다중프로세서시스템에 대한 파이프라인방식 메모리 접근제어의 설계와 그 효율분석

(A Design of Pipelined Memory Access Control for Multiprocessor Systems and its Evaluation)

金 鼎 斗*, 孫 潤 求**

(Jung Doo Kim and Yoon Koo Sohn)

要 約

본 논문은 밀결합 다중프로세서 시스템에 대한 프로세서와 메모리의 버스 인터페이스를 위한 새로운 기법으로서 파이프라인식 메모리 접근방식을 제안하고 있다. 공유버스가 시스템의 병목이므로 메모리 액세스과정을 파이프라인화하는 모델을 제안하고, 離散時間 Markov 모델에 의하여 效率을 分析한 결과 현저한 效率向上이 있음을 보여주고 있다.

Abstract

This paper proposes a pipelined memory access method as a new technique for a bus interface between processors and memories in tightly coupled multiprocessor systems. Since the shared bus is bottle neck of the system, model of pipelined access to memory has been developed. Results of the evaluation by the discrete time Markov model showed a significant improvement of the efficiency.

I. 序 論

메모리시스템의 주요 設計目標 중의 하나는 프로세서의 要求에 메모리시스템의 帶域幅을 一致시키는 데 있다. p개의 프로세서가 主메모리 시스템을 共有하는 多重프로세서시스템에서는 主메모리에 대한 액세스要求가 p倍로 增加하므로 主메모리를 m개의 獨立된 모듈들로 分割하고 番地를 分割된 모듈들에 分配하는 인터리브(interleaved) 메모리方式을 採擇함으로써 帶域幅을 一致시킬 수 있다.[1]

프로세서와 메모리間 結合方式으로서 傳統的으로 單一共有버스가 使用되어 왔으나, p개의 프로세서와 m개의 메모리모듈간에 必要한 액세스經路의 數는 p.m으로서 p와 m이 커지면 幾何級數的으로 增加하

*正會員, 曉星女子大學校 自然大學 電子計算學科
(Dept. of Computer Science, Hyosung Women's Univ.)

**正會員, 嶺南大學校 電算工學科
(Dept. of Comp. Eng., Yeungnam Univ.)

接受日字 : 1988年 2月 2日

므로 시스템의 規模가 커짐에 따라 버스에 병목현상이 일어난다. 이에 대하여 多段階스위치網이 適切한 代案으로 널리 認定되고 있는데, 여러가지 스위치가 提案되어 있으나 스위치칩의 pin數 制限으로 인하여 VLSI技術進步의 利點을 適用하지 못하는 缺點이 있어 多重마이크로프로세서시스템의 경우, 스위치의 費用이 프로세서와 메모리의 費用보다 커져 費用面에서 均衡을 잃게된다. 이에따라 多重버스 또는 한개의 共有메모리에 單一 共有버스를 통하여 小數의 마이크로 프로세서를 結合시켜 cluster를 이루고, 이들을 階層적으로 結合한 階層構造(hierarchical) 시스템을 構成하는데, cluster에 所屬된 마이크로프로세서의 數가 많을수록 프로세서간의 論理的 平均距離가 짧아지며, 프로세서간 通信이 簡便해지는 利點이 있어 竝列度가 높은 結合方式이 要求된다.

竝列度가 높고 複雜度가 比較的 낮은 것으로 L-M 메모리 構造가[2] 있으나, 마이크로프로세서의 連續된 메모리 要求間의 平均 時間間隔(average inter-request time)은 일반적으로 메모리週期 보다 크므로 大型프로세서의 경우보다는 낮은 메모리 帶域幅이 要求되며 高價의 크로스바스위치網이 적당치 않으므로, 多重마이크로프로세서 시스템에서는 單一共有버스를 通하여 多數의 프로세서가 多數의 메모리 모듈들을 竝列로 액세스할 수 있는 結合方式이 바람직하다.

多數의 프로세서가 單一 共有버스를 통하여 竝列액세스를 하기 위하여 버스의 情報傳達時間이 半導體메모리의 액세스週期와 프로세서의 메모리 要求時間間隔보다 짧은 점을 이용하여 프로세서의 메모리 액세스 過程을 여러 段階로 區分하고 파이프라인(pipeline)化 하여 메모리 要求當 버스의 使用時間을 最小化할 수 있다.

본 硏究에서는 그러한 竝列메모리시스템을 考案, 設計하고 그 效率을 評價한다.

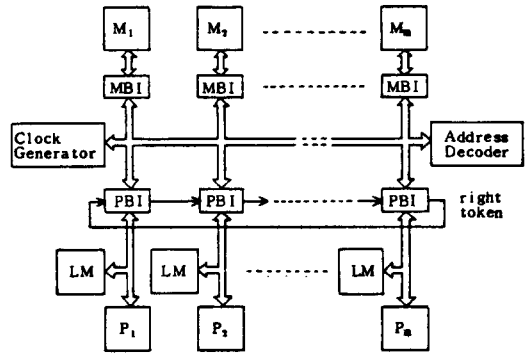
Ⅱ. 시스템 모델

그림 1은 전체 시스템의 構成圖이다.

프로세서와 메모리의 特性은 機種에 따라 매우 多樣하고 그에 따라 竝列메모리시스템의 設計方法도 달라지므로 본 硏究가 意圖하는 設計方法이 可能한 限 一般性을 갖기 위하여 다음과 같이 假定한다.

1) 각 프로세서와 각 메모리의 특성은 同一하다.

