計算機アーキテクチャ　145－6
（２００１．　１１．　２９）

レイテンシ耐性を有するコンピュータシステムの研究

孕石 裕昭*、宮坂 和幸*、清水 尚彦**
* 東海大学 大学院 工学研究科
** 東海大学 電子情報学部

* {1aepm041,0aepm049}@keyaki.cc.u-tokai.ac.jp
** nshimizu@keyaki.cc.u-tokai.ac.jp

コンピュータシステムにおけるメモリレイテンシの改善は演算性能に追いつく程には成されておらず、プロセッサとメモリ間の性能格差が深刻な問題となっている。

レイテンシ隠蔽技術は性能差が開くほど複雑度が増大する傾向にある。

ここでは科学技術計算分野で有効な方法としてバッファを用いたレイテンシ隠蔽技術とそれによって実現されるコンピュータシステムについて述べる。

キーワード：レイテンシ隠蔽技術, バッファ, メモリシステム、スループット

A stady of the computer system that is intended to have latency tolerance.

Hiroaki Haramiishi* , Kazuyuki Miyasaka* , Naohiko Shimizu**
* Graduate school of Engineering, Tokai Univ.
** Dept, Communication Engineering, Tokai Univ.

In the computer system, the improvement of a memory latency has not been accomplished as the processor performance improvement. The serious performance gap has been greater and greater.

In this paper, we propose a computer system that is realizing the latency-tolerance using a buffer as an effective method for High Performance computing.

Keywords : Latency, Buffer, Memory system

1 はじめに

コンピュータシステムの性能を最大限に引き出すためにはメモリ性能が重要となってくる。コンピュータシステムの性能を向上させるためには演算性能の向上とともにメモリスループットの向上も重要である。

現在のコンピュータシステムにはDRAMが用いられおり、データバス幅の拡張、動作周波数の向上などによりスループットの向上をはかっているが、DRAMのみではプロセッサの演算性能に見合うスループットは十分には得られていない。

このメモリレイテンシの隠蔽方法にはキャッシュ、データプリフェッチ、Out-of-Order など様々な方法がに用いられている。しかしキャッシュはハードウェアによるコンシステンシの管理が必要であり、マルチプロセッサ構成などではキャッシュパージによるシステムバスの占有が問題となっている。Out-of-Order 実行については相対レイテンシの増大に伴い複雑度が増大する傾向にある.

我々の研究ではメモリ-プロセッサ間のデータをキャッシュのようなコンシステンシを保つことがなく、ソフトウェアから明示的に管理できるバッファを用いることでレイテンシを隠蔽する Scalable Latency Tolerant Architecture(SCALT) というものを提案している。

本論文ではバッファを用いたレイテンシ隠蔽アーキテクチャと DRAM へのアクセス最適化を行うメモリシステムの設計について述べる。

メモリシステムの設計には NTT で開発された PARTHENON システム (HDL に SFL) を用いた。

2 システム構成時の問題点

コンピュータシステムにおける基本的な要素の内、ここではメモリシステムについて検討する。

システムの演算性能は動作周波数、並列度の向上にともない増加し続けているが、メモリ速度は演算性能ほどには性能が向上していない。これにより相対レイテンシは増加している。

メモリシステム性能がボトルネックとなるものにはソートや FFT、ポインタ指向アプリケーションなどのようにメモリアクセス-演算比の大きいもの、メモリアクセスが連続的でないものなどが挙げられる。

これに限らずプログラムは問題が大きくなるにつれ、メモリのアクセス範囲が広く、キャッシュメモリの効果が薄れる傾向にある。

またキャッシュメモリはプロセッサの動作周波数向上とともにアクセス時間を短縮しなければならないため、キャッシュメモリ容量はそれほど増加していない。

キャッシュメモリによる性能向上が期待できないアプリケーションではメインメモリのスループットがシステム設計時の重要な問題となる。

2.1 メモリシステムとの親和性

コンピュータシステムで広く用いられている DRAM は集積度とともに高スループット化が進んでいる。

代表的な技術には次のようなものが挙げられる

- クロック同期
- バースト転送
- 1 チップマルチバンク化
- クロックの多相化
- キャッシュ

これらの技術により連続したアドレスへのメモリスループットはピーク性能が期待できるほどに向上されている。

例として SDRAM における複数のバンクへの分散アクセスタイミングチャート図 1 に示す。SDRAM のバースト長は 4 としている。

DRAM へのアクセスは図 1 のようにバンクアクティブ、Read/Write、データの転送、プリチャージという 4 段階のアクセスを必要とする。この間、各バンクは 1 つのリクエストしか受け付けることはできない。そのため SDRAM 以降の DRAM では 1 チップ内に 4 以上のバンクがあり、各々のバンクが独立した動作を行うことによりアクティブ、プリチャージ時間を隠蔽することが可能となっている。

