計算機アーキテクチャ　104－3
システムソフトウェアと
オペレーティング・システム　62－3
（１９９４．　１．　２７）

Mach3.0 実装ボードにおけるシステムバスの測定

金原 弘幸　　丸山 輝幸

株式会社 リコー 先駆技術研究所

我々は、MBus と SPARC プロセッサを使ったハードウェアプラットフォームを開発し、Mach 3.0 を移植した。このプラットフォームの性能評価の一環として、MBus モニタボードを開発し、MBus トランザクションを測定した。簡単な画像処理を想定したテストプログラムを作成して Mach 3.0 上で実行させ、コピーバックモードとライトスルーモードにおける MBus トランザクションを測定したので、その結果について本稿で報告する。

Measurement of the systembus transaction on Mach 3.0 implemented board

Hiroyuki KINBARA　　Teruyuki MARUYAMA

RICOH Co., Ltd. Advanced Technology Development Center.

This paper describes the measurement of MBus transaction which we adopted for our hardware-platform with SPARC processor. We also ported Mach3.0 to this platform, and developed an MBus monitor board. The measurement was done for copy-back mode and write-through mode with the MBus monitor board, by executing simple image processing test program on Mach 3.0 .

1 はじめに

コピアやファックスといったOA機器の高機能化、カラー化にともない多量のデータがオフィスで扱われるようになってきた。更にこれら多量のデータは、1つのOA機器の中の処理に留まらず広くネットワークを介した分散環境で処理されることが望まれている。そのためにはより多くのデータを高速に処理が行なえるハードウェア・プラットフォームと、それをサポートするソフトウェア・プラットフォームが必要とされている。

このようの状況の中で我々は、将来のOA機器のベースとなるプラットフォームの開発を段階的に進めている。常に高機能化に対応して適切なプラットフォームを提供していくためには開発したプラットフォームを正確に評価し次ステップに活かしていかなければならない。

今回我々はプラットフォームとしてMBus[1]とSPARCプロセッサ[2]を使ったボードを開発し、OSとしてMach3.0[3]を移植した。そのプラットフォームの評価の一環として、システムバスの動的な使用率の測定を行なった。それはシステムバスの動的なトラフィックの測定により、多量データの高速処理に対応するだけのバスのバンド幅があるか、プロセッサの高速化によりプラットフォームの性能を更に上げられるかを知るためである。

システムバスの使用率はロジックアナライザなどの市販の計測器で測定することも考えられるが、長時間のデータをサンプリングすることができず、高性能の専用システムにおいても性能評価支援ツールとしてソフトウェアモニタ、ハードウェアモニタの試作が報告されている[4]。今回、我々もプラットフォームのシステムバスをモニタするための専用ボードを開発した。

システムバスの使用率に影響を及ぼすと考えられるものにキャッシュモードがあげられる。一般的にコピーバックとライトスルーを比較すると、コピーバックではメモリのバンド幅をそれほど必要とせず、バスのトラフィックを減らすと言われている[5]。そこで、我々は簡単な画像処理のテストプログラムを作成し、コピーバックとライトスルーにおけるバスの使用率を実測した。

本稿では、はじめにプラットフォームについて説明を行ない、続いてシステムバスの使用率を測定するためのMBusモニタについて説明し、2種類のキャッシュモードを整理した後、バストランザクションの測定について考察を行なう。

2 Mach3.0実装ボード

ハードウェア・プラットフォームとしてSPARCプロセッサを使ったクロック40Mhzのボードを開発し、ソフトウェア・プラットフォームとして分散OSMach3.0をそのボード上に移植した。

このプラットフォームの上でシステムバスの使用率の測定を行なった。

2.1 ハードウェア・プラットフォーム

今回我々は、分散環境における多量データの高速処理に対応できることをを目的として、SPARCプロセッサを使ったハードウェア・プラットフォームを開発した。図1にこのハードウェア・プラットフォームの構成を示す。