2) 프로세서의 메모리 要求로부터 read 데이타를 래치할 때까지의 時間을 메모리 要求時間이라 하며, 메모리 要求時間은 한개 以上의 클럭週期로 構成된다.



where,
M_i : shared memory module
P_i : processor
LM : local memory
PBI : processor bus interface
MBI : memory bus interface

그림 1. 시스템構造
Fig. 1. System organization.

3) 프로세서에게 待機(wait) 信號가 들어오면, 待機信號가 失效될때까지 프로세서의 實行이 中止된다.

4) 프로세서의 메모리 要求時間間隔을 tr, 클럭週期를 tc, 半導體 메모리의 액세스時間을 tx, 버스週期를 tb라면 $tb \leqq tc \leqq tx \leqq tr$의 關係가 成立한다.

5) 競合調整時間(arbitration time)을 ta라면 ta, tb, tc, tx, tr 서로간에는 整數倍의 關係가 있어서 $a = ta/tu$, $b = tb/tu$, $c = tc/tu$, $x = tx/tu$, $r = tr/tu$가 모두 整數인 tu가 存在한다.

6) 메모리리프레쉬(refresh), 프로세스 同期化機構, 入出力등은 省略한다.

파이프라인化 하기위한 시스템의 共有메모리액세스過程은 다음과 같다.

1) 競合調整(arbitration)

프로세서들이 競合調整器(arbiter)에 버스의 使用을 要求하면 그들중 하나를 選擇하여 承認(grant)한다.

2) 要求傳達

競合調整器로부터 承認을 받으면 要求(番地, read/write 信號)를 버스를 통하여 行先(destination)모듈에 傳達한다.

3) Write data 傳達

Write의 경우, write 할 데이타를 버스를 통하여 行先모듈에 傳達한다.

4) 承認(acknowledge) 傳達

行先모듈이 遊休(idle)狀態이면, 3)과 同時에 承認信號를 버스를 통하여 原프로세서에 傳達한다.

5) 액세스

行先모듈이 遊休狀態이면, 3) 과 同時에 메모리 액세스를 시작한다.

6) Read data 傳達

Read의 경우, 읽은 데이타를 버스를 통하여 原프로세서에 傳達한다.

시스템의 특성을 (a, b, c, x, r) 로 表記하면, (a, b, x) = (1,1,3) 인 시스템에서 각 프로세서 모듈의 공유메모리 액세스를 파이프라인화 하기 위한 豫約表 (reservation table) 는 표 1 과 같다.[3]

표 1. 豫約表
Table 1. A reservation table.

stage \ time	1	2	3	4	5
arbiter	×				
address, read signal bus		×			
acknowledge bus			×		
write data bus			×		
access			×	×	
read data bus					×

파이프라인이 效率的으로 動作하기 위해서 連續的인 入力의 흐름이 必要한데 競合調整器는 多數의 프로세서로부터의 無作爲한 要求에 대하여, 順次的인 一連의 承認을 행하므로 파이프라인의 이러한 要求를 滿足시킨다. 액세스 段階에서, 액세스중인 모듈이 選擇되면 要求가 拒否되어 自動的으로 順序化되어 表 1은 表 2와 같이된다. 表 2의 파이프라인의 到着時間間隔 (latency) 은 1이다.

표 2. 豫約表 (表 1의 變形)
Table 2. Modified reservation table.

stage \ time	1	2	3	4	5
arbiter	×				
address, read signal bus		×			
ack, write data bus, access			×		
read data bus					×

프로세서의 클럭사이클을 s=c/b 개의 타임슬로트로 區分하고 p개의 프로세서를 s 그룹으로 나누어 각 그룹에 割當된 프로세서들에 그림 2와 같이 클럭을 供給하면, 그룹당 [p/s] 개의 프로세서가 割當되고 그룹내에서만 競合을 하므로 競合調整時間이 짧아지고 버스競合이 緩和되는 利點이 있다.

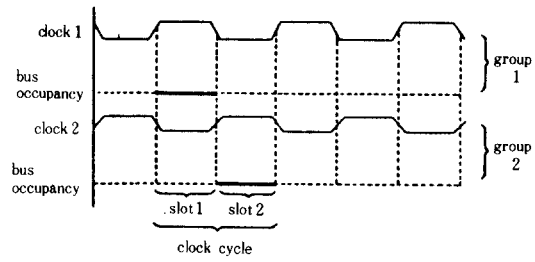
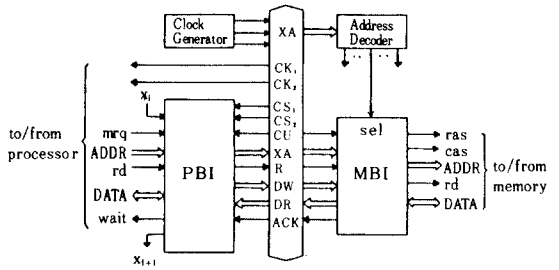


그림 2. (b, c) = (1,2) 인 경우의 클럭 및 버스占有時間圖
Fig. 2. Clock and bus occupancy timing chart for the system of (b, c) = (1,2).

