卒業論文

題目

階層タイル形式格納キャッシュメモリのシミュレータによる性能評価

指導教員

近藤利夫

2016 年

三重大学 工学部 情報工学科
コンピュータアーキテクチャ研究室

小池　竜馬 (409826)

内容梗概

科学技術計算や画像処理の高速化には，高効率の 2 次元データ処理が必要不可欠である．しかし，ラスター走査順に割り当てられた 2 次元配列データに対し，ライン単位でアクセスする既存のキャッシュメモリでは，2 次元の空間的な局所性を活かせないことから，アクセス効率の低下を来し，それが高速化のボトルネックとなっている．これに対し 2 次元の局所性を最大限に活かすことを目指した格納方式の Z 順配置 [1][2] が提案されていた．しかし，この方式では列方向のアクセスは効率化されるものの，行方向のアクセス効率が逆に低下してしまう問題がある．そこで，当研究室では Z 順配置ベースの 3 レベル階層タイル形式にスキュード配列形式を組み合わせることで，従来のライン単位アクセスとタイル単位アクセスを両立するライン・タイル両アクセス対応の新構成キャッシュメモリを提案している．本研究ではこのキャッシュメモリのシミュレータの実装，評価を行った．シミュレータは SimpleScalar[3] を改造することで実装した．ただし，卒研の範囲内で改造が終えられるよう，実装機能は 3 レベルの階層タイル形式に限定した．最上位のタイルの形状として縦長の長方形と正方形の 2 種類試した結果，正方形の形状でヒット率を行列乗算で 2%，ボックスフィルタで 0.6%，それぞれ改善する理想的な Z 順配置と同等のアクセス性能が得られることを明らかにできた．

Abstract

High-speed scientific computing and image processing need efficient two-dimensional data processing. However, a conventional cache memory based on Raster order unit line access cannot utilize 2-D spatial locality so that, it has poor 2-D access efficiency and causes bottleneck to accelerate two-dimensional data processing. In order to solve this problem, Z-Morton order layout that can maximize the utilization of 2-D spatially locality had been proposed in recent years. Though this method can improve the data access speed in the vertical direction the performance of data access in the horizontal direction decreased. In this paper, we proposed a new cache memory with both unit line and unit tile accessibility based on 3 level Z-order tiling data layout and multi-bank memory structure supporting skewed array store scheme. We developed a cache simulator based on Simplescalar and evaluated the performance of proposed cache in this study. We only implemented the 3 level Z-order tiling data layout into the cache simulator. Simulation result shows that the proposed 3 level Z-order tiling data layout provide less cache miss ratio compared with raster scan order at matrix multiplication (2%), convolution filtering (0.6%) and it also has similar performance to the Z-Morton order.

目 次

図 目 次

表 目 次

1　まえがき

1.1　背景

近年，コンピュータの性能が著しく向上しているにも関わらず，科学技術計算や画像処理に代表される 2 次元データ処理の高速化の要求が高まっている．しかし，2 次元データ処理でのキャッシュメモリへのアクセスは，既存のライン単位でのアクセスするキャッシュメモリでは，ブロック単位のアクセスがボトルネックとなり非効率である．さらに，ブロック単位のアクセスアドレスが不規則になる場合には，アドレス計算の複雑化や，ブロック単位データがキャッシュラインに収まりきらないために並列アクセスが不能になり，キャッシュブロッキングがうまく機能しなくなってしまう．これらに対処すべくブロック単位アクセスに特化した Z-order と呼ばれる格納方式が提案されているが，列方向への高効率なアクセスとの両立はできていない．そのため，2 次元的な局所性を活かし，行・列両方向に効率的にアクセス可能な新構成キャッシュメモリの開発が求められている．

1.2　研究目的

当研究室では，行・列方向へ効率的にアクセスすることを目指したライン・タイル両アクセス対応の新構成キャッシュメモリが提案されている．しかし，実装の複雑さから性能評価を行えていない．本研究では，既存のシミュレータを改造することで，新構成キャッシュメモリの性能を評価し，新構成キャッ

シュメモリの問題点を明らかにすると共に，その原因を明らかにする．そして，その明らかにした原因を基に改善法を示す．また，新たに 2 種類のアドレス変換によるアドレス割り当てを提案・比較評価し，提案キャッシュメモリに最適なアドレス割り当てを明らかにする．

2　先行研究

2.3 節に挙げる研究は 2.1 節の手法や 2.2 節の格納方式を基に当研究室において行われているものである．

2.1　キャッシュブロッキング

キャッシュブロッキング法 (cache blocking) は，多くのアプリケーションにおいてメモリ帯域幅のボトルネックを回避できる一般的な最適化手法である [4]．具体例として，大きな行列サイズの行列積をあげる．大きな行列積計算を小さなブロックサイズの行列積に分割することで，キャッシュに収まるデータの再利用性が高まり，データの参照局所性が向上し，メモリー帯域幅への負担を減らすことができる．を図 2.1(a) に通常の行列積の処理，2.1(b) に簡略したキャッシュブロッキング法に基づく行列積の処理法を示す．2.1(a) では N が大きくなると，処理するブロックサイズがキャッシュメモリサイズの限界を超えるため，配列 Z，Y，X についてデータ局 6 所性が無くなり，キャッシュ容量性ミスと競合性ミスが頻発し，データアクセス速度の低下が起きる．ブロッキング法を利用する 2.1(b) では，行列積がブロック幅 B の領域 ($B \times B$) ごとになされており，配列 Z，Y，X のデータアクセスがブロックで局所化されている．この $B \times B$ のブロック単位をキャッシュに収めることができれば，データアクセスの速度向上が期待できる．

