상태를 측정하였다.

Ⅱ. 密結合 멀티프로세서 시스템의 구성

컴퓨터 시스템은 密結合(tightly-coupled) 시스템과 疏結合(loosely-coupled) 시스템으로 대별되고 密結合 시스템은 共有 메모리(shared memory) 구조, 스타구조, 계층구조등이 있으며 本 論文에서 具現한 시스템은 共有메모리 구조에 속한다. 共有메모리 구조는 물리적 연결 방법에 따라 時分割 버스구조, 멀티포트 메모리 구조, 크로스바 스위치구조로 分類되며 특징은 표 1과 같다.

표 1. 멀티프로세서 하드웨어 구조에 따른 비교
Table 1. Comparison of Multiprocessor Hardware Organization

분 류	특 징
시분할 버스구조	* 가장 단순한 구조이며 저렴한 가격으로 구성이 가능하다. * 기능적 장치등을 쉽게 제거하거나 추가할 수 있다. * 어느 한계이상의 기능적 장치의 증가는 시스템의 성능을 감소시킬 수 있어 소규모 시스템에 적합하다. * 시스템의 능력은 버스 전송속도에 의해 제한되며 버스의 고장은 전체 시스템에 치명적이다.
크로스바 스위치 구조	* 가장 복잡한 구조를 가지며 스위칭 로직과 제어 로직이 스위치안에 포함되어 있기때문에 기능적 장치들은 간단해진다. * 서로 다른 메모리 모듈을 동시에 액세스하므로 총 전송률이 높으며, 좋은 시스템 효율을 이룰 수 있다. * 크로스바 스위치내의 여유와 segmentation을 통해 스위치와 시스템의 신뢰도를 증가시킬 수 있다. * 시스템의 분할이 쉽고 기능이 좋지 않은 부분은 시스템의 기능에 영향을 주지않고 쉽게 제거할 수 있다.
멀티포트 메모리 구조	* 대단히 높은 전송율을 얻을 수 있다. * 프로세서는 고유의 기억장치를 확보하여 다른 프로세서가 액세스하지 못하도록 할 수 있다. * 기억장치안에 제어회로, 스위치 회로등이 내장되어 있으므로 기억장치의 가격이 비싸다. * 시스템의 구조를 변경하기 어렵다.

표 1에서와 같이 時分割버스 구조는 구성이 간단하며 저렴한 가격으로 구성이 가능하고 소규모 시스템으로부터 필요에 따라 시스템의 확장이 용이하다는 장점이 있고 전송률이 높은 버스와 PE 자체에 로칼메모리를 둠으로써 時分割버스 使用을 爲한 경합을 줄임으로 단점을 보완하면 소수의 프로세서를 硏究하는 産業應用 시스템에서는 높은 性能向上을 이룰 수 있다.

따라서 本 論文에서는 時分割(共通) 버스 구조를 채택하였으며 各各의 PE에 자체의 로칼메모리를 두어 이를 적절히 使用하고, 傳送速度가 빠른 VMEbus[7]를 時分割 버스로 使用하였다.

本 멀티프로세서 시스템은 VME/10 開發 시스템의 後部에 있는 VMEbus 콘넥터에서 擴張 보드와 연결 케이블을 통하여 VMEbus를 rack까지 연장시켜 VME/10 開發 시스템과 멀티프로세서 시스템을 연결하였다.

VME/10에서 開發된 멀티프로세서 시스템의 運營體制

는 이 시스템에서 컴파일되고 링크되어 VMEbus를 통해 共通 메모리로 전송하여 實驗하였다. 이를 爲하여 VERSAdos의 Kernel인 RMS 68K[8]를 利用한 傳送 프로그램을 開發하여 프로그램 및 데이타를 하드디스크 또는 플로피디스크로부터 共通 메모리로 직접 傳送이 可能하도록 하였다.

1. 하드웨어의 구성

멀티프로세서 시스템은 그림 1과 같이 4대의 PE와 相互通信을 爲한 2개의 共通 메모리 모듈로 구성하였으며 各各의 PE는 로칼메모리와 독자적인 클럭을 갖고 있다. 時分割 버스로는 非同期式으로 융통성이 높고, 非멀티플렉싱 方式이기 때문에 버스라인數는 많으나 傳送速度(bandwidth : 57Mbytes/sec)가 빠른 VMEbus를 利用하였다. 이 VMEbus는 IEEE P1014標準規格으로 권고되고 있다.

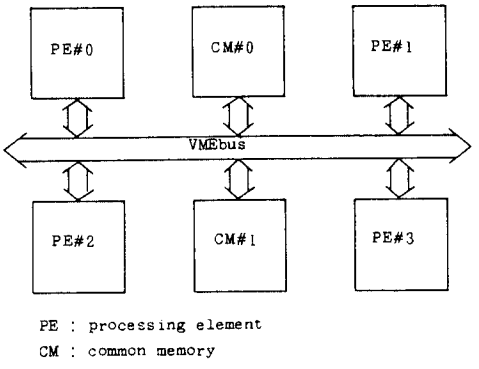


그림 1. 멀티프로세서 시스템의 블럭圖
Fig. 1. Block Diagram of Multiprocessor System.

