卒業論文

題目

動き探索処理用SAD演算命令の制定

指導教員

近藤 利夫 教授

2013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

渡邊 敬太(409860)

内容梗概

2016 年にスーパーハイビジョンの試験放送が予定されるなど，近年動画像の高精細化が進んでいる．この高精細化に伴い，符号化処理に必要な処理量が大幅に増大している．

符号化処理の大半は動き探索処理が占めている．動き探索処理では，類似度を調べるために符号化対象画像と参照画像間でブロックマッチングを行う．このブロックマッチングにより，SAD 値を求めている．また，最小の SAD 値を求めるために最小値抽出がおこなわれている．そのため，動き探索処理の高速化には高速かつ高効率な SAD 演算命令と最小値抽出命令が必要である．しかし，現行の汎用プロセッサに組み込まれている動き探索処理用の命令は現在主流となっている追跡型の動き探索に適していないため，これらを高速に処理できない．

そこで，この問題点の解決に向けて，本研究では動き探索処理を高速に処理可能な最小値抽出命令と，高効率な処理を可能とした新たな動き探索法を提案した．また，従来手法と提案手法に必要な動き探索処理のステップ数の評価を行った．その結果，最大 1.47 倍処理速度が向上し，動き探索処理の高速化が行われた．

Abstract

In 2016, the test broadcast of Ultra High Definition Television is scheduled, and the high definition of video image has made progress in recent years. By this high definition, the throughput required for the encoding processing is greatly increasing.

The motion estimation processing occupies most of the encode prosessing. The motion estimation does the block matching between the current picture and the reference picture to calculate similarity measure. This block matching calculates the SAD. Further, the minimum value extraction is executed to take the minimum SAD. Therefore, high speed and high efficient SAD operation instruction and minimum value extraction instruction are necessary for fast motion estimation. However, the instructions for motion estimation within embedded on the general purpose processors are not suitable to the main current motion estimation of tracking type, these are unable to process at high speed.

Hence, in order to solve this problem, in this study, I proposed minimum value extraction instruction and new motion estimation for speeding up. I evaluated the number of steps to the motion estimation required for the conventional method and the proposed method. As a result, the performance of processing speed improved by maximum 1.47 times and it was sped up the motion estimation.

目 次

図 目 次

表 目 次

1　はじめに

1.1　研究背景

一般的に使用されているハイビジョンの 16 倍の画素数を有するスーパーハイビジョンの試験放送が 2016 年に予定されるなど，近年動画像の高精細化が進んでいる．これに伴い，動画像の圧縮符号化に要求される処理量が増大している．この符号化処理の大半は動き探索処理によって占められている．

また，現在主流である符号化圧縮規格 H.264/AVC[1][2] では，7 種類の可変ブロックサイズを用いることで，より精度の高い探索を行っている．これらの可変ブロックサイズの使用もあって，動き探索処理の演算量が大幅に増加している．そこで，符号化処理にかかる時間を短縮するために動き探索処理の高速化が必要である．

1.2　研究目的

符号化処理の演算量の大半を占める動き探索処理では，主に参照画像ブロックと符号化対象ブロックの類似度を求めるため，SAD(Sum of Absolute Difference) 演算と，最小値位置検出が行われている．そのため，動き探索処理の高速化には SAD 演算と最小値位置検出を高速化する必要がある．

また，当研究室では高効率の動き探索法として，データ再利用の面で優れた追跡型スクエアサーチを使用する．x86 プロセッサには，SSE4(Streaming SIMD Extensions 4) として，動き探索を高速に処理するための SIMD(Single Instruction stream Multiple Data stream) 命令に，SAD 演算命令 MPSADBW と最小値抽出命令 PHMINPOSUW が含まれている．しかし，この二つの動き探索用命令は，スクエアサーチに適合性がなくほとんど役に立たない．そこで，この追跡型の動き探索処理をより高速かつ高効率に処理することを可能とする SAD 演算命令と最小値抽出用命令の実現を目指す．

2　動き探索とその高効率処理の核となる命令への要求

2.1　動き探索

動き探索では，符号化対象画像と参照画像を使用し，ブロックマッチングを行うことで探索を行う．このブロックマッチングは，符号化対象画像と参照画像の類似度を求めるために使用されている．また，類似度の計算には主に差分絶対値和 (SAD) 演算と最小値抽出が使用されている．そのため，SAD 演算と最小値抽出の高速化を行うことで，動き探索処理を高速化させることが可能である．

