卒業論文

題目

タイル・ライン変換機能を備える新構成キャッシュメモリの設計

指導教員

近藤利夫 教授

2012年

三重大学 工学部 情報工学科
計算機アーキテクチャ研究室

立野　篤 (409834)

内容梗概

動画像符号化処理における動き探索に代表されるように，画像処理のデータアクセスは2次元の局所性が高く，2次元の近傍アクセスが頻発する．また，DCT 等ではその演算の性質から列方向に連続したデータアクセスが必要になる．しかし，従来のキャッシュメモリ構成ではそれらの効率的なアクセスは困難である．

本研究室では，タイル形式によるアドレス割り当てを行うことで，2次元的なデータアクセスを効率的に行う手法が提案されてきた．しかし，通常の命令はラスタ形式を前提としており，タイル形式に対応していないという問題点がある．

そこで，本研究ではタイル形式に対応し，ロード時にタイル単位の 90 度回転を可能とする新構成のキャッシュメモリを提案，設計した．その結果，従来のキャッシュメモリ構成と比較して低容量・高性能を実現できることを示した．

Abstract

The data access in image processing such as motion estimation for video encoding has high two-dimensional locality. Thus two-dimensional access locality also frequently occurs in the signal processing. For example, DCT, FFT and so on need continuous data access in the column direction from the nature of operations. However, such accesses are inefficient to a conventional cache memory.

In this laboratory, we have proposed an efficient method of the two-dimensional data access by address assignment of the tile form. Though normal instructions are suitable for raster format, they do not correspond to the tile format.

Therefore, in this study, I have proposed and designed a new cache memory that corresponds to the tile format and enables 90-degree rotation of each tile at load time. As a result, I have shown that the new cache memory can perform low capacity and high performance compared to the conventional cache memory.

目 次

図 目 次

表 目 次

1　はじめに

画像処理におけるデータアクセスは 2 次元の局所性が高く，2 次元の近傍アクセスが頻発する．動画像符号化処理の大半を占める動き探索はその一例である．また，DCT や FFT 等では，その演算の性質から行方向と同じ割合で列方向に連続したデータアクセスが必要になる．しかし，従来のキャッシュメモリ構成では 2 次元近傍アクセスや列方向の効率的なアクセスは困難で，それによるオーバーヘッドがボトルネックとなり，演算の高速化の障害となっている．

そこで，本研究では 2 次元の近傍アクセスや列方向アクセスを効率的に行うことで，メモリアクセスに要する時間を短縮し，高速化を図ることが目的となる．標準的なキャッシュメモリ構成に大きな変更を加えることなく，データ転送を効率化する画像処理に適したキャッシュメモリの実現を目指す．

2　動き探索と2次元DCT

2.1　動き探索

動画像符号化処理では，圧縮率を高めるために動き探索が行われる．動き探索は，マクロブロック単位でブロックマッチングを行い，参照画像の中で符号化対象ブロックと最も似ている箇所を探し出し，動きベクトルを算出する．この算出した動きベクトルを用いて，動いた差分のみを圧縮することにより，冗長なデータを減らすことが出来，画素値をそのまま符号化するのに比べて大幅な圧縮率の向上が見込める．

動き探索では，1画素の動きベクトルを算出するために，その1画素に対してブロックマッチングを繰り返す．さらに，符号化対象画像全体に対して行われるため，動き探索の演算量は動画像符号化処理の大半を占めており，演算量の低減化が求められている．

2.2　2次元DCT

動画像符号化処理では圧縮率を高めるために，動き探索の他に2次元離散コサイン変換 (DCT) が行われる．DCTとは画素を別の情報である周波数に変換するものである．一般に，人間の眼は低周波成分に対して敏感であり，高周波成分に対しては鈍感であるという特性を持っている．

その特性を利用し，高周波成分を削除してしまうことで，画質の劣化を感じさせることなく圧縮率の向上が行える．

しかし，2次元DCTの処理には多くの積和演算が必要となる．その効率的な処理方法として，2次元DCTでは，まず行方向に対して1次元DCTを行い，得られた結果に対して今度は列方向に1次元DCTを行っている．このようにすることで，それぞれのデータに対して直接2次元DCTを行うよりも効率的なものとなっている．

3　従来キャッシュ構成の問題点

3.1　2次元データ処理

前述したように動画像符号化の動き探索においては，2次元の近傍アクセスが頻発する．これは，動きベクトルを算出する際，ブロック単位での比較を繰り返すためである．

通常，画像データはラスタ走査順でメモリに格納されているため，行方向以外では必要な部分のみのアクセスができず，無駄な領域のデータ転送が発生する．データへはキャッシュライン単位でアクセスするため，必要な部分のみへのアクセスが行えない．図 3.1 に示したように，要求データが 8x8 のブロックでキャッシュラインが 32byte の場合，8 回のアクセスが必要となり，28x8 画素分の無駄な転送が発生する．また，行方向でもキャッシュラインを跨いだ場合は 16 回のアクセスが必要となり，ロードするが使用しないデータの領域は更に増加することになる．そして，ラスタ形式の場合，8x8 のブロックに必要なアクセス回数は平均 9.75 回となる．このように，従来のキャッシュ構成ではメモリへのアクセス回数が増加し，処理時間増加の大きな要因となっている．

