計算機アーキテクチャ 125－16
（1997. 8. 21）

FDUMA 共有メモリ・アーキテクチャにおけるバス・アービトレーション

森垣 利彦　弘中 哲夫
児島 彰　藤野 清次

広島市立大学 情報科学部 情報工学科

近年，プロセッサと DRAM を 1 つの LSI 上に混載することでメモリバンド幅を広げる研究が行われている．しかし，この方法ではベクトル処理的な用途以外では得られるメモリバンド幅を有効に活用できず，On Chip Multiprocessor などの共有メモリとして利用しにくい．そこで我々はこの問題を解決するメモリ・アーキテクチャとして，*FDUMA* マルチポートメモリシステムを提案している．本稿では，現在開発中である *FDUMA* メモリシステムの試作機で用いるバス・アービトレーションについて述べ，その後ソフトウェア・シミュレータによる *FDUMA* メモリシステムの特性評価を行う．

Bus Arbitration for FDUMA Shared Memory Architecture

Toshihiko MORIGAKI　Tetsuo HIRONAKA
Akira KOJIMA　Seiji FUJINO

Department of Computer Engineering, Faculty of Information Science,
Hiroshima City University

Many research are done on deriving high memory bandwidth by merging the DRAM and logic on one chip. This merged DRAM/logic chip is effective for vector-style processing. Although it is not suitable for shared memory archtecture like the On Chip Multiprocessor. We proposes an memory archtecture, *FDUMA Multiport Memory System*, to solve the problem. This paper describes the method of bus aribitrations, that we use to impliment the prototype FDUMA Memory System. The results of performance evaluation done by software simulation are also presented.

1　はじめに

近年メモリとプロセッサを結ぶメモリバスがLSIのピンネックの原因となることが問題になっている．この問題を解決する一つの方法として現在着目されているのがDRAM混載ロジックであり現在さまざまな研究が行われている[PAC+97][村上 96]．つまり，DRAM混載ロジックが実現できるようになったことでDRAMとプロセッサを同一LSI内に実現することが可能になり，LSIのピンネックを避けてメモリとプロセッサのメモリバス幅を広げることが可能になった．しかしながら，このような方法を用いるだけでは大きな性能向上を達成することは困難である．これは次のような事柄が問題となるためである．

a.単純にバス幅を拡張するだけでは連続アドレスへのアクセスのみが高速化される．そのため，メモリバンド幅を有効に使用できるアプリケーションが限定され，ベクトル処理的な用途以外ではメモリバンド幅を有効に活用できない．

b.メモリ容量に制限が生じる．高いメモリバンド幅でアクセスできるのはLSI内のメモリのみである．そのため大容量のメモリを必要とするようなアプリケーションを用いる場合，従来のキャッシュと同様にデータ入れ換えのオーバヘッドが問題となってくる．

本稿では，上記の問題を解決する *FDUMA* (Functional memory with Distributed Unified Memory Access) 共有メモリ・アーキテクチャ[弘中 97] の概要とその実現方式について述べた

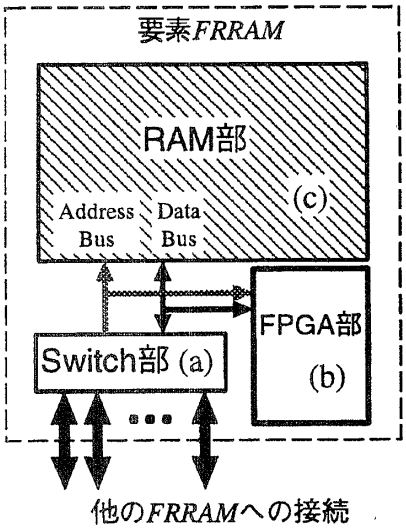


図 1: *FRRAM* の概念図

後，*FDUMA* においてメモリアクセスの際に発生するバスの競合をアービトレーションするための，バス・アービトレーション方法の提案を行い，最後にシミュレーションによる評価結果を示す．

