計算機アーキテクチャ 83−40
（１９９０．７．１８）

高並列計算機CAP-IIのメッセージコントローラ

清水 俊幸 石畑 宏明 堀江 健志

(株) 富士通研究所

高並列計算機CAP-IIのプロセッサエレメント(セル)を構成するマイクロプロセッサ(SPARC-IU)とキャッシュメモリ,大容量メモリ,I/Oデバイスをインタフェースする LSI, メッセージコントローラ (MSC) を開発した. 数値シミュレーションや映像生成を対象としたCAP-IIの特徴を踏まえ，通信デバイスの転送能力ににあった十分な量のデータをセルの計算能力を損なうことなく供給できるようにした. MSCには, キャッシュコントローラも集積するため, その動作情報を利用したメッセージ送信(ラインセンド)も実現した. 本報告では,MSCが提供するこれらの機能について述べる.

A Message Controller for a Highly Parallel Processor, CAP-II

Toshiyuki SHIMIZU, Hiroaki ISHIHATA and Takeshi HORIE

Fujitsu Laboratories LTD.

1015, Kamikodanaka, Nakahara-ku, Kawasaki 211, Japan

We developed a message controller (MSC) for a highly parallel processor, CAP-II. The MSC realize interface among a microprocessor (SPARC-IU), cache memories, dynamic RAM modules and I/O devices. It is designed to supply enough data to I/O devices without penalties to calculation. Its design is based on CAP-II architecture, which handles image generations and numerical simulations. A cache controller, which is also incorporated in the MSC, makes it possible to execute a special message transfer (line send). We present the architecture and performance of the MSC.

1 はじめに

高並列計算機CAP-IIのプロセッサエレメント (セル) を構成するにあたり, 32 bits マイクロプロセッサ (SPARC-IU) とキャッシュメモリ, 大容量メモリ, I/O デバイスをインタフェースする LSI , メッセージコントローラ (MSC) を開発した.

MSC を設計するにあたって,

1. 数値シミュレーションや映像生成を対象
2. 64 ~ 1024 台のセルを実装
3. パケット化したデータストリームによるセル間通信
4. 大容量コピーバック方式のキャッシュメモリ

というCAP-IIの特徴[1] を踏まえ, 通信デバイスの転送能力[3,4] ににあった十分な量のデータをセルの計算能力を損なうことなく供給することを目標とした.

並列計算機では, 送受信するデータはセル内に規則性をもって, あるいは全く規則性を持たずに分散している. これらのデータを通信をおこなう毎にメッセージに組み立てると, そのオーバヘッドが問題となる. 特に, セルの計算能力が高くなったり, セルの台数が多くなったりした場合, 大量に発生するメッセージを処理するためのオーバーヘッドが無視できなくなる. そこで, このようなオーバーヘッドを減らすための工夫が必要となる.

MSCでは, セル間の効率の良いメッセージ交換を実現するために, 種々の形態の DMA を用意した. MSC には, キャッシュコントローラも集積するため, その動作情報を利用したメッセージ送信 (ラインセンド) を実現した. 本報告では, MSC が提供するこれらの機能について述べる.

2. においてCAP-IIが対象とするアプリケーション のデータアクセスパターン について考察し, メッセージの構成手順を検討する. 3. では, 従来技術でそのアプリケーションを実現した場合に, 効率上どのような問題が生じるかを検討する.4. において今回実現した転送方式について述べ,5. でその効果について検討する. 最後に, まとめをおこなう.

2 データアクセスパターン

CAP-IIでは, アプリケーションとして, 映像生成, 数値シミュレーション等を考えている. これらは, 大規模行列の求解および大域データへのアクセスに特徴づけられる. 行列, 大域データどちらもデータ量が大きいため, セルそれぞれがデータを分割して保持する必要がある.

行列の求解には反復法, 直接法があるが, データを分割保持した場合, どちらも求解時には大量のデータ (メッセージ) がセル間を移動する. このため送受信されるメッセージがどのように構成されているかがポイントとなる. また, 大域データに対するアクセスに対してはどのようなメッセージを送受すれば良いかを検討しなければならない.

2.1 データの分割形式

大規模行列をセルで規則的に分割して保持することを考える.

例えば, 8×8 の配列 A を4台のセルで分割して 4×4 の配列 LA に保持する場合の分割形式は, 図1のようになる.