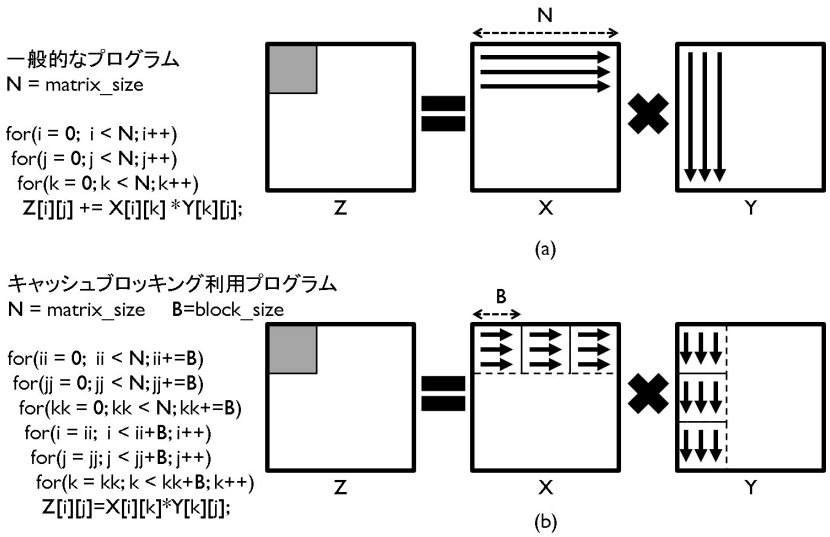

図 2.1: (a) ブロック無しの行列積 (b) ブロッキングされる行列積

2.2 Z-order

Z-order とは，2 次元の局所性を最大限に活かすことを目指した格納方式である．図 2.2 にラスター順のアドレス付け，図 2.3 に Z-order のアドレス付けを示す．通常のキャッシュメモリは図 2.2 に示すように，ラスター順のアドレス付けでデータを格納するのに対し，Z-order では $N \times N$ の正方形にデータを格納する．図 2.3 に示すように，メモリアドレスをタイル形式でデータアクセスできるように変換することで，通常のキャッシュメモリに比べて列方向のアクセスを効率的に行える．しかし，従来キャッシュメモリと比較して行方向アクセスが非効率になる問題を抱えている．この問題の改善のために，任意の 2 のべき乗サイズの 2 次元配列のラスター順アドレスを Z-order のアドレス付けに変換するアドレス変換法が提案されており，これまでのラスター順のアドレス付けをそのまま変換できるものの，ハードウェア規模が