2 *FDUMA* メモリシステム

2.1 特徴

FDUMA 共有メモリ・アーキテクチャは1節であげた目的を達成するため，プロセッサ間結合網とメモリを一体化したメモリ・アーキテクチャであり，次のような特徴を持つ．

a. *FDUMA* メモリシステムは，見通しのよいプログラミングモデルが提供可能な共有メモリ型マルチプロセッサを実現するマルチポートメモリシステムである．

b. プロセッサからの指示によりメモリ内のFPGA部をプログラムし，そのメモリ内のデータに対して単純な演算やデータの移動をプロセッサの動作と並行して行うことが可能である．

c. 単純な構造を2次元的に繰り返す構造により実現することも可能であるため，大規模VLSIアーキテクチャとしても向いている．

以下にこれらの特徴を実現するためのメモリシステムの基本アーキテクチャについて述べる．

2.2 基本アーキテクチャ

FDUMA 方式は，*FRRAM* (Functional Routing RAM) をネットワークスイッチとして用いてさまざまな結合網（クロスバ，メッシュ，トーラス，ハイパーキューブ，オメガ網等）を実現し，このトポロジにより同時アクセスが可能な多バンク構成，パイプライン・メモリアクセスを可能にするインタリーブメモリ構成を実現する．ただし，ここでは実装を簡単にするため，メッシュを用いたメッシュ型 *FDUMA* についてのみ議論する．*FRRAM* は図1に示すように (a) 結合網を構成する Switch 部，(b) プログラマブルな機能ユニットである FPGA 部，および (c) 主記憶を構成する RAM 部から構成されている．

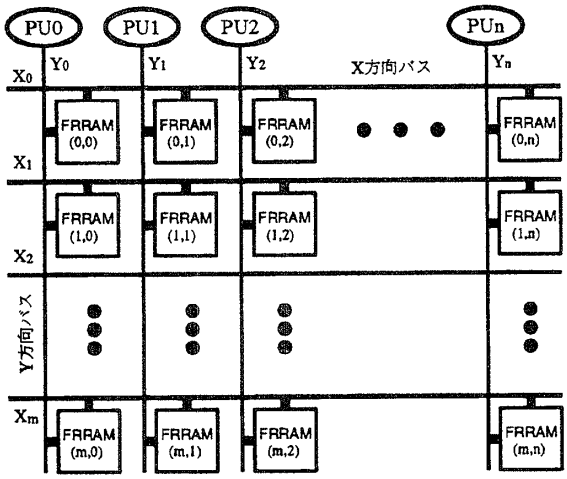


図 2: メッシュ型 *FDUMA* の概念図

このような *FRRAM* を複数相互に接続したものが，*FDUMA* メモリシステムである．図2にメッシュ型に *FRRAM* を配置した *FDUMA* メモリシステムの例を示す．

2.3 メモリアクセス動作

メッシュ型 *FDUMA* メモリシステムは図2に示すように *FRRAM* をX方向バス，Y方向バスの交点にそれぞれ配し，メモリシステムに接続するプロセッサは必要数だけ各Y方向バスに接続される．図1に示す Switch 部で Y 方向バス，X 方向バス間のデータの中継，および，RAM 部，FPGA 部へのデータの受渡しを実現する．各 *FRRAM* 上のメモリはシステム全体で一意の物理アドレスを用いてアドレシングされ，すべてのプロセッシング・ユニット (Processing Unit; PU) からアクセス可能である．

3 バス・アービトレーション

FDUMA において複数の PU からの同時アクセスを可能にするため，バスのアービトレーションを行う必要がある．また，現在試作している *FDUMA* は信号線数を制限するため，アドレスとデータが同じバスを使用する．そのためアドレスとデータの競合もアービトレーションする必要がある．本節はまず 3.1項で問題の定義を行ったの