メモリシステム性能はアクセス時間の短縮が重要な要素となり、各バンクへのアクセスの分散がそのままメモリシステムの性能となる。

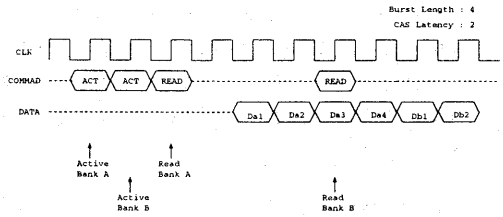

図 1: SDRAM タイミングチャート

2.2 プロセッサアーキテクチャに対する要求

メモリシステムはプロセッサから発行されるメモリリクエストを受け付け、DRAM 上記の技術を有効に活用できるようにメモリアクセスを並び変えることで高いスループットを期待できる。しかしプロセッサは load/store は発行順に処理することを期待するが、この順序依存関係が特にランダムアクセスの場合にメモリシステムのアクセス最適化を妨げる要因となる。プロセッサ側も順序依存の変更に対応しているが、これらの管理にアドレスを用いるため load/store の同時発行数はそれほど多くはならない。プロセッサからの順序依存の無いの同時発行数が増加するということはそれだけメモリシステム内にリクエストが増加し、メモリへのアクセス最適化が増加することを意味する。

アクセス最適化を行う要素数によるランダムアクセス性能を図 2 に示す。図 2 ではシーケンシャルアクセスに対するランダムアクセス性能を最適化キューの数による所要サイクル数の変化を示す。

図 2 より CPU からの同時発行数 (最適化要素数) が増加するに従い、データ転送に要するサイクル数が連続的なアクセスに近づく。これは図 1 のようにキュー内のリクエストをバンクごとにアクセスを分散することによってバンクコンフリクトを低下させることが有効であることを示している。

3 Scalable Latency Tolerant Architecture(SCALT)

SCALT アーキテクチャは実装が容易であり、かつ現在のアーキテクチャとの親和性の高いアーキテクチャとなっている。

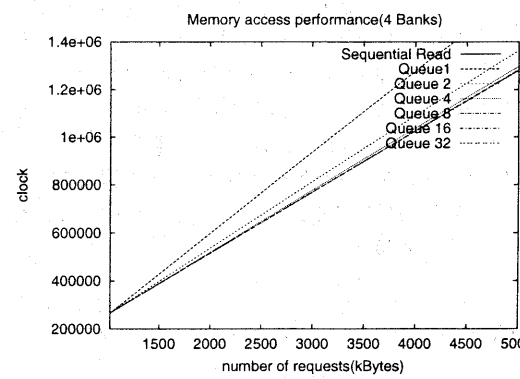

図 2: 最適化要素数によるメモリアクセス性能

SCALT アーキテクチャではキャッシュメモリではなく、現在のキャッシュメモリと同様の SRAM 技術を用いたバッファメモリによってメモリレイテンシを隠蔽する。キャッシュとバッファとの相違点は記憶されているデータがメインメモリのコピーであるか、そうでないかということである。つまりバッファはメモリとの一貫性を保つ必要が無くなる。

バッファの 1 エントリはキャッシュライン長とし、バッファ用命令を用いてバッファーメモリ間でエントリ単位のデータ転送を行う。

3.1 アーキテクチャ決定考慮点

3.1.1 load/store アーキテクチャとの親和性

SCALT は従来型の load/store アーキテクチャに対して数命令の追加で実現できる。追加された命令は従来の load/store 命令とほぼ同様の動作であり、この命令の実装のためにプロセッサのメモリパイプラインが大きく異なるということはない。

1 回のアドレス変換でエントリ単位のデータが移動するため、従来型の RISC のアドレス計算パイプラインをそのまま利用できる。また、キャッシュリフィル/パージ論理とデータパスを共有でき、キャッシュとバッファはともに load/store によってアクセスする。そのためバッファはソフトウェアから明示的に管理するために仮想アドレスにマッピングされる。

バッファへのアクセスは TLB により管理され、キ

ャッシュへのアクセスを阻害することは無い。そのためTLBからキャッシュへのプロセッサのクリティカルパスとなる部分に余計な論理は入らない。

SCALTではバッファのエントリを用いた"tag"という識別データを用いた同時発行管理を行う。"tag"はバッファのエントリ番号であり、8bitの情報であるため、同時発行数が増加してもプロセッサの設計を困難にすることは無い。

3.1.2 バッファのソフトウェアコンテキスト化

バッファのソフトウェアによる保有を前提とすることで、効率的な制御が可能となる。

キャッシュメモリではハードウェアにより記憶領域が管理されており、またメモリとの一貫性が必要なためアドレス、状態を保持しなければならない。しかし、SCALTにおけるバッファではハードウェアによる管理情報はデータが存在するかどうかを1bitのフラグでのみ判定するため、ハードウェアが簡潔になる。

