4J－01

マルチプロセッサ MTA/TSM の開発

松岡 孝，原口 恵美子，岩根 雅彦
九州工業大学

1. はじめに

マルチプロセッサの同期通信オーバーヘッドを削減するために同期通信用メモリ TCSM[1]が開発されている．TCSM では同期通信の高速化のみを目的としており，共有変数の生存期間を考慮していなかった．これを改善するための共有変数用データメモリ TSM[2]と TSM を搭載したマルチプロセッサ MTA/TSM を開発した．ここでは，MTA/TSM の概要と簡単な性能評価の結果を報告する．

2. TSM の概要

2.1 TSM の構成

TSM の構成を図 1 に示す．TSM は CAM(Content Addressable Memory)で構成されており，CAM の 1 つのエントリは，タグ，データ，カウント，PT フラグ，PM フラグの各フィールドからなっている．タグは各エントリを識別するためのものであり，データフィールドはデータを格納，カウントフィールドはデータの読み出し回数を指定する．PT フラグは 1 のときデータが恒久変数，0 のとき一時変数であることを示す．恒久変数とはタスクの生成時に TSM に割り当てられ，タスクの消滅まで開放されない変数を意味し，一時変数とは変数の定義時(値の代入時)に TSM に割り当てられ，通信終了時(カウント 0)に開放される変数を意味する．PM フラグは恒久変数のモードを切り替えるフラグで，1 のとき通信モード，0 のとき通常モードを表し，プログラム実行中に切り替えることができる．

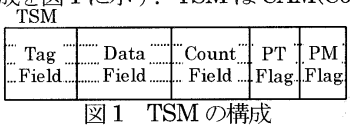


図 1　TSM の構成

2.2 TSM の基本動作

読み出し動作および書き込み動作についてまとめたものを表 1 に示す．読み出しヒットではそのエントリのデータを読み出し，カウントを 1 デクリメントする．読み出し後にカウントが 0 になった場合，そのエントリが一時変数であれば開放し，恒久変数であれば開放しない．また，そのとき書き込みブロックされているプロセッサのブロック解除を要求する．読み出しミスでは読み出しを行ったプロセッサのブロックを要求する．書き込みヒットではタグ，データ，カウント，PT，PM を指定されたエントリに書き込み，読み出しでブロックされているプロセッサのブロック解除を要求する．書き込みミスでは書き込みを行ったプロセッサのブロックを要求する．リセット動作では，タグが一致したエントリのカウントまたは PT フラグをリセットする．

表 1　TSM の読出し/書き込み動作

Tag	PT Flag	PM Flag	Count	Read TSM	Write TS	Write Place
Hit	0	—	0	Read Miss	Write Hit	PSEL
			non zero	Read Hit	Write Miss	—
	1	0	—	Read Hit	Write Hit	Hit Entry
		1	0	Read Miss	Write Hit	Hit Entry
			non zero	Read Hit	Write Miss	—
Miss	—	—	—	Read Miss	Write Hit	PSEL

Hit Entry：タグが一致したエントリ，PSEL：優先順位の高い空エントリ

3. プログラミングモデル

3.1 メモリ空間

図2に示すようにTSMはメモリ空間の一部にマッピングする．TSM へのアクセスにはメモリアクセス命令(mov 命令)を用いて行い，そのアドレス指定部にタグ，カウント，PT，PM を埋め込む．

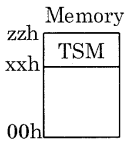
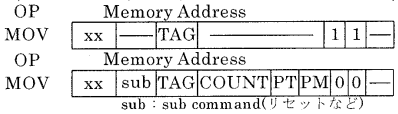


図 2 メモリ空間



3.2 変数割り当て

同一タスクに属するスレッド間で協調処理を行う場合，スレッド間の共有変数を TSM に割り付けて扱う．

型宣言文　real shared perm　a；
　　real shared temp　x；

によって変数 a は恒久変数，変数 x は一時変数として扱われる．また TSM への変数割り当ての際に，タグにタスク ID，共有変数 ID を持たせることにより，各タスクで TSM の保護が自動的に行える．さらに TSM のエントリは動的に割り当てられるのでエントリの有効利用および割り当てにおけるオーバーヘッドが削減できる．

4. TSM を用いた同期通信

図 3，4 に恒久変数および一時変数を用いた同期通信の例を示す．スレッド 1 がスレッド 2 を生成したとする．恒久変数を用いた同期通信では，始め非同期モードとして生成された恒久変数は読出し/書き換えは自由に行うことができる．この恒久変数を同期モードに切り替えるとカウントが 0 の間は読出しがブロックされる．その後，有効データが書き込まれるとそのカウント数だけ読み出

すことができる．ただし，カウントが 0 になっても開放されない．一方，一時変数を用いた同期通信では，スレッド 2 が先に読み出しを行ってもまだその変数は生成されていないのでブロックされる．変数が一時変数として生成されるとそのカウント数だけ読み出すことができるが．カウントが 0 になるとその変数は消滅してしまう．