1) Processing Element

PE는 모노보드 컴퓨터로서 16비트 MC68000 CPU를 使用하였고, 시스템 클럭은 8MHz이며, VMEbus와 연결 가능하도록 버스 requester와 VMEbus 인터페이스를 갖는다. 이 모노보드 컴퓨터는 그림 2와 같이 프로그램이 가능한 타이머를 내장하고 있으며, 이 칩에는 3개의 16비트 카운터/타이머가 있다. 이 중에서 한개의 타이머를 멀티프로세서 시스템의 RTC(real time clock)를 제공하기 爲하여 使用하였다.

로칼 메모리는 ROM 64KB, RAM 48KB이며 ROM에는 VMEbus 3.3이 수정된 모니터 프로그램(32KB)과 프로세서 색인 번호(PID)등 멀티프로세서 운영체제를 저장하였다. I/O채널은 외부확장을 위한 8비트 비동기식 메모리맵I/O와 4개의 인터럽트를 提供할 수 있고, 로칼I/O를 爲하여 RS232 포트를 포함하고 있다.

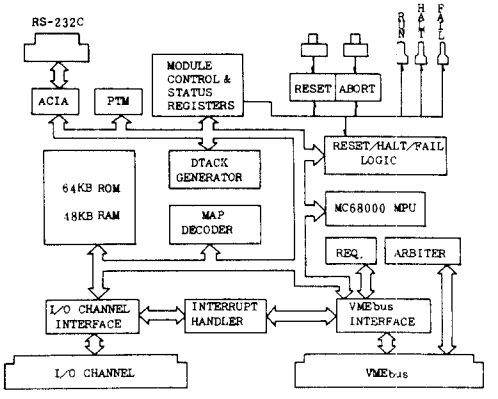


그림 2. PE의 블럭圖
Fig. 2. Block Diagram of PE.

인터럽트 기능으로는 7레벨 우선순위 인터럽트 處理機能이 로칼그룹과 共通버스그룹으로 區分된다.

共通 메모리를 엑세스 하고자 할 때 時分割 버스를 使用하기 爲한 버스 要求機能은 레벨 0에서 레벨 3까지 4개의 레벨로 構成하였으며, 레벨 3이 가장 높은 優先順位를 갖는다. PE#0은 레벨 2, PE#1은 레벨 1, PE#2와 PE#3은 레벨 0으로 하였으나 PE#2와 PE#3은 데이지 체인으로 연결하여 PE#2가 PE#3보다 높은 優先順位를 갖도록 하여 버스 要求에 對한 競合問題를 해결하였다.

버스 使用權을 획득한 후에 버스를 解除하는 方式은 4種類가 있으나 本 시스템에서는 現在의 버스 사이클과 병행하게 버스 仲裁가 可能한 RWD(release when done) 方式을 使用하였다.

2) 共通 메모리

共通 메모리 모듈은 64KB를 갖는 다이내믹 RAM으로 構成하였고 VMEbus에 연결되어 있으므로 VMEbus에 접속된 모든 프로세서로부터 데이타의 入出力이 可能하다. 어드레싱 범위는 $B00000_{16}$~$B1FFFF_{16}$까지 128KB이다.

VMEbus 인터페이스 시스템은 데이타 傳送버스(DTB), 優先順位 인터럽트버스, DTB仲裁버스, 유틸리티 等 4그룹의 버스 信號線과 裝置들을 接續하기 爲한 기능적 모듈들로 構成되며, 그림 3에 멀티프로세서 시스템의 機能的 接續狀態를 나타내었다. DTB는 데이타, 어드레스 및 制御信號等을 傳送하며, 各各의 PE는 共通 메모리에 있는 데이타를 액세스하기 爲하여 DTB를 使用한다. 各 PE는 DTB마스터를 내장하고 있으며, 共通 메모리 모듈은 DTB 슬레이브를 내장하고 있다. 여기서 마스터는 데이타를 傳送하기 爲하여 데이타 傳送버스를 制御할 수 있으며, 슬레이브는 마스