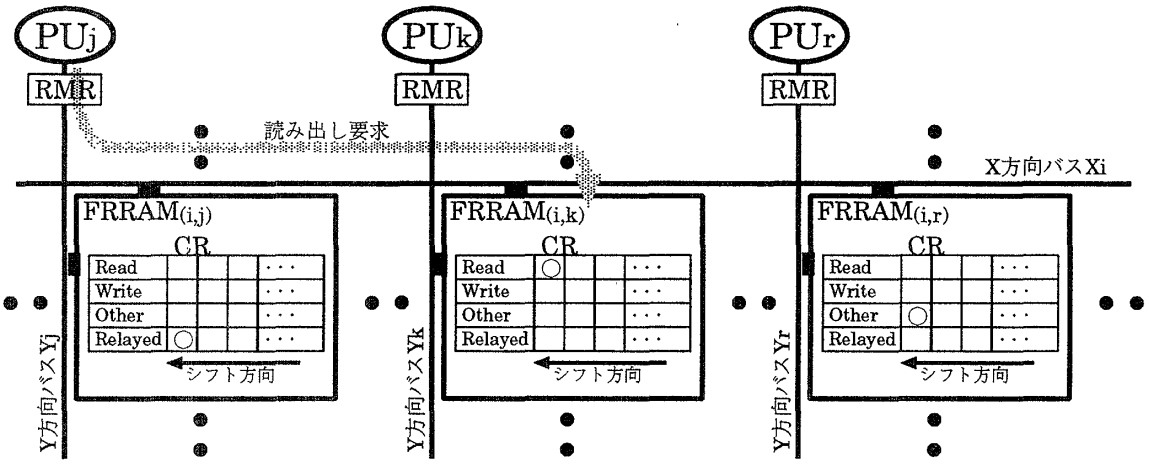


図 3: *FRRAM* の状況把握レジスタの動作

ち，次の3.2項でX方向バスのアービトレーションの方法について述べ，3.3項で *FRRAM* からのデータの読み出しについて述べる．3.4項ではデータ読み出しと書き込み動作の違いについて述べ，最後に3.5項でY方向バスしか用いない場合についての説明を行っている．

3.1 データの読み出しにおける競合

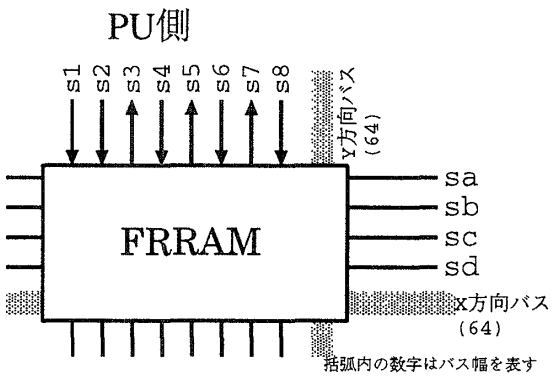
図3のように，PU_j に $FRRAM_{(i,k)}$ からデータの読み出しの必要が生じた場合を例に説明を行う．まず PU_j は $FRRAM_{(i,k)}$ に読み出すデータのアドレスを送るが，このときX方向バス X_i を使う必要がでてくる．ここで同時に他のPUが X_i を使おうとしている場合，X方向バスのアービトレーションを行わなければならない．以下にアービトレーションの方法を述べる．

ここで，図4はアービトレーションおよびデータの転送を行う際に必要となる信号線である．図中の信号線の矢印の向きは信号を伝える方向である．以下，文中で具体的に信号線をあげて説明する時は，図4の信号線名を用いる．

メモリアクセスを行うPUはY方向バスとX方向バスの中継点にあたる *FRRAM* に対して，X方向バスを使用するための予約を信号線s1,s2によって出す．PU_j の場合は中継点は $FRRAM_{(i,k)}$ となる．X_i 上の予約を受けた *FRRAM* 間でアービトレーションを行い，X_i を使用するPUの順番を決定する．以下 *FRRAM* 間でのアービトレーションの詳細を述べる．

3.2 X方向バス・アービトレーション

X方向バスのアービトレーションの概念図を図5に示す．まず，4個の隣り合う *FRRAM* の中でPUからのX方向バスの使用予約を受けたものから，アービタにより1つ選ぶ．この操作は1ステージで行われる．



