計算機アーキテクチャ 149－16
（２００２．８．２３）

メモリアクセス列の最適化を行うメモリインタフェース

小川 周吾† 平木 敬†

†東京大学大学院 情報理工学系研究科 コンピュータ科学専攻

要 旨

近年の CPU 処理速度の大きな伸びに対し，メモリのアクセス速度の向上は小さい．その結果，CPU と外部との速度の差が増大し，メモリなどの外部のデバイスへのアクセスがシステム全体の速度を制限する．さらに，複数プロセッサでメモリを共有するシステムでは，複数の CPU が同時にメモリへのアクセス要求を発生させるため，アクセス速度に関してより十分に考慮する必要がある．
本稿では複数プロセッサ環境でメモリアクセスを高速化する方法として DDR SDRAM のバンク並列性の利用を提案する．実験として複数の CPU とメモリ，その間を結ぶインタフェースのシミュレーションを行い，シミュレータ上のプログラムの実行結果から性能を評価する．この実験結果に基づいて，メモリアクセスの高速化に適したインタフェースの設計の方向性について提案する．

Memory Interface for Optimizing Memory Access Sequences

Shugo Ogawa† Kei Hiraki†

†Graduate School of Information Science and Technology, University of Tokyo

Abstract

Compare with the increase of CPU speed, the decrease of memory latency is slow. As a result, memory latency often limits the performance of whole system. Furthermore, we need to regard memory access speed more carefully on multiprocessor system because more than two processors access memories at the same time.
In this paper we propose the method for multiprocessor computer to make use of the parallelism of banks in DDR SDRAM. We made simulators which simulates several CPUs, memory and the interface connecting them. We evaluate the performance of memory accesses on the results of programs which we run, and we propose the better interface which can reduce the access latency.

1 はじめに

既に指摘されているように，今日の計算機の性能の向上において最も大きな障害となっているのは外部デバイスへのアクセス，特にメモリアクセスの速度である．近年，CPU は動作周波数の向上，アーキテクチャの改良により動作速度を向上させ続けている．

しかしながら，プロセッサの目覚ましい速度向上と比較すると，メモリの動作速度向上は緩やかである．現在，計算機のメインメモリとして多く用いられている DRAM の動作周波数は CPU の動作周波数に比べると低い．また，DRAM はその構造上アクセスする際にアドレスを Row と Column に分けて指定する必要があり，さらにアクセス後に Precharge を行う必要があるためアクセスがより遅くなる．

メモリアクセスの速度を補う手法の一つとしてキャッシュメモリが用いられる．キャッシュメモリを置くことによって頻繁にアクセスするデータを高速にアクセスすることを可能にし，また SMP ではキャッシュを用いることによりメモリのトラフィックが減少する [1] ことで性能を向上させることができる．

しかしキャッシュが必ずしもヒットするとは限らず，また，アプリケーションによってはキャッシュは効果が薄い [5]．キャッシュミスが発生した場合はメモリにアクセスをするのでメモリのアクセス速度，メモリアクセスのスループットの向上がシステムの性能向上につながる．

通常のプログラムでは load/store 命令が頻繁に出現するため，メモリアクセスの遅れによってデータ依存関係や命令フェッチの遅れが発生して全てのプログラム実行が停止し，性能の低下を招く．

SMP では状況がより困難である．SMP では，メモリを複数のプロセッサで共有するために 1 つのメモリに対して複数の CPU から同時にアクセスが発生し，プロセッサの内部と外部の速度差はより深刻となる．更に，複数のプロセッサ，メモリ，各種入出力デバイスを制御するためにそのデータパスはより複雑になり，メモリアクセスに要する時間を増大させる要因となる．即ち SMP ではメモリアクセスの速度がシングルプロセッサの計算機以上に要求され，メモリアクセス速度の低下が計算機の性能をシングルプロセッサ時以上に制限する．

本稿では複数プロセッサのシステムにおけるメモリインタフェースの改善，特にメモリのバンク並列性を利用したインタフェースを提案し，システム全体の性能に与える影響をシミュレータを用いた実験によって検証する．実験では SMP 環境のシミュレータを用いて，シミュレータの各プロセッサ上で SPEC CPU2000 INT のベンチマークプログラムを動作させて性能を計測する．また，プロセッサ数，動作周波数，インタフェースの違いによる性能の比較を行う．