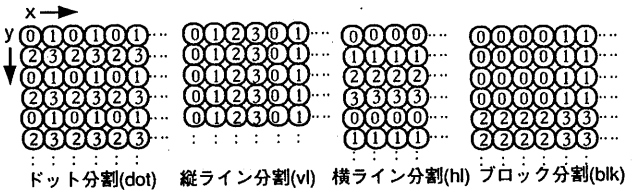

図 1: いろいろなデータ分割形式

これら, 規則的な分割の他に, 粒子シミュレーション等における粒子データの分割保持や, 疎行列を分割保持するために, リスト構造でデータを保持することがある[2] .

2.1.1 大域データのアクセス

大域データを参照しながら, 計算をおこなうシミュレーションにおいて, 隣接のデータをもとに反復計算をおこなうタイプのものでは, 配列要素の上下左右の値が知れれば良い. 大域データをセルで分割 (ブロック分割) して保持した場合, 保持している配列の袖の部分を転送する必要がある. (●) が配列の袖である.

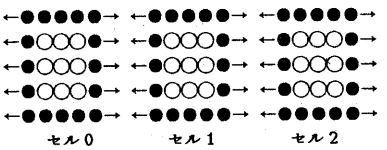

図 2: 袖の転送

ODD-EVEN 法では, 前の計算の結果 (○) をもとに新しい点 (●) の値を更新する. 次のステップでは (●) を元に (○) の値を計算する. このため, 図3のような通信が起こる.

上記2 つの例では, あるセルへ送出するデータは一定間隔をおいて存在する. 例えば, 前者でセル 1 から 2 へ送出

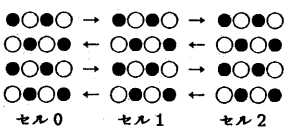

図 3: ODD-EVEN 法

べきデータは $LA[j][3](j = 0..3)$, 後者はODD,EVENのフェーズをph(0,1)とすれば $LA[2j+ph][3](j = 0,1)$ となる.

2. 1. 2 データ分割形式の変換

行列方程式の解法の1つにADI法がある. 2次元のADI法では, X方向の演算とY方向の演算を交互におこなうことにより解を求める. このとき, 各方向の演算においてデータに対するアクセスがローカルになるように, まえもってデータの移動をおこなっておけば効率が良い. このため, 縦ライン分割と横ライン分割の相互変換をおこなうメッセージが使用される.

負荷分散をはかるためにドット分割して演算を行なった後, 結果を取り出したり, 表示するために, 縦ライン分割や, ブロック分割に変更する場合がある. これらの相互変換のためのメッセージを構成することも有用である.

2.2 細かなメッセージ

並列計算機では, コントロールメッセージ等の細かなメッセージが, 非同期的に大量に発生する. また, 大規模疎行列をセル分割して保持した場合, 配列要素をセル同士で通信する場合など, 細かなメッセージが多く発生する.

3 従来方式による問題

先に述べた, 分散形式の変換や, 配列の袖の転送に必要なデータは, どれもかたまりになっていないため, 直接DMAを起動するには効率が悪い.

このため, ソフトウェアによってメッセージを組み立てた後, DMAを起動する. 受信側のセルでは, まとまって受信したデータを適切な場所に展開する. この組み立てや展開は, CPUの仕事でありオーバーヘッドとなる.

非同期的に発生する細かなメッセージには, コントロールメッセージ等, バッファリングすることにより, 遅延が生じることを嫌うメッセージもある. これらは, 従来はポーリングで送出する以外に手段はなかった.

CAP-IIでは, メッセージとして, 1ワードのヘッダ + データ + エンドビットという形式を採用している. このヘッダをもDMAで送り出すには, メッセージをバッファ領域で組み立てる必要がある.

受信セルにおいては, メッセージはヘッダ付で受信される. この場合, メッセージ全てをDMAで直接データ領域に受信することはできない. ヘッダは何らかの方法で取り除いてやる必要がある.

4 MSCの機能

分散形式の変換や, 袖の転送を少ないオーバーヘッドでDMAするために, DMAのパラメータ設定の自由度を上げ, 多様なアクセスシーケンスを実現する. リスト構造のデータに対してもハードウェアでアクセス可能とする. このことにより, メッセージ作成のためのバッファリングの必要をなくす.

細かいメッセージについては, キャッシュされているメッセージデータをフラッシュするのと同じ操作により転送する機構を用意する. このことにより, 明示的にメッセージ送出のタイミングを指定することと, 転送起動時に生じるオーバーヘッドを軽減する.