大幅に増える上に，ロードレイテンシ増大の要因となる欠点があった．

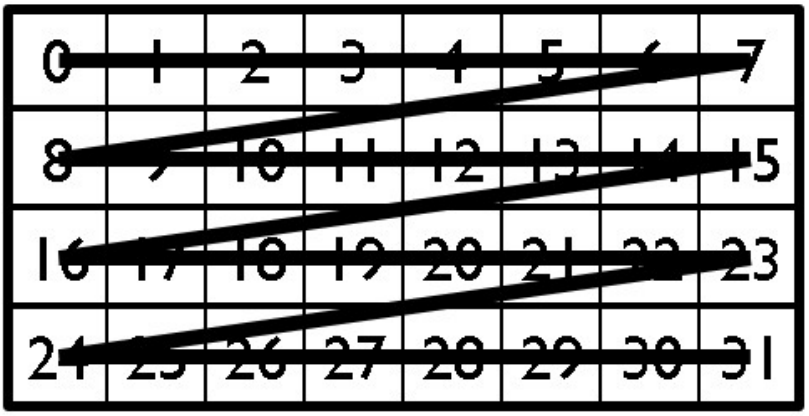

図 2.2: ラスター順のアドレス付け

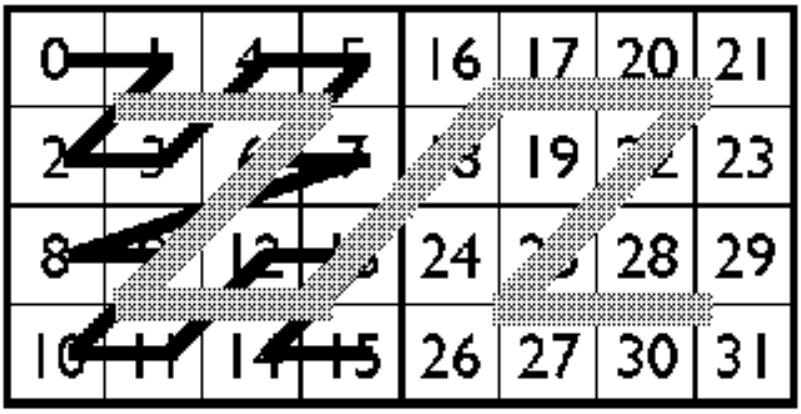

図 2.3: Z-order のアドレス付け

2.3 ライン・タイル両アクセス対応キャッシュメモリ

行・列方向の高アクセス効率を実現するため，当研究室では新構成のキャッシュメモリを提案している [5]．図 2.4 に簡単なハードウェア構成を示す．図 2.4 中のアドレス変換器により，プロセッサから出されたラスター順のメモリアドレスをタイル形式のアドレスへ変換し，非整列アクセスする場合アドレ

ス変換器からはアドレスタグ・アドレスインデックスを並列に生成する．このとき変換されるアドレス割り当てを 2.5 に示す．このアドレス割り当ては，図 2.5 に示すように Z-order を改良した 3 レベルの階層タイル形式格納を採用しており，最上位階層のタイルを横幅を 64KB に固定している．最下層のタイルをスモールタイルと呼び，その 1 つ上の階層のタイルをラージタイルと呼ぶ．スモールタイルは通常のキャッシュラインの 32Byte と同サイズに，ラージタイルはキャッシュメモリと同サイズの 4KB に設定している．アクセス順としては各タイル内をラスター順にアクセスする．また，図 2.4 に示す通りキャッシュメモリのタグメモリ・データメモリを 4 バンクに分け，キャッシュメモリへの格納方法を工夫することよりライン・タイル両アクセスを可能にしている．図 2.6 にキャッシュメモリへのデータの格納方法を示す．図 2.6 に示すとおりメインメモリにデータをタイル形式に格納し，キャッシュメモリにロードする際スモールタイル内のデータを各バンク内にずらして格納する．各バンクからデータをロードする際に 1 サイクルしかかからないのでラインデータもタイルデータも 1 サイクルでのロードが可能になる．タグデータに関しても各バンク内のタグデータを入れ替えることで列方向への非整列アクセスを可能にしている．アドレス変換可能なラスター順領域の横幅を 64KB に固定することで，アドレス変換のハードウェア規模低減とロードレイテンシの増加を抑えている．しかし，実装上の複雑さから性能評価が行えていない問題がある．

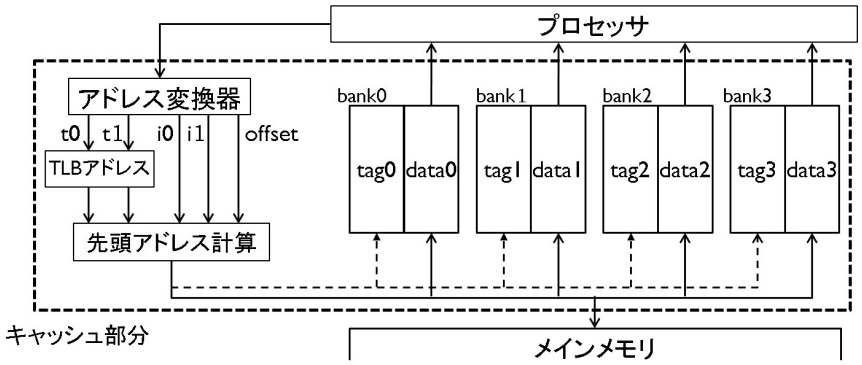

図 2.4: 新構成キャッシュメモリの構成

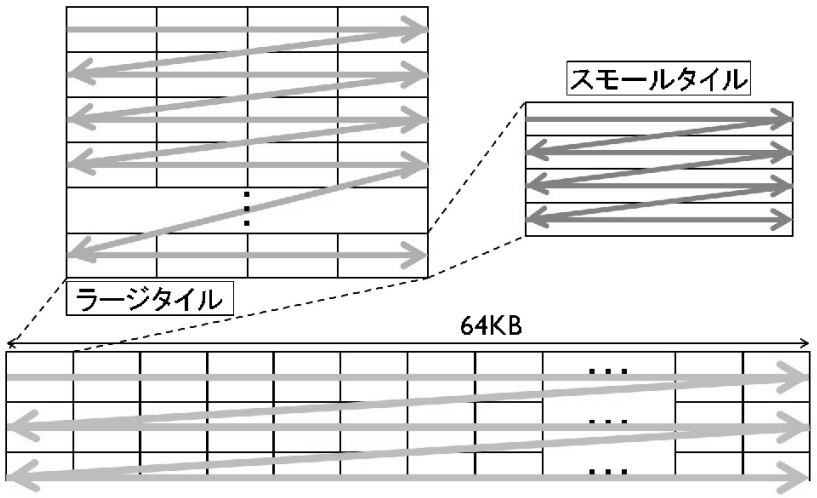

図 2.5: アドレス割り当て

3　シミュレータの実装上の問題点とその解決法

シミュレータを作製する際 SimpleScalar をベースに実装した．前述の Z-morton を用いたキャッシュメモリの性能評価を行う上で，SimpleScalar が採用されていたため先行研究に則り本研究においても採用した．この章ではシミュレータの基にした SimpleScalar，及び実装に際して生じた問題点とその