以降，2 章においてメモリアクセスの高速化手段について述べる．3 章では今回の実験についての説明，4 章ではその結果について述べ，5 章でまとめる．

2 メモリアクセスの高速化手法

2.1 メモリアクセスの高速化

メモリの特徴を利用してメモリアクセスの並列度を高める手法は，メモリへのアクセスを高速化するアプローチの一つである．メモリアクセスの並列度を高めることでメモリアクセスのスループットを向上させ，メモリに直接触れる時間が短縮される．また，メモリアクセスを高速化するアプローチには，データパスを改善することで 1 回のメモリアクセスに要する時間を減少させる手法がある．データパスの改善はメモリにアクセスする時間を減少させるのではなく，プロセッサのアクセス要求がメモリまでより短時間で伝わるようにする，或いはアクセス結果がより短時間で伝わるようにすることでプロセッサが要求を出してから結果が返るまでの時間を短縮する．

本稿では以上の高速化手法を併用したインタフェースを用いた場合の性能を比較し，メモリアクセスの高速化に効果的なアプローチを検証する．性能の比較に用いた手法について以下で述べる．また，以下ではメモリは現在多くの計算機で利用されている DDR SDRAM を使用すると仮定する．

2.2 バンク並列性を利用した高速化

DDR SDRAM にアクセスするためにはアドレスを Row, Column の順番に 2 段階で別々に指定する必要がある．その後，同じメモリの別の row にアクセスするためには Precharge を行う必要がある．DDR SDRAM へのアクセスは以上の複数の手順を踏む必要があるためアクセスレイテンシは大きい．データパスの改善によらずに DDR SDRAM へのアクセス手順の改善によってアクセス速度を向上させるには，アクセスの並列性を高めることでスループットを高める手法を提案することが必要になる．

本稿では DDR SDRAM のバンクの並列性を利用した高速化手法を提案する．既に論文 [2] で SDRAM のバンク並列性を利用するスケジューリングの提案がされているがメディアプロセッサでの適用を仮定したものである．本稿では複数の汎用プロセッサを持つシステムでの適用による性能の改善について検証する．

DDR SDRAM は 4 つのバンクに分かれ，それぞれのバンクが互いに別々の状態を持つことができ，個々のバンクを別々に操作することができる．また，row, column 2 回のアドレスの指定，precharge などは一連の動作として連続して一度に実行する必要はなく，途中に別のバンクへの操作を挟んで任意の間隔で実行することができる．

これを利用してあるアクセスに対して次の動作を行うまでの待ち時間の間にバンクの異なる別の部分

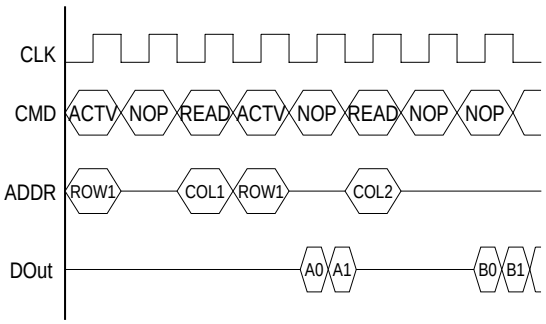

図 1: DDR SDRAM への通常のアクセスの様子

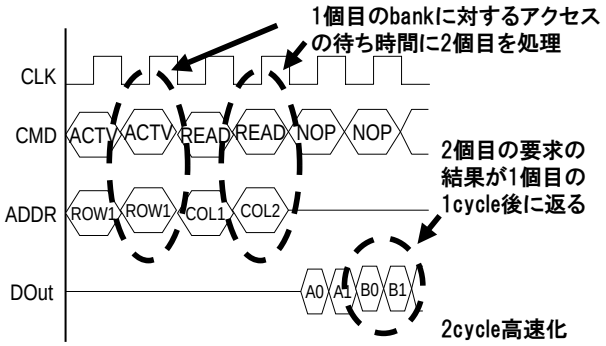

図 2: 異なるバンクへの操作を並列化した DDR SDRAM へのアクセス

にもアクセスを行うことが可能になる．また row が同一のアクセスについては直接 column を指定することによって高速化することができる．これらをスケジューリングによって効率的に利用することにより動作の並列性を高めることが可能である．