データ受信においては, コントロールメッセージなどのデータとはタイプの異なるメッセージについては, 別のメッセージとして, 自動的にバッファリングされる機構を用意する.

メッセージの形式を考慮して, 送出するデータとは別にヘッダを埋め込む機構 (プリアンブルの送出機構) を用意する. 受信時にはヘッダを別な領域に保存することができる. メモリのアライメントによらず最小のバスサイクルでデータ転送が可能になる機構を用意する.

4.1 とびとびの転送

図4に示すようにメモリ上に規則的に散在するデータをメッセージに組み立て, デバイスに送り出したり, デバイスから送られてくるメッセージをメモリ上に規則的に展開する機能をストライドDMAと呼ぶ. 図4において, 網のかかっている部分が実際に転送されるデータである. この部分は, パラメータ addr,size,hskip,vskip,vcnt,hcnt を用いて指定する.

このアクセスパターンによって配列の袖の転送や, データ分割形式の変更がデータのコピーなしに実現できる. 例えば, $x \times y$ 台のセルで $h \times v$ の配列のデータ分割形式の変換を行なうには, これらのパラメータを表1のように設定すればよい.

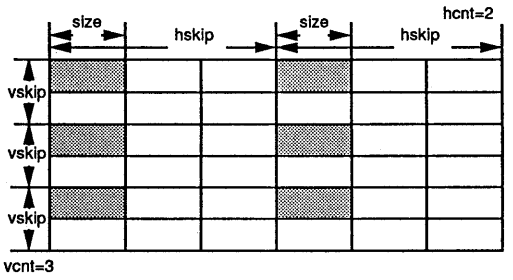

図 4: とびとびに存在するデータに対するアクセス

表 1: 分割形式の変換パラメータ

A⇔B	A 側 size,hskip,hcnt	B 側 size,hskip,hcnt
hl⇔ vl	$1, xy, hv/x^2y^2$	$h/xy, h, v/xy$
dot⇔ vl	$1, y, hv/x/y^2$	$h/xy, h/x, v/y$
hl⇔ blk	$h/x, h, v/y^2x$	$h/x, hy, v/y^2x$
dot⇔ blk	$h/x^2, h/x, v/y^2$	$1, h/x^2, h/x^2, hy/x^2, v/y^2$
vl⇔ blk	$h/x^2y, h/x, v/y$	$1, h/xy, hv/x^2y^2$
hl⇔ dot	$1, h/xy, hv/x^2y^2$	$h/x^2, hy/x^2, v/y^2$

(注) dot⇔blk 変換では vskip,vcnt まで指定する.

4.2 リスト転送

リスト転送は, 図 5に示すように, ポインタ TBLP で指された配列の要素が指すデータ (網が掛かっている部分) を転送するものである.cnt で指定された回数転送をおこなう. この機能によって粒子データ等の不規則に分割されたデータを 1 つのメッセージとして送出できる.

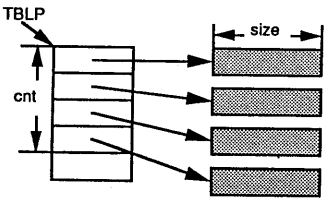

図 5: リストの転送

データの毎にパケットの終了を示すエンドビットを付加する機能も持つ. また, 図 6のように, ポインタの配列の中に 1 ワードのデータを埋め込むことができる.

この機能により, [ヘッダ] + [データ] + [エンドビット] という構成のメッセージを複数送受信することができる.

4.3 ベクトル転送

FORTRAN で用いるような, リストベクトルの転送をサポートする. 配列 $A[], B[]$ において, $B[A[i]] : (i = 0..HCNT-1)$ で指定するデータに連続的にアクセスする.

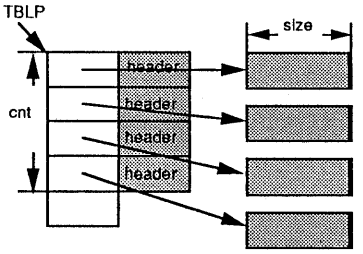

図 6: ヘッダつき転送

リスト転送で設けたように, $A[i](imod2 == 1)$ の部分に埋め込んだデータを併せて転送する機能も提供する.

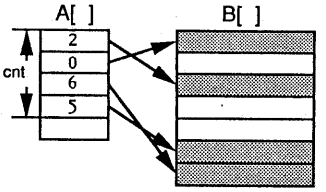

図 7: ベクトル転送

4.4 プリアンブルの転送

上記 3 種類の転送 (とびとびおよびリスト転送, ベクトル転送) に先立って数ワードのヘッダ (プリアンブル) を送ったり受けたりすることができる.

