計算機アーキテクチャ 78－4
（１９８９．９．１９）

並列計算機テストベットATTEMPTの実装と評価

天野　英晴
慶應義塾大学理工学部

並列計算機テストベットATTEMPTは最大20台のプロセッサから成る小規模バス結合型マルチプロセッサである。そのアーキテクチャは、統合設計された同期機構とキャッシュを持ち、バスにはIEEE Futurebusを用いている点に特徴がある。本報告では現在稼働中のATTEMPT0のハードウェア構成について述べ、同期機構とFuturebusインタフェースの性能評価を行う。

The implementation and performance estimation of a multiprocessor test-bed ATTEMPT

Hideharu Amano
Faculty of Science and Technology, KEIO University

The multiprocessor test-bed ATTEMPT is a small scale bus-connected multiprocessor which consists of 20 processors. The communication mechanism of ATTEMPT is designed based on a concept of "Cache with Synchronization Mechanism", which provides powerful support for various kinds of communications in multiprocessors using IEEE Futurebus.
The implemenation of the prototype called ATTEMPT0 is described, and the performance of its communication mechanism is briefly estimated.

1. はじめに

並列計算機テストベットATTEMPT(A Typical Testing Environment of MultiProcessing sys Tems)は、並列計算機の研究者に対し、優れた環境を、個人的に用いられるワークステーション程度の価格で供給する目的で開発された。

ATTEMPTはソフトウェア的にもハードウェア的にも完全に開放されたシステムであり、拡張性に優れたプロセッサと、規格化されたバスを持つ。

安価に密結合を実現するため、ATTEMPTは図1に示すように共有バスを用いている。ボードはプロセッサボード、主記憶ボード、VMEインタフェースボードの3種類であり、今の所はディスクは持たず、SUN等のUNIXに基づくワークステーションのバックエンドとして機能する。各プロセッサ上にはこのための簡単なソフトウェアが用意されている。

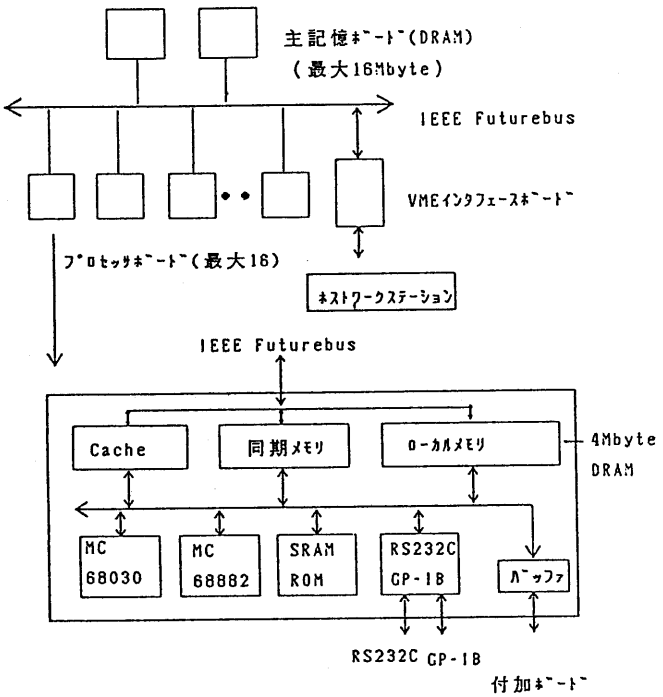

図1　ATTEMPTの構成

各プロセッサはローカルメモリ及びボード内インタフェース(RS232C,GP-IB)を持ち、単独に他のホストマシンに接続し動作することができる。また、ボード毎に付加ハードウェアを取り付け、拡張を可能にするためシステムバスを延長できるように配慮されている。　マルチプロセッサとしてのATTEMPTの特徴は以下の2点である。

(1)様々なプロセッサ間交信を統合的に実現する交信機構[1]を持つ。

(2)拡張性、柔軟性を高めるため、システムバスにIEEE Futurebus[2]を用いている。

以下ATTEMPTの交信機構を簡単に紹介し、現在稼働中のプロトタイプATTEMPT0のハードウェアの詳細を述べる。

2. ATTEMPTの交信機構

現在典型的なバス結合型のマルチプロセッサにおいては、各プロセッサはバスを介して大容量の共有メモリをアクセスする。この場合、バスの混雑を低減し、アクセスタイムを小さくするためにキャッシュの使用は不可欠であり、複数のキャッシュに分散されたデータの一貫性を保つための様々なプロトコルが提案されている。これに対し、プロセッサ間の同期機構はスピンロックやTest&Set等簡単なものが多く、プロセッサ間の相互作用のほとんどはキャッシュを介した共有メモリ上で行われる。