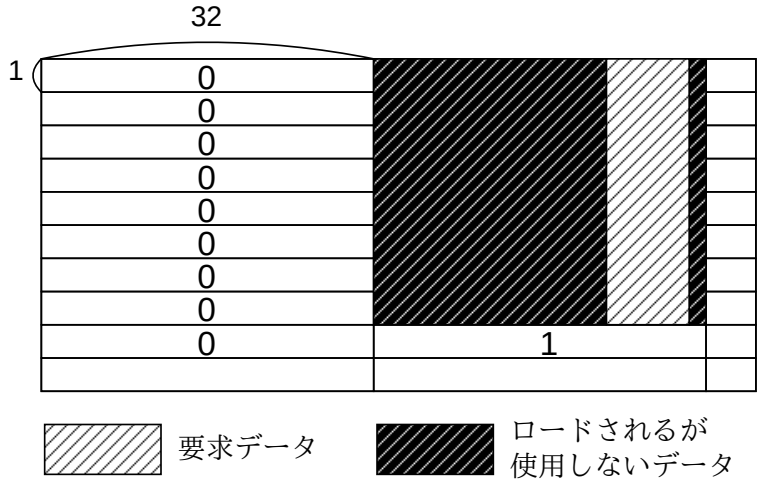

図 3.1: ラスタ形式の非効率性

3.2 列方向アクセス

動画像符号化処理の 2 次元 DCT では，列方向への DCT 処理のために転置を行う．その際に列方向へのアクセスが必要になる．データは基本的に行方向に連続して格納されているため，行方向へのアクセスは効率的に行えるが，列方向に対してはデータに飛び飛びでアクセスしなければならず，行方向よりも多くのキャッシュラインにアクセスすることになる．例えば 4x4 の行列を転置する場合，最低でも 4 回のアクセスが必要となり，従来キャッシュでは効率的なアクセスが行えないという問題がある．

4　タイル形式による解決とその要求条件

4.1　タイル形式

通常の画像データのアドレス割り当ては行方向に連続したものである．そのため，2 次元的なデータアクセスには非効率的であった．

そこで，タイル形式で格納することにより，1 つのキャッシュラインへのアクセスで 2 次元的なデータアクセスを可能とすることで，ラスタ形式に比べて 2 次元の近傍アクセスに適した割り当てとなる [1]．8x4 のタイル形式でのメモリ空間へのアドレス割り当ての例を図 4.2 に示す．

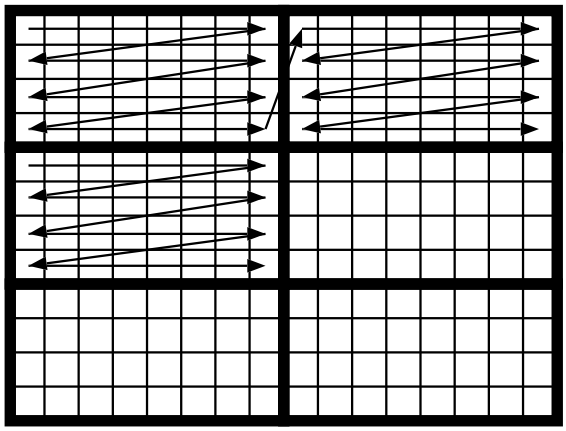

図 4.2: タイル形式のアドレス割り当て

8x4 のタイル形式で格納すると，8x8 のブロックに必要なアクセス回数は，2 回，3 回，4 回，6 回の場合が存在し，平均アクセス回数は 5.16 回となる．図 4.3 に示したように，図 3.1 と比較して無駄な領域の転送も少

ない．このように，タイル形式での格納により，2 次元的なデータアクセスを可能とし，ラスタ形式と比較して効率的なアクセスが行えるようになる．

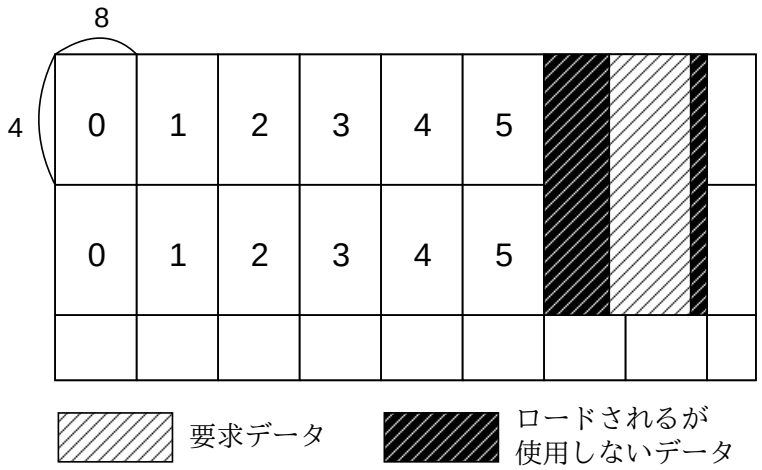

図 4.3: タイル形式による効率化

4.2 要求条件

しかし，通常の命令はラスタ形式で格納されていることを前提としており，タイル形式での格納を想定していない．そのため，目的のデータのアドレスを送っても，タイル形式で格納されているために目的のデータとは違うデータを参照してしまう．また，データは最初はラスタ形式で格納されているため，タイル形式でのアクセスを行うには，ラスタ形式からタイル形式に変換する必要がある．このため，タイル形式データ

の処理には，ラスタ形式からタイル形式への変換機能と，タイル形式対応のアドレス生成機能が必要になる．

5　提案キャッシュの設計

5.1　概要

前述のとおり，従来のキャッシュ構成では，2 次元の近傍アクセスや列方向アクセスを効率的に行うことは困難である．そこで，本研究では 4 章の要求に応え得るライン・タイル変換機能を備えるキャッシュに，レジスタロード時にタイル単位の 90 度回転を可能とするパーミュテーションネットワーク [2] を付加する新構成のキャッシュメモリを設計する．