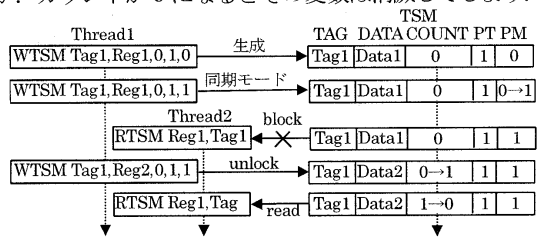


図 3 恒久変数を用いた同期通信

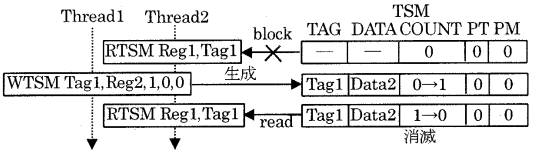


図 4 一時変数を用いた同期通信

5. マルチチップマルチプロセッサ MTA/TSM

5.1 システム構成

TSM の機能および基本性能の評価のためにマルチチップマルチプロセッサ MTA/TSM を開発した．図 5 に MTA/TSM のシステム構成を示す．Intel 社の 486DX2

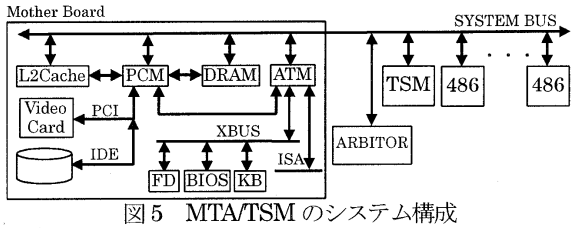


図 5 MTA/TSM のシステム構成

プロセッサを 8 台，TSM，集中型バスアービタ(ARBITOR)を搭載しており，これらをドータボードとして構成し，市販の PC/AT 互換機用マザーボードのプロセッサソケットに接続してある．TSM, ARBITOR は Xilinx 社の FPGA(XC4013E)を用いて実装した．486DX2, TSM, ARBITOR，チップセット(PCM，ATM)，メモリ，L2 キャッシュは単一のシステムバスに接続されている．システムバスの調停は ARBITOR が行い優先度は回転式である．また，L1 キャッシュはライトスルー，L2 キャッシュはライトバックであり，キャッシュの無効化はライトインバリデート方式である．現在の TSM の仕様はタグ 8 ビット，データ 32 ビット，カウンタ 3 ビット，PT1 ビット，PM1 ビット，エントリ数は 16 である．

5.2 基本性能評価

TSM およびメモリを用いて 1 対多通信，相互排除，バリア同期を行った実験結果を図 6，7，8 に示す．実験では 1 つのスレッドを 1 台のプロセッサに割り当てた．いずれの場合も，TSM を用いた方がよい結果を得られた．メモリでは，フラグを用いて同期を行っておりそのフラグチェックのバスサイクルが多いことがこのような結果につながっている．

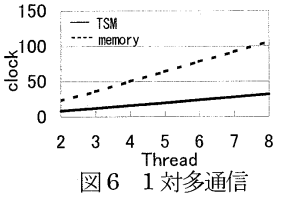


図 6 1 対多通信

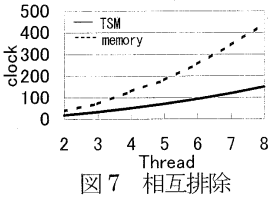


図 7 相互排除

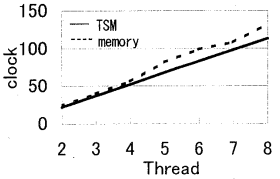


図 8 バリア同期

次に図 9 に示すループプログラムを doacross の形で TSM，メモリを用いて並列化したもの，および X[i]を求める第 1 ループと A[i]を求める第 2 ループに分割し並列化したものを実験した．そのシリアル版に対する速度向上比を図 10 に示す．メモリでは，データ生成を示すフラグを用いてデータ生成を保証している．ループ分割では第 1 ループと第 2 ループ間にメモリを用いたバリア同期を挿入している．TSM はシリアルに対して 1.2 倍程度の速度向上が得られた．メモリではフラグチェックによるバスサイクルの増大，ループ分割ではバリア同期のオーバーヘッドによって図 10 に示すような結果となった．また，いずれの場合も台数効果が得られていないのは細粒度並列処理であり処理時間のほとんどがバスサイクルであるためである．

```
int X[N],Y[N], Z[N], A[N];
for( i = 1; i <= N; i++){
   X[i] = (Y[i] + Z[i])*Y[i];
   A[i] = X[i – 1] + 1;
}
```

図 9 ループプログラム

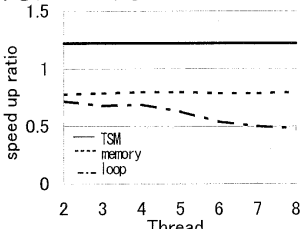


図 10 doacross(N=8)

7. むすび

実験結果から TSM を搭載したマルチプロセッサ MTA/TSM を用いての 1 対多通信，相互排除，バリア同期，ループプログラム共に TSM を用いた方がメモリよりもよい結果を得た．今後，エントリの増加，各種ベンチマークによる評価を行う．

参考文献

[1]岩根，山脇，田中，“マルチプロセッサオンチップにおける CAM を用いた同期通信用メモリ，”電子情報通信学会，論文誌 D1，2000(採録決定)

[2]岩根，“タグ付き共有変数用メモリ設計メモ，”九州工業大学 1999.1(研究室内部資料)