Ⅲ. 設 計

본 論文에서는 (a, b, c, x) = (1,1,2,3) 인 並列 메모리시스템을 設計한다. 그림 3은 프로세서의 버스인터페이스 (以下 PBI 라 한다) 와 메모리모듈의 버스인터페이스(以下 MBI 라 한다) 雙을 中心으로 버스, 競合調整器, 番地復號器 (address decoder), 클럭信號發生器의 블록圖이다.



where,

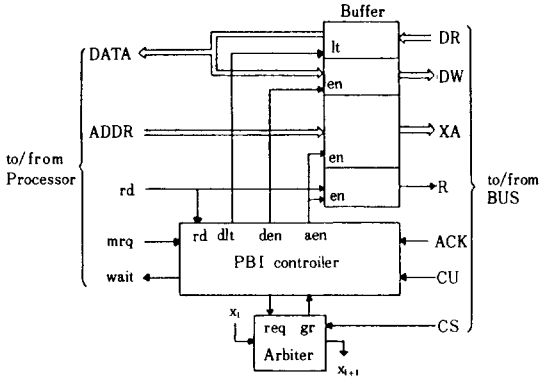
CU	: unit clock	XA	: address bus
$CK_{1,2}$	: processor clock	DW	: write data bus
$CS_{1,2}$	: slot identifying clock	ACK	: acknowledge
mrq	: memory request	DR	: read data bus
rd	: read/$\overline{\text{write}}$ signal	R	: read/$\overline{\text{write}}$ bus
x_i, x_{i+1}	: arbitration token	sel	: select memory module
wait	: wait processor	ras	: row address select
		cas	: column address select

그림 3. PBI/MBI 雙의 블럭圖
Fig. 3. Block diagram for a pair of PBI and MBI.

1. PBI

PBI는 그림 4 와 같이 버퍼, 制御器(controller)및 競合調整器 셀(cell) 로 構成된다.

PBI 制御器는 프로세서, 버스, 競合調整器와 制御信號를 授受하고 PBI 버퍼를 制御한다. 그림 5 는 표 1 의 豫約表로부터 表現한 PBI 制御器의 狀態遷移圖이다.



where, lt : latch
en : enable
aen : address enable
den : data enable
dlt : data latch

그림 4. PBI 블럭圖
Fig. 4. PBI block diagram.

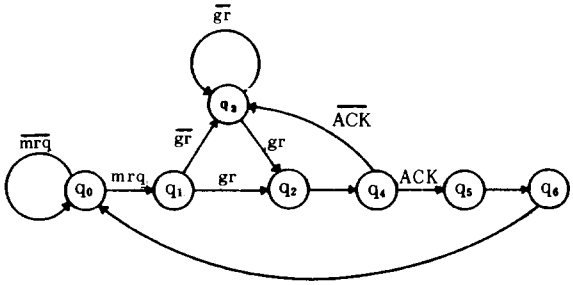


그림 5. PBI 制御器의 狀態遷移圖
Fig. 5. State transition diagram of PBI controller.

初期狀態인 q_0에서 共有메모리要求가 들어오면 q_1으로 遷移한다. q_1에서는 競合調整器에 버스의 占有權을 要求하여, 競合調整器로부터 承認(gr) 信號가 들어오면 q_2로 遷移한다. q_2에서는 番地, rd 信號를 각각 XA 및 R 버스에 내고 q_4로 遷移한다. q_4에서는 write의 경우, write 할 데이타를 DW 버스에 내고 버스로부터 承認(ACK) 信號가들어오면 q_5로 遷移한다. q_5에서는 read의 경우, 읽은 데이타를 래치하고 q_0 상태로 되돌아 감으로써 한 週期가 끝난다. q_1에서 競合調整器로부터 承認(gr) 信號를 얻지 못하면, q_3으로 遷移하여 承認(gr) 信號가 들어올 때까지 q_3에서 머문다. q_3에서는 버스要求信號(req)를 냄과 동시에 프로세서에 待機(wait) 信號를 낸다. q_4에서 行先메모리모듈이 액세스 중이어서 承認(ACK) 信號를 얻지 못하면 待機(wait) 信號를 Flip-Flop 에 記憶시킴과 동시에 q_3으로 遷移한다.

承認(ACK) 信號를 얻을 때까지 $q_3-q_2-q_4$ 루프를 反復하다가 q_4에서 承認(ACK) 信號가 들어오면 q_5로 遷移함과 동시에 Flip-Flop에 記憶된 待機(wait) 信號를 無效化 한다. 따라서, 버스의 使用承認이나 메모리모듈의 액세스承認을 얻을 때까지 프로세서는 待機狀態에 있게된다. 각 出力信號의 論理式은 다음과 같으며 그림 6 은 그로부터 設計한 論理回路圖이다.

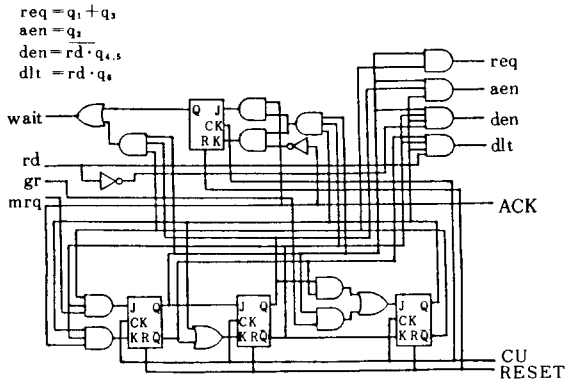


그림 6. PBI 制御器
Fig. 6. PBI controller.

競合調整器의 機能은 多數의 프로세서가 限定된 버스의 獨占使用을 要求할때 그중 하나를 選擇하여 承認해 주는 것이다. 競合調整器는 매우 多樣한 方式이 提案되어 있으므로 設計하려는 시스템의 特性에 맞는 方式을 選擇하여야 한다. 選擇基準으로는 競合調整速度, 優先順位, 費用, 信賴性, 擴張性등을 考慮하여야 한다. 여기서는 進行波를 利用한 ring arbiter 중 가장 速度가 빠른 權利토큰 沮止形(right token stopping) IRA(inverter ring arbiter)를 採擇

한다.[8]

Ring arbiter의 特徵은 回路가 簡單하면서도 競合調整速度가 빠르며 擴張性이 좋고 要求프로세서가 同一한 優先順位를 가지는 利點이 있는 반면 信賴性이 낮다.