5.2　構成

図 5.4 に示されるように提案キャッシュは，タイル形式に対応するためのアドレス変換部，データを保存するためのデータ部，スキューアレイ形式で格納するためのバレルシフタと整列化制約解除及びタイル単位の 90 度回転をサポートするパーミュテーションネットワークで構成される．今回，データ部はダイレクトマップで動作するものとした．

5.3　動作内容

演算器から送られてきたアドレスを変換し，それぞれのバンクに変換したアドレスを送り，ミス/ヒットを判定する．バンク内にデータが揃っていれば，それぞれのバンクからロードしてきたデータを演算器とバン

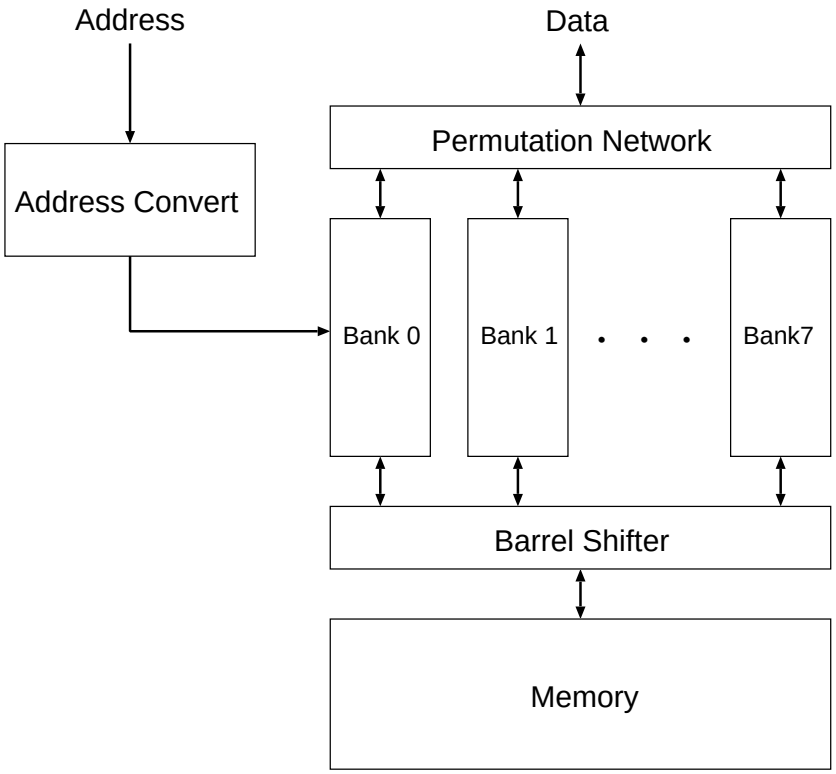

図 5.4: 提案キャッシュの構成

ク間のパーミュテーションネットワークで並び替え，結合，シフトすることで整列化制約を解除する．目的のデータが存在しないバンクがあった場合，下位の記憶階層とのリプレースを行う．下位の記憶階層からデータを読み出す際にバンクとメモリ間のバレルシフタでデータの並びを入れ替え，スキュードアレイ形式に対応した形で格納する．

提案キャッシュはキャッシュライン単位の転送とタイル単位の転送を行うことができ，タイル形式のアドレス割り当てに対応しているため，2次

元の近傍アクセスを効率的に行うことができる．

5.4 整列化制約解除機構

キャッシュラインの制約上，目的のデータを演算器側のレジスタの所定の位置に入れるために，余分な処理が発生することがある．そこで，先行研究としてキャッシュを複数バンクに分けることで，並列にアクセスすることを可能とし，目的のデータを取り出せるようにすることで，キャッシュラインの制約にとらわれないデータ転送を可能とするキャッシュが研究されている [3]．今回の新構成キャッシュにも，これと同様の機能を付加する．

具体的には，それぞれのバンクからデータを読み出し，読み出したデータを正しい順番に直して結合することで，整列化制約の解除を行うものである．2 バンクでのデータ読み出しの例を図 5.5 に示す．

5.5 アドレス変換

4 章で記述したように，ラスタ形式のアドレスからタイル形式での格納に対応したアドレスに変換する必要がある．ラスタアドレスからタイルアドレスへの変換はビットの並び替えで可能である．

画像の横幅が 2048 の場合の例を図 5.6 に示す．ラスタアドレスの 0 か

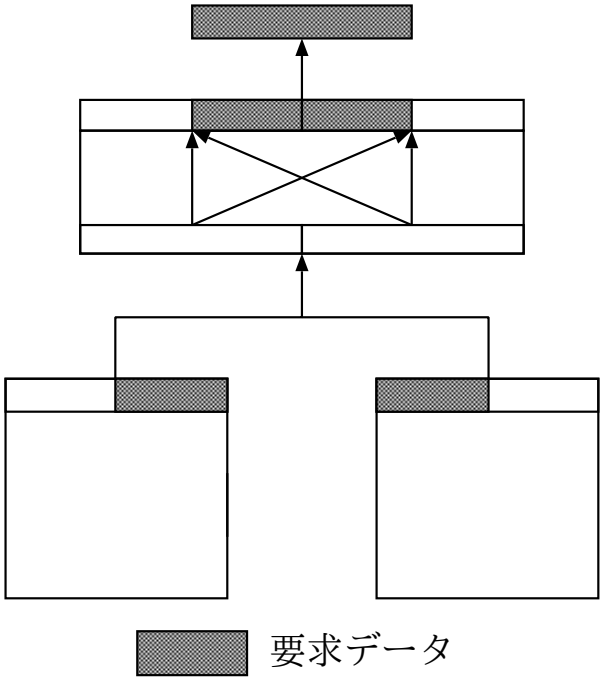

図 5.5: 整列化制約解除機構を持つ従来キャッシュ

ら 2 ビット目がラスタ形式で格納されている画像の 8 行分を，11 から 12 ビット目が 4 列分を管理しているため，11 から 12 ビット目を 3 から 4 ビット目とすることで，タイル形式に対応したアドレスとなる．