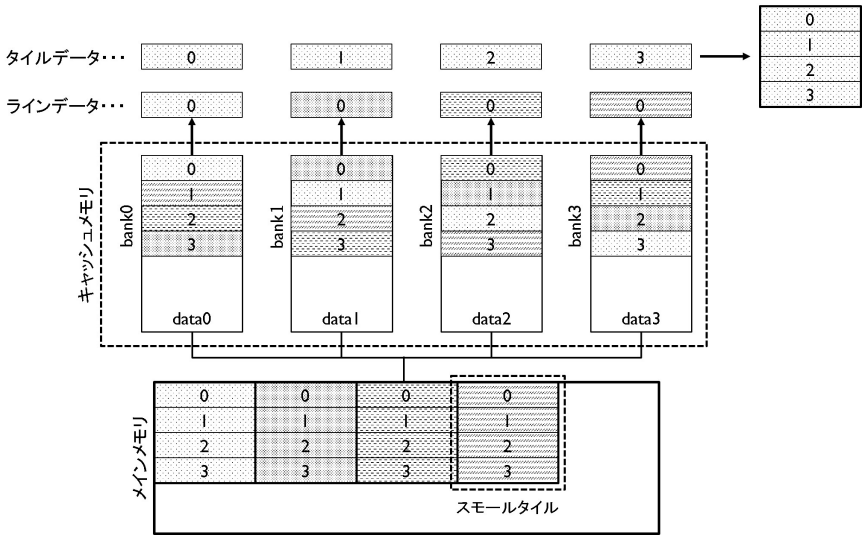

図 2.6: データのキャッシュメモリへの格納方法

解決法について述べ，実装内容の詳細は 4.1 節で述べることとする．

3.1 SimpleScalar

SimpleScalar とはオープンソースで提供されいているマイクロプロセッサの命令セットアーキテクチャをシミュレートするプログラムであり，キャッシュメモリやプロセッサなどの各機能の性能を評価することも可能である [3].

3.2 ライン・タイル両アクセス対応キャッシュメモリの実装

本研究に関連する SimpleScalar の構成を図 3.7(a)，改造後の構成を図 3.7(b) に示す．構成の中で主に改造する箇所としては L1 キャッシュとメインメモリの機能であり，通常アドレスを 3 レベル階層タイル形式のアドレスへ変換するためにプロセッサと TLB・L1 キャッシュの間にアドレス変換器の機能を

追加する．機能として SimpleScalar にライン・タイル両アクセス対応キャッシュメモリを実装する上で変更・追加が必要な内容は以下の通りである．

- アドレス変換機構の追加
- ヒットミス判定機構の改造
- タグメモリ・データメモリのバンク化
- SimpleScalar・新構成キャッシュメモリ・Z-order のモード切り替え機能の追加

モード切替機能は効率的な評価のために必要となるもので新構成キャッシュメモリの性能に関わりのないものである．

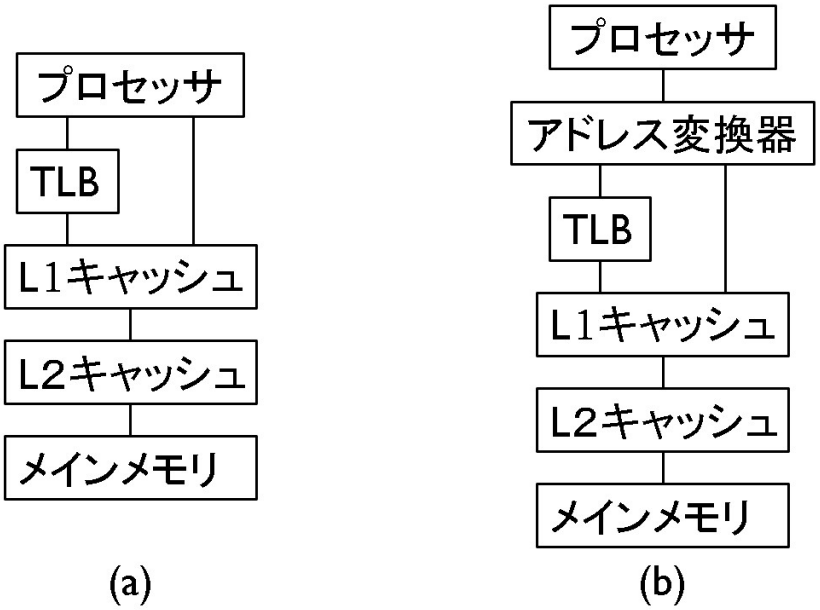

図 3.7: (a)SimpleScalar の構成　(b) 改造後の構成

3.3 実装上の問題点と解決法

前述の内容を実装する際に問題点が 4 点発覚した．この節では発覚した問題点とその解決法，または妥協点を述べることとする．

- 新構成キャッシュメモリのライン・タイルアクセスの切り替え方法が確立されていない点．並列アクセスは諦め，3 レベル Z-order 格納の効果検証に絞り，ラスター順の従来の格納法と理想的な Z-order 格納法とを比較し評価する．これにより，リプレイスを 8×4 サイズのタイル単位で行うことで非整列アクセスによる使用しないサブラインをアクセスするために発生する余分なリプレイスが含まれないため新構成キャッシュメモリに比べミス率は低めになるという差異が発生する．