現在動き探索は，高効率な動き探索法として追跡型の動き探索が主流である．この追跡型の動き探索は，ブロックマッチングの走査範囲を必要に応じて移動させながら探索を繰り返すことで，最も類似度の高いブロックを検出することが可能な探索法である．追跡型動き探索には，探索中心とその上下左右 4 個のブロックのブロックマッチングを行うダイヤモンドサーチや，探索中心とその周囲 8 個のブロックのブロックをマッチングを行うスクエアサーチなどがある．探索中心と周囲の探索点を記したダイヤモンドサーチとスクエアサーチの図をそれぞれ図 2.1 と図 2.2 に示す．

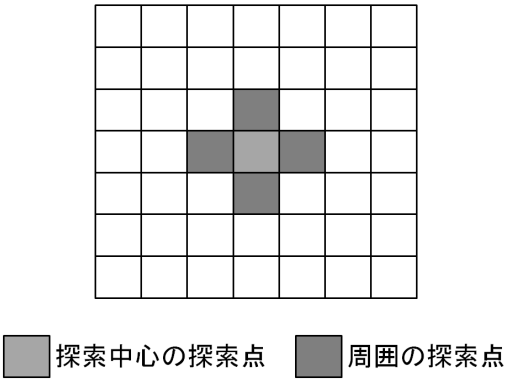

図 2.1: ダイヤモンドでのブロックマッチング

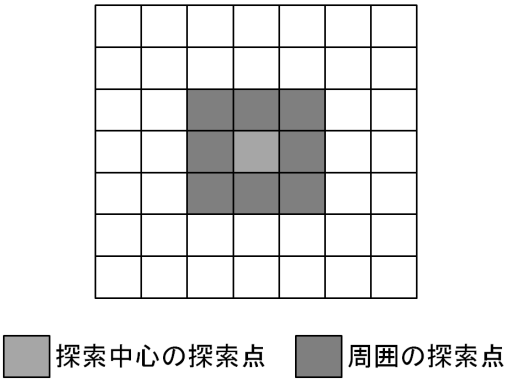

図 2.2: スクエアサーチでのブロックマッチング

また探索点を 1 画素空間空けることで，高速化を図るスモールヘキサゴンサーチがある．探索中心と周囲の探索点を記したスモールヘキサゴンサーチの図を図 2.3 に示す.

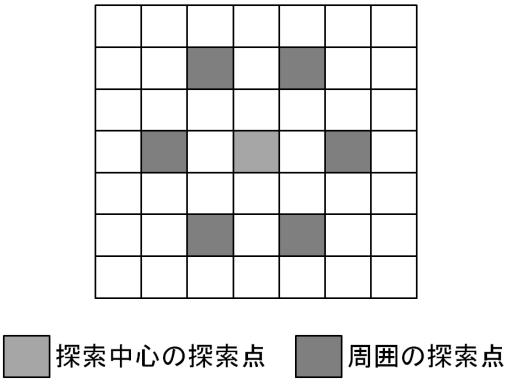

図 2.3: スモールヘキサゴンサーチでのブロックマッチング

2.2 スクエアサーチ

当研究室では追跡型の動き探索としてスクエアサーチを使用している．スクエアサーチでは，探索中心のブロックとその周囲の 8 個のブロックから SAD 値を求め，その結果から最小の SAD 値を持つ点を次の探索中心として最小 SAD 値が探索中心になるまで繰り返す．これにより，最も類似度の高いブロックを検出することが可能である．スクエアサーチのアルゴリズムを図 2.4 に示す。

スクエアサーチはダイヤモンドやヘキサゴンなど他の動き探索法と比較し，画像データの再利用性が高く，データの読み込みによるオーバーヘッドが少ないため，動き探索を高速に処理するのに向いている．