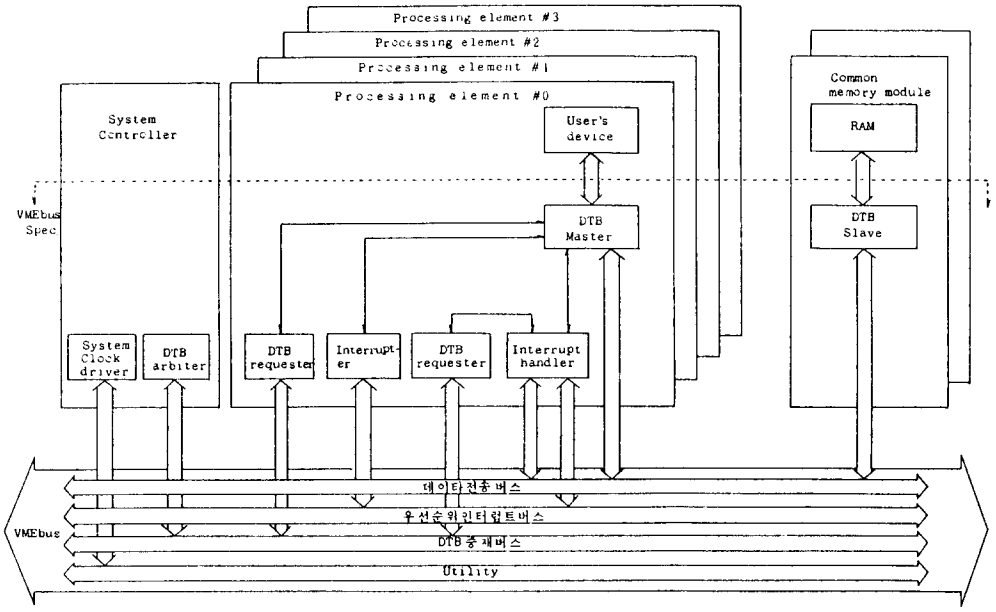


그림 3. 멀티프로세서 시스템의 機能的 接續圖
Fig. 3. Functional Interconnection of Multiprocessor System.

터에 依하여 發生된 데이타 傳送動作에 應答할 수 있는 機能的 모듈이다.

PE간의 通信은 各 PE의 DTB 마스터가 requester를 通하여 DTB를 要求한다. 그후 시스템 컨트롤러의 DTB仲裁機(arbiter)로 부터 使用權을 획득하면 DTB를 通하여 필요한 데이타를 共通 메모리에 傳送하거나 共通 메모리로부터 읽어온다.

2. 운영체제의 구조 및 이식

본 운영체제는 MTOS(multitasking operating system)를 수정 이식하여 멀티태스킹 및 멀티프로세싱이 가능하게 하였으며 다음과 같은 기능을 갖게 하였다.

- 다중처리
- 우선순위 스케쥴링
- 동적 메모리 관리
- 인터럽트 처리

운영체제는 Kernel, 하드웨어 인터페이스 및 소프트웨어 인터페이스로 구성된다. 그림 4에 운영체제를 수정이식하는 과정을 나타내었으며, Porting guide line의 설정은 target machine의 구조와 working environment 등이 이에 속한다. 처음 PE는 bare machine으로써 PE의 초기화를 위하여 VMEbus 3.3을 수정한 모니터 프로그램을 개발하였으며, 모든 소프트웨어는 VME/10에서 개발되어 시분할 버스를 통해 공통메모리로 전송하여 실험하였다. 운영체제에 관련된 주요

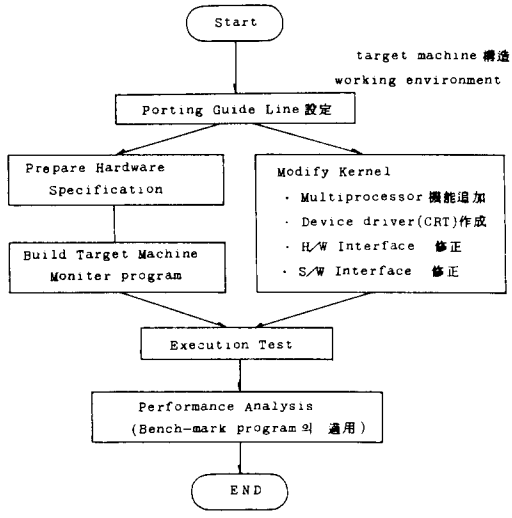


그림 4. 운영체제의 移植과정
Fig. 4. Porting Procedure of Operating System.

하드웨어 사양은 다음과 같다.

- 구 조 : 시분할 버스 구조
- PE의 수 : 4 대
- 공통 메모리 : 128KB ($B00000_{16}$~$B1FFFF_{16}$)
- 로칼 메모리 : RAM (48KB : 00000_{16}~$00BFFF_{16}$)
  ROM (64KB : $F00000_{16}$~$FOFFFF_{16}$)

- RTC 인터럽트 : 레벨 6
- RTC클럭주기 : 5 msec
- CRT 인터럽트 : 레벨 5
- 버 스 중 재 : 4 레벨 우선순위

각 PE의 PID와 우선 순위 레벨은 表 2 와 같다.

표 2. PE의 PID와 우선순위 레벨
Table 2. PID and Priority Level of PE.

PE No.	PID	Priority level
0	0	2
1	1	1
2	2	0
3	3	0

다음 과정은 Kernel의 수정으로 멀티 프로세싱 기능을 추가하기 위하여 총 26개의 모듈중 20개의 모듈을 수정하였으며, 이중에 시스템의 초기화와 멀티 프로세서 시스템에 중요한 영향을 미치는 相互 排除를 그림 5 와 6 에 나타내었다.