このように，画像の幅が 2 の累乗ならば，複雑な処理が必要なく，ラスタ形式からタイル形式への変換を簡潔に行うことができる．

5.6 スキュードアレイ

複数バンクに分けた場合，図 5.7 に示すように，通常の格納方式だとロードまたはストア時に 1 つのバンクにアクセスが集中し，並列に動作

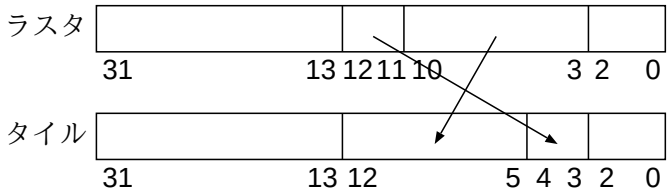

図 5.6: アドレス変換

しないことがある．図 5.7 には通常の格納方式からラインデータを読み出す例を示しているが，要求データが同じバンクに格納されているために，右端のバンクに対して 3 回のアクセスが必要になり，データの転送を滞らせてしまう．

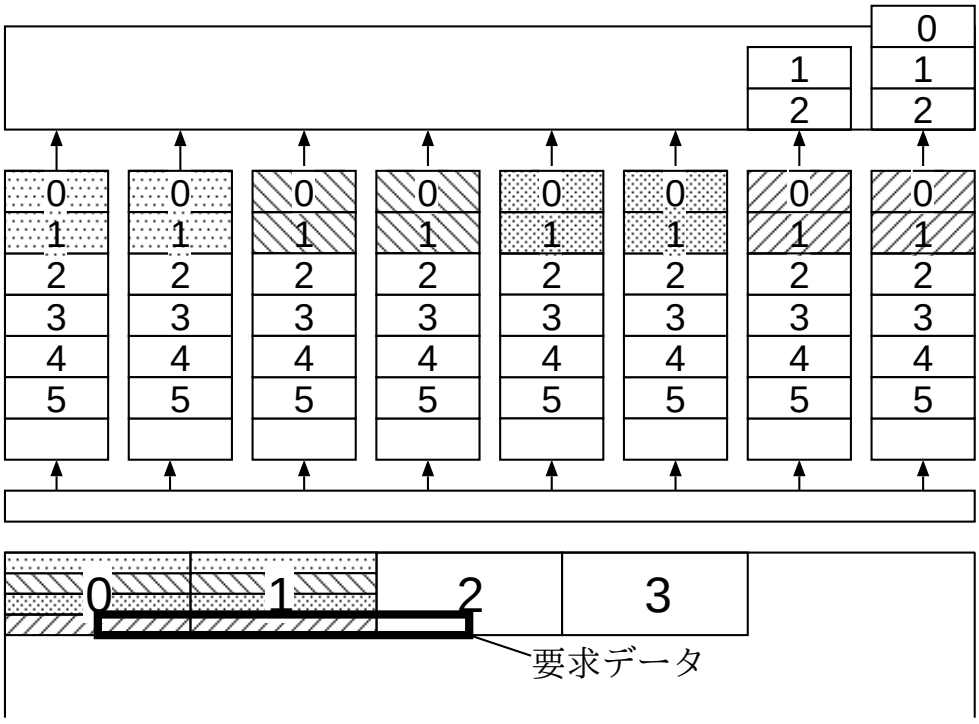

図 5.7: スキュードアレイでないタイル格納

そこで，図 5.8 のようにデータを格納する際にデータを並び替えて，各番号のタイル構成ラインをずらしたスキューアレイで格納する．しかし，ずらして格納しているために読み出す際にもデータの並びを元に戻す必要がある．その機能をバレルシフタとパーミュテーションネットワークに付加する．

スキューアレイで格納することで，ロード及びストアの際に，特定のバンクにアクセスが集中することを防ぎ，それぞれのバンクに大して並列にアクセスすることが可能となる．

また，図 5.8 には 0〜2 のタイルにまたがる 16 画素幅のラインでロードする例を示している．

5.7　スプリットインデックス

動画像処理では，画像のある地点からその近傍を参照することが多い．その際，ダイレクトマップ方式やセットアソシアティブ方式で格納すると，キャッシュの競合が頻発する可能性がある．フルアソシアティブ方式で格納すると，キャッシュの競合は発生しなくなるが，その制御のために，ダイレクトマップ方式やセットアソシアティブ方式と比較して，非常に膨大なものとなってしまう問題点がある．

図 5.8: スキュードアレイ形式でのタイル格納とそこからのライン読み出し

そこで，回路規模の増加を抑えつつ，キャッシュの競合を減らす手法として，スプリットインデックス [4] を提案キャッシュに組み込む.

通常だと競合が発生しないのは画像の水平方向に対しての 1 次元的なものだが，それを 2 次元的なものにすることによって，2 次元の近傍アクセスが多い画像処理において，競合の発生回数を低減することができる.