しかし，現行の汎用プロセッサは 9 点分の SAD 値 144bit を効率よく処

理できる命令がサポートされていない．そこで，演算幅が 2 のべき乗である現行の汎用プロセッサのデータパス向けに，スクエアサーチを改良する必要がある．

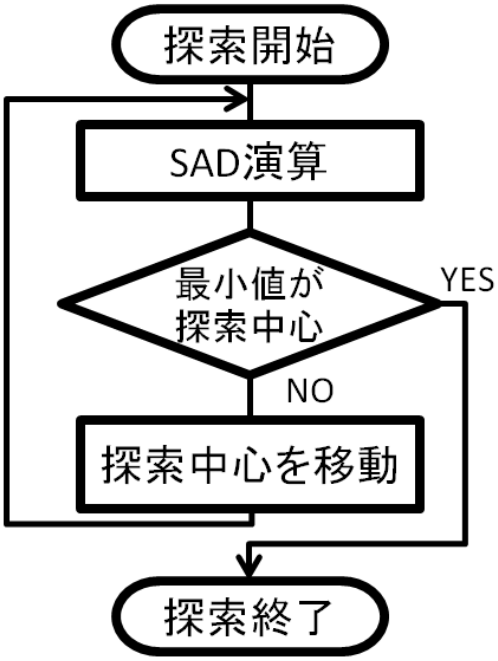

図 2.4: スクエアサーチのアルゴリズム

2.3 差分絶対値和演算

動き探索内のブロックマッチングでは，符号化対象画像と参照画像の類似度を調べるために，SAD 値の計算をおこなっている．符号化対象画

像ブロックを X，参照画像ブロックを Y とした 4x4 のブロックサイズにおける SAD を求めるための計算例を図 2.5 に示す。

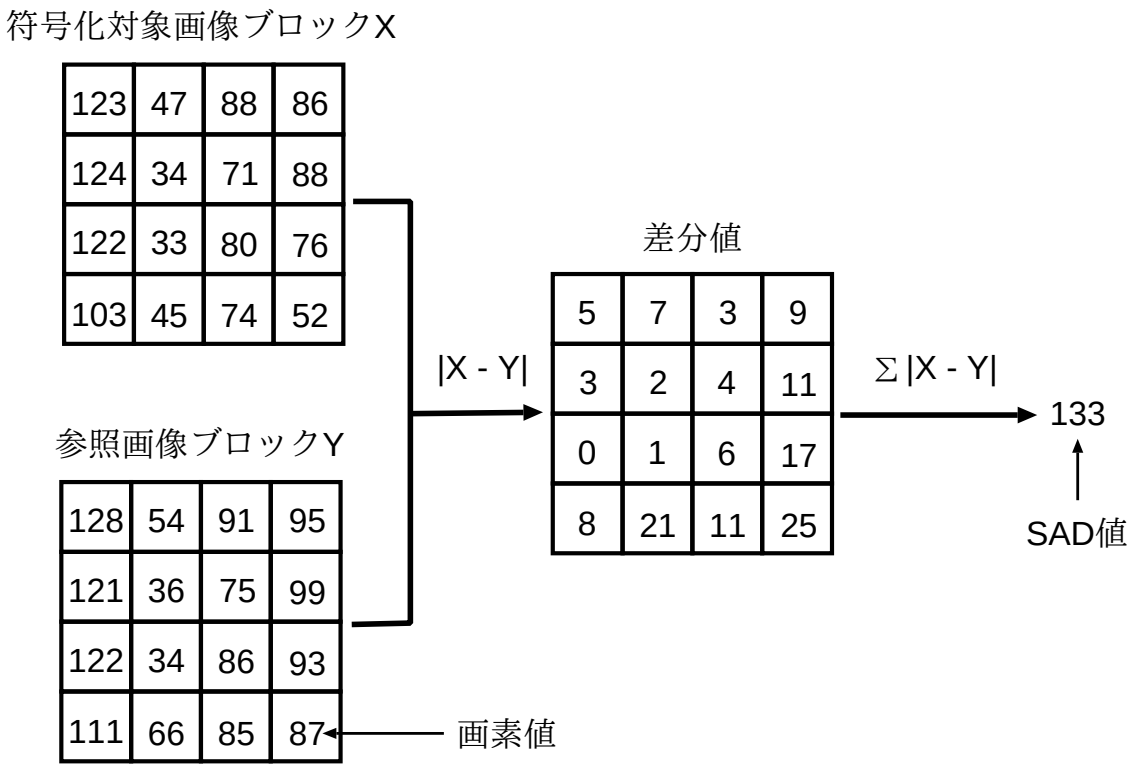

図 2.5: SAD 値演算例

SAD 値が小さければ小さいほど、2 つのブロック間の類似度は大きい。動き探索では、この SAD 値を求める処理と最小値の位置の検出を行う処理が必要である。

2.4 可変ブロックサイズ対応の差分絶対値和演算

現在，主流となっている符号化圧縮規格の H.264/AVC では，7 種類の可変ブロックサイズを用いることで，より精度の高い動き探索を行っている．可変ブロックサイズには，16x16，16x8，8x16，8x8，8x4，4x8，4x4 のブロックサイズが用意されている．可変ブロックサイズの例を図 2.6 に示す．