このプラットフォームは、システムバスとしてMBus、I/OバスとしてSBus、及び8ビット標準I/Oバスを採用している。

- MBusは、320Mバイト/秒の大きなバンド幅を持つマルチプレクス型の同期バスである。64ビットのデータ幅を持ち40MHzで動作する。マルチ・プロセッサ・システムへの対応も考えられており、キャッシュ・コヒーレンシ・プロトコルもサポートしている。
- SBusは、32ビットのデータ幅を持ち最大25MHzで動作するバスである。SBusは拡張性が高く、市販あるいは内製のSBus拡張ボードをSBusスロットに接続することでシステムの拡張が容易に行なえる。
- 8ビット標準I/Oバスは、標準的なPROM、リアルタイムクロック、シリアルコントローラを接続した低速用I/Oバスである。

次に本ハードウェア・プラットフォームを構成するMBusモジュールについて説明する。

- SPARCプロセッサ・モジュール
  40MHzで動作するSPARCチップセットを使用した。IU(Integer Unit, CY7C601)、FPU(-Floating Processor Unit, CY7C602)、MMU(-

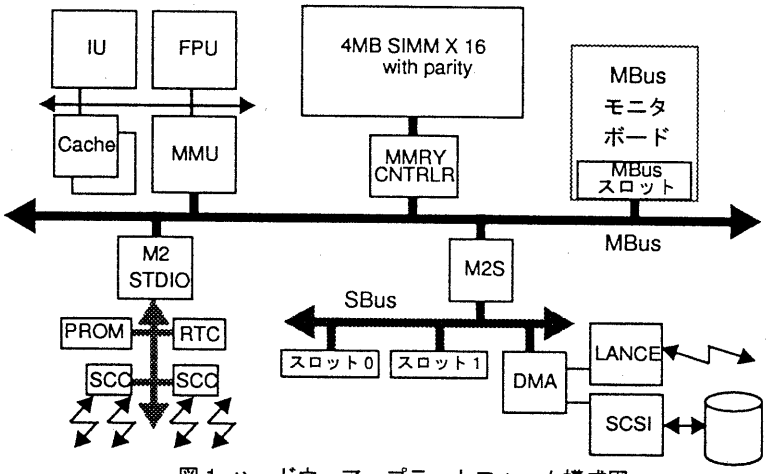


図1 ハードウェア・プラットフォーム構成図

Cache Controller &Memory Management Unit, CY7C604) 及びインストラクション/データの区別のない 64K バイトのキャッシュメモリにより構成される。

- メイン・メモリ・モジュール
  4M バイトの SIMM を 16 個用いてパリティ付きの 64M バイトの主記憶を実現している。
- 標準 I/O モジュール
  標準の PROM,RTC(Real Time Counter) 及び 2 つの SCC(Serial Comunication Controller) を含み、4 シリアル・ラインをサポートしている。またシステムの割り込みを管理している。
- SBus モジュール
  SBus 空間を MBus 空間にマップすることにより、SBus 空間を MBus 上のモジュールとして扱う。この SBus モジュールには、Ethernet をサポートするための LANCE と、SCSI コントローラが用意されている。また 2 つの SBus スロットに拡張ボードを接続することにより容易に I/O システムを拡張することができる。
- MBus スロット/MBus モニタ
  MBus モジュール・ボードを接続するスロットで、今回、システムバスの測定を行なうための MBus モニタ・ボードを組み込んでいる。

2 2 ソフトウェア・プラットフォーム

今回ソフトウェア・プラットフォームとして分散 OS Mach3.0 を選び、ハードウェア・プラットフォーム上に移植した。Mach3.0 は MK83（マイクロカーネル）と UX42（UX サーバ）で構成されている。