3.2 専用命令

SCALTプロセッサにおけるバッファ専用命令は次の3命令がある。

- Buffer fetch

  バッファ-メモリ間のデータ転送命令であり、1命令で1エントリ(現在の実装ではキャッシュライン長)の単位でデータを転送する。ディスティネーションとなるバッファエントリとメインメモリのアドレスを指定し、データの到着はエントリの1bitのフラグをチェックすることで確認でき、プロセッサの後続の命令実行を妨げることはない。

- Buffer store

  バッファ-メモリ間のデータ転送命令であり、1命令で1エントリをメインメモリへ書き込む。ソースとなるバッファエントリ、メインメモリのアドレスを指定する。この命令も後続の命令実行を妨げることは無い。

- Buffer check

  Buffer fetch命令発行後、バッファエントリにデータが到着したのかどうかを確認する命令であり、この命令によってデータの到着したものから演算するようソフトウェアから制御することができる。

専用命令の動作を図3に示す。

また、バッファのメモリ空間へのマッピングとメモリアクセス命令の動作を図4に示す。

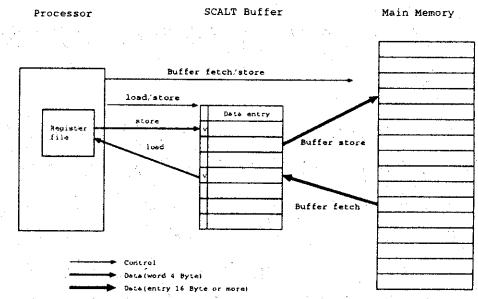

図3: 専用命令の動作 (Buffer fetch/store)

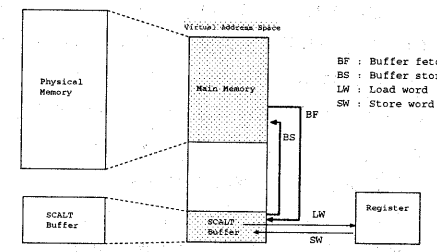

図4: メモリ空間と命令の動作

SCALTアーキテクチャにおけるプロセッサの構成については参考文献[1],[2],[3]に詳細が述べられている。

4 システムコントローラ

大規模な数値計算やランダムなメモリアクセスを行うプログラムに向けたコンピュータシステムの重要な要素としてシステムコントローラ(メモリシステム)が挙げられる。

ここではCPUからの同時発行数の増加に対応することができるシステムコントローラについて述べる。

4.1 概要

SCALT 用システムコントローラ (以下 SSC) はプロセッサ、外部 I/O などから発行されるメモリリクエストを処理することを主な目的としている。

構成図を図 5 に示す。

SSC はアトミックな動作に対応する細粒度のロック機構と DRAM へのアクセスの最適化がを行う。また現在のシステム構成ではキャッシュメモリもあり、これに対応するためのキャッシュパージ、順序依存のあるリクエストにも対応している。

SSC の各ブロックについては参考文献 [4] に説明がある。

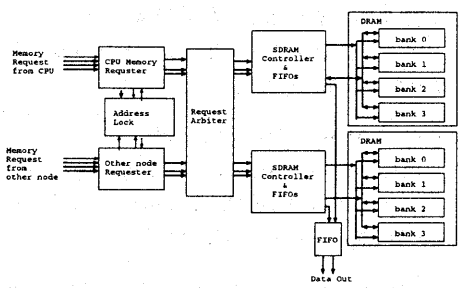

図 5: SSC 構成図

4.2 メモリアクセス最適化

SSC では DRAM へのアクセス最適化にバンク分散最適化を行っている。アクセス最適化を行うとき、各バンクへアクセスを分散することは重要であり、そのためのメモリアドレスの割り当てが重要となっている。

プロセッサからのメモリリクエストは 32bit で送られ、Read アクセスの場合、必ずエントリ長でのアクセスが行われる。

4.2.1 メモリへのアドレスの割り当て

メモリアドレスの対応についてはエントリ長、バス幅、バンク数、モジュール数などに依存するが、ここではエントリ長 16Byte、読みだしバス幅 32bit、バンク数 4、モジュール数 1 とした場合の SDRAM モジュールへのアドレスの割り当て例を図 6 示す。

図 6 において、Word Address とは SDRAM のデータ幅が 8 Byte 単位となる部分のアドレスである。このアドレスの割当の場合、1 エントリの読みだしのバースト転送長は 2 となる。図 1 から分かるようにバースト長が短くなるにつれ、バンクアクティブ、プリチャージのオーバーヘッドが増加する。