- 評価項目としてキャッシュコンフリクトミス率のみではなく，キャッシュメモリに対するアクセスサイクル数を加えると改造にかかる時間が大幅に増加してしまい評価・考察に対する時間確保が難しくなる点．本研究において重視されているのは，キャッシュコンフリクトミス率の比較なのでアクセスサイクル数の評価は諦める．

- SimpleScalar のデータアクセスが 4Byte 単位で行われるため非整列アクセスが発生しない点．タイル格納の評価に絞るためこの項目も本研究では見送る．

- タイル形式に格納する機能，もしくは命令の追加が困難である点．シ

ミュレータの機能としての追加は諦め，評価用のプログラムを工夫する

ことによって実現，解決する．解決法については 4.2 節にて述べる．

4　実装・評価方法

本研究においてのシミュレータの具体的な実装内容を 4.1 節で述べる．3.3 節で述べたタイル形式格納においての問題点の解決法を 4.2 節で述べる．

4.1　シミュレータの実装

実装しなければならない機能については 3.2 節で述べたが，3.3 節で述べたとおり変更点がいくつか出たため，実際に実装したものについてこの節では述べることとする．まず，タイルアクセスするためのアドレス変換機構を追加した．本研究においてはアドレス変換を 2 種類使用し，そのミス率の違いを検証する．アドレス変換の変換パターンを図 4.8,4.9 に示す．2 種類のアドレス変換の違いはラージタイル (タグ領域) の形である．図 4.10(a) はアドレス変換 1 によりラージタイルを縦長の長方形，図 4.10(b) はアドレス変換 2 によりラージタイルを正方形に変換したものを示している．このアドレス変換器はプロセッサから出たアドレスを L1 キャッシュとメインメモリへ送る前に通すよう実装した．次にキャッシュメモリ内のタグメモリとデータメモリをそれぞれ 4 バンク構成とし，各バンクでタグデータが管理できるようにミス・ヒット判定の機構も改造した．さらに，一つのシミュレータで複数のアドレスパターンをシミュレートできるように，アドレス変換 1・アドレス変換 2・SimpleScalar・Z-order へのモードを切替えられる機構も追加した．

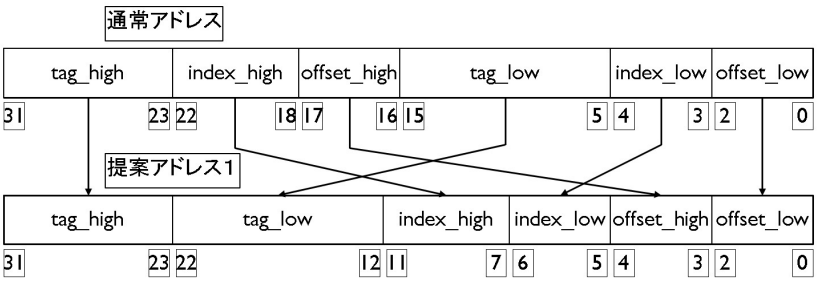

図 4.8: アドレス変換 1

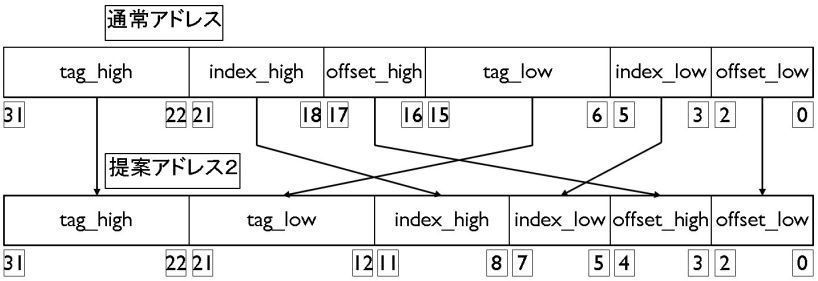

図 4.9: アドレス変換 2

4.2 評価用プログラム

3.3 節で述べたタイル形式格納については評価用プログラム内で対応する．図 4.11 に 2 次元配列変数の定義方法を示す．図 4.11 に示す 2 次元配列 A のように malloc 関数を用いてキャッシュメモリの横幅 64KB に合わせた領域を確保し，確保した領域内のアドレスを 2 次元配列 B，C の各行の先頭アドレスが 64K ずつずれるように与えることで，擬似的にタイル形式に格納する．このように格納した配列を用いた評価用プログラムでタイルアクセスキャッシュメモリの評価を行う．ただし，ラスター順格納したものとタイル格納したものの比較をするために，改造を施していない SimpleScalar を用いて実行する際は通常格納の 2 次元配列を用いた．なお性能評価は以下の 2 つのプロ

