デュアル・バス型マルチプロセッサ・システム

4C-9

小原盛幹[1]、佐藤豪芳[2]、ワタリ・シゲル[3]、永松礼夫[3]、吉田紀彦[4]、森下巖[3]

[1]日本アイ・ビー・エム株式会社　サイエンス・インスティチュート
[2]株式会社　東芝　[3]東京大学工学部　[4]九州大学工学部

1　はじめに

我々は共有バス型アーキテクチャのマルチプロセッサ・システムとして、デュアル・バス型の方式を提案した［1］。ここではこの方式で実際に試作したハードウェアとその性能について述べる［2］。

本システムは、並列に処理されるプロセス間で同期をとるための（メッセージと呼ばれる）小容量のデータを送るメッセージ・バスと、それ以外の大容量のデータを送るデータ・バスの2つのバスを共有する、共有バス型マルチプロセッサ・システムである（図1）。

実際に試作したシステムは、4台のコンピュータ・モジュール（CM）から構成されている。各CMはCPU（MC68000，8MHz）、ローカル・メモリ、メッセージ・バスのコントローラなどからなっている。CPUとローカル・メモリには、既製品のVMEバス規格のボードを使用した。また、各CPUはプログラムとデータを自分のローカル・メモリに持っていて、独立に命令を実行する。

2　デュアル・バス方式

2．1　メッセージ・バス

各CMは送信用、受信用のメイル・ボックスを持っている。CPUはメイル・ボックスへの読み書きによって、他のCPUとの通信を行う。メッセージ・バスはこのメイル・ボックス内のメッセージを転送する。

メッセージ・バスのコントローラは送信コントローラ（Sender）、受信コントローラ（Receiver）、Arbiterから構成されている。Senderは送るべきメッセージがあると、Arbiterに対して送信要求を出し、送信許可を受け取るとバスにメッセージを出力する。メッセージ転送の最初のサイクルでは宛先プロセスのIDが送信される。Receiverはバスを常に監視していて、自分宛のメッセージがあれば、自分の状態を返送する。Receiverが受信可能状態であれば、Senderはメッセージの続きを送信し、Receiverはメッセージを受信用メイル・ボックスに取り込む。Receiverが受信不能状態であれば、Senderは送信をやめる。後者の場合、通常は再送信を行う。

メッセージ・バスはなるべく高速であることが望ましい。CPUのバス・サイクルとメッセージ・バスのサイクルの時間差を吸収するため、受信用、送信用にそれぞれメッセージ・バッファを持つことにした。これは前述のメイル・ボックスに相当するが、メッセージ1つ分のスロットのみを持つ。2番目以降のメッセージはCPUのローカル・メモリ上に置かれる。

Arbiterはトークン・リング方式を採用した。バスの予約権を示すトークンがチェインのリング上を回っている。トークンを受け取ったCMはバスの要求があるときにはバスの予約を行う。バス要求がないか、バスの使用を開始すると、トークンを次のCMに回す。ここでバスの予約を行っているのは、バス使用中に次の使用者を決定するためである。

実際に試作した回路では、トークンは62．5nsで次のCMへ伝わる。バス幅は1word（16bit）、バス上の転送時間は125ns／wordである。また、論理デバイスとして主にLS－TTLを使用した。ただし、一部高速性を要求されるところにS－TTL、FAST－TTLを使用している。ICの個数は1コントローラ（Sender、Receiver、Arbiter）当り、約60個である。バッファにはアクセス・タイム35nsのスタティックRAMを使用している。

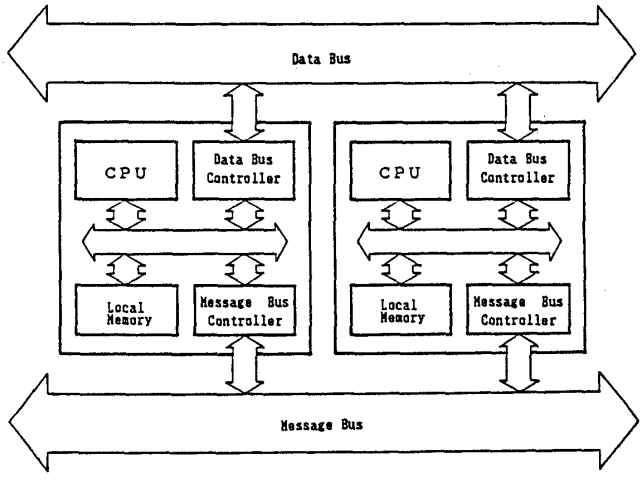

図1　デュアル・バス型マルチプロセッサ・システム

Design of Dual Bus Multiprocessor System

M.Ohara[1], T.Satoh[2], S.Watari[3], L.Nagamatsu[3], N.Yoshida[4], I.Morishita[3]
[1]Science Institute, IBM Japan, Ltd.　[2]TOSHIBA CORPORATION
[3]University of Tokyo　[4]Kyushu University