2.3 データパスの高速化

多くの計算機システムでは CPU とメモリが直接接続されているのではなく I/O をはじめとする他のトランザクションを処理するためにバッファ，スイッチがデータパス中に存在する．途中経路でのレイテンシは全てのメモリアクセスに関して一律に影響を及ぼすので，メモリアクセスが多くデータパスが複雑な環境である SMP 環境ではデータパスによるレイテンシが大きな問題となる．

前項に加え，本稿ではデータパスの高速化を DDR SDRAM のバンク並列性を用いた高速化と共に適用した際のメモリアクセスの性能について検証する．本稿で用いたデータパスの高速化を以下に挙げる．

2.3.1 read/write 別のバッファ

CPU が命令を実行する際に load 命令に関してはメモリアクセスの後にその結果を利用するため，メモリアクセスの処理時間が CPU の性能に影響を及ぼす．一方 store 命令を実行する場合は，命令の結果が返ってくることがなく，またメモリに書き込んだデータは次に同一のアドレスから読み出しが行われない限りは使用されることはないため，store 命令の処理時間が CPU の性能に与える影響は load 命令よりも小さい．この事実を利用して load 命令を優先的に実行するようにすることによりメモリアクセスの高速化が可能になる．

実際にデータパスに加える変更は，CPU 内部のプログラム実行で直接データ依存関係に影響する load を後に同一アドレスを読むまで依存関係に影響が起こらない store よりも優先して処理する機構として，メモリに対する読み出し要求と書き込み要求を別々のバッファで保持するようにする．そしてメモリの読み出し要求の格納されたバッファを優先的に処理するように制御を行う．

しかし，書き込みを行った同一アドレスに対する読み出し，及び書き込み要求が後から発生した場合には順序関係を保たなければハザードが発生する．書き込みに関しては書き込みを in-order で処理することでハザードが発生することはなくなるが，読み出しに関してはアドレスの比較を行うことでハザードを回避する必要があり，読み出し要求をメモリに送る前にアドレスの比較を行いハザードを回避する．

2.3.2 結果のバイパス

CPU がメモリの読み出しを行った際に，バッファ中にまだ処理されていない以前の同一アドレスへの書き込み結果が残っている場合はその結果を直接メモリアクセスの結果として返すようにすることでアクセス時間を短縮することができる．Writeback キャッシュを実装した CPU の場合にはキャッシュラインから追い出されて書き戻されるデータがバッファに残っている場合に効果的である．

2.3.3 CPU へのバスの割り当て

バス共有型の SMP システムではメモリアクセスが複数の CPU から行われるので，バスを割り当てるアルゴリズムがメモリアクセスの性能に影響を与える．

CPU へのバスの割り当ては CPU によって割り当てに差が生じないようにする必要がある．しかし，命令間のデータ依存関係の距離の観点ではメモリアクセスの要求が同一 CPU から連続して発生した場合には，可能な範囲で連続して処理を行うことによってプログラムのデータ依存関係が早く解決できる可能性がある．

本稿では CPU の割り当てに関してラウンドロビン方式を利用した方法と，ラウンドロビン方式を用いながらも一定数の連続アクセスを優先させる方法の 2 種類のアルゴリズムを用いる．

2.3.4 内部スイッチの改良

CPU とメモリをはじめ，外部デバイスにつながるバスを結ぶコントローラでは各方面からの大量のバストランザクションを処理する必要がある．これに対応するためにコントローラ内部ではこれらはスイッチによって結ばれている．この内部スイッチの改良によって高速化を図ることが可能であるが，本稿ではメモリアクセスの高速化に着目するため，内部スイッチのアーキテクチャについての議論は行わない．

3 評価環境

3.1 アーキテクチャモデル

3.1.1 全体概要

バンク並列性を利用した高速化，データパスの改善による高速化を行った際のメモリアクセス性能を比較検証するため，複数プロセッサ，メモリとこれらを結ぶインタフェースチップから構成されるアーキテクチャを仮定する．