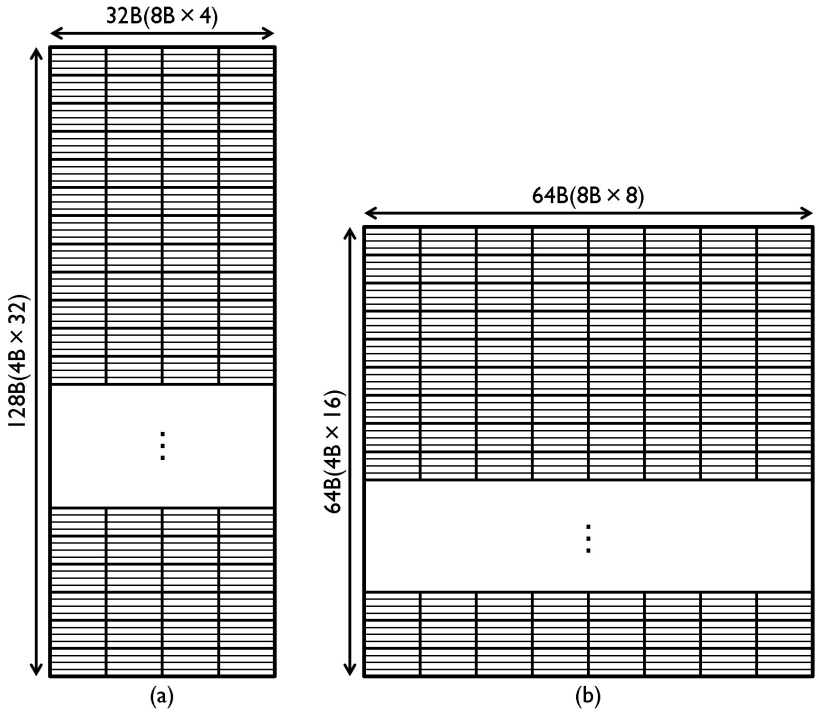

図 4.10: (a) アドレス変換 1 のラージタイル　(b) アドレス変換 2 のラージタイル

グラムで実行した.

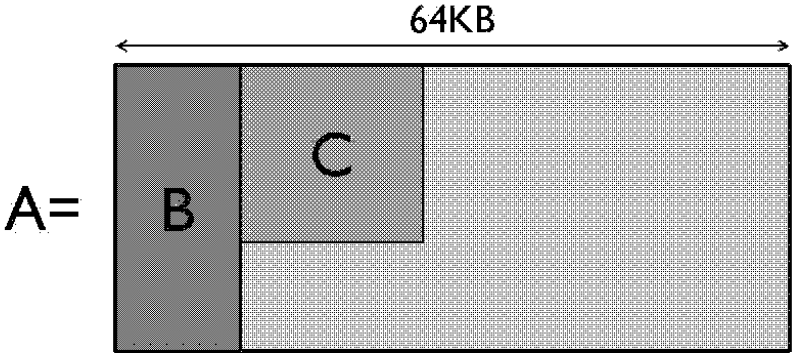

図 4.11: 2 次元配列の領域確保

4.2.1 行列積

一般的な行列積を計算するプログラムであり，キャッシュブロッキング手法を用いるか否かが切り替えられ，配列サイズ・ブロックサイズともに行・列両サイズ指定可能である．転置を行わずに行方向・列方向に連続にアクセスするのが特徴である．使用した 2 次元配列のサイズ表 4.1，ブロッキング手法使用時のブロックサイズを表 4.2 に示す．

X×Y		
128×128	256×256	512×512
1024×128	128×1024	1024×1024
2048×128	128×2048	2048×2048
5120×128	128×5120	-

表 4.1: 行列積の配列サイズ

X×Y				
0×0	2×2	4×4	8×8	2×4
4×2	4×8	8×4	8×16	16×8

表 4.2: 行列積のブロックサイズ

4.2.2 畳み込み演算

画像処理などで使用されているフィルタリングの一種で，フィルタリングする注目画素の近傍画素の平均値を求める畳み込み演算である．こちらもキャッシュブロッキング手法を用いるか否かを指定可能．使用する配列のサイズは近年動画像で主流である 16:9 のものを使用し，ブロックサイズ，フィルターサイズを組み合わせる．それぞれのサイズを表 4.3，表 4.4，表 4.5 に示す．

X×Y			
512×288	1024×576	1280×720	2560×1440

表 4.3: ボックスフィルタの配列サイズ

X×Y		
0×0	4×2	8×4

表 4.4: ボックスフィルタのブロックサイズ

5　性能評価

5.1　評価

評価をするにあたり提案アドレス 1，提案アドレス 2，Z-order，SimpleScalar の 4 つを比較した．提案アドレス 1，提案アドレス 2 はそれぞれ 4.1 節で示したアドレス変換 1，アドレス変換 2 を指し，Z-order はタイル形式格納の代表として，SimpleScalar はラスター順格納の代表として用いた．評価結果を以下の図に示す．図 5.12，図 5.13 は行列積のプログラムを実行したときのミス率を示しており，図 5.12 は配列サイズ毎に，図 5.13 はブロックサイズ毎に，それぞれ平均をとった． 図 5.14，図 5.15，図 5.16 はボックスフィルタのプログラムを実行したときのミス率の値を示しており，図 5.14 は配列サイズ毎に，図 5.15 はブロックサイズ毎に，図 5.16 はフィルタサイズ毎に，それぞれ平均をとった．