시스템의 初期化는 4 개의 PE중 PE # 0 가 시스템 변수 및 공통 메모리를 초기화하고 다른 PE들은 초기화가 끝날 때까지 대기 상태에 있으며 초기화 완료 신호(INIFLQ)에 의하여 각 PE는 자신의 로칼 메모리와 로칼 디바이스, RTC 등을 초기화하도록 하였다.

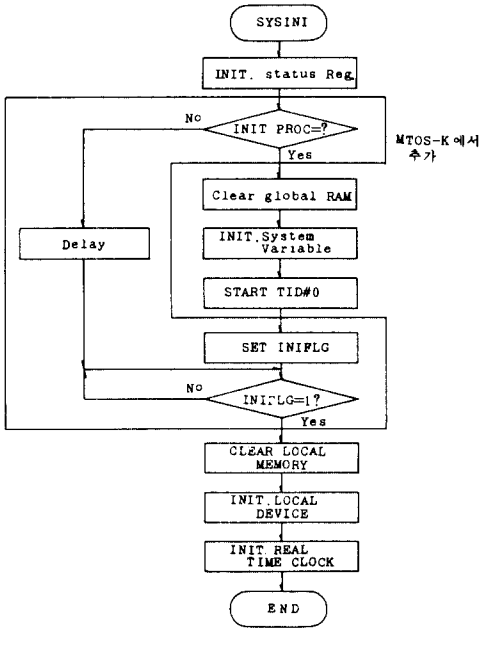


그림 5. 운영체제의 초기화 순서도
Fig. 5. Initialization of the Operating System.

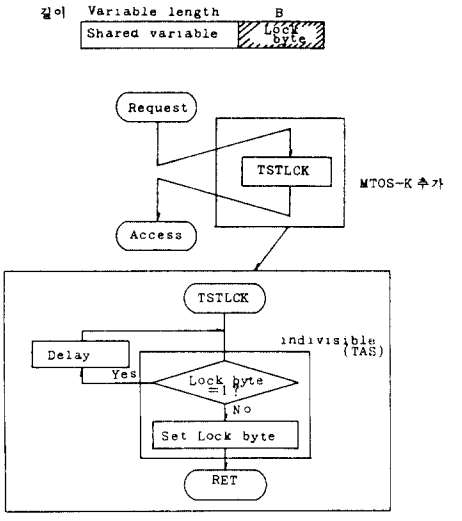


그림 6. 공유변수에 대한 相互排除
Fig. 6. Mutual Exclusion for Shared Variables.

공유 데이타(shared data)에 대한 相互 排除는 "TAS"라는 read-modify-write 명령을 이용하였으며, 각 共有 데이타는 1 바이트의 lock 바이트를 두어 lock 바이트의 상태에 따라 共有 데이타를 사용하고자 하는 프로세서는 액세스하거나 기다린다. 따라서 共有 데이타를 액세스하고자 하는 프로세서는 항상 TSTLCK서브루틴을 實行하도록 하였다.

PE간의 同期(synchronization)는 EF(event flag), 세마포어, MB(mailbox message)에 의하여 可能하며, 태스크의 상태 천이도를 그림 7 에 나타내었다.

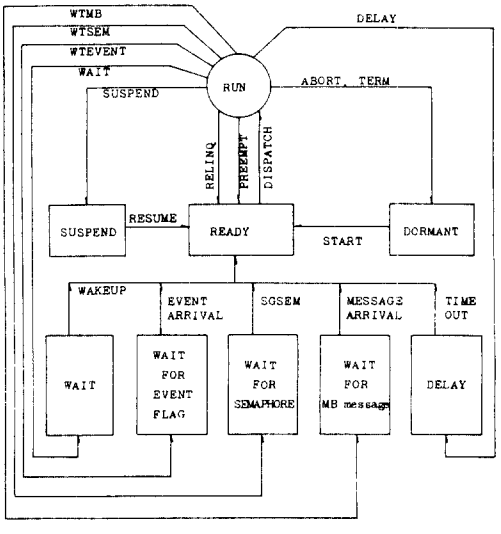


그림 7. 태스크의 狀態 천이도
Fig. 7. State Transition Diagram of Task.

실행 테스트는 다이내믹 디버거가 포함되어 있어 break point 설정과 trace, 각 태스크의 상태를 확인할 수 있으며, 진단 프로그램을 이용하여 EF, 세마포어, MB, 메모리 관리, 타이머 관리등의 실행을 확인하였다. 운영체제를 移植했을 때의 전체적인 메모리 배열은 그림8과 같다.

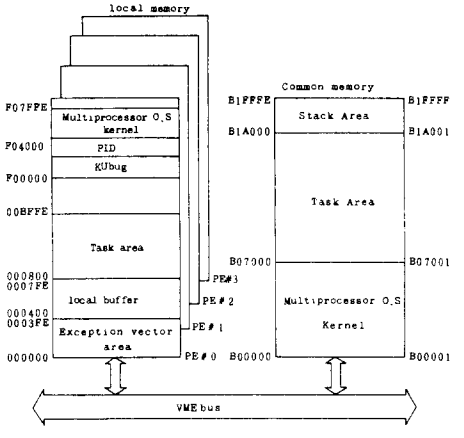


그림 8. 운영체제의 메모리 排列
Fig. 8. Memory Map of the Operating System.