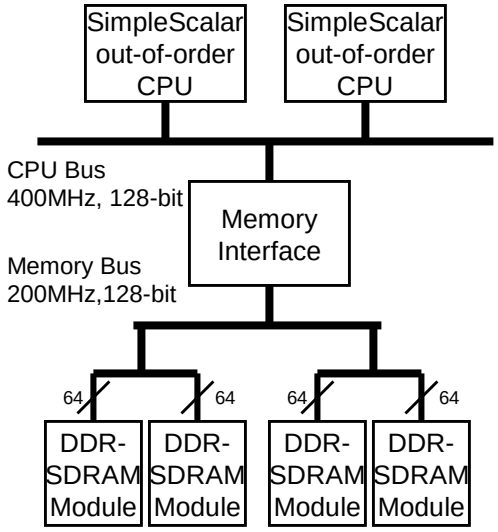

図 3: シミュレーションするアーキテクチャの全体ブロック図

まず，CPU とインタフェース間のバスは単一であり全ての CPU で，一度にアクセス要求を出すことのできる CPU は 1 つに限られる．CPU とメモリインタフェース間のバスはアドレスバスが 32-bit，データバスは 128-bit を仮定している．load/store されたデータはこのバスを通じて CPU とインタフェースとの間を流れる．

各 CPU から出された要求はインタフェース内部のメモリコントローラに到達し，バッファに格納されて処理される．

システムの周波数はバス及びインタフェース内部では 400MHz，DDR SDRAM チップでは 200MHz を仮定する．

CPU に関する設定は以下の通りとする．

- 動作周波数: 2.0GHz, 4.0GHz, 6.0GHz
- 命令の out-of-order 実行をサポート
- Cache line : L1 32-byte, L2 64-byte
- outstanding なメモリアクセス数は最大 4 個

CPU のアーキテクチャは SimpleScalar の標準アーキテクチャを使用する．

3.1.2 メモリ

メモリは 128Mbit, 200MHz, CAS Latency=3 の DDR SDRAM チップを仮定した．これを片面 16 個実装した 512MB のモジュールを仮定し，このモジュール 2 つに対して一度に 64-bit ずつ 128-bit 単位でアクセスすると仮定する．

物理アドレス空間の割り当ては，図に示したように下の 4-bit が一度にアクセスを行う word を表し，その上の 10-bit がバースト転送，連続アクセスが可能な column のフィールドを表す．更にその上に row, bank 番号，モジュール番号を割り当てる．

3.1.3 メモリインタフェース

CPU とメモリとの間を結ぶインタフェースの構成は図 4 に示したとおりである．

まず，CPU から要求が 1 つずつ入ってくる．要求は図にあるリクエストバッファに格納される．ここから順番に内部スイッチを通ってメモリモジュールへのアクセス用バッファに格納される．このバッファは読み出し用，書き込み用に分離されており，要求は割り振られて格納される．このバッファからさらにメモリチップ内部の各バンクに対応するバッファに要求が転送されて格納される．このバッファに格納された要求は，メモリのアクセス幅の単位で分割されて処理される．例えば，64-byte のアクセスであれば 16-byte 単位のアクセス 4 個に分割されて処理される．このバッファ内部ではメモリアクセスのレイテンシが削減されるように同一 column のアクセスを優先する処理を行う．そしてこのバッファのアクセス要求に従ってメモリに対してアクセスが行われ，読み出しの場合には結果を格納するバッファに蓄えられ，そして先程分割されたアクセス要求は再構成される．そして再構成されたアクセスの結果は CPU にデータを送るためのバッファに格納され，順番に CPU に返される．