これに対しATTEMPTでは、強力な機能を持つ同期機構シンクロナイザと、これに適合したキャッシュコヒーレンシィプロトコルであるKeioプロトコルを組み合わせることにより、メッセージ転送、セマフォア、シグナル等様々なプロセッサ間相互作用を総合的に実現する。以下この交信機構について簡単に述べる。

2.1. 同期機構シンクロナイザ

シンクロナイザは割り込み機構を伴った同期メモリであり、メモリのロック、セマフォア、メッセージのマルチキャスト等を効率よく実現する。

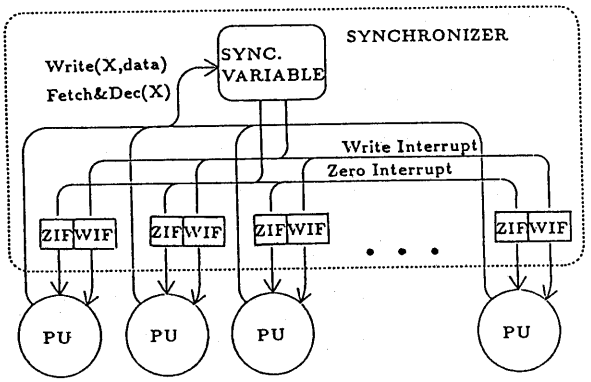

図２　シンクロナイザの基本モデル

図２にシンクロナイザの基本モデルを示す。シンクロナイザは基本単位であるシンクロナイザユニットの集合（プロトタイプATTEMPTOでは32Kbyte）である。各シンクロナイザユニットは、同期変数と割り込みフラグ（書き込みによる割り込みと、零割り込み）から構成される。フラグはプロセッサ毎に設けられ、独立に、セット、リセットされる。シンクロナイザは以下に示す制御命令を持つ。

・Write(X,data)

同期変数Xがロックされていなければdataが書き込まれる。このとき、書き込み割り込みフラグがセットされているプロセッサに割り込みが発生すると共に領域がロックされる。

・Fetch&Dec(X)

同期変数Xを読み込むとともに不可分に１減少させる。その結果が０になる場合零割り込みフラグがセットされているプロセッサに割り込みが発生する。この時領域がロックされている場合、ロックは解除される。一度領域が０になるとそれ以上減少は行われない。

・Read(X)

同期変数の値が読み込まれる。

・Lunlock(X)

同期領域のロックをプロセッサ毎に解除する。同期操作に関係するプロセッサがすべてLunlock命令を実行すると領域のロックは解除される。

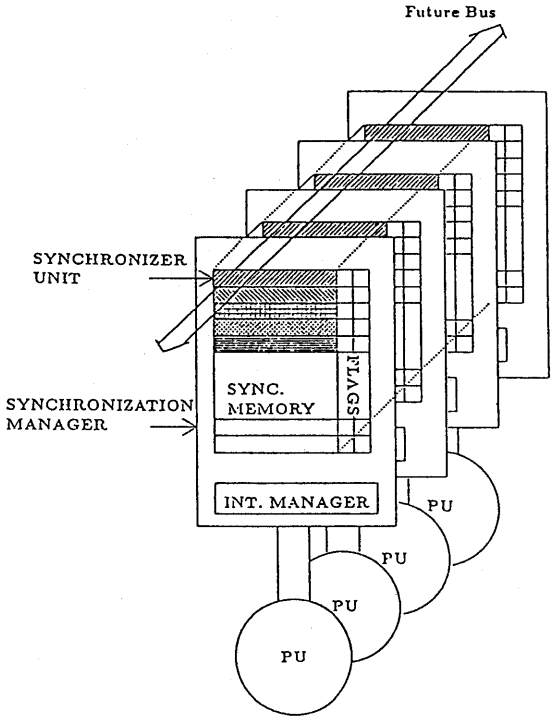

図３　シンクロナイザの実装

バスの混雑の軽減のため、シンクロナイザは図３のように分散実装される。各プロセッサは同期メモリとフラグからなるシンクロナイザ制御部を持つ。同期領域は一種のマルチリードメモリとして働き、同一アドレスに書き込まれた（またはFetch&Decされた）データは全ての制御部の同期メモリの同一アドレスに書き込まれる。このことによりRead命令とLunlock命令はバスを用いることなしに実行される。