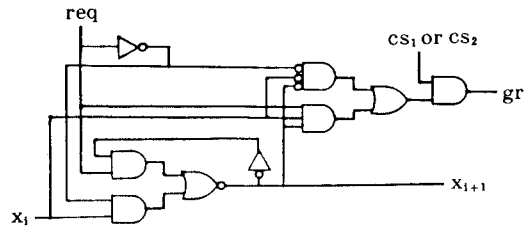


그림 7. IRA 셀
Fig. 7. IRA cell.

競合調整時間은 토큰의 現 位置와 要求 셀의 相對的 距離에 따라 決定되나, 最大競合調整時間은 ⌈q/s⌉·tp(단, p : 프로세서의 數, s : 클럭당 타임슬로트 개수, tp : 셀의 傳達遲延時間) 이 된다. 또, 토큰의 흐름과 프로세서의 要求는 서로 非同期的으로 動作하므로 토큰 到着과 要求메세지의 到着時刻이 接近하면 셀이 相當時間동안 擬似安定狀態(meta stable)에 들어가 시스템에 故障이 發生할 수 있다. 따라서 總競合調整時間을 다음 조건을 만족하게 한다.

ta > max{⌈p/s⌉·tp, ts} (단, ts : 擬似安定狀態 持續時間)

2. MBI

MBI는 그림 8과 같이 MBI 버퍼와 MBI 制御器 및 承認論理(acknowledge logic)로 構成된다.

MBI 制御器는 버스, 半導體메모리, 番地復號器, 承認論理와 制御信號를 授受하고 MBI 버퍼를 制御한다. 그림 9는 表 1의 豫約表로부터 表現한 MBI制御器의 狀態遷移圖이다. 初期狀態인 q_0에서는 承認論理에遊休信號를 내고, 選擇(sel) 信號가 들어오면 q_1으로 遷移함과 동시에 番地버스와, R 버스로부터 각각 番地, rd信號를 래치한다. q_1에서는 行番地選擇(ras : row address select) 信號를 내고 q_2로 遷移함과 동시에 write의 경우, DW버스로부터 write할 데이타를 래치한다. q_2에서는 q_1에 이어 行番地選擇信號를 계속 내며, 列番地選擇(cas : column address select) 信號와 rd信號를 내고 q_0로 遷移함으로써 한 週期를 끝낸다. Read의 경우, 읽은 데이타를 q_2로부터 q_0로 遷移할때 버퍼에 래치시키고 q_0에서 DR버스에 낸다. 각 出力信號의 論理式은 다음과 같으며 그림10은 그로부터 設計한 論理回路圖이다.

그림11의 承認論理는 番地復號器로부터 選擇信號(sel)가 들어왔을때 모듈이 액세서중이 아니면 즉, 그림 9의 狀態遷移圖에서 MBI 制御器가 q_0 狀態에 있을때 버스에 承認信號를 낸다.

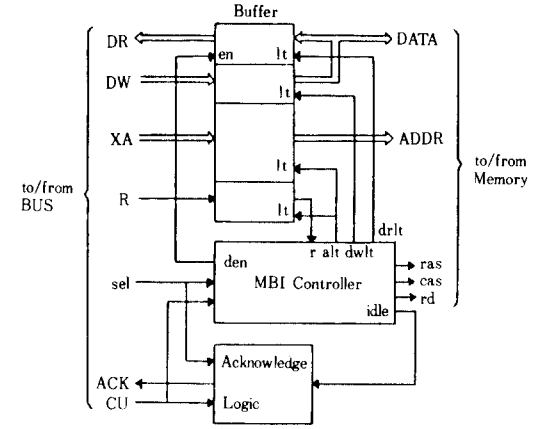


그림 8. MBI 블럭도
Fig. 8. MBI block diagram.

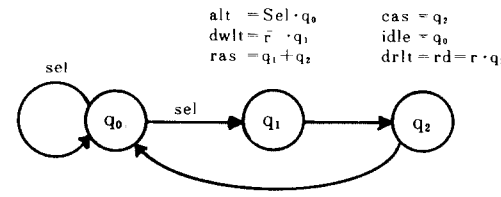


그림 9. MBI 制御器의 狀態遷移圖
Fig. 9. State transition diagram of MBI controller.

Ⅳ. 效率 分析

프로세서의 單位時間當 命令實行能力에 대한 실제 實行命令의 數를 效率이라 한다. 버스에 接續된 프로세서 및 메모리모듈의 增減에 따른 프로세서의 效率變化를 分析함으로써 費用/性能面에서 適切한 프로세서 및 메모리모듈의 數를 決定할 수 있으며 이를 基本單位로 多重버스, 階層버스등의 시스템을 構成할 수 있다. 그림12는 전체 시스템의 待機行列閉回路網이다.

본 연구의 시스템은 單一共有버스를 假想하지만 Ⅱ

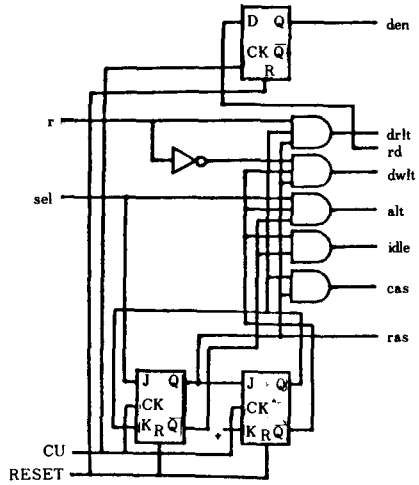


그림10. MBI 制御器
Fig. 10. MBI controller.