Ⅲ. SPN에 의한 모델링과 性能 評價

M. K. Molly가 제안한 stochastic petri net(SPN)[10]을 이용하여 본 논문에서 具現한 멀티프로세서 시스템의 모델링 및 性能 分析을 하였다.

일반적 Petri Net는 표지소(place)의 유한 집합P, 천이의 유한 집합T, 방향 호의 유한 집합A, 토큰 표시의 유한 집합M으로 구성되어 있다. 여기에 각 천이에 관련된 점화율의 유한 집합G를 추가하여 SPN을 다음과 같이 정의한다.

$$SPN \triangleq (\mathbf{P}, \mathbf{T}, \mathbf{A}, \mathbf{M}, \mathbf{G}) \quad (1)$$

여기서 $\mathbf{P} = (p_1, p_2, p_3, \cdots, p_n)$

$\mathbf{T} = (t_1, t_2, t_3, \cdots, t_n)$

$A \subset (\mathbf{P} * T) \mid (T * \mathbf{P})$

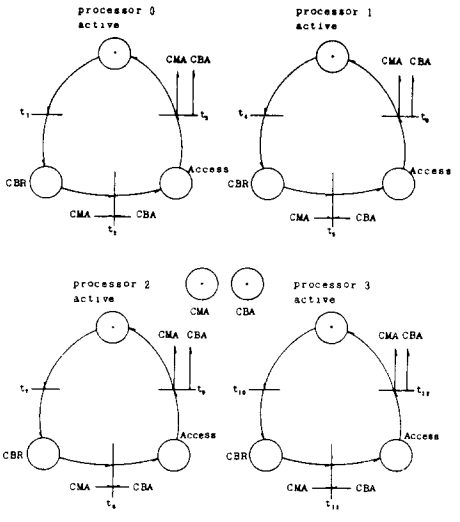
$\mathbf{M} = (m_1, m_2, m_3, \cdots, m_n)$

$G = (g_1, g_2, g_3, \cdots, g_n)$ 이다.

멀티프로세서 시스템을 모델링하기 위하여 프로세서의 상태를 다음과 같이 정의한다.

1) Active : 프로세서가 로칼 메모리를 사용할 때
2) Acessing : 프로세서가 공통 메모리를 사용할 때
3) Waiting : 프로세서가 공통 메모리를 사용하기 위하여 기다릴 때

본 시스템을 SPN을 이용하여 모델링하면 그림9와 같다. 처음에 모든 프로세서는 로칼 메모리 액세스 상태에 있으며, 이 상태에서 공통 메모리 액세스를 요구할 때 프로세서는 BR(bus request) 상태가 된다. 여기서 공통 버스가 利用 可能하고 공통 메모리가 액세스 가능하면 공통 메모리 액세스 상태가 된다. 그렇지 않으면 BR 상태에서 기다리게 된다. 이와같은 시스템의 상태변화는 토큰을 運用하므로써 이루어지며, 그림9의 토큰 분포는 初期 狀態를 나타낸다.



CBA : Common bus available
CBR : Common bus request.
CMA : Common memory available.

그림 9. 멀티프로세서 시스템의 SPN모델
Fig. 9. SPN Model of Multiprocessor System.

각 천이에 관련된 점화율은 다음과 같이 정의한다.

$$g_1 = g_4 = g_7 = g_{10} = \lambda \quad (2)$$

$$g_3 = g_6 = g_9 = g_{12} = \mu \quad (3)$$

$$g_2 = g_5 = g_8 = g_{11} = \alpha \quad (4)$$

λ는 프로세서가 액티브 상태에 있다가 공통 메모리를 사용하기 위하여 공통 버스를 요구하는 율(rate)로서 로칼 메모리 액세스율(local memory access rate)로 정의한다. μ는 프로세서가 공통 메모리 사용후에 다시 액티브 상태로 되돌아오는 율(rate)로서 공통 메모리 액세스율(common memory access rate)로 정의한다. α는 버스 중재와 관련이 있으며 프로세서가 액티브 상태 및 액세스 상태에 있는 시간보다 버스 중재와 해제에 걸리는 시간이 매우 작으므로 α는 무시할 수 있다. t_2, t_5, t_8, t_{11}은 충돌 가능하며 동시에 버스 요구가 일어날 경우 t_2의 우선 순위가 가장 높고 t_{11} 이 가장 낮아 PE #0이 버스 사용권을 획득한다. 8대와 16대의 프로세서를 使用하는 경우는 그림9의 확장으로 볼 수 있으며 같은 方法으로 모델링할 수 있다. 이

시스템의 시뮬레이션을 하기 위하여 다음과 같이 정의한다.