さらに、制御部に対するアクセス競合を減らすため、各シンクロナイザユニット毎に登録フラグを設ける。このフラグがセットされていないユニットに対しては、書き込みもFetch&Decも実行されず、割り込みも発生しない。

シンクロナイザを用いて、大容量のメッセージをマルチキャストする例を図４aに示す。シンクロナイザXに対して、送信側のプロセッサ（プロセス）は、あらかじめ零割り込みフラグを、受信側のプロセッサ（プロセス）は書き込み割り込みフラグをセットしておく。送信側プロセッサは共有メモリ領域にメッセージを書き込ん

でからWrite(X,n)を実行する。ここでnは受信プロセッサ（プロセス）の数である。受信側は書き込み割り込みによりメッセージが受信可能になったことを知り、受信後Fetch&Dec(X)を実行する。全ての受信者が受信を終了したときにFetch&Decの結果は0になり送信側に零割り込みがかかる。このことにより送信側は全受信者が受信を終了したことを知り、次のメッセージがあれば再び転送を行うことができる。

メッセージが単一ワードであった場合、図4bのようにメッセージバッファに共有メモリを用いずシンクロナイザを用いることができる。この場合メッセージ自体はWrite(Y,message)により、一回のバス利用で送られるので効率が向上する。

さらに、受信者が全部受信を終了した場合の送信者に対する割り込みを省略すると図4cのように、全体の操作が1回のバス利用で実現できる。この方法は科学技術計算における反復法のように、アルゴリズムの上で送信者と受信者の同期がほぼとれている場合、非常に有効に働く。しかし送信側が次のメッセージを書き込もうとしたとき、全ての受信者が受信を終了していなかった場合、送信側は何度も失敗を繰り返し、バスを無駄に使用することになる。

```
{ Here, n is the number of receivers}
Sender(X)=
   begin
     wait until interrupt caused by Fetch&Dec(X) of Receivers;
     write a message to the region in the main memory;
     Write(X,n)
   end
Receiver(X)=
   begin
     wait until interrupt caused by Write(X,n) of Sender;
     read the message from the region in the main memory;
     Fetch&Dec(X)
   end
```

図4a 大規模なメッセージマルチキャスト

```
{initially the receiver should register synchronizers}
{ Here, n is the number of receivers}
Sender(X)=
   begin
     repeat v ← Write(X,message) until v = success
   end
Receiver(X)=
   begin
     wait until interrupt caused by Write(X,n) of Sender;
     Read(X);
     LunLock(X)
   end
```

図4b 単一長データのマルチキャスト
（送信側への割り込み付き）

```
{initially the receiver should register synchronizers}
{ Here, n is the number of receivers,

    X is used for synchronization
    and Y is used to store the message data}
Sender(X)=
   begin
     wait until interrupt caused by Fetch&Dec(X) of Receivers;
     Write(Y,message);
     Write(X,n)
   end
Receiver(X)=
   begin
     wait until interrupt caused by Write(X,n) of Sender;
     Read(Y);
     LunLock(Y);
     Fetch&Dec(X);
     LunLock(X)
   end
```

図4c 単一長データのマルチキャスト
（送信側への割り込みなし）

2.2. Keioプロトコル

Keioプロトコルはライトバックプロトコルの一種であり、各ブロックはI(Invalid)、CE(Clean Exclusive)、CS(Clean Shared)、DE(Dirty Exclusive)、DS(Dirty Shared)の5状態を持つ。表1にその状態遷移を示す。このプロトコルの特徴は以下の通りである。

1)書き込み無効化型である。

すなわち書き込み時に他のキャッシュが同一ブロックのコピーを持っている場合、無効化される。ATTEMPTにおいては、頻繁に交換される共有領域は、シンクロナイザがサポートする。このため、キャッシュでは実際の交換があまり行われないコード等の共有を効率よく行うことを目的とする。このためには無効化型が有利である。

2)キャッシュ対キャッシュ転送を用いている。

ブロックを必要とした時はできる限りキャッシュが供給する。ATTEMPTではキャッシュはSRAMを用いており、DRAMを用いた主記憶に比べ4-5倍高速であり、キャッシュ間転送が有利である。

3)メッセージ転送用の機能を持つ。

無駄なキャッシュ、主記憶転送を省略するためメッセージ転送用の命令を持つ。ATTEMPTではこれらの命令を用いることにより、主記憶共有型を採らない場合、キャッシュをメッセージバ