信号線一覧	
s1: X方向バス番号(*a)	s5: 要求受理(1)
s2: X方向バス予約信号(1)	s6: RW用X方向バス番号(*a)
s3: 次サイクル要求実行(1)	s7: 次サイクルRW実行(1)
s4: Y方向バス上のFRRAMへの要求割り込み(1)	s8: Y方向バス上のFRRAMへのRW(1)
sa:アービトレーション用(*b)	sc: 次サイクル X方向バス使用状態(1)
sb: 要求受理(1)	sd: RW可能状態(1)

*a:X方向バスの本数が表現可能なビット幅　*b:アービタ1つにつき2ビット必要　R:読み出し W:書き込み

図 4: メモリアクセスに使用する信号線

次に前の操作で選んだFRRAM4個の間で再び同様に1つ選ぶ．このように次にX方向バスを使用可能なFRRAMを決定する．つまりそのFRRAMと同じY方向バスにあるPUを決定することになる．4個から1個を選ぶ操作を1ステージで行いパイプライン化することで，大規模なシステムにおいてもアービトレーションによるオーバヘッドを軽減できる．

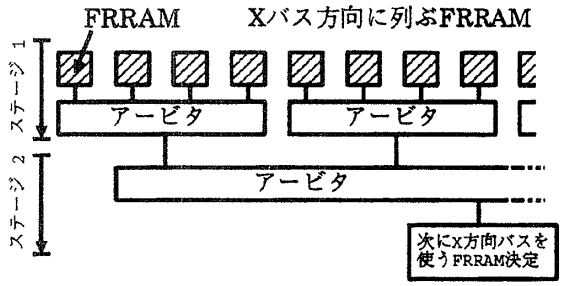


図 5: X 方向バス・アービトレーション概念図

3.3 *FRRAM* からのデータの読み出し動作

ここではすでに前項の X 方向バス・アービトレーションにより $FRRAM_{(i,j)}$ が X_i の使用権を得たものとする．つまり PU_j が X_i を使用して，読み出し要求を出すことが可能になったとする．

$FRRAM_{(i,j)}$ は使用権を得ると s1 が自分を指しており，次サイクルに X_i かつ Y_j が使用可能であるとき，s3 によって次サイクルで要求が可能であることを PU_j に知らせる．また，このとき $FRRAM_{(i,j)}$ が次サイクルで Y_j と X_i の中継を行うことが決定する．

s3 によって要求が可能であることを受けとった PU_j は読み出すアドレスをバスに流す．このとき Y_j および X_i 上の全ての *FRRAM* は，バスに流れるアドレスが自分自身に対する読み出し要求であるかどうかを判断する．以下，要求が受理されない場合と受理される場合に分けて説明する．

a.要求が受理されない場合：読み出し要求を受けた $FRRAM_{(i,k)}$ がすでに他の PU によって読み出し要求を受けており，要求を貯めておくためのキューが 埋まっている場合．このとき $FRRAM_{(i,j)}$ および PU_j は要求が受理されなかったことを s5 に受理信号が流れないことで知る．これにより，PU_j には再度要求を出す必要が生じ，$FRRAM_{(i,j)}$ では X 方向バスアービトレーションの最上位階層の予約を保持する必要が生じる．

b.要求が受理される場合：$FRRAM_{(i,k)}$ が他の PU によって読み出し要求を受けていない，もしくは受けていてもキューに空きがある場合．$FRRAM_{(i,k)}$ は s5 に受理信号を流す．

PU_j は s5 の受理信号を受けとると，それ自身が持つ**要求記憶レジスタ（RMR）**に以下の情報を記憶する（図 3参照）

- 要求を出した *FRRAM* がある X 方向バス番号
- 書き込みか，読み出しか
- Y 方向バスのみを使用するアクセスか否か

また，*FRRAM* には，**状況把握レジスタ (CR)** と呼ぶレジスタがある．ここで要求が流れた X 方向バス上にあるすべての *FRRAM* は以下の 4 つの項目のうちそれ自身が該当する情報を保存する．