[プリアンブル]+[ブロックデータ]

4.5 リングバッファによる受信

ブロック DMA 以外に, 非同期的に生じるデータ転送要求に対して応答する機能を用意した. これをバッファレシーブと呼ぶ.

バッファは, リングバッファとなっており, バッファオーバーフローは, ハードウェアで監視される. オーバーフロー時には, 割り込みによって CPU に通知するか, またはデータの読みだしによって, バッファに受信領域が生じるまでデータ転送を停止する.

MSC のレジスタ読むことによって, これまでアクセスしていた領域を解放し (受信バッファとして DMA に返し), 同時に次の有効データが存在するアドレスを得る. データが存在しない時には戻り値として 0 を返し, 読み込むべきメッセージが無いことを知らせる.

4.6 アドレス指定受信

ヘッダに示されているインデックス (10 ビット) をもとに生成したアドレスに, 受信したデータを書き込む機能を提供する. これをアドレスレシーブと呼ぶ. DMA デバイスは,

アドレス指定受信用のパケットが到達した事をMSCに知らせると同時にそのメッセージのインデックスを出力する[3].

MSCではインデックスをデータパケットサイズで乗じ,更にBASEのアドレスを加えることによって,実際に書き込みをおこなうアドレスを得る.この機能によって全てのプロセッサが全てのプロセッサに対して通信を行なうの時に,送り手のプロセッサ別にデータを分離して受信することがハードウェアで実現できる.

4.7 キャッシュラインの直接送信

サイズの小さい(キャッシュにのっているような)メッセージの転送にDMAを用いるとDMA設定のオーバヘッド,キャッシュのフラッシュのオーバヘッドが問題となる.

セルは大容量のコピーバックのキャッシュを備える.スヌープ機能は無い.このため,キャッシュされているデータをDMA可能な領域(メインメモリ)に書きださなければない.そこで,キャッシュをフラッシュする操作と同等な操作でキャッシュラインのメッセージをデバイスに送出する機構を実現した.これをラインセンドと呼ぶ.

指定したデータがキャッシュ上に無い場合もDMAが自動的に起動される.図8において矢印はデータの移動を示す.LSHITはキャッシュにデータが存在した時,LSMISSはキャッシュ上にデータがなかった時である.比較のために通常のキャッシュフラッシュ(FLUSH)も示す.

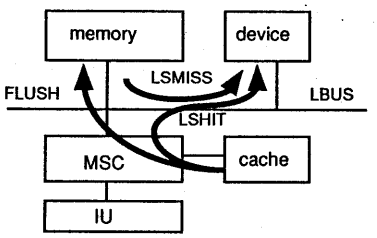

図8: ラインセンドにおけるデータ移動

4.8 DMAリクエストおよびDMA転送

MSCが接続されるバス(LBUS)のデータ転送単位は,byte, half word, word, double word, cache lineである.キャッシュラインサイズは4word (16byte)である.word以上はサイズが大きくなるほどバンド幅が高い.このうちMSCは,word, double word, cache lineをデータタイプとしてサポートする.

DMAデバイスは,転送効率を高めるために,キャッシュラインサイズ以上のFIFOを持っている.デバイスはFIFOの状態を表2のようにデコードしてDMAリクエストとする.

表2: DMAリクエストのデコード

デバイスへの書き込み	デバイスからの読みだし
1 word以上の空きがある	1 word以上のデータがある
2 word以上の空きがある	2 word以上のデータがある
line size以上の空きがある	line size以上のデータがある

FIFOの状態デコードを利用してハードウェアによりバスの転送効率を最大にすることができる.残りの転送データ量,アドレスアライメントを考慮し,まとまった単位でバスアクセスが生じるようにする.例えば,アドレス4から8ワード分の転送を指定した場合,[1ワード]+[2ワード]+[4ワード]+[1ワード]という転送が行なわれる.アドレス4はワードアライメント,アドレス8はダブルワードアライメントである.

デバイスからメモリへの書き込みに対しては,パケットエンドを知る手段がないため,データが存在していればバスサイクルを開始する.

5 評価

表3にMSCの諸元を示す.