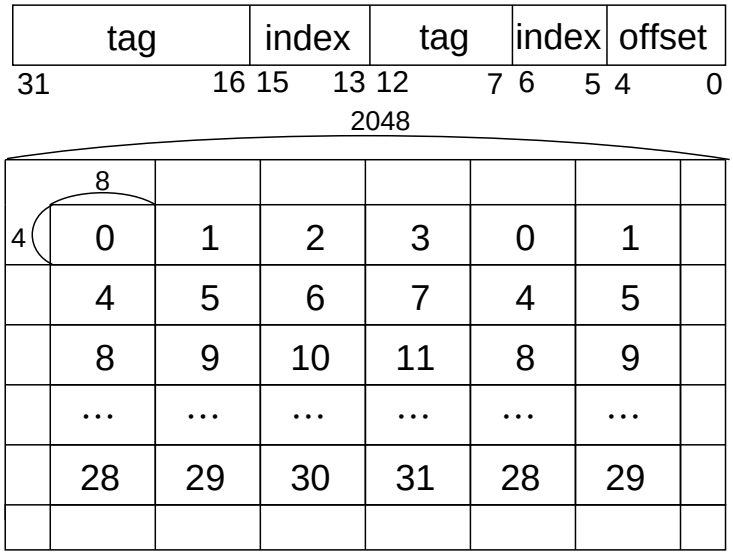

図 5.9: スプリットインデックス

6　性能評価

6.1　転置にかかるサイクル数

図 6.10 に 4x4 の転置の例を示す．タイル形式のままロードし，並び替えによって転置を行う．また，4x4 の転置を組み合わせることで更に大きなサイズの転置が可能である．

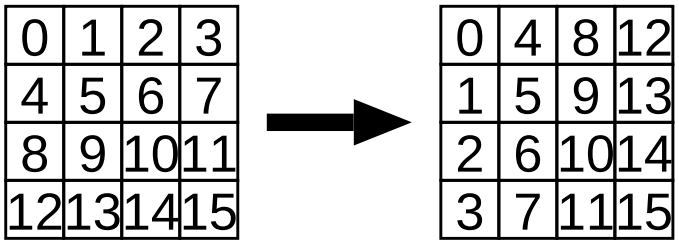

図 6.10: 4x4 ブロックの転置

ロードとストアを含めた 4x4 の転置に要するサイクル数を，1 命令 1 サイクルと仮定し，従来キャッシュと提案キャッシュそれぞれの場合で求めたものを表 6.1 に示す．

従来キャッシュでは 4 列分のロードとストアに必ず 4 サイクル必要となる．また，ロードしたデータの並び替えに 8 サイクルかかり，合計で 16 サイクルとなる [5]．提案キャッシュの場合，タイル形式でのを行うことができ，ロードと同時に並び替えを行うので，従来キャッシュに比べて少ないサイクル数で転置を行うことができる．

表 6.1: 4x4 転置サイクル数

	従来キャッシュ	提案キャッシュ
8bit データ	16(MMX)	2
16bit データ	16(MMX)	4
32bit データ	16(SSE2)	8

16bit データの DCT にかかるサイクル数は，従来キャッシュの場合 49 サイクルとなる．これに対して提案キャッシュの場合は 38 サイクルとなり，従来キャッシュに比べて約 1.3 倍高速化できる．

6.2 動き探索におけるサイクル数

6.2.1 評価方法

評価は動画像符号化ソフトウェアに提案キャッシュの機能シミュレータを組み込み，動き探索におけるデータ読み込みにかかるサイクル数を求め，整列化制約解除機能を持つ従来キャッシュと比較した．詳細なエンコード条件は表 6.2 のとおりである．また，各メモリへの読み込みサイクル数は表 6.3 のように仮定する．

6.2.2 評価結果

動き探索アルゴリズムに拡張テンプレートを用いた場合の評価結果を図 6.11 に示す．拡張テンプレートでは，1KB 及び 2KB といった，低容

表 6.2: エンコード条件

項目	内容
入力画像 (ITE 標準画像)	HorseRace(540〜570)
入力画像サイズ	1920x1056, 30fps
フレーム数	30
エンコーダ	JM11.0
シーケンス構成	M=2(IBBPBBP…)
ブロックサイズ	16x16, 16x8, 8x16, 8x8, 8x4
探索アルゴリズム	拡張テンプレート [6], EPZS

表 6.3: メモリ構成

	容量	サイクル数
提案キャッシュ	1KB〜32KB	1cycle
2 次キャッシュ	256KB	10cycle
主記憶	制限無し	200cycle

量時に大幅にサイクル数を低減することができた．また，キャッシュ容量を増やさずとも，高性能を実現することができた．しかし，4KB よりも大きなキャッシュ容量になると，従来キャッシュとの差はほとんど見られないようになった．この原因としては，拡張テンプレートは元々，他の動き探索に比べてデータの再利用性が高く，探索に必要なデータがほぼキャッシュ内に収まってしまい，従来キャッシュも下位の記憶階層とのリプレースがほとんど行われなくなったためであると考えられる．