Read:自分に対して読み出し要求があった（図 3の $FRRAM_{(i,k)}$ の CR 参照）．

Write:自分に対して書き込み要求があった（書き込み時に使用）．

Other:同じ X 方向バス上にある他の *FRRAM* に対する要求があった（図 3の $FRRAM_{(i,j)}$ の CR 参照）．

Relayed:自分が Y 方向バスと X 方向バスの中継を行った（図 3の $FRRAM_{(i,r)}$ の CR 参照）．

RMR および CR は FIFO として動作する．$FRRAM_{(i,j)}$ は，データの転送準備完了後この CR の内容に従って PU_j へのデータの転送を行うため，新たに帰路のアービトレーションを行う必要がない．これにより，*FRRAM* から X 方向バスへアクセスする場合，アービトレーション・レイテンシは生じない．以下が各 *FRRAM* の動作を示す．

i.CR の最も古い情報が "Read" である $FRRAM_{(i,k)}$ は，次に X_i にデータを流すことができることを示している．データの転送準備が完了する 1 サイクル前に sc に信号を流し，sd により次サイクル X 方向バス使用信号を確認後，X_i にデータを流す．

ii."Relayed" である $FRRAM_{(i,j)}$ は sc に信号が流れたとき，s6 が自分を指しているならば sd および s7（書き込み時に使用）に信号を流す．また，次サイクルで X_i と Y_j を中継する．

iii."Other" である *FRRAM* は何も行わない．

転送が終了後，図中に示すシフト方向に CR をシフトさせ最も古い情報を消す．これにより次に X 方向バスが使用可能な *FRRAM* が一意に決定し，かつ使用する Y 方向バスも同時に判明する．

3.4 *FRRAM* へのデータの書き込み

$FRRAM_{(i,k)}$ への書き込みも読み出しの場合と同様に行う．書き込みでは，読み出し要求の代わりに書き込み要求を出す点が異なる．読み出し要求が読み出し先アドレスだったのに対し，書き込み要求は書き込み先アドレスである．要求は，CR に登録され順番が来るのを待ち sc に信号を出す．PU は s7 によって *FRRAM* が書き込みが可能となることを知り，次サイクルから書き込む

データをバスに流す．

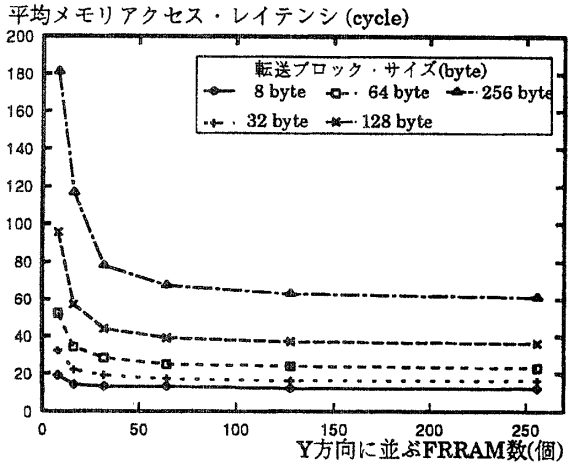


図 6: *FRRAM* 数とメモリアクセス・レイテンシの関係

3.5 Y方向バスのみを使用するメモリアクセス

PUと同一Y方向バス上に存在する *FRRAM* に対するアクセスは前述の方法とは異なる．X方向バスを使用しないため，X方向バスの競合を避けるためのアービトレーションを行う必要がない．よって，X方向バス使用予約を出す必要がなく，要求を受けとっても *FRRAM* はCRに記憶しない．以下，読み出しと書き込みに分けて説明する．

- 読み出し：PUが次サイクルにおいて，読み出しおよび書き込みを行わないとき，PUと同じY方向バス上の *FRRAM* に対する要求を行うことをs4によって伝える．これにより，このY方向バス上のすべての *FRRAM* は次サイクルにおいてX方向バスへの中継を行わない状態となる．そのときPUはY方向バスに要求を流す．要求が受理されたらPUはRMRにY方向バスのみを使う読み出し要求を行ったことを保存する．