この Mach3.0 の移植を行なうためにカーネギメロン大学から Mach のソースコードを入手し、ハードウェア依存部分は mips 版を参考に作成を行なった。作成した箇所は主に割込み処理、メモリ管理モジュール、コンテキスト切替え処理、デバイスドライバ、およびカーネルデバッガである。

ここではシステムバスの測定に関連すると思われるメモリ管理とキャッシュコントロールについて MMU の機能と合わせて説明する。

使用している MMU はオンチップの 64 エントリの TLB を持ち、64K バイトのダイレクトマップ仮想キャッシュ、ツリー構造のテーブルウォークをサポートしている。また MMU はコピーバックとライトスルーの 2 種類のキャッシュモードをサポートしている。このモードは MMU の初期化の時点でコントロールレジスタの設定により切替えることができる。

仮想アドレスから物理アドレスへの変換テーブルは、全て物理メモリ上に置かれ、TLB ミス時 MMU により参照 (テーブルウォーク) される。

テーブルウォークは MMU 内のコンテキストレジスタにセットされたルートテーブルのアドレス (コ

ンテキストテーブル) からレベル 1、レベル 2、レベル 3 ページテーブルの順に PTE が見つかるまで行なわれ物理アドレスが生成される。なお PTE は物理ページ番号、アクセスレベル (プロテクション)、キャッシュ可、変更ビット、参照ビットからなる。

メモリ管理モジュールでは、レベル 3 テーブルまでの 4 つのテーブルを使い仮想アドレスから物理アドレスへの変換を行なっている。ただし SCC、SCSI、LANCE 等のデバイスのレジスタはコンテキストテーブルとレベル 1 テーブルのみで参照することができるようにマップしている。

3 MBus モニタ

システムバスの使用率を測定するために、我々は MBus モニタと呼ぶ MBus の状態をモニタリングし、バス使用パターンをカウントする専用のハードウェアと、これをサポートするためのソフトウェアの開発を行なった。

3.1 MBus モニタ・ボードのハードウェア構成

MBus モニタ・ボードは、ハードウェア・プラットフォームのシステムバスである MBus の使用率等の測定を行なう MBus モジュールボードであり、プラットフォームの MBus スロットに接続して MBus のモニタリングを行なうことができる。

MBus モニタ・ボードは図 2 に示すように MBus インターフェース部、タイム・モニタ部と複数の測定エレメントからなるマスタ・モニタ部により構成されており、内部バスにより接続されている。

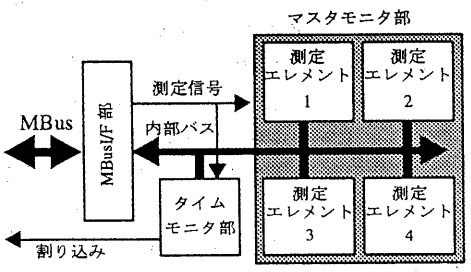


図 2 MBus モニタ構成図

以下に構成要素の概要を示す。

- MBus インターフェース部

MBus マスタから本ボードへのリード/ライト・アクセスを受けて、タイム・モニタ部及びマスタ・モニタ部の測定エレメントの内部レジスタのリード/ライト・コントロール信号を出力し、内部バス、MBus 間のデータのやりとりを行なう。また MBus トランザクションより測定信号を作りだし、トランザクション情報とともにタイム・モニタ部及びマスタ・モニタ部に対して出力する。

- タイム・モニタ部

MBus インターフェース部からの時間測定信号を受け、モニタリング開始からの経過時間の測定及びモニタリング中のバス・ビジー時間の測定を行なう。タイム・モニタ部は、それぞれの測定を行なうカウント・レジスタにより構成されており、これらのカウントレジスタが、MBus のクロック周期である 25ns 単位で測定を行なう。これらのレジスタを読み込むことでそれぞれの測定値が得られる。ただしこれらの測定値は実測定値を 4 ビットシフトした値で 400ns 単位の値を示している。例えばモニタの結果得られたバス・ビジー時間の測定値が 16 であったならば、モニタリング中のバスのビジー時間として

$$16 \times 400\text{ns} = 6.4\ \mu\text{s}$$

が得られる。

またモニタリングの時間を設定することにより、設定時間が経過した後にモニタリングを停止させるとともに割り込みをかけることができ、これにより実時間ベースのバス・モニタリングが可能である。