1) 프로세서는 액티브 상태후에 공통 메모리의 액세스를 요구한다.
2) 각 프로세서의 로칼 메모리 액세스율과 공통 메모리 액세스율은 서로 무관한 지수분포 함수이다.
3) 프로세서가 공통 메모리 사용을 요구했을 때 버스가 사용가능하고 공통 메모리가 액세스 가능하면 즉각적으로 Path가 설정된다(지연없음).
4) Path가 설정될 수 없으면 프로세서는 대기 상태에 있다.
5) 공통 메모리 액세스가 완료되면, 공통 메모리와 공통 버스가 즉시 해제된다

시뮬레이션을 위한 입력 변수는 평균 로칼 메모리 액세스 시간($1/\lambda$)과 평균 공통 메모리 액세스 시간($1/\mu$)이며, 이를 이용하여 시스템 負荷率(systemload factor) ρ를 다음과 같이 정의한다.

$$\rho \triangleq \frac{\text{평균 공통메모리 액세스시간}(1/\mu)}{\text{평균 로칼메모리 액세스시간}(1/\lambda)} = \frac{\lambda}{\mu} \quad (5)$$

따라서 ρ는 λ와 μ에 의하여 결정된다. 식(6)과 같이 정의되는 processing power **P**는 이용되는 프로세서 갯수의 평균치로서 速度 改善과 비례관계를 갖는다.

$$\mathbf{P} = \sum_{n=1}^{N} \frac{T_n - W_n}{T_n} \quad (6)$$

여기서 T_n은 프로세서 n이 처리한 총시간, W_n은 프로세서 n이 대기 상태에 있은 총시간, N은 프로세서의 수이다.

2 대의 프로세서로 多重處理할 경우에는 SPN과 등가인 markov chain(MC)의 상태 천이도에 의한 解析的인 方法으로도 구하였다. 식(7)과 같이 2 N-tuple 에 의하여 시스템의 狀態를 定義할 수 있다.

$$(m_1, s_1; \cdots; m_i, s_i; \cdots; m_N, s_N) \quad (7)$$

표 3. 멀티프로세서 시스템의 Processing Power
Table 3. Processing Power of the Multiprocessor System.

프로세서수 / ρ / 알고리즘	2 대		4 대	8 대	16대
	MC	SPN	SPN	SPN	SPN
0.01	1.9997	2	3.997	7.992	15.959
0.05	1.9976	1.997	3.967	7.832	14.549
0.1	1.9910	1.990	3.876	7.229	10.500
0.2	1.9677	1.995	3.818	5.567	5.997
0.4	1.8970	1.868	2.989	3.484	3.601
0.8	1.7377	1.693	2.245	2.333	2.539
1.0	1.6667	1.621	2.009	2.038	2.210

표 4. 시스템 負荷率에 따른 프로세서의 效率
Table 4. Efficiencies of Processors for the Different System Load Factors.

프로세서수 / ρ	2 대	4 대	8 대	16대
0.01	1	0.999	0.999	0.997
0.05	0.998	0.992	0.979	0.909
0.1	0.995	0.969	0.904	0.656
0.2	0.968	0.904	0.696	0.375
0.4	0.934	0.747	0.436	0.225
0.8	0.847	0.561	0.267	0.156
1.0	0.811	0.502	0.255	0.138

S_i는 프로세서 i의 상태로써 다음과 같은 값을 갖도록 정의하였다.

2 : 액티브 상태(Active)
1 : 액세싱 상태(Accessing)
0 : 대기상태(Wait)

또한 m_i는 프로세서 i가 使用하는 메모리모듈의 index 이며 다음과 같은 값을 갖도록 정의하였다.

2 : 공통 메모리 使用
1 : 로칼 메모리 使用
0 : 비 사용

이와같은 상태 값을 이용하여 2 대의 프로세서를 사용하는 경우의 MC 상태 천이도를 그림10에 나타내었다. 그림10은 대칭성에 의하여 더욱 간소화할 수 있으며 이에 의한 **P**는 식(8)과 같이 구해진다.

$$\mathbf{P} = \frac{(2 + 2 * \rho + \rho^2)}{(1 + \rho + \rho^2)} \quad (8)$$

SPN에 의한 시뮬레이션 결과가 해석적인 방법으로 구한 결과와 3.5% 이내의 차이를 나타내고 있으므로 SPN에 의한 분석 방법의 타당성을 입증하고 있다.

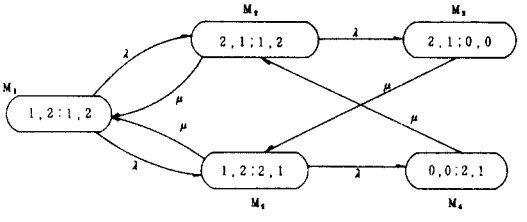


그림10. MC 상태천이도
Fig. 10. MC State Transition Diagram.