このインタフェース内部及びバスの動作周波数は 333MHz を仮定する．この内部のバッファを遷移する際にそれぞれ 1 サイクル以上の処理時間がかかると仮定する．

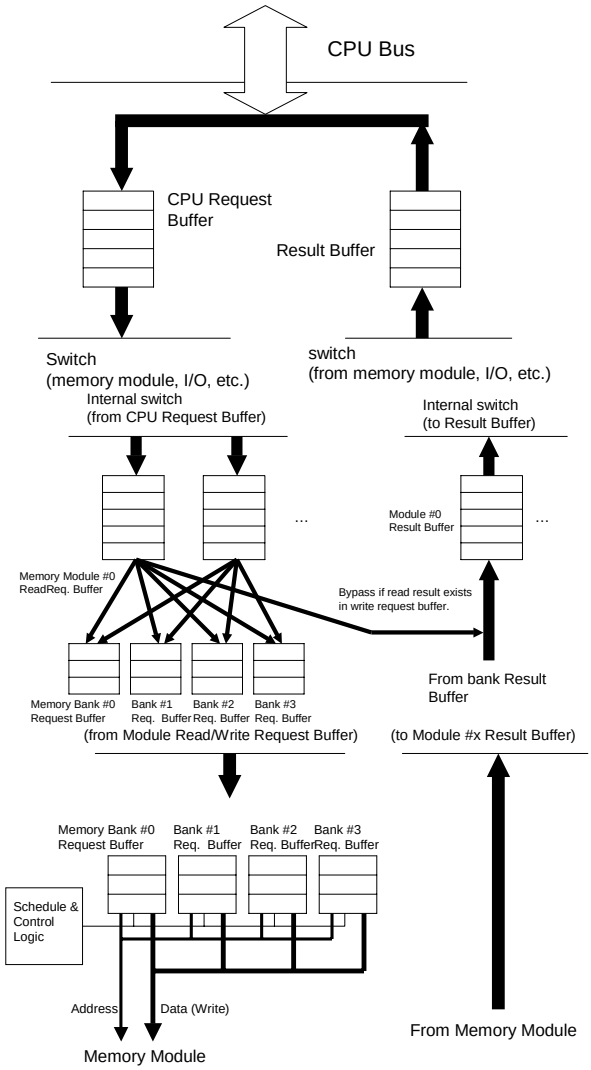

図 4: メモリインタフェースのブロック図

3.2 シミュレータ

CPU のシミュレータとして SimpleScalar 3.0c を用いた．これが CPU の 1 個に対応する．これらの間のメモリアクセスとそのレイテンシの計算及び同期を行うために外部に新たなプログラムを作成し，これらの間で通信を行い，CPU 毎に別のプログラムを並列に実行することでシミュレーションを行い，データを収集した．

性能評価に用いたプログラムは SPEC CPU2000 INT の 175.vpr, 181.mcf をテストデータを用いて各プロセッサ上で実行し，その実行時間，メモリアクセス時間を計測する．アーキテクチャ，プロセッサ数，プロセッサの動作周波数による性能の変化のデータを収集した．

3.3 予備実験

本稿では予備実験としてシングルプロセッサ時にメモリアクセスを 1 つずつ順番に処理した場合とバンク並列性を利用したインタフェースを用いた場合とで比較，検証を行う．予備実験でのアーキテクチャはプロセッサ，メモリのバス幅は 128-bit から 64-bit に，メモリアクセス単位は 128-bit から 64-bit を仮定する．また，プロセッサの動作周波数は 3.33GHz，メモリの動作周波数は 166MHz，メモリインタフェースの動作周波数は 333MHz を仮定する．