また，SAD 演算にかかるサイクル数と比較して，データ読み込みにか

かるサイクル数のほうが大幅に多く，SAD 演算よりもデータアクセスが高速化の障害となっていることがわかる．

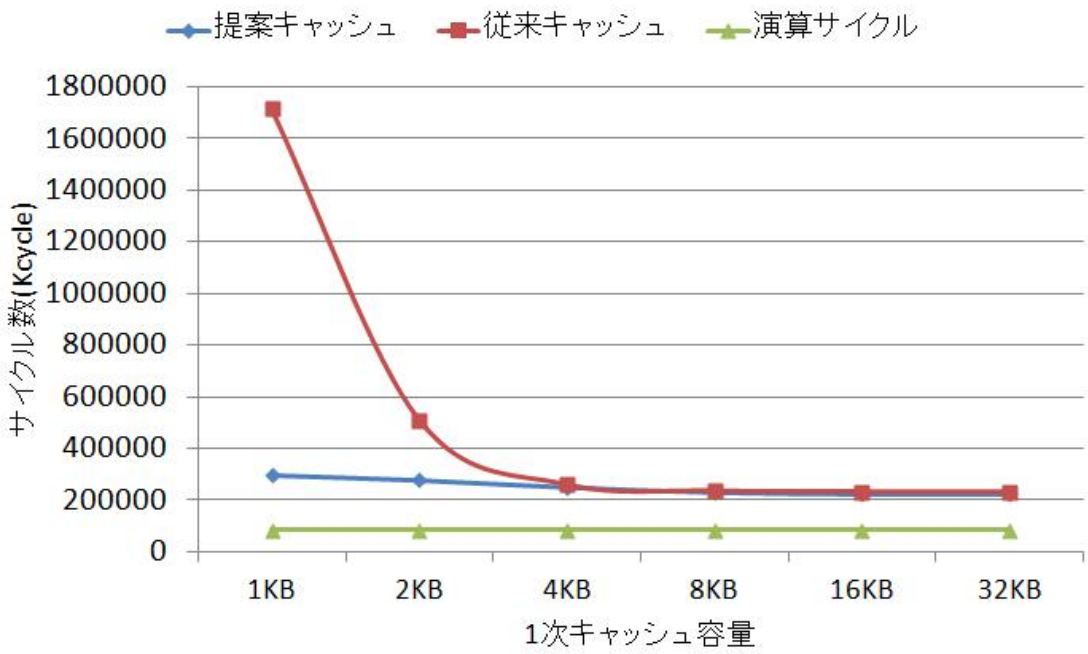

図 6.11: 拡張テンプレートでの評価結果

図 6.12 に EPZS での評価結果を示す．EPZS では，どのキャッシュ容量に対しても従来キャッシュと比較して，サイクル数を低減することができた．中でも提案キャッシュの効果が大きく見られたのは，拡張テンプレートと同様に低容量時である．また，提案キャッシュは従来キャッシュの半分の容量で，同程度以上の性能が出ていることがわかる．

EPZS においても，SAD 演算にかかるサイクル数と比較して，データ

読み込みにかかるサイクル数のほうが大幅に多く，SAD 演算よりもデータアクセスが高速化の障害となっていることがわかる．

拡張テンプレートと EPZS どちらの場合も低容量時に高性能を実現できたのは，タイル形式を用いることで，余分なデータ領域へのアクセスが減るため，余分なデータの入れ替えが発生し難くなり，多くのサイクル数を要する下位の記憶階層までデータを取りに行く必要が少ないためであると考えられる．

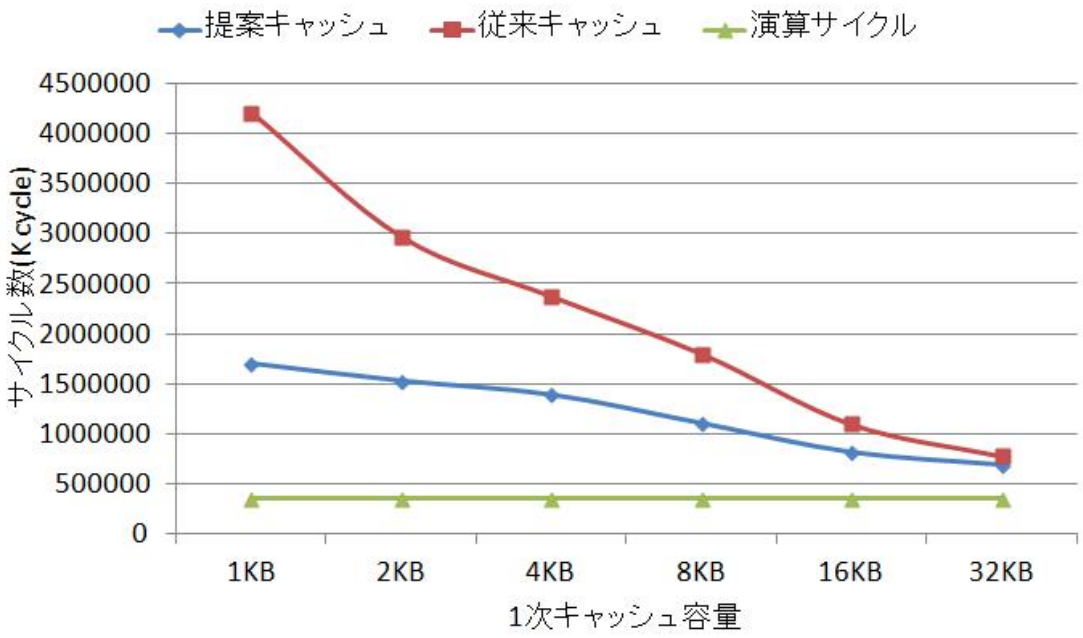

図 6.12: EPZS での評価結果

6.3　ハードウェア規模

今回，転置機能を持たず，1 つのバンクのキャッシュラインが 8 バイトで 4 バンク合計 1KB，16 画素 1 ライン形式でのロードのみに対応したものを設計した．図 6.13 に概要を示す．