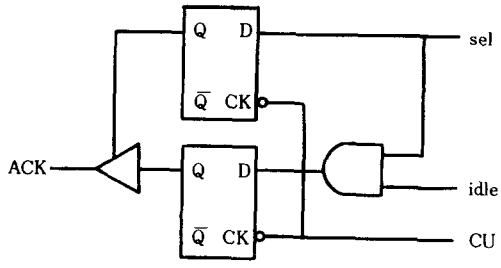


그림11. 承認論理
Fig. 11. Acknowledge logic.

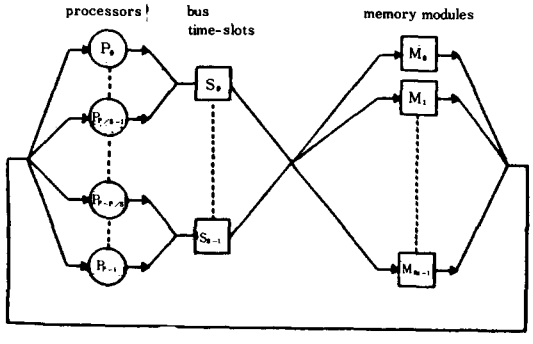


그림12. 시스템의 待機行列閉回路網
Fig. 12. Closed queueing network of the system.

장에서 언급한 바와 같이 각 그룹에 속한 프로세서들은 割當된 타임슬로트에서만 버스 使用權을 얻을 수 있으므로 s 개의 버스를 갖고 버스待機時間이 s 배인 시스템과 같다.

프로세서와 메모리모듈간 共有버스結合方式의 效率에 관해서는 지금까지 많은 논문에서 연구되어 왔다. Fung[6] 은 process graph의 reference모델을 定義하고, 그것을 多重버스시스템의 버스競合과 메모리衝突의 影響分析에 適用하였으며, Marsan[7] 은 여러가지 形態의 單一버스시스템의 性能을 分析, 比較하기 위하여 Markov 모델을 開發하였는데, 메세지傳達時間동안 프로세서가 버스 및 메모리를 占有하는 시스템을 對象으로 하였다. Mudge[8] 는 多重버스시스템의 메모리衝突影響을 分析하기 위한 離散時間 semi-Markov 모델을 提示하였는데 임의의 要求時間間隔과 서비스時間을 許容하므로 凡用性이 높다. 그러나, 본 연구의 시스템의 論理的構造는 그림12와 같이 單一共有버스도 아니며, 多重버스도 아니므로 以上의 論文에서 硏究한 結果를 그대로 適用할 수 없다.

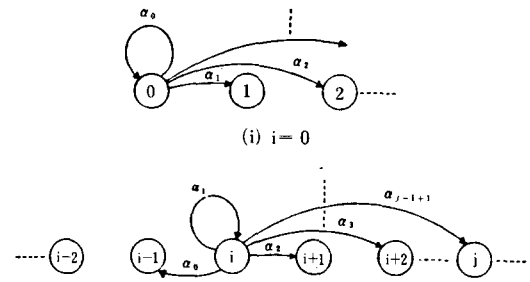


그림13. M/D/1 Markov 連鎖의 狀態遷移圖
Fig. 13. State transition diagram for the M/D/1 Markov chain.

프로세서의 共有버스占有時間別로 시스템을 分類하면 占有時間이

i) 메세지傳達時間
ii) 프로세서의 메모리要求時間(inter-request time)
iii) 메모리週期(memory cycle)
iv) 레지스터간 傳達時間

인 네가지 경우로 나누어진다. 이중, i), ii), iii) 인 경

우는 버스 占有時間이 메모리週期 以上이므로 하나의 버스에 하나의 메모리 모듈로써 充分하다. i) 의 경우는 서비스시간이 메세지의 길이에 의해 결정 되므로 여기서는 먼저 ii), iii) 인 경우의 共有메모리衝突로 인한 效率變化分析모델을 開發하고 이것을 iv) 의 경우에 擴張適用한다

1. 單一共有메모리시스템의 경우

프로세서의 메모리액세스 要求時間間隔의 單位時間 tu에 대한 比率을 r이라 하고 總 메모리要求중 共有메모리要求의 比率을 γ라하면 共有메모리 즉 共有버스 平均要求時間間隔(smr) 및 共有메모리모듈 要求率(λ) 는 각각 식(1),(2)와 같다.

$$smr = r/\gamma \qquad (1)$$

$$\lambda = \gamma/r \qquad (2)$$

共有메모리에 대한 要求時間間隔이 指數函數的으로 分布한다고 假定하면, 프로세서의 버스 및 메모리 占有時間은 一定하므로 p 개의 프로세서와 메모리와의 관계는 M/D/1//p(exponentially distributted interrequest time/deterministic service time/single server //p population)型 Markov連鎖가 된다.[9] 시스템에 들어오는 n번째 要求를 Rn이라하고 Rn이 서비스를 받고 시스템을 떠나는 순간에 남아있는 要求의 계수를 q_n이라 하면, 그림13은 q_n을 狀態變數로 가지며 서비스가 끝나 Rn이 떠나는 순간에 遷移하는 M/D/1 Markov連鎖의 狀態遷移圖이다.