ッファとして用いることが可能になる。

3. ATTEMPT0の構成

現在稼働しているプロトタイプATTEMPT0はFuturebusインタフェースがLSI化されていないため、ボード面積の関係でKeioプロトコルは実装されていない。表２にATTEMPT0のハードウェア仕様を示す。キャッシュはNECのコントローラμPD71641を用い、128Kbyte、4wayセットアソシアティブ、ライトスルー制御である。各プロセッサは4MbyteのDRAMメモリを持ち、ジャンパの切り替えによりローカルメモリとしても共有主記憶としても使用できる。

ATTEMPTの交信部は、図５に示すようにFuturebusインタフェース、シンクロナイザ制御部、キャッシュ制御部、DRAM制御部の４つのブロックから構成される。各部の仕様を表２示す。Futurebusインタフェースは受動的に動作し、信号線の一時記憶、バックプレーンとの電気的レベルの変換を行い、他の制御部はFuturebus上の信号線を直接的に制御する。転送は全てブロードキャスト、4エッジハンドシェークの形式を取る。2エッジハンドシェークはATTEMPT0ではサポートされていない。

3.1. Futurebusインタフェース

バッファ部、データラッチ部、アービトレーション部から構成される。バッファ部はバックプレーンに対する信号の授受を行う。Futurebusの信号線は特殊はオープンコレクタで、2Vでプルアップされる。バッファはTrapezoidal Driver/Receiver NS3897を使用し、受信側は常に信号線をモニタしている。

信号線のうち、データ線に相当するAD<0..31>、CM<0..5>、ST<0..3>はデータラッチ部で一時記憶される。Futurebusにおいてはアドレス転送とデータ転送が完全に分離されてマルチプレクスされている。このため、これらのデータを格納するために、送／受、アドレス／データあわせて４組のレジスタを用いている。コマンド情報CM<0..6>とステータス情報ST<0..4>は必要な部分がエンコード・デコードされ、シンクロナイザ、キャッシュ、DRAMの制御部に接続されている。

アービトレーション部はFuturebusインタフェースの中で唯一インテリジェンスを持つ部分で、他の制御部からマスタとしての交信を行うための要求が出されると、バスのマスタ権を取るまで、アービトレーション要求を出し続ける。

3.2. シンクロナイザ制御部

シンクロナイザ制御部は受信部と送信部が完全に非同期的に動作する。図６に示すように、受信時、アドレス領域がシンクロナイザ領域であることがFuturebusインタフェースで判定された場合、まず登録フラグがチェックされ、登録されていた場合、ロックフラグがチェックされる。領域がロックされていればステータス線のST0(Busy)をアクティブにして送信側にそのことを通知する。ロックされていなければ、データの転送が行われ、割り込みフラグが立っていれば割り込みが発生する。割り込み要求は重なった場合、割り込み要求キューに入れられる。、

Fetch&DecはWriteとほぼ同様のシーケンスで行われ、同期メモリから読みだされ、１減少された値がバス上に出力される。ハードウェア量を減らすためATTEMPT0ではFetch&Decは下位8bitについてのみ有効である。

3.3. キャッシュ制御部とDRAM制御部

同期の損失を減らすため、キャッシュコントローラ(μPD71641)はMPU(MC68030)と同一クロック(20MHz)で動作する。キャッシュ制御部はキャッシュコントローラからの制御信号に基づき、Futurebusの信号を制御する。

DRAM制御部はDRAMがローカルメモリとして用いられた場合は使われず、共有メモリとして用いられた場合のみ働く。主記憶として独立動作するため、独立したレジスタを介してデータを授受する。

表１　Keioプロトコルの状態遷移

from state	Local Event (from processor)				External Event (from bus)		
	Read	Write	Msg Write	Swap out	Inv	Res	Rep
I	from the other cache → CS,DS from memory → CE	Read & Write	CE(Inv) (Mark)	—	I	I	I
CE	CE	DE	CE(Mark)	if Marked write back	I	CS	—
CS	CS	DE (Inv)	CE(Inv) (Mark)	if Marked write back	I	CS	—
DE	DE	DE	CE(Mark)	write back	I	DS	—
DS	DS	DE (Inv)	CE(Inv) (Mark)	write back	I	DS	CS