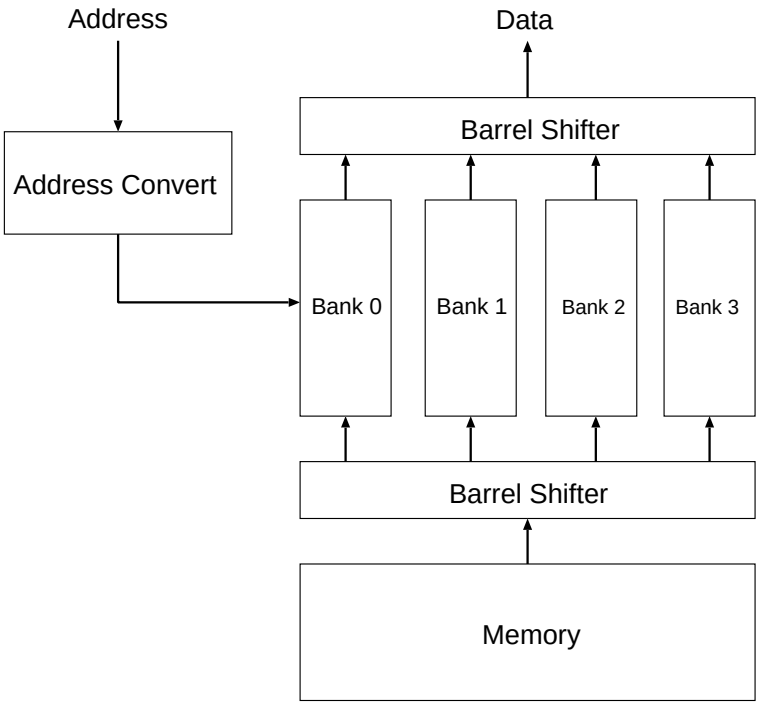

図 6.13: 設計したキャッシュメモリ

6.3.1　動作内容

演算器とデータ部間のバレルシフタは，もとのアドレスの 4 から 5 ビット目がタイル形式内のどの行か，12 から 13 ビット目がどのバンクにアク

セスするかを表しているので，これらを用いて正しい並びに戻す．さらに，1 ビット目から 3 ビット目によって整列化制約解除のために何ビットシフトするかを決定する．

データを格納する際に使用するバレルシフタでは，下位の記憶階層に送るアドレスの 6 と 7 ビット目が，行方向に 4 つ分のタイルを管理しており，これによってデータの並びを決定する．

16 画素幅でロードする場合，最大 3 つのキャッシュラインにアクセスすることになるが，それぞれでミス/ヒットを判定しなければならない．そこで，図 6.14 のように，元のアドレスに 8 と 15 を加算した後にタイル形式に変換したものを生成する．8x4 のタイル形式を想定しているので，8 を加算して生成したものは必ず隣のタイルにアクセスすることになる．15 を加算して生成したものは，16 画素幅の最後尾がどこを指しているかということで必要になってくる．この最後尾は，先ほどの 8 を加算して生成したアドレスと同じタイルを指す場合と，更に隣のタイルを指す場合がある．同じタイルを指している場合，このアドレスにより得られるミス/ヒット判定結果は無視する．

生成したアドレスをどのバンクに送るかは，バレルシフタと同様にもとのアドレスの 4 から 5 ビット目と 12 から 13 ビット目により決定でき

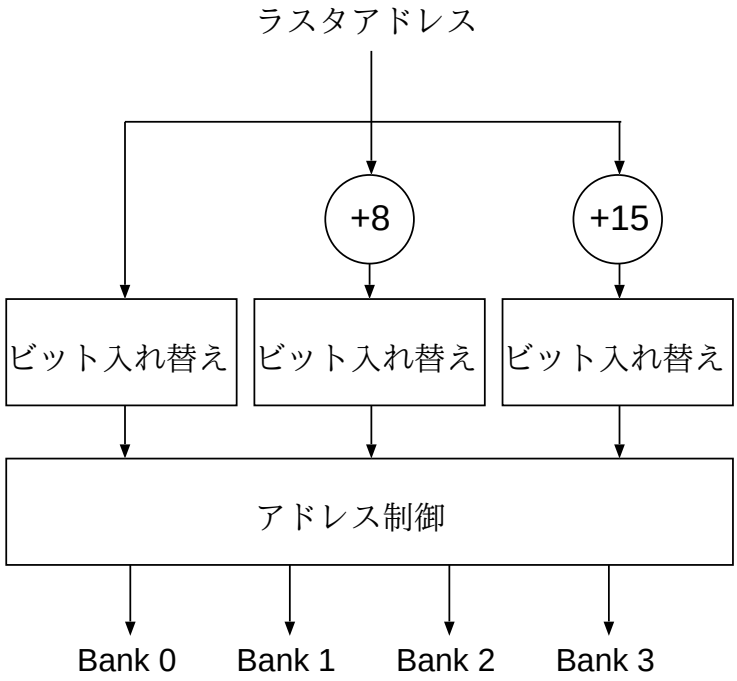

図 6.14: アドレス変換部詳細

る．そのパターンをまとめたものを表 6.4 に示す．addr がラスタアドレスをタイルアドレスに変換したもので，addr8 と addr15 が 8 と 15 を加算し，タイルアドレスに変換したものである．

表 6.4: アドレスの送り先

{[4:3], [12:11]} の値	addr	addr8	addr15
0000, 0111, 1010, 1101	Bank0	Bank1	Bank2
0001, 0100, 1011, 1110	Bank1	Bank2	Bank3
0010, 0101, 1000, 1111	Bank2	Bank3	Bank0
0011, 0110, 1001, 1100	Bank3	Bank0	Bank1

ラスタアドレス 1814h にアクセスする例を図 6.15 に示す．1814h に 8

と 15 を加算したものはそれぞれ 181Ch と 1823h となる．これらをタイル形式に変換すると，addr:005Ch，addr8:007Ch，addr15:009Bh となる．このアドレスをどのバンクに送るかは，先ほどの表 6.4 により決定される．この場合，1814h の 5 と 4 ビット目，13 と 12 ビット目を結合した値は 1011b なので，それぞれのアドレスを Bank1 と Bank2 と Bank3 に送る．各バンクで，送られてきたアドレスの 32 から 17 ビット目と 13 から 8 ビット目を結合したものをタグとして比較し，ミス/ヒットを判定する．ミスしていた場合，そのアドレスを下位の記憶階層に送ることで，欲しい部分のデータのリプレースを行う．全てヒットとなれば演算器にヒット信号を送る．