Ⅳ. Benchmark 프로그램의 적용 및 실험결과

Quicksort, FFT, 行列 곱셈등의 Benchmark 프로그램을 최대 4 대의 PE를 사용하여 병렬처리하였다. 이를 위하여 각각의 알고리즘을 단일 프로세서 시스템

에서 4개의 태스크로 分割하여 멀티태스킹 처리하였으며, 이 프로그램을 그대로 멀티 프로세서 시스템에 적용하고 각각의 PE에 태스크를 할당하여 2대의 PE와 4대의 PE로 처리했을 때의 速度改善(speedup)을 比較하였다. 速度改善은 식(9)와 같이 정의된다.

$$\text{速度改善} = \frac{\text{단일 프로세서 시스템에서의 실행시간}}{\text{멀티프로세서 시스템에서의 실행시간}} \quad (9)$$

각 태스크는 모든 프로세서에게 broadcast하여 태스크 단위에서 負荷均等分配가 可能하도록 하였으며, 일부의 PE가 故障을 일으켜도 다른 PE가 대치하여 실행할 수 있도록 하였다.

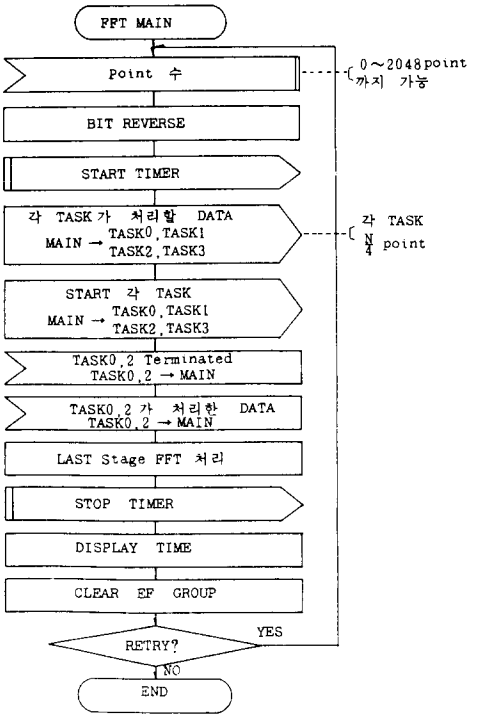
표 5. 멀티프로세서 시스템에서의 처리시간과 속도개선

Table 5. Execution Time and Speed up in Multiprocessor System.

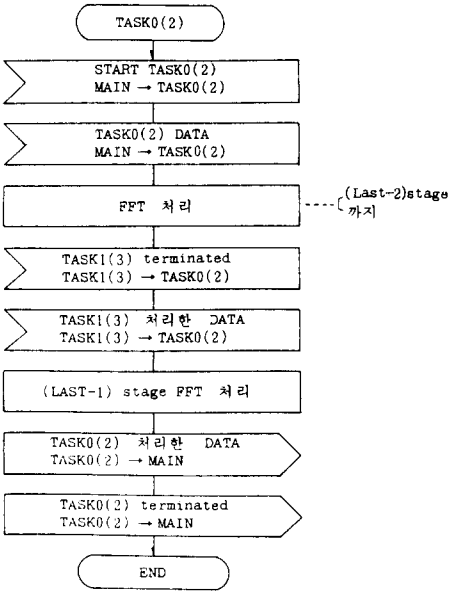
알고리즘 \ 태스크 / 측정 / PE數		local 처리시간 (sec)	local 속도개선	common 처리시간 (sec)	common 속도개선	비 고
quicksort	1	3.28	1	4.89	1	데이타數 (1024)
	2	1.89	1.74	3.12	1.57	
	4	1.06	3.09	1.79	2.73	
FFT 연 산	1	3.37	1	4.01	1	512 point 연 산
	2	1.97	1.71	2.04	1.52	
	4	1.08	3.12	1.49	1.69	
행 렬 곱 셈	1	1.76	1	2.63	1	(8×100) * (100×8)
	2	0.95	1.86	1.60	1.64	
	4	0.53	3.29	0.93	2.83	

각 태스크는 MC68000 어셈블리어로 작성하였으며, 처리결과를 표5에 나타내었다. 처리시간의 측정은 本 運營體制에서 제공하는 타이머 관리기능을 이용하여 msec 單位까지 時間測定이 가능하도록 타이머 모듈을 作成하여 이용하였다. 태스크들을 공통 메모리만을 사용하여 실행한 결과와 PE자체의 로칼 메모리에 두고 실행한 결과를 각각 측정하였다. 공통 메모리만을 사용할 경우 모든 태스크와 데이타를 공통 메모리에 두고 각 PE는 자기에게 할당된 태스크만을 처리할 수 있다. 로칼 메모리를 사용할 경우 각 PE는 로칼 메모리에 모든 태스크를 copy 하여야 하며 공통 메모리에는 共有 데이타만을 둔다. 이 경우에 각 PE는 로칼 메모리에 중복된 태스크 코드를 갖으므로써 메모리의 비 효율적인 사용이 발생하나 시분할 버스의 사용을 줄임으로써 처리 속도를 개선할 수 있다. FFT의 분할 처리과정을 附錄 1에 SDL로 나타내었다. 주 태스크와