表3: MSC諸元

チャネル数	4 + バッファレシーブ×2 + アドレスレシーブ + ラインセンド
転送モード	ストライド,リスト, ベクトル(ヘッダ送出機構付)
動作周波数	25MHz
転送速度	40MB/s (最大)
その他機能	キャッシュコントローラ ブートストラップ機能
ゲート数	42,000ゲート

5.1 DMAの性能評価

何種類かのセル構成において,512×512要素の配列をドット分散から,縦ライン分散へ変換する例題を,シミュレータで実行しその速度を測定した.ネットワークで消費される時間はどのセルに対する通信も一定としているが,転送データ量が多いので問題ない.送受信ともネットワークには常に書き込み可能であるとした.このためネットワークの混み方等の条件は加味されない.

(1) ソフトウェアによってステータスをポーリングしながら, データの送信, 受信を実現する. (2) 従来の DMA を用いて転送をおこなう. (3)MSC の機能を用いて転送をおこなう. 以上の結果を表 4に示す.

表 4: ドット→縦ラインの変換時間

セル構成	(1)	(2)	(3)
32 × 32	65.1	99.2	29.7
16 × 16	152	192	30.3
8 × 8	518	604	72.8
4 × 4	1960	2270	266.0

(単位 1000 cycle)

結果から, MSC に付加したパラメータにより転送が効率化されたことがわかる.

5.2 ラインセンドの性能評価

デバイスが常に書き込み可能である状態で, データを送り出すのに必要な時間を表 5に示す.

表 5: データアクセスに要する時間

アクセスの種類	データサイズ	時間	転送速度
LSHIT	32byte	600ns	26.7MB/s
LSMISS	32byte	640ns	25.0MB/s
FLUSH	32byte	480ns	33.3MB/s
IO_READ	4byte	480ns	8.33MB/s
IO_WRITE	4byte	400ns	10.0MB/s

ラインセンドを用いずにポーリングによって転送を実現した場合, 最小でも 1 回のステータスリード, 4 ワードの書き込みが必要であり, IO_READ+IO_WRITE*4 = 2480ns の時間を必要とする. これはラインセンドに比較して約 4 倍である.

DMA の最大転送速度はデバイスへの書き込みで 25MB/s である (書き込みが遅いのはメモリの読み出しにウエイトが入るため) . 全てをフラッシュして転送するとすれば, 1 ワードあたり 120ns+160ns = 280ns の時間を要する. これは, ラインセンドに比較して約 1.8 倍である.

ここで注意しなければならないのは, DMA を用いた場合, DMA 実行中は CPU は他の演算を行なえるのに対して, ポーリング, ラインセンドでは行なえないということである.

以上のことから, 小量のキャッシュにのっているようなメッセージについてはラインセンドが効果的に活用でき, 大量のデータに対しては DMA を用いるのが適当であることがわかる.

6 まとめ

高並列計算機のセルに使用する MSC の機能, および性能について検討をおこなった. 種々のパターンのデータ転送をハードウェアで実現することにより, 効率的なメッセージ転送が実現された.

とびとびの転送は, 並列計算機においてデータを分割保持し計算をおこない, 途中で全体的な同期をとり, データ交換を行う場合等に有効に活用できる. 例えば多次元の FFT や,ADI 等の計算においてデータの分割形式を変更する場合に利用できる.

また, リスト転送, ベクトル転送はユーザプログラムが使用しているデータ構造をそのまま転送の対象として (コピーすることなしに) 使用できる. バッファ受信は, 非同期的なメッセージ受信を効率良くハンドリングできる. アドレス指定受信はセルのコンフィグレーション (マッピング) とデータ分割を意識して, 多対多の通信をおこなう時に活用できる. ラインセンドは, 非同期的に発生する細かなデータの送信イベントをオーバヘッドを最少に処理できる.

実現した転送方式は, 以上の効果が期待できる. しかし, 例えば大きなデータブロックが, キャッシュ可能な領域にマップされていて, そのデータブロックを転送する場合等を考えると, プロセッサがその領域をソフトウェアでフラッシュしなければならない. これらのオーバヘッドや, バスの使用効率, 通信の混雑度等について, 今後実際にアプリケーションを構築して実験を行っていく.

参考文献

1) 石畑他, “高並列計算機 CAP-II の構成とメモリシステム”, 本研究会投稿予定

2) 佐藤他, “並列計算機 CAP による分子動力学計算”, 信学会研究会資料, CPSY89-24,pp.57-62,1989

3) 堀江他, “高並列計算機 CAP-II のルーティングコントローラ”, 本研究会投稿予定

4) 加藤他, “高並列計算機 CAP-II のブロードキャストネットワーク”, 本研究会投稿予定