- 測定エレメント (マスタ・モニタ部)

MBus インターフェース部からのトランザクション情報と測定信号を受け、このトランザクション情報をカウント条件として、条件別のトランザクション回数のカウントを行なう。各測定エレメントは、それぞれの測定を行なうカウント・レジスタにより構成されており、これらのレジスタの読み込みによりそれぞれの条件別の測定値が得られる。

タイムモニタ部及びマスタモニタ部の測定エレメントは汎用の Complex PLD で実現されているの

で、HDLプログラムによりフレキシブルにカウント条件を設定することが可能であり、さまざまな測定項目をモニタリングすることができる。これにより、MBusトランザクションの広範囲のモニタリングが可能である。以下に設定可能なカウント条件を示す。

- MBusトランザクション・マスタ
  MBusトランザクションを起こしたバス・マスタ
- MBusトランザクション・タイプ
  MBusトランザクションがリード・アクセスかライト・アクセスかのタイプを示す
- MBusトランザクション・サイズ
  MBusトランザクションのサイズは、バイト／ハーフ・ワード／ワード／ダブル・ワード／16バイト・バースト／32バイト・バースト／64バイト・バースト／128バイト・バーストのいずれかを示す
- MBus Memory InHibited
  Mbusトランザクションにおいて、MIH(Memory InHibit)信号がアサートされたかを示す
- MBus cache block SHared
  MBusトランザクションにおいて、MSH(MBus cache block SHare)信号がアサートされたかを示す

また、これらの測定条件を組み合わせることで、より細かな条件下の測定が可能となる。例えば、MBusトランザクション・マスタがMID(MBus Module IDentifier signal)=Eのバス・マスタであるという条件、MBusトランザクションがリード・アクセスであるという条件、MBusトランザクション・サイズがワードであるという条件を組み合わせることで、MID=Eのバス・マスタがワードサイズのリード・アクセスを行なった回数を測ることができる。

3.2 MBusモニタの制御

MBusモニタの制御は全てソフトウェアから行なっている。よくハードウェアの制御は、デバイスドライバを作成しハードウェアからの割込みによりデバイスドライバで制御する方法で行なわれる。MBusモニタ・ボードでシステムバスの状態を測定している場合、MBusモニタ・ボードから割込みがデバイスドライバに入ってくるまでの時間に必要以上のトランザクションがMBusモニタ・ボードに測定されてしまう。この不必要な測定を極力減らし、測定精度を上げるために我々は割込み駆動のデバイスドライバを作成せずに、MBusモニタ・ボードのレジスタを仮想アドレスにマップし、直接アクセスすることとした。

またカーネル内の測定だけでなくユーザレベルでの測定を行なうため、ユーザからのアクセスも可能となるように仮想アドレスへのマップを行なった。

マップしたアドレスに対しアクセスすることでMBusモニタ・ボードを制御できる。

3.3 測定項目

測定対象のハードウェア・プラットフォームはユニ・プロセッサ・システムであり、そのプロセッサ・モジュールがMBusに対して行なったトランザクションの測定を行なった。以下のすべての項目はMBusモニタ・ボードのタイム・モニタ部及び測定エレメントの内部カウント・レジスタにおいて測定されるものであり、これらの測定項目の測定値は、モニタリング開始からの計数値として得られる。