α_k는 다음의 要求Rn에 대한 서비스기간중에 發生할 要求의 數 v_{n+1}이 k개일 確率이며 식(3)과 같이定義된다.

$$\alpha_k \triangleq P[v_{n+1} = k] \qquad (3)$$

i = 0 즉 $q_n = 0$인 경우, j개의 要求가 發生하면 q_j로 遷移하게 되며, $i \geq 1$ 즉 $q_n \geq 1$인 경우, R_{n+1}의 서비스기간동안 j 개의 要求가 發生하면 서비스가 끝나는 순간 서비스를 받은 要求는 시스템에서 벗어나므로 q_{j-i+1}로 遷移하게 된다.

p_j^n을 R_n이 시스템을 벗어나는 순간 $q_n = j$일 確率이라면 p_j^{n+1}은 식(4)와 같다.

$$p_j^{n+1} = p_0^n \alpha_j + \sum_{i=1}^{P} p_i^n \alpha_{j-i+1} \qquad (4)$$

一段遷移確率(one step transition probability) a_{ij}를 식(5)로 정의하면, 遷移確率매트릭스(transition probability matrix) $A = [a_{ij}]$ (i, j = 0, 1, 2…p)는 다음과 같다.

$$a_{ij} \triangleq p[q_{n+1} = j \mid q_n = i] \qquad (5)$$

$$A = \begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_P \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_P \\ 0 & \alpha_0 & \alpha_1 & \alpha_2 & \cdots & \alpha_{P-1} \\ 0 & 0 & \alpha_0 & \alpha_1 & \cdots & \alpha_{P-2} \\ 0 & 0 & 0 & \alpha_0 & \cdots & \alpha_{P-3} \\ \cdots & & & & & \\ \cdots & & & & & \\ 0 & 0 & 0 & 0 & \cdots & \alpha_0 \ \alpha_1 \end{bmatrix}$$

시스템이 q=k인 狀態에 있을 때 共有메모리를 要求할 수 있는 프로세서의 個數는 p−k이므로 메모리 모듈에 대한 要求率 λ_m는 식(6)과 같다.

$$\lambda_m = (p-k)\lambda \qquad (6)$$

이 要求가 指數函數的으로 分布한다고 假定하였으므로 α_k는 식(7), (8)과 같이 된다. 식(7)은 프로세서의 버스占有時間이 ii)의 메모리要求時間과 같을 경우이며, 식(8)은 iii)의 메모리사이클과 같을 경우의 α_k이다.

$$\alpha_k = \frac{(\lambda_m r)^k}{k!} e^{-\lambda_m \cdot r} \qquad (7)$$

$$\alpha_k = \frac{(\lambda_m x)^k}{k!} e^{-\lambda_m \cdot x} \qquad (8)$$

또, (4)식에 의하면 p_j^{n+1}은 식(9)와 같다.

$$\mathbf{P}^{n+1} = \mathbf{P}^n \mathbf{A} \qquad (9)$$

단, $\mathbf{P} = [P_0, P_1, P_2, \cdots\cdots]$

$\mathbf{P}^n$은 n의 값에 無關하므로 $\mathbf{P}^n$을 定常確率벡터 $\widetilde{\mathbf{P}}$로 代置하면 식(9)는 식(10)으로 표현된다.

$$\widetilde{\mathbf{P}} = \lim_{n \to \infty} \mathbf{P}^n \mathbf{A} \qquad (10)$$

여기서, 식(10)은 다음과 같이 反復計算法에 의하여 구해진다.

i) 매트릭스 **A**를 구한다.

ii) **P**의 初期값을 設定한다. (여기서는 $P_0 = 1$, $P_1 = P_2 = \cdots = P_P = 0$으로 하였다.)

iii) $\mathbf{P}' = \mathbf{P} \cdot \mathbf{A}$에 의하여 새로운 **P**의 값을 計算한다.

iv) 새로 計算된 $\mathbf{P}'$의 값과 과거의 **P**와의 差異가 許容誤差以內에 들 때까지 iii)을 反復한다.

q=k인 狀態에서는 k−1개의 프로세서가 待機狀態에 들므로 共有메모리를 待機할 프로세서수의 期待값 w_m, 프로세서의 共有메모리待機率 P_{wm} 및 效率η는 각각 식(11), (12), (13)과 같다.

$$w_m = \sum_{k=1}^{P} (k-1) P_k \qquad (11)$$

$$P_{wm}=w_m/p \tag{12}$$

$$\eta=1-P_{wm} \tag{13}$$

그림14는 γ와 프로세서의 個數p의 增加에 따른 效率η의 變化를 그린것이다.

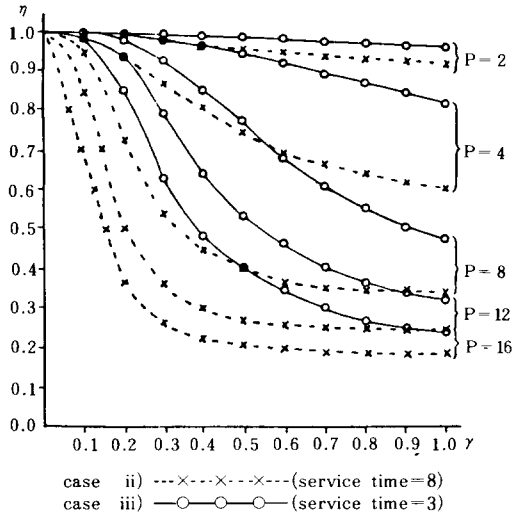


그림14. 單一共有메모리시스템에서 메모리衝突로 인한 效率低下

Fig. 14. Efficiency degradation due to memory conflicts for single shared memory.