注目画素からの距離			
5	7	9	17

表 4.5: ボックスフィルタのフィルタサイズ

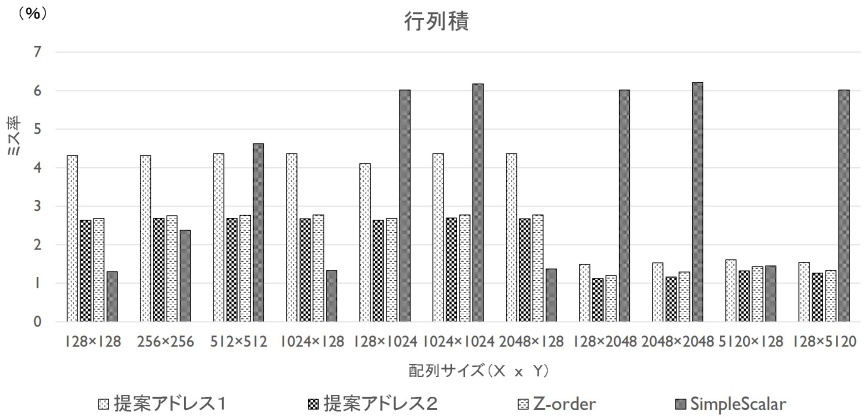

図 5.12: 行列積 (配列サイズ平均)

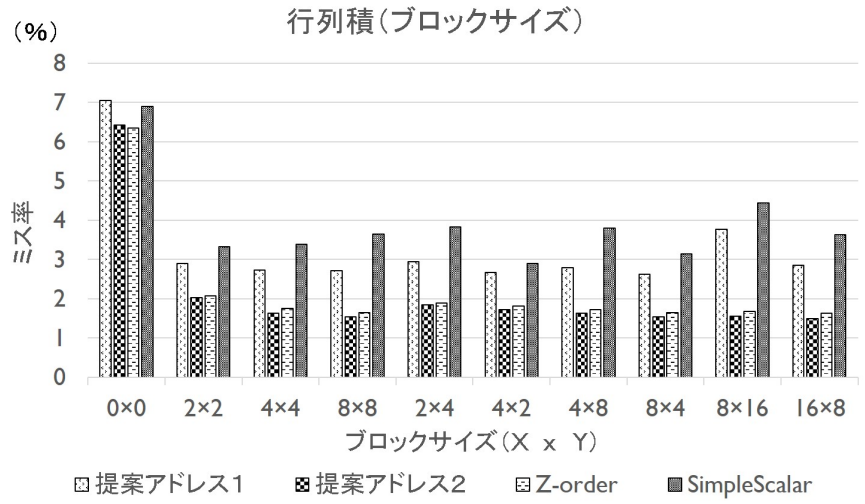

図 5.13: 行列積 (ブロックサイズ平均)

5.2 考察

提案アドレス 2 は Z-order と同程度のミス率に抑えることができた．スモールタイルの形状の違いにより多少の差が出てはいるが，やはりタグ領域が同

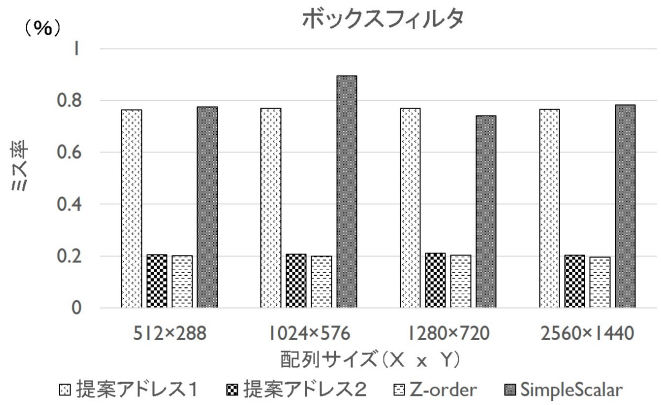

図 5.14: ボックスフィルタ (配列サイズ平均)

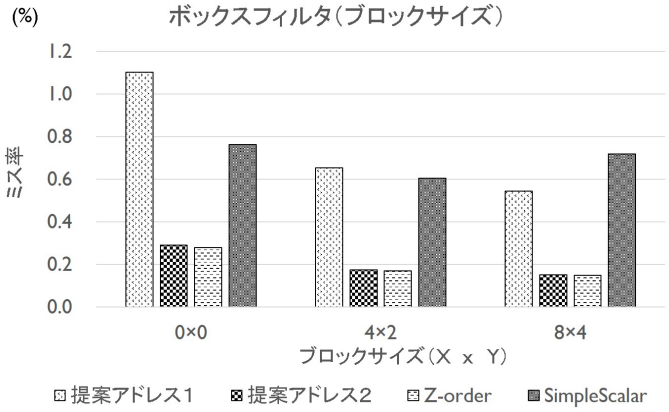

図 5.15: ボックスフィルタ (ブロックサイズ平均)