- モニタリング時間
  モニタリングの開始からの経過時間
- バス・ビジー時間
  システムバスがバス・マスタによるトランザクションによりビジーとなっている時間。
- CPU_RN
  プロセッサ・モジュールがメモリ・アドレス空間に対してノン・バースト・リード・アクセスした回数。このアクセスは、一般にプロセッサ・モジュールがテーブル・ウォークする時に行なわれる。
- CPU_RB
  プロセッサ・モジュールがメモリ・アドレス空間に対してバースト・リード・アクセスした回数。このアクセスは、一般にプロセッサ・モジュールのキャッシュがミスした時のキャッシュ・ラインの詰め込み動作として行なわれる。

- CPU_WN

  プロセッサ・モジュールがメモリ・アドレス空間に対してノン・バースト・ライト・アクセスした回数。このアクセスは、一般にライトスルー時のデータ・ライトにおいて行なわれる。

- CPU_WB

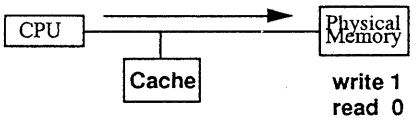
  プロセッサ・モジュールがメモリ・アドレス空間に対してバースト・ライト・アクセスした回数。このアクセスは、一般にコピーバック時のデータ・ライトにおいて行なわれる

4 コピーバックとライトスルーモード

我々はある条件下でコピーバックとライトスルーにおいてMBusのトランザクションを測定した。ここでは2つのキャッシュモードで、物理メモリがアクセスされる状況について、特にシステムバスの測定に影響すると思われるライトアクセスのキャッシュミスの場合について整理しておく。

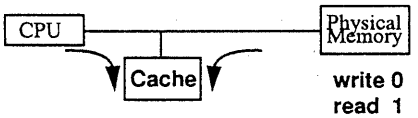
(1). ライトスルーモード

データを物理メモリへ書き出し、キャッシュは更新しない。



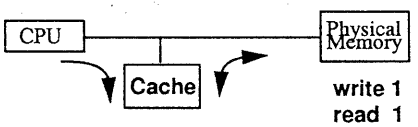
(2). コピーバックモード、クリーン

書き込みを行なうキャッシュラインを物理メモリから読み込み、キャッシュ上のデータを更新する。



(3). コピーバックモード、Modified

書き込みを行なうキャッシュラインのModifiedデータを物理メモリへ書き出した後、キャッシュラインを物理メモリから読み込み、キャッシュ上のデータを更新する。



以上のようにコピーバックではキャッシュミス時には、キャッシュの内容を更新した後ライトアクセスを行なうため、物理メモリからの読み込みの回数が増える。

また本ハードウェア・プラットフォームでは、CPU及びキャッシュとMBusを介した物理メモリ間のデータのやりとりを32バイトのライトバッファ、リードバッファを用いて行なう。ライトバッファはMBusへのデータの書き出しにおいて用いられ、MBusライト・トランザクションによる待ち時間を無くしている。また、リードバッファはMBusからキャッシュへのデータの読み込みにおいて用いられる。

5 システムバスの測定

我々は多量のデータを扱う画像処理に適したプラットフォームの開発を目指している。そこで簡単な画像処理のテストプログラムを作成し、システムバスの測定を行なった。MBusモニタボードに測定項目を指定してモニタリングをスタート、ストップさせ、その間の測定値を読み込むことで所望の測定値を得ることができる。ここでは、テストプログラムの実行時間、実行中のバスビジー時間、およびバースト・リード(CPU_RB), ノン・バースト・リード(CPU_RN), バースト・ライト(CPU_WB)、ノン・バースト・ライト(CPU_WN)といった各MBusトランザクションのパターンを次のような異なるキャッシュモードとデータのアクセス順序を組合せた4つの条件下で求めた。

- コピーバックとライトスルー
- 1K × 1Kの2次元の画像データを想定し、データをアドレス順(横方向)に処理していく場合と1Kバイトおき(縦方向)に処理していく場合(それぞれキャッシュのヒット率が変わる)