2. 竝列메모리시스템의 경우

m개 메모리에 대한 프로세서의 要求가 確率的으로 同一하다고 假定하면, 어떤 프로세서의 어떤 메모리모듈 Mi에 대한 要求率 λ는 식(14)가 된다.

$$\lambda=\gamma/(r\cdot m) \tag{14}$$

한개 以上의 프로세서가 Mi를 要求하면 그중 한개만 承認되고 나머지는 待機狀態에 들게된다. 프로세서가 Mi를 待機할 確率을 P_{wm}이라면 Mi이외의 다른 모듈을 待機할 確率도 同一하므로 어떤 프로세서가 Mi이외의 메모리모듈을 待機할 確率은 $(m-1)P_{wm}$이 된다. 따라서, 실제 Mi에 대한 要求率 λ는 식(15)와 같이 減少된다.

$$\lambda=\gamma\{1-(m-1)P_{wm}\}/(r\cdot m) \tag{15}$$

식(15)는 다른 其有메모리를 待機할 確率을 감안한 것이나 버스를 待機하고 있을 確率도 고려하면, 프로세서가 버스의 타임슬로트를 待機하고 있을 確率을 P_{wm}라면 實際의 要求率λ는 식(16)과 같이된다.

$$\lambda=\gamma\{1-(m-1)P_{wm}\}(1-P_{wb})/(r\cdot m) \tag{16}$$

프로세서의 메모리모듈 Mi에 대한 要求가 指數函數的으로 分布된다고 假定하면 p개의 프로세서와 Mi와 關係도 M/D/1//p型 Markov連鎖가 된다.

Mi에 대한 어떤 要求가 서비스를 받고 시스템을 벗어난 순간, 남아있는 要求의 個數가 k개일 確率을 Pk라면 Mi를 待機할 프로세서의 期待값w_m 및 其有메모리 待機確率P_{wm_1}'은 각각 식(17), (18)이 된다.

$$w_m=\sum_{k=1}^{P}(k-1)P_k \tag{17}$$

$$P_{wm1}'=w_m/P \tag{18}$$

이상의 一連의 식에서 P_{wm_1}'은 P_{wm_1}과 函數關係에 있으므로 反復計算法에 의하여 $P_{wm_1}'=P_{wm_1}$이 되는 점이 Mi에 대한 待機確率이 된다. 시스템에는 m개의 共有메모리 모듈이 있으므로 總 共有메모리에 대한 待機確率 P_{wm}은 식(19)와 같이 된다.

$$P_{wm}=m\cdot P_{wm_1} \tag{19}$$

버스待機確率 P_{wb}는 다음의 過程으로 구해진다. 타임슬로트를 競合하는 프로세서의 個數는 p/s, 要求率은 λ로서 指數函數的으로 分布한다고 假定하면, 버스의 實效서비스時間은 s·b로서 一定하므로 이 시스템도 M/D/1// 型 Markov連鎖가 된다. 프로세서가 其有메모리를 待機하고 있는 狀態를 wait狀態, 그렇지 않은 狀態를 active狀態라 하면, active狀態일때 프로세서의 버스타임슬로트에 대한 要求率 λ_a는 식(20)과 같으며, wait狀態일 때는 每 s.b時間마다 타임슬로트를 要求하므로 버스타임슬로트에 대한 要求率 λ_w는 식(21)과 같다.

$$\lambda_w=\gamma/r \tag{20}$$

$$\lambda_w=1/(s\cdot b) \tag{21}$$

따라서, 프로세서의 타임슬로트에 대한 要求率 λ_0는 식(22)와 같이 된다.

$$\lambda_0=\lambda_a(1-P_{wm})+\lambda_w\cdot P_{wm} \tag{22}$$

遷移確率매트릭스 A는 다음과 같으며 p/s개 프로세서의 타임슬로트에 대한 實際 要求率 λ_b 및 α_k는 각각 식(23), (24)와 같다.

$$\mathbf{A}=\begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{p/s} \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_{p/s} \\ 0 & \alpha_0 & \alpha_2 & \alpha_2 & \cdots & \alpha_{p/s-1} \\ 0 & 0 & \alpha_1 & \alpha_1 & \cdots & \alpha_{p/s-2} \\ 0 & 0 & 0_0 & \alpha_0 & \cdots & \alpha_{p/s-3} \\ \cdot & \cdot & \cdot & \cdot & \cdots & \\ \cdot & \cdot & \cdot & \cdot & \cdots & \\ 0 & 0 & 0 & 0 & \cdots & \alpha_0\ \alpha_1 \end{bmatrix}$$

$$\lambda_b = \left(\frac{p}{s} - k\right)\lambda_0 \tag{23}$$

$$\alpha_k = \frac{(\lambda_b \cdot s \cdot b)^k}{k!} e^{-\lambda_b \cdot s \cdot b} \tag{24}$$

버스에 대한 어떤 要求가 서비스를 받고 시스템을 떠난, 순간, 남아있는 要求의 個數가 k개일 確率을 Pk라면, 共有버스의 타임슬로트를 待機할 프로세서의 期待값 w_b, 共有버스待機確率 P_{wb}'는 각각 식(25), (26)과 같다.

$$w_b = \sum_{k=1}^{\frac{p}{s}} (k-1) P_k \tag{25}$$

$$P_{wb}' = s \cdot w_b / p \tag{26}$$

以上의 式에서 P_{wb}'는 P_{wb}와 函數關係에 있으므로 反復計算法에 의하여 $P_{wb}' = P_{wb}$ 되는 점이 共有버스 待機確率이 된다. 以上의 共有메모리待機確率과 共有버스待機確率에 의하여 프로세서의 效率 η는 식(27)와 같다.

$$\eta = (1 - P_{wm})(1 - P_{wb}) \tag{27}$$

그림(15)는 (b, s, r, x) = (1, 2, 8, 3)인 시스템에서 p= 4, 8, 12, 16인 경우의 γ 및 m의 增加에 따른 η의 變化를 그린것이다.