  RMRが自分の順番となるのを待ち，s6でデータ読み出し時は *FRRAM* のあるX方向バス番号を，s8でY方向バスのみを使うアクセスであることを示す．この *FRRAM* の読み出し準備が完了次第，s7によりPUにそれを知らせる．これで，次サイクルからデータの読み出しが開始される．
- 書き込み：PUは書き込み要求を出す前に，まずY方向バスのみを使った書き込みを行なうことをRMRに登録する．PUがRMRに従って順にデータ転送を行い，自分の順番が来るのを待つ．次に読み出しと同様に要求を出し，そのまま続けて次サイクルからデータもY方向バスに流すことで書き込みを行う．

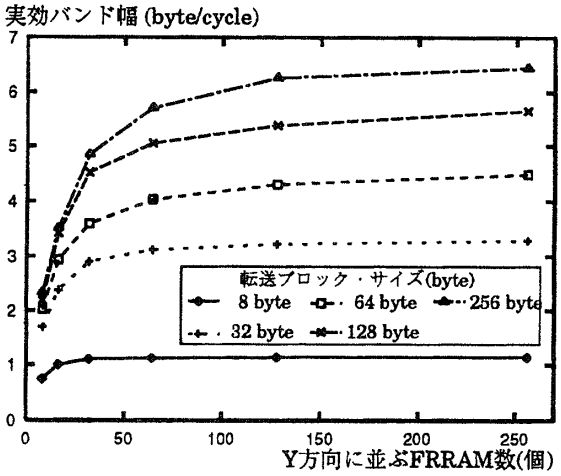


図 7: *FRRAM* 数と実効バンド幅の関係

4 シミュレーションによる評価

本節ではメッシュ型 *FDUMA* メモリシステムにおける構成が性能に及ぼす影響を，ランダムにデータをブロック転送することによって評価する．

4.1 *FRRAM* 数が性能に及ぼす影響

PUおよびX方向の *FRRAM* を16個に固定し，Y方向の *FRRAM* 数を変化させることによる平均メモリアクセス・レイテンシ，およびメモリバンド幅への影響をソフトウェア・シミュレータを用いて調べる．また，ブロック転送サイズの違いによる性能への影響も同時に調べる．

FRRAM 数を増加させた場合，図6に示す通りある程度まではレイテンシの短縮がみられ，また図7で示す通りバンド幅の増加もみられる．これは *FRRAM* 数の増加により，Y方向の *FRRAM* 数が少ないときにボトルネックとなっていたX方向バスの通信量が緩和されるためである．しかし，*FRRAM* 数の増加を続けると今度はY方向バスの使用率が限界に近くなるため性能が向上しなくなる．この評価を基に，Y方向の *FRRAM* 数をX方向の2倍から4倍程度用意した，Y方向に伸びた長方形のシステムを構成するのがコスト，性能の両方の面で効率がよいといえる．また，転送するブロック・サイズに注目すると，ブロック・サイズを増加させるとバンド幅が向上し減少させるとメモリアクセス・レイテンシが向上する．これより使用するアプリケーションの用途にあわせてメモリアドレス割り付けを最適化する必要があることがわかる．