テストプログラムの構成は3つの部分からなる。

- 1Mバイト(1024 × 1024、1バイト/画素)の入力画像データを読み込む
- メモリ上の入力画像データに3 × 3のマスク演算を行ない、メモリ上に同サイズの出力画像データを作成する
- 1Mバイトの出力画像データをファイルへ書き出す

この画像処理部分では、1画素1バイトで、縦1024画素、横1024画素(1Mバイト)づつの領域を入出力用に確保し、入力画像よりある注目座標の周辺を含む9画素を読み込み、出力画像の1画素を計算しメモリへ書き込む。

更に、画像処理部分を次の様に修正したテストプログラムも作成した。

- 出力画像データの注目画素の値に、入力画像データに3×3のマスク演算を行なった結果を加え、出力画像データの注目画素の値とし出力する。

今回の測定ではI/Oを除く処理能力を調べるために、これらののテストプログラムの入力画像から出力画像を作成する画像処理部分のみの測定を行なった。

この2つのテストプログラムを実行して得た結果で変化すると思われるMBusのトランザクションの内容を以下にあげる。初めのテストプログラムをA、後のテストプログラムをBとする。

(1). ライトスルーで横方向に画像処理を行なう場合

はじめに出力画像を読み込むデータサイズ分キャッシュミスの回数が増える。キャッシュミス時には物理メモリからバースト転送で32バイト読み込まれるため、出力画像データ1Mバイトを読み込む回数1M/32 = 32k回増えることになる。

(2). ライトスルーで縦方向に画像処理を行なう場合

キャッシュミスが多く起こる。テストプログラムBでは出力画像データを1度読み込むため1Mバイトのキャッシュミスが発生する。また画像処理方向が縦のため各画素の演算処理ごとにキャッシュミスが発生し、1024×1024回読み込みのバースト転送も増える。

(3). コピーバックの場合

ライトスルーでは物理メモリへの書き込みでのキャッシュミス時には、キャッシュを更新せず物理メモリのみに書き出すが、コピーバックではキャッシュを更新、すなわち物理メモリから読み込んだ後、書き出す。従ってライトスルーに比べコピーバックは読み込みのバースト転送が1回づつ多く行なわれる。このためコピーバックではテストプログラムA,BともMBusのトランザクションは変化しない。

テストプログラムAを実行した時の測定結果を図3に、テストプログラムBを実行した結果を図4に示す。

図3、4において横軸の1文字目はキャッシュモードの1文字目を表しWはライトスルーCはコピーバックを示している。2文字目は画像処理の方向を示している。Hは横方向に対し処理を行ない、Vは縦方向に処理を行なったことを表している。

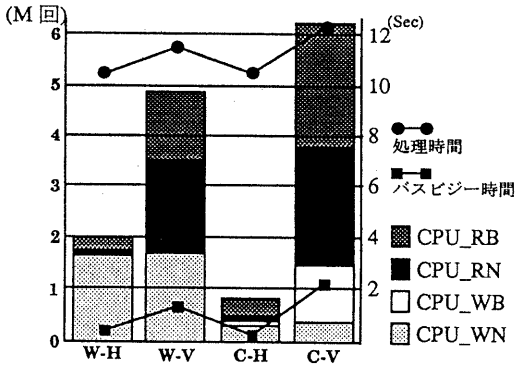


図3テストプログラムAのトラフィック

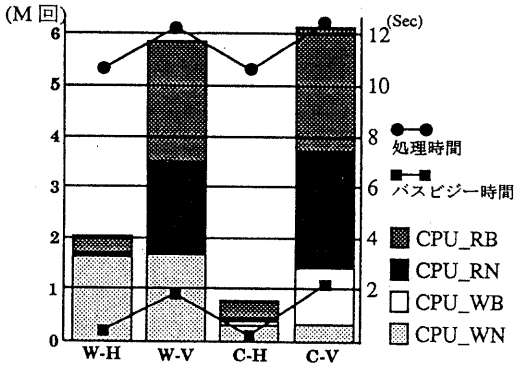


図4テストプログラムBのトラフィック

図3と図4の結果を比べると、MBusのトランザクションのうち大きく変化したものはライトスルーの読み込みバースト転送であり、W-Hでは約32000回、W-Vでは約108万回増加し予想通りの結果が得られている。その他の読み書きの回数とコピーバックでの回数は2%以内の変化であった。これは誤差範囲と考えられ、コピーバックでの増減はないことも確かめられた。