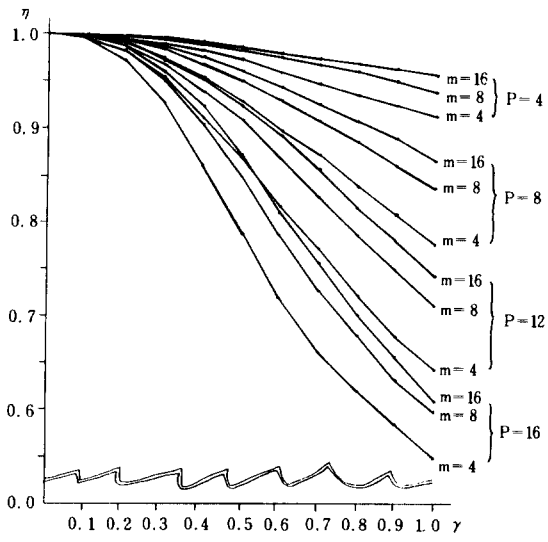


그림15. 인터리브메모리, 파이프라인버스 시스템의 메모리衝突 및 버스競合으로 인한 效率低下

Fig. 15. Efficiency degradation due to shared memory conflicts and bus contentions for the interleaved memory and pipelined bus system.

V. 結 論

單一共有버스의 多重마이크로프로세서 시스템에서 프로세서의 메모리액세스 過程을 파이프라인化하여 프로세서의 버스占有時間을 最小化함으로써 버스의 利用率을 높이는 方法을 考案, 設計하고 그 效率을 分析하였다. 最小버스占有時間은 버스를 통한 버퍼간 情報傳達時間이 되므로 버스의 電氣的特性에 따른 버스遲延時間을 논하였다.

效率分析結果, $\gamma \leqq 0.4$ 에서 $\eta \geqq 0.9$를 許容範圍로 基準하면, 프로세서의 버스占有時間이 메모리要求時間과 같을 경우 그림14에서 p=2인 경우만 許容値 以內에 들며, 메모리週期와 같을 경우 $p \leqq 4$의 경우만 許容値 以內에 든다. 그러나 그림15의 인터리브 메모리方式에서 버스를 파이프라인化할 경우, $p \leqq 16$, $m \geqq 8$ 이면 $\gamma \leqq 0.4$에서 $\eta \geqq 0.9$이므로 그림14의 單一共有메모리와 比較하면 懸隔한 效率向上을 보인다.

버스를 파이프라인化하기 위하여는 PBI 制御器, MBI 制御器 등 약간의 하드웨어만이 追加되므로 낮은 費用으로 높은 效率의 多重마이크로프로세서시스템의 實現이 可能하다.

그러나 버스占有時間의 短縮으로 인하여 外部의 電氣磁氣的影響을 크게 받으므로 信賴性에 問題가 있으며, 이를 解決하기 위하여 穴長技法을 追加할수 있으나 그 結果 하드웨어量이 增加하는 問題가 따르게 된다. 따라서, 본 연구의 파이프라인버스 方式은 높은 信賴性을 要求하지 않으면서 廉價로 高速處理를 必要로 하는 應用分野에 適合하다.

參 考 文 獻

[1] K. Hwang, Computer Architecture and Parallel Processing, Mcgraw-Hill, p. 52, 1984.

[2] F.A. Briggs, "Organization of Semiconductor Memories for Parallel-Pipelined Processors," *IEEE Trans. Comp.*, pp. 162-169 Feb. 1977.

[3] P.M. Kogge, The Architecture of Pipelined Computers, Mcgraw-Hill, p. 21, 1981.

[4] 會和將容, "進行波を 利用した高速リングアービタの一構成法," 日本電子通信學會 論文誌 (D), pp. 519-524, 1983. 5.

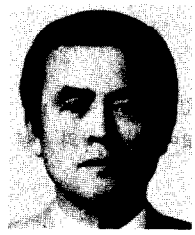
[5] R.V. Balakrishnan, "The Proposed *IEEE 896 Futurebus-A Solution to the Bus Driving Problem," IEEE Micromag.*, p. 23-27, Aug. 1984.

[6] K.T. Fung, "On the Analysis of the Memory Conflicts and Bus Contention in Multiple-Microprocessor System," *IEEE Trans. Comp.*, pp. 28-37, Jan. 1979.
[7] M.A. Marsan, "Markov Models for Multiple Bus Multiprocessor Systems," *IEEE Trans. Comp.*, pp. 239-248, Mar. 1982.
[8] T.N. Mudge, "A Semi-Markov Model for the Performance of Multiple Bus Systems," *IEEE Trans. Comp.*, pp. 934-942, Oct. 1985.
[9] L. Kleinrock, Queueing Systems. Wiley Interscience, pp. 53, 1975.

著 者 紹 介

金 鼎 斗(正會員)

1932年 5月 19日生. 1958年 경북대학교 사범대학 물리과 졸업 이학사. 1971年 경북대학교 교육대학원 졸업 교육학석사학위 취득. 1977年, 1987年 영남대학교 대학원 졸업 공학석사, 공학박사학위 취득. 1966年～1967年 미국 국무성초청 TDP 참가; Oregon 주립대학에서 연수. 1973年～1974年 Colombo Plan 참가, 일본 CEC에서 연수. 1981年～ 1982年 미국 Southern Illinous 대학에서 연수. 1983年～현재 효성여자대학교 전자계산학과 교수. 주관심분야는 계산기구조, system software, 학습 system개발 등임.



孫 潤 求(正會員)

1932年 2月 15日生. 1956年 서울대 공대 전기공학과 졸업 공학사학위 취득. 1974年 영남대학교 대학원 전자공학과 공학박사학위 취득. 1966年～현재 영남대학교 전산공학과 교수. 주관심분야는 Multiprocessor System 임.