２．２　データ・バス

今回の試作システムで使用した既製品のＣＰＵボードでは、ローカル・メモリをＶＭＥバスからアクセスできない。このため、ＶＭＥバス上の共有メモリを経由してＣＭ間のデータ転送を行っている。しかし、ローカル・メモリをＶＭＥバスからアクセスでき、ローカル・メモリ間でＤＭＡ転送ができるようにすれば、ＶＭＥバス上でより高速なデータ・バスを実現することができる。

３　メッセージ・バスの評価

メッセージは最初、ＣＰＵのローカル・メモリ上に組み立てられる。このため、ＣＰＵ間のメッセージ通信は次の手順で行われる。

1）ローカル・メモリ上のメッセージをメッセージ・バスのバッファに転送する。
2）メッセージ・バスを経由して、相手のＣＰＵのメッセージ・バスのバッファに転送する。
3）相手のＣＰＵが、バッファ上のメッセージを自分のローカル・メモリに読み込む。

2）は、メッセージ・バスのコントローラによって行われるが、1）と3）はソフトウェアによって行われる。1）～3）を合計すると、ＣＰＵ間の通信時間は３．８８μｓ／wordであった。さらに、メッセージ・バスにはコンテンションが発生する。シミュレーションによって、コンテンションのある場合の通信時間を求めた。図２はメッセージの発生間隔やＣＭの個数を変化させたときの通信時間である。ここで、メッセージの発生間隔は、最大メッセージ発生間隔（Ｉmax）以下の正値をとる一様乱数である。曲線ＡはＩmaxを５０μｓとしてＣＭの台数を変化させたときの通信時間を示している。

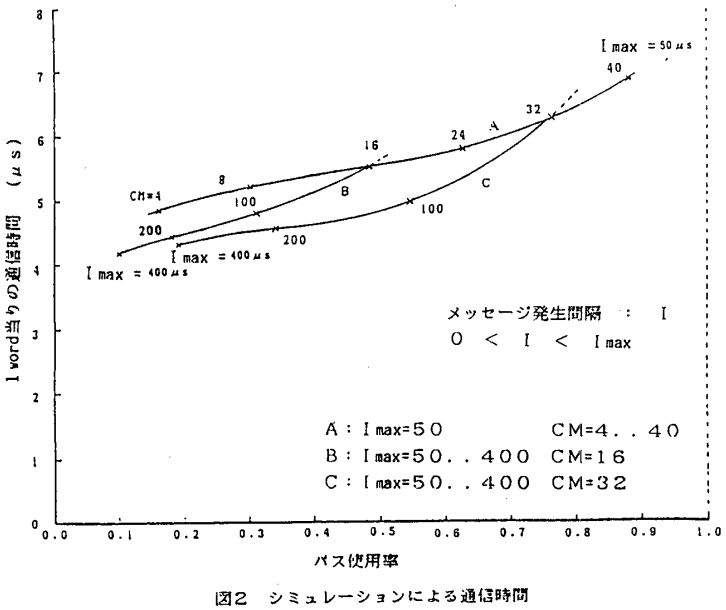

図2　シミュレーションによる通信時間

また曲線Ｂ、Ｃは、それぞれＣＭが１６台、３２台の場合に、Ｉmaxを変化させたときの通信時間を示している。時間ＩはＣＰＵがＯＳやプロセスを実行する時間であるから、５０μｓは現在使用している６８０００としては非常に短い。また通常のＴＴＬを使った場合、ファンアウトの制約から、いろいろ工夫を行ったとしても、１つのバスに接続できるＣＭの台数は高々３２台である。このため、パラメータを変えても曲線Ａと曲線Ｃにはさまれた領域の中にある。最悪の場合でも通信時間がコンテンションのない場合の２倍以上になることはない。

４　システムとしての評価

本システムの上に、通信制御、プロセス管理などを行うＯＳのカーネルを実現してみた［３］。このＯＳでは、メッセージ通信にblocking send方式を採用し、メッセージの送信や受信のたびにプロセス切り換えを行っている。コンテンションがない場合、ＯＳによるＣＰＵ間のメッセージ往復時間は８４５μｓ（メッセージ長＝７word）であった。

ここで、ＣＰＵ間往復の転送時間は、５４．３μｓ（３．８８μｓ／word × 7word × 2）であり、全通信時間の６％を占めるに過ぎない。プロセスの切り換えやその他のオーバーヘッドが大きいことがわかる。

５　おわりに

メッセージを転送するという意味では、十分な性能のメッセージ・バスを試作することができた。今後の課題は、通信にともなうＯＳのオーバーヘッドをなるべく少なくするようなハードウェアを提供することである。たとえば、現在バッファはメッセージを１つまでしか持つことができないが、全てのメッセージをコントローラのバッファ上に持ち、メッセージの管理を高速かつ容易にするなどの方法が考えられる。

参考文献

［１］ワタリ、吉田、永松、森下："メッセージ／データ通信用　２重共有バス型マルチプロセッサシステム"、第３２回情報処理学会全国大会論文集、5Q-6(1986)
［２］小原："共有バス型マルチプロセッサシステムの研究"、東京大学工学部計数工学科卒業論文(1986,3)
［３］ワタリ："Design of an Operating System Nucleus for Shared Bus Multiprocessor Systems"、東京大学工学部計数工学科修士論文(1986,3)