次にテストプログラムAを実行して得た結果図3について考察を行なう。処理時間とバスビジー時間

をそれぞれのモードで比較すると、ほぼ比例していることが分かる。この処理は画像処理部分のみの測定であるため、処理時間からバスビジー時間を引くことで演算処理時間が得られる。今回のテストプログラムでは横方向の処理で全体の処理時間に比べバスビジー時間の割合が約5%以内であり、プロセッサを高速化することで全体の処理時間をかなり短縮することができると言える。

ライトスルーとコピーバックの比較ではキャッシュミスの回数が多い場合(縦方向の処理)には、2つのモードとも読み込みの回数が増加している。更にコピーバックではキャッシュミスによるバーストリードのために、ライトスルーに比べキャッシュミスがより多くなるためにテーブルウォークのためのテーブル参照も増えている。

キャッシュのヒット率が高い場合(横方向の処理)では、バスビジー時間はライトスルーに比べコピーバックでは約6割になっており、キャッシュのヒット率が上がると、コピーバックの方がバスの使用率を少なくすることができる。しかし全体の処理時間にはほとんど差が認められない。これは我々が使用したMMUのライトバッファが書き込みに対し有効に働き、CPUがMBusトランザクションを待つ必要がなく時間短縮できたと考えられる。

一般的にはコピーバックの方がバスのトラフィックを減らせ、性能上も有利であると思われているようであるが、ここで使用したテストプログラムのように比較的大きなデータに対して一様に処理していく場合はライトバッファが有効に働き、ライトスルーとコピーバックとで大差ないという結果が得られた。更に、キャッシュのヒット率が悪い場合は却ってコピーバックの方がバス使用率、処理速度ともに劣るということも確かめられた。

いずれにせよ、ここでのテストプログラムによる測定では、MBusの使用率はもっとも良い場合で約2%、最悪の場合でも約18%であり、プロセッサの高速化によりプラットフォームの性能向上が充分期待できる。

6 おわりに

本稿では、Mach3.0を実装したハードウェア・プラットフォームの性能評価の一環として行なったMBusの使用率の測定について報告した。まず測定のツールとして開発したMBusモニタ・ボードによりバスのトランザクションが容易に測定できることを示し、また簡単な画像処理テストプログラムを実行させた実測値からMBusには充分余裕があることが確かめられた。

今後更に、MBusモニタ・ボードを使ってより多くの実際の画像処理プログラムを実行させた場合や、I/O処理と組み合わせた場合のMBusの測定やプラットフォームの処理速度の測定評価を行ない、その結果を更に高性能のプラットフォームの開発に役立てていく予定である。

7 参考文献

[1] SPARC MBUS interface Specificaton",Sun Microsystems,Inc. (1991

[2] CYPRESS SEMICONDUCTOR.,"SPARC RISC USER'S GUIDE"

[3] 乾 和志、菅原 圭資: "分散OS Machが分かる本"、日刊工業新聞社(1992)

[4] 喜連川 優、鈴木 和宏 他:"スーパデータベースコンピュータ(SDC)における性能評価支援ツールの構築とそれによる評価" 情報処理学会論文誌、Vol.34, No.4, pp793-803(1993)

[5] John L Henessy & David A Patterson: "Computer architecture a quantitative approach"、Morgan Kaufmann Publishers,Inc. (1990)