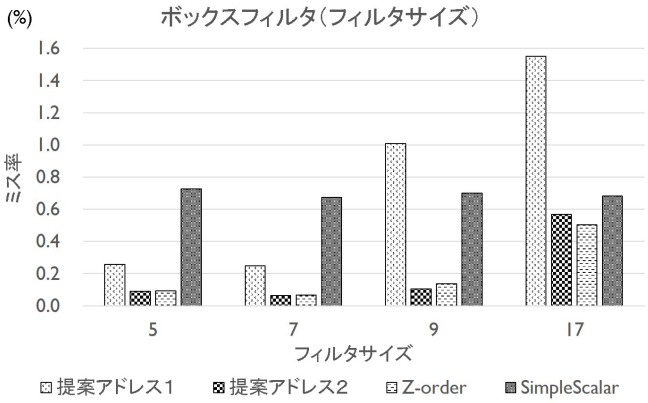

図 5.16: ボックスフィルタ (フィルタサイズ平均)

じ形状のため大きく差が出ることはなかった．しかし，列方向に大きいタグ領域を持っている提案アドレス 1 では，行方向のアクセスに対応できずミス率が増加してしまった．行列積の結果に注目すると配列サイズが大きくなるか，列方向へのサイズが大きいものに関して提案アドレス 1，2 ともにミス率が低くなっているのがわかる．ブロックサイズに注目すると全てにおいて提案アドレス 2 は SimpleScalar よりミス率が低いことがわかる．このことから通常格納のものより提案アドレスのものの方がキャッシュブロッキング手法が有効だということがわかる．ボックスフィルタの結果に注目すると全てにおいて提案アドレス 2 及び Z-order が SimpleScalar よりミス率が低いことがわかるが，大きな差は見られない．この原因としてボックスフィルタはフィルタサイズ内をラスター順にアクセスするためタイル形式格納のアドレスではうまく対応できないこと，さらにラスター順アドレスではフィルタサイズ内の列方向へのアクセスにうまく対応できないことが挙げられる．つまり，フィルタサイズ内の一度のアクセス内にタイル形式格納，ラスター順格納に不得手なアクセスが含まれるためそれぞれに大きな差がなかったのだと考えられる．しかし，その不得手なアクセスも頻発しているわけではないので全体としてミス率が上昇にはつながらなかった．

6　あとがき

本研究はタイルアクセスキャッシュメモリのシミュレータを実装し、評価した．シミュレータの実装は、新構成キャッシュメモリの問題点，具体的にはタイル格納とライン・タイル間のアクセス切り替えへの対応が卒研の範囲内では困難であることから，タイル形式 格納の有用性の確認に的を絞た構成で行った．その結果，新構成キャッシュメモリのタイルアクセスにより理想的な Z 順アクセスと同等の性能が得られることを明らかにできた．具体的には，タグ領域のタイルの形状として縦長の長方形と正方形の 2 種類試し，正方形の形状でヒット率を行列乗算で 2%，ボックスフィルタで 0.6%，それぞれ改善する結果となった．ただし，2 次元データの格納方法としてはキャッシュの機能ではなく評価用プログラムを工夫し実現したため，今後シミュレータへの機能実装が必要である． また，今回使用したプログラムは至って単純なプログラムのみになってしまったため，様々なアクセスパターンを持つプログラムについても検証を進めて行く必要がある．

謝辞

本研究を進めるに当たり，終始熱心な御指導，御助言を頂きました近藤利夫教授，佐々木敬泰助教，深澤祐樹さんに感謝いたします．また，研究に協力して頂いた王宝康さん，水野篤さん，ならびにコンピュータアーキテクチャ研究室の皆さんに感謝します．

参考文献

[1] Chatterjee S. , Lebeck A.R., Patnala P.K., Thottehodi M. “Recursive Array Layouts and Fast Parallel Martix Multiplication.” IEEE Transactions on Parallel and Distributed Systems(IEEE TPDS) 13(11), 1105-1123(2002).

[2] Thiyagalingam J., “ Alternative Array Storage Layouts for Regular Scientific Programs.” PhD thesis, Department of Computing, Imperial College, London, U.K.(2005).

[3] SimpleScalar LLC,http://www.simplescalar.com/terms.html, [2016 年 3 月 4 日アクセス].

[4] Lam Monica D., Edward E. Rothberg, Michael E. Wolf. “The cache performance and optimizations of blocked algorithms.” ACM-SIGARCH Computer Architecture News. Vol. 19. No. 2. ACM, 1991.

[5] 王宝康，“タイル・ライン両アクセス機能を備えた新構成キャッシュメモリの提案”，三重大学大学院修士論文，2015 年 3 月.

A　プログラムリスト

B　評価用データ