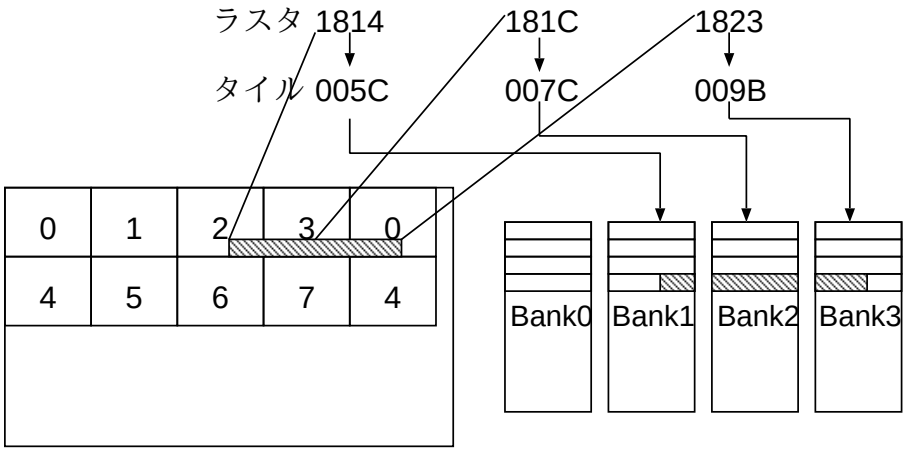

図 6.15: アクセス例

6.3.2 結果

ハードウェア規模は2入力のNAND換算でのゲート数として算出した．結果としてデータを保持するSRAM部分を除いたものは93499ゲートとなった．

また，特別な機能を持たない通常のダイレクトマップキャッシュを設計し，同様に求めた結果は15505ゲートとなった．

今回設計したものは，ハードウェア規模の小さく済むダイレクトマップキャッシュと比較して約6倍となった．データを格納する際と，読み出す際にデータの並び替えを行うが，その処理の部分が原因で大きくなったものと考えられる．

6.4 考察

提案キャッシュを用いることでタイル形式に対応し，従来キャッシュに比べて動画像処理を効率的に行えることを示した．

転置に関しては従来キャッシュよりも高速化できることを示した．データサイズが大きくなると，提案キャッシュでもロードとストアの回数が増加してしまうため転置にかかるサイクル数が増加してしまっているが，並び替えに要するサイクル数が少ない分従来キャッシュに比べて高速化が可

能である．

動き探索における評価では，探索アルゴリズムにより提案キャッシュの有効性に差異はあるものの，従来キャッシュよりも低容量で高性能を実現できている．しかし，SAD 演算に対してデータアクセスにかかるサイクル数のほうが非常に多く，より一層の高速化が求められる．今回はダイレクトマップで評価を行ったが，セットアソシアティブにすることでより高速化できる可能性がある．

また，ハードウェア規模は様々な機能を追加した結果，増加してしまったが，このような制御に必要な部分よりも SRAM 部分に必要なハードウェア規模のほうが一般的に大きいため，低容量で高性能を実現できているので，低容量で良いのであれば，結果としてハードウェア規模は小さくなる可能性もある．しかし，提案したキャッシュは 4x4 転置機能なども実装しようとすると，今回設計したものよりも更にハードウェア規模が増大してしまうため，今後設計する際はなるべくハードウェア規模が増大しないように設計していく必要がある．

7　おわりに

本研究では，画像処理における 2 次元データ処理に適した，新構成キャッシュを提案した．新構成キャッシュを用いることで，画像処理で多用される転置を 8 倍高速化できることと，低容量でもデータ転送にかかるサイクル数を軽減でき，高性能を実現できることを示した．

しかし，現状の設計では画像の横幅が 2048 にしか対応できておらず，今後は横幅の可変性を確保することや，転置機能を有したパーミュテーションネットワークを設計していく必要がある．

また，今回提案したキャッシュを用いたとき，現状のままのプログラムでは正しく動作しないため，提案キャッシュに対応するコンパイラを作成する必要もある．

謝辞

本研究を進めるにあたり，様々なご指導を頂きました近藤利夫教授，大野和彦講師，佐々木敬泰助教に深く感謝いたします．また，様々な面でお世話になった計算機アーキテクチャ研究室の皆様に感謝いたします．

参考文献

[1] 猪俣 匠. 動き探索処理に適したロードバッファの研究. 三重大学，2012.

[2] A. Waksman. A Permutation Network. Journal of the ACM, 15(1):159-163, January 1968.

[3] H. Chang, W. Sung. Efficient Vectorization of SIMD Programs with Non-aligned and Irregular Data Access Hardware. in CASES '08: Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems, pp. 167-176, 2008.

[4] S. Yoon, S.-I. CHAE. Cache optimization for H.264/AVC motion compensation. ISSN:1745-1361, 0916-8532. IEICE Transactions on Information and Systems E91-D(12), 2902—2905 (IEICE)

[5] W.-Y. Lo, D. P.-K. Lun, W.-C. Siu, W. Wang, J. Song. Improved SIMD Architecture for High Performance Video Processors. Circuits and Systems for Video Technology, IEEE Transactions on, 21(12):1769—1783, december 2011.

[6] 菅谷知大, 佐々木敬泰, 大野和彦, 近藤利夫. 拡張テンプレート複数併用法と探索区域予測を組み合わせる H.264 対応の高効率動き検出法. Proc. 情報処理学会研究報告, オーディオビジュアル複合情報処理, 2008.