Inv : Invalidation by the writing

Res : When the cache is responsible and supplies the block

Rep : Another DS cache is written back to main memory

表２　ATTEMPTOのハードウェア仕様

最大20ﾌﾟﾛｾｯｻ IEEE Futurebus 使用
プロセッサボード仕様
CPU: MC68030 FPU: MC68882 ﾛｰｶﾙﾒﾓﾘ ROM: 128Kbyte SRAM: 512Kbyte DRAM: 4Mbyte (DRAMは主記憶に切り換え可) ｲﾝﾀﾌｪｰｽ GP-IB RS232C
ｼﾝｸﾛﾅｲｻﾞ: 32Kbyte ｷｬｯｼｭ: 128Kbyte 4-Way set associative Block size: 64byte

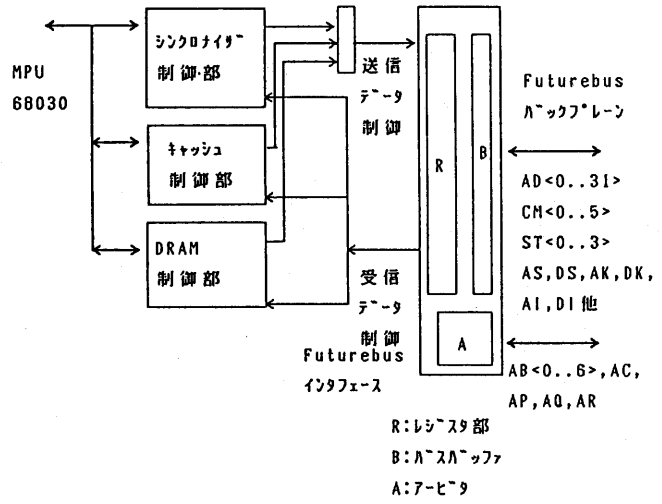

図５　ATTEMPTOの交信部ブロック図

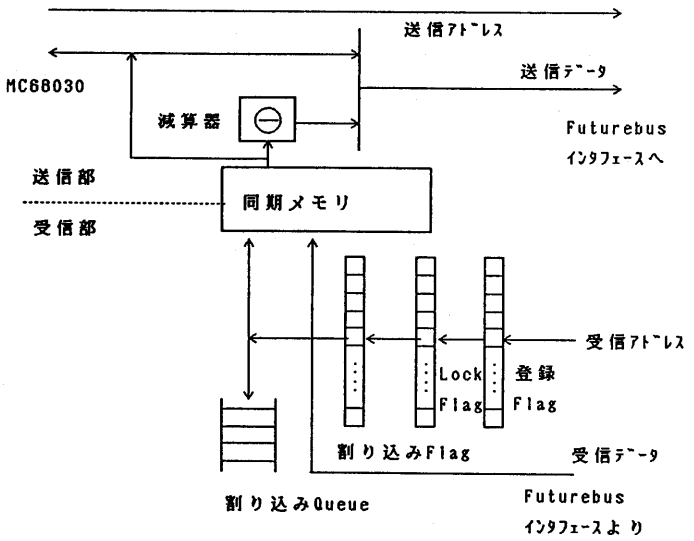

図６　シンクロナイザのブロック図

表３　ATTEMPTOの交信部の動作時間

ｱｰﾋﾞﾄﾚｰｼｮﾝ時間		360nsec(待時間は考慮外)
シンクロナイザ	Write(X,data) Fetch&Dec(X) Read(X)他Flag操作	270nsec 300nsec Waitなし(20MHz MC68030)
キャッシュ	ヒット時読みだし ﾘﾌﾚｰｽ 書き込み	Waitなし(20MHz MC68030) 1010nsec (64byte Block) 210nsec

4. 性能評価

転送部のハードウェアは表3に示す性能を実現することができた。全ての制御用のコントローラは30MHzのクロックで動作する。アービトレーションでかなりの時間がかかるのは、Future busで定義されたシーケンスを実行するためで一度マスタになった後は他のプロセッサから要求がなければ、アービトレーションなしで転送をし続けることができる。

5. おわりに

現在のところATTEMPTOのプロトタイプはアプリケーションプログラムが安定動作するに至っておらず、全体としての評価をここで報告することはできなかった。バスの状態等をモニタするハードウェア評価ボードを設計中であり、アプリケーションプログラムを含めた詳細な評価を行う予定である。

参考文献

[1]H.Amano, T.Terasawa and T.Kudoh "Cache with synchronization mechanism", Proc. of The 11th IFIP Congress Aug. 1989.

[2]"IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus," March.1988.