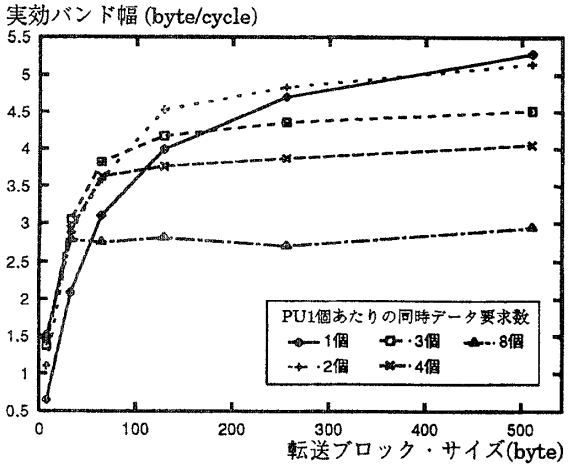


図 8: 輻輳制御によるレイテンシの変化

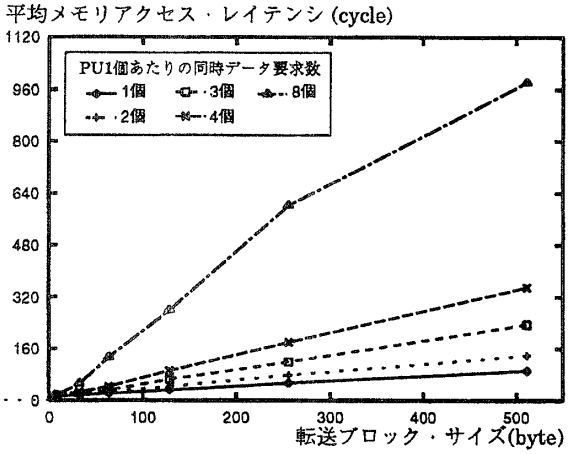


図 9: 輻輳制御によるバンド幅の変化

4.2 輻輳制御が性能に及ぼす影響

PU がメモリアクセスを行うとき，データ転送をせずして要求のみを出し続けると 1 つの PU が多くの *FRRAM* を占有する．他の PU が その *FRRAM* へのアクセスを行うとき，最悪 1 つの PU が現在占有しているすべての *FRRAM* に対するデータ転送を待たなければデータ転送を開始できない．本項では，この問題を避けるために各 PU の要求数を制限する輻輳制御を行うが，どの程度行うのが効率が良いのかを評価する．評価は 16PU，*FRRAM* が x 方向に 16 個 Y 方向に 32 個のシステムで行った．

図 8より単位転送ブロック・サイズが 8 から 64 (byte) 程度では PU1 個あたりが *FRRAM* に対して同時に出せる要求の数（同時データ要求数）は 3，64 から 256 (byte) の間では 2，それ以上では 1 にするのが性能がよい．また，図 9より同時データ要求数が少ないほど平均メモリアクセス・レイテンシが小さく性能がよいことがわかる．この 2 つの図より，同時データ要求数は広い範囲で高い性能を示している 2 を採用するのが望ましいといえる．

5 まとめ

本稿では *FDUMA* メモリシステムのアービトレーション方式について述べ，その方式をソフトウェア・シミュレーションにより評価した．当初試作を計画していた *FRRAM* は正方形に配置したシステムであった．しかし本評価から，期待する性能を得るには少なくとも X 方向に並べた *FRRAM* 数の 2 倍の *FRRAM* を Y 方向に用意すべきであることがわかった．輻輳制御では同時データ要求数の制限を 2 にすれば，各ブロック・サイズで平均してパフォーマンスがよいことがわかった．今後，本評価をもとに *FDUMA* メモリシステムの詳細設計を行っていく予定である．

謝辞

日頃ご討論戴く広島市立大学 情報科学部 情報工学科の高山毅助手，およびコンピュータシステム講座の諸氏に感謝致します．

参考文献

[PAC+97] David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberley Keeton, Christoforos Kozyrakis, Randi Thomas, and Kathy Yelick. Intelligent ram (iram): Chips that remember and compute. In *1997 IEEE International Solid-State Circuits Conference*, February 1997.

[弘中 97] 弘中哲夫，森垣利彦，児島彰．FDUMA 共有メモリ・アーキテクチャにおけるバス・アービトレーション． 第 5 回 FPGA/PLD Design Conference 予稿集, pp. 223–231, Tokyo, Japan, June 1997.

[村上 96] 村上和彰，岩下茂信，宮嶋浩志，白川暁，吉井卓． メモリ－マルチプロセッサ一体型 aspp(application - specific standard product) アーキテクチャ：ppram. Technical report, 信学技法 CPSY96-13, 1996.