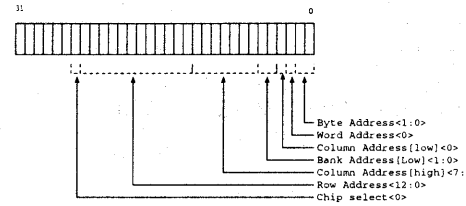

図 6: SDRAM への物理アドレスマッピング

4.2.2 アクセス最適化

プロセッサからの同時発行数が多いため、システムコントローラ内部に最適化キューを設けバースト長が短く、ランダムアクセスに耐性を持たせることで、ピークスループットに近づけることが可能である。

SSC では最適化は 2 段階で行われ、バンクコンフリクトを回避するようにリクエストを行う (図 7,8) SSC では 41 のリクエストを受け付けるが、そのうち 8 エントリに対して図 8 のようなアクセス分散最適化を行う。最適化キュー数の決定は図 2 と回路規模、動作周波数の制約から 8 としている。

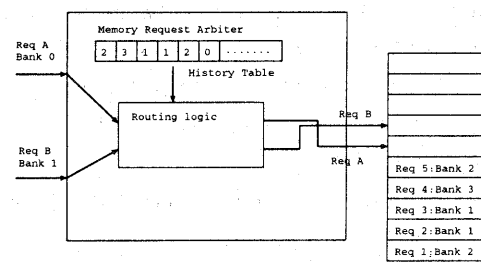

図 7: アクセス最適化 (第 1 段階)

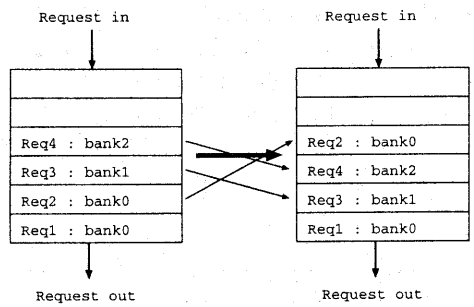

図 8: アクセス最適化 (第 2 段階)

4.3 最適化による順序依存関係

メモリアクセスをインオーダーに行わないときの問題点として RAW、WAW 等の順序依存関係の破壊が問題となる。これらの依存関係は同一アドレスへのメモリリクエストを最適化する場合の問題点となる。バンク分散最適化は同一バンクへのメモリリクエストに対してのみ順序関係の変更を行う。1 つのメモリアドレスは必ず同一のバンク、モジュールにアドレスが割り当てられ、またこれは時間によって変化することもない。したがって、これらの問題は原理的にあり得ない。そのためこれらの依存関係を監視する必要もなく、設計が簡単になる。

5 まとめ

現在の SCALT アーキテクチャコンピュータシステム設計 の内、プロセッサ、システムコントローラの HDL による RTL レベルでの設計は動作検証、性能評価段階となっている。

そしてコンパイラ (GCC)、アセンブラ (GAS) を SCALT アーキテクチャに移植する作業を並行しておこなっている。

2002 年度内にこれらを用いて Linux を載せることを目標としている。

SCALT アーキテクチャにおけるバッファはソフトウェアによる管理が必要となるため、コンパイラ技術、ライブラリなどに対する調査、研究が必要であり、これらについても今後検討していく必要がある。

参考文献

[1] 清水尚彦
スケーラブル レイテンシ トレラント アーキテクチャ, JSPJ Sig Notes Vol.97 No.21, 1997

[2] 三竹大輔、清水尚彦
変動するメモリレイテンシに対応するプロセッサ, 東海大学紀要 (工学部), Vol39, No.1, 1999

[3] Naohiko Shimizu, Kazuyuki Miyasaka, Hiroaki Haramiishi
Design of A Memory Latency Tolerant Processor(SCALT), MEDEA, 2001

[4] 孕石 裕昭, 宮坂 和幸, 清水 尚彦
レイテンシ耐性を有するコンピュータシステムの研究, SWoPP, 2001

[5] Sally A.McKee, Wm. A. Wulf
A Memory Controller for Improved Performance of Streamed Computations on Symmetric Multiprocessor, IPPS '96

[6] Bharat Chandramouli, John B. Carter, Wilson C. Hsieh, Sally A. McKee
A Cost Framework for Evaluating Integrated Restructuring Optimizations, IEEE International Conference on Parallel Archtectures and Compilation Techniques, 2001

[7] David Kroft
Looku-free Instruction Fetch/Prefetch Cache Organization, IEEE, 1981

[8] 天野 英晴
並列コンピュータ, 昭晃堂, 1996

[9] John L. Hennessy, David A. Patterson
Computer Architecture: A Quantitative Approach MORGAN KAUFMANN

[10] Cray Performance of the Cray T3E Multiprocessor,

[11] DEC, Alpha 21164 Microprocessor Data Sheet, 1994

[12] AMD, AMD-751 System Controller Data Sheet 2000