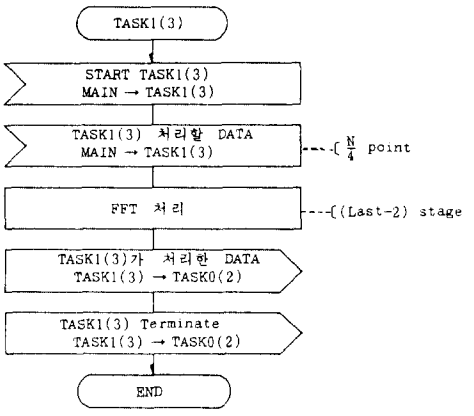
附錄 1. FET의 4 대 並列處理 順序圖



(a) 주 태스크 處理 順序圖



(b) 종 태스크 0, 2處理 順序圖



(c) 종 태스크 1, 3處理 順序圖

4 개의 종 태스크로 구성되어 있으며 주 태스크만 PE #0에서 實行하도록 하였고 종 태스크들은 각 PE에 broadcast하여 어떤 PE에서든지 實行이 가능하도록 하였다. 表 5 에서 알 수 있듯이 공통 메모리만을 사용할 경우에 速度改善은 2 대의 PE로 처리할 때 평균 1.58, 4 대의 PE로 처리할 때 평균 2.75배 이었다. 로칼 메모리를 사용하였을 경우에는 2 대의 PE로 처리할 때 평균 1.77, 4 대의 PE로 처리할 때 평균 3.17배의 速度改善이 있었다.

평균 처리시간은 로칼 메모리를 사용할 경우에 공통 메모리를 사용할 때 보다 약 40% 단축되었다.

V. 結 論

버스 構造를 갖는 멀티프로세서 시스템의 단점을 보완하기 爲하여 各各의 PE에 自體의 로칼 메모리를 두어 各 태스크를 로칼 메모리상에서 實行하고 共通 메모리를 액세스할 때만 時分割 버스를 使用하도록 함으로써 處理速度가 더욱 빠르게 됨을 確認하였다.

本 시스템의 運營體制로는 multitasking OS(MTOS)를 멀티태스킹과 멀티프로세싱이 可能하도록 수정하여 移植하였다. 運營體制의 Kernel은 共通 메모리에 두었으며 모든 PE가 共有하도록 하여 자유로운 情報交換이 可能하도록 하였고 記憶場所의 共有로 컴퓨터의 效率을 증가시켰다.

SPN에 依한 시뮬레이션 結果에 依하면 時分割 버스 構造는 시스템 負荷率에 따라 프로세서 數의 증가는 프로세서의 效率을 하락시키게 되므로 주어진 負荷率에 따라 적당한 數의 프로세서를 결정하는 것이 바람직하다.

Benchmark 프로그램을 적용한 결과 로칼 메모리를 使用할 경우의 處理時間은 共通 메모리를 使用할 때 보다 平均40% 단축되었으며, 速度改善은 2 대, 4 대의 프로세서를 使用할 때 各各 1.77배, 3.17배로 나타났다. 따라서 태스크를 스케쥴하는데 걸리는 時間과 태스크간의 同期等을 考慮하여 적당한 數의 태스크로 分割하여 並列處理하면 本 멀티프로세서 시스템에서 좋은 速度改善을 이룰 수 있으며, 특정한 産業應用 시스템에서 高速 實時間 處理를 要하는 컴퓨터 시스템으로 活用이 可能할 것이다.

參 考 文 獻

[1] J.L. Baer, "Multiprocessing systems," *IEEE Trans. Comp.* vol. C-25, Dec. 1976, pp. 1271-1277.

[2] J. Deminet, "Experience with multiprocessor algorithms," *IEEE Trans. Comp.*, Vol. C-31, no.4, pp.278-288, April 1982.

[3] D. Towsley, "Approximate models of multiple bus multiprocessor system," *IEEE Trans. Comp.,vol.C-35,* no.3, pp.220-228. Mar. 1986.

[4] N.J. Dimopoulos, "On the structure of the homogeneous multiprocessor," *IEEE Trans Comp. Vol.C-34,* no.2, pp.141-150, Feb. 1985.

[5] G. Conte and D. Pelcorso, "TOMP80-A multiprocessor prototype," *Proc. EURO-MICRO81,* Paris France, Sept. 1981.

[6] 김덕신, 김영천외, "다중프로세서에 의한 신호처리에 관한 연구," 한국 전자통신연구소 연구보고서, 1985.

[7] VMEbus manufacturers group, "VMEbus specification manual", Aug. 1982.

[8] Motorola inc., "VERSAdos overview", May 1984.

[9] Industrial programming inc., "A Multitasking operating system for the 68000."

[10] M.K. Molly, "Performance analysis using stochastic Petri Net," *IEEE Trans. Comp.*, Vol.C-31, no.9, pp.913-917, Sep. 1982.

[11] H. Kirrmann, "Events and interrupts in tightly-coupled multiprocessors," *IEEE Micro*, pp.53-66, Feb. 1985.

[12] S.S., "Scheduling multipipeline and multiprocessor computers," *IEEE Trans. Comp.*, vol.C-33, no.7, pp.637-645, July 1984. *