4 実験結果

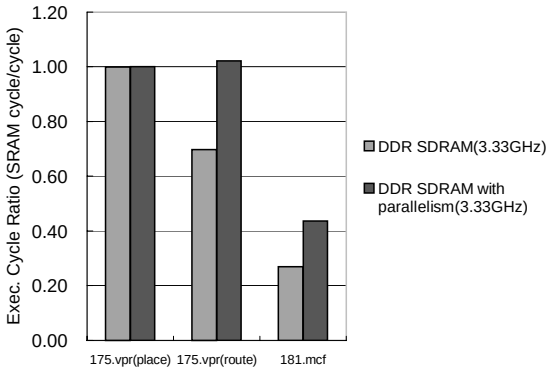

図 5: バンク並列性利用の有無による実行時間比較

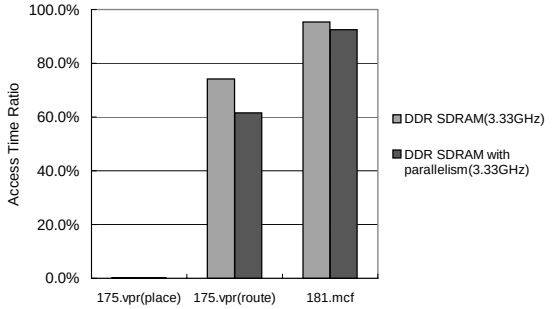

図 6: メモリアクセス時間の実行時間に占める割合

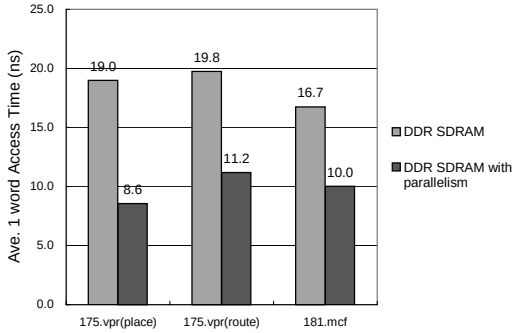

図 7: 64-bit あたりのメモリアクセスの実時間

メモリアクセスを順番に処理した場合及びバンク並列性を利用した場合の結果は図 5 である．グラフ

では 175.vpr の placement 部分では両者の性能に差は表れていないものの，175.vpr の routing 部分及び 181.mcf ではバンク並列性を利用している場合の方が高い性能を示す．175.vpr の placement で両者の性能に差が見られないのは図 6 で示されているように他のプログラムに比べてメモリアクセスに要している時間が短いことによる．175.vpr の placement 部分ではメモリアクセスを行っている時間の割合が他のプログラムと比較すると小さい．実行時間に限らず実際に DDR SDRAM へのアクセスに要した時間についてもバンク並列性は効果を示す．実際に DDR SDRAM に対する 64-bit のメモリアクセスに要した平均に時間は図 7 の通りであり，全てのプログラムでバンク並列性の利用によりメモリアクセス時間が短縮される．

この実験結果から，メモリインタフェースでのバンク並列性の利用はメモリアクセス時間の短縮，システムの性能向上に有効であることが分かる．

5 まとめ

本稿では SMP におけるメモリアクセスの高速化するインタフェースとしてバンク並列性を利用する手法を提案した．また，データパスの改善による高速化手法を提案した．これらの高速化手法を用いて改良したインタフェースと，用いていないインタフェースとを比較すると，改良したインタフェースでは改良をしていないものに比べて高い性能を示すことが実験から示され，バンク並列性の利用はメモリアクセスを高速化し，システムの性能向上に有効であることが示された．また，バンク並列性の利用によるメモリアクセスの高速化は，メモリアクセスの回数が多いプログラムでより効果があることが示された．

今後はより多くのプログラム，インタフェースアーキテクチャでの性能の比較検証，より正確なシミュレーションを実現するシミュレータの改良を行う予定である．

参考文献

[1] J. R. Goodman: Using Cache Memory to Reduce Processor-Memory Traffic, ISCA 1983, pp.124-131

[2] Scott Rixner, William J. Dally, Ujval J. Kapasi, Peter Mattson, and John D. Owens: Memory Access Scheduling, ISCA 2000, pp.128-138

[3] Sung I. Hong, Sally A. McKee, Maximo H. Salinas, Robert H. Klenke, James H. Aylor, Wm. A. Wulf: Access Order and Effective Bandwidth for Streams on a Direct Rambus Memory, HPCA 1999, pp.70-79

[4] Vinodh Cuppu and Bruce Jacob: Concurrency, Latency, or System Overhead: Which Has the Largest Impact on Uniprocessor DRAM-System Performance?, ISCA 2001, pp.62-71

[5] Perl and R. Sites: Studies of Windows NT Performance using Dynamic Execution Traces, Proceedings of the Second USENIX Symposium on Operating Systems Design and Implementation, 1996, pp.169-184

[6] Doug Burger and Todd M. Austin: The SimpleScalar Tool Set, Version 2.0, Tech. Rep. 1342, University of WisconsinMadison, CS Department, June 1997

[7] JEDEC Solid State Technology Association: Double Data Rate (DDR) SDRAM Specification, http://www.jedec.org/, 2000

[8] Elpida Memory Inc.: Preliminary Data Sheet E0086H10, http://www.elpida.com/pdfs/E0086H10.pdf, 2001

[9] Elpida Memory Inc.: SDRAM の使い方, http://www.elpida.com/pdfs/J0123N11.pdf, 2001

[10] Elpida Memory Inc.: DDR SDRAM の使い方, http://www.elpida.com/pdfs/J0234E20.pdf, 2002

[11] Micron Technology, Inc.: 256Mb: x4, x8, x16 DDR SDRAM, http://download.micron.com/pdf/datasheets/dram/256Mx4x8x16DDR_C.pdf, 2001

[12] Micron Technology, Inc.: 256Mb: x4, x8, x16 DDR333 SDRAM Addendum, http://download.micron.com/pdf/datasheets/dram/256Mx4x8x16DDR333_B.pdf, 2001

[13] Micron Technology, Inc.: 128Mb: x4, x8, x16 DDR SDRAM, http://download.micron.com/pdf/datasheets/dram/128Mx4x8x16DDR_C.pdf, 2001

[14] Micron Technology, Inc.: 128Mb: x4, x8, x16 DDR400 SDRAM Addendum, http://download.micron.com/pdf/datasheets/dram/128Mbx8DDR400.pdf, 2002