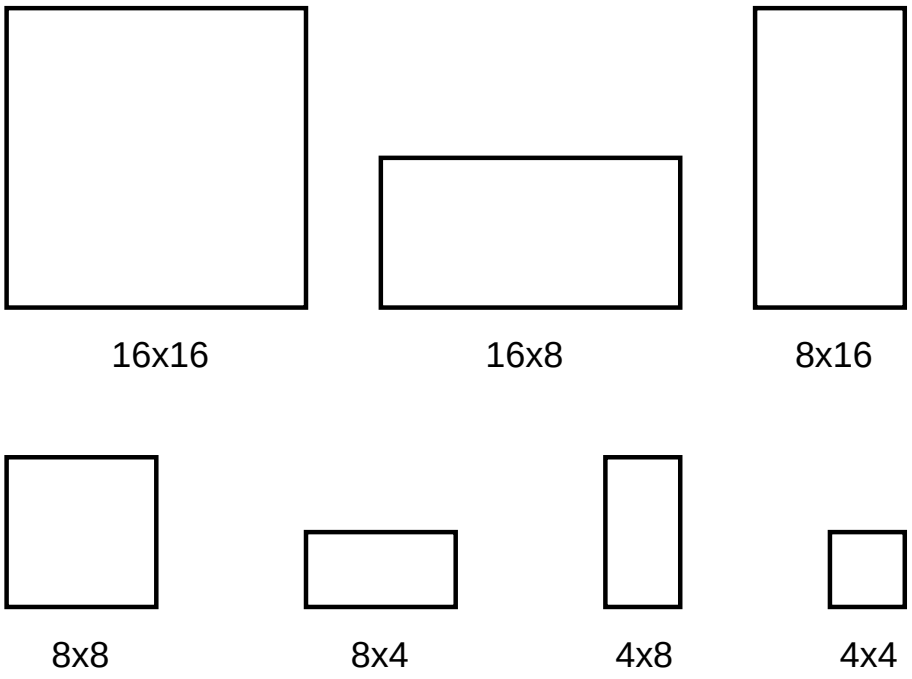

図 2.6: 7 種類の可変ブロックサイズ

4x4 の可変ブロックサイズの SAD 値を足し合わせることで，他のブロックサイズの SAD 値を求めることが可能である．そこで当研究室では，選

択されたブロックサイズのSAD値を求める際に，包含されるブロックサイズを4x4のブロックサイズのSAD値を足し合わせていくことで，同時に求める．これにより，動き探索の演算量を低減することが可能である．最大のブロックサイズである16x16のブロック探索を行う場合，4x4のブロックサイズのSAD値を16個演算する必要がある．そのため，動き探索の高化には高速な4x4のブロックサイズのSAD演算が必要である．4x4のブロックサイズから，他のブロックサイズを生成する例を図2.7に示す．

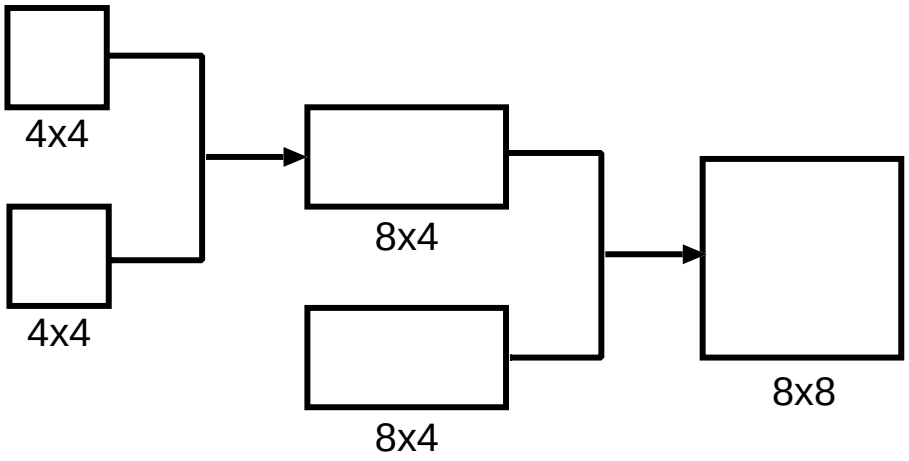

図 2.7: 可変ブロックサイズの生成

3　x86 プロセッサの動き探索用命令とその問題点

3.1　動き探索命令の概要

x86 プロセッサでは SIMD(Single Instruction Multiple Data：単一命令/複数データ) 演算が使用されている．SIMD 演算は 1 命令で複数のデータに対して処理をおこなうことができる演算方式であり，画像処理などの同一の演算を繰り返す処理の高速化に役立つ．SIMD 演算の例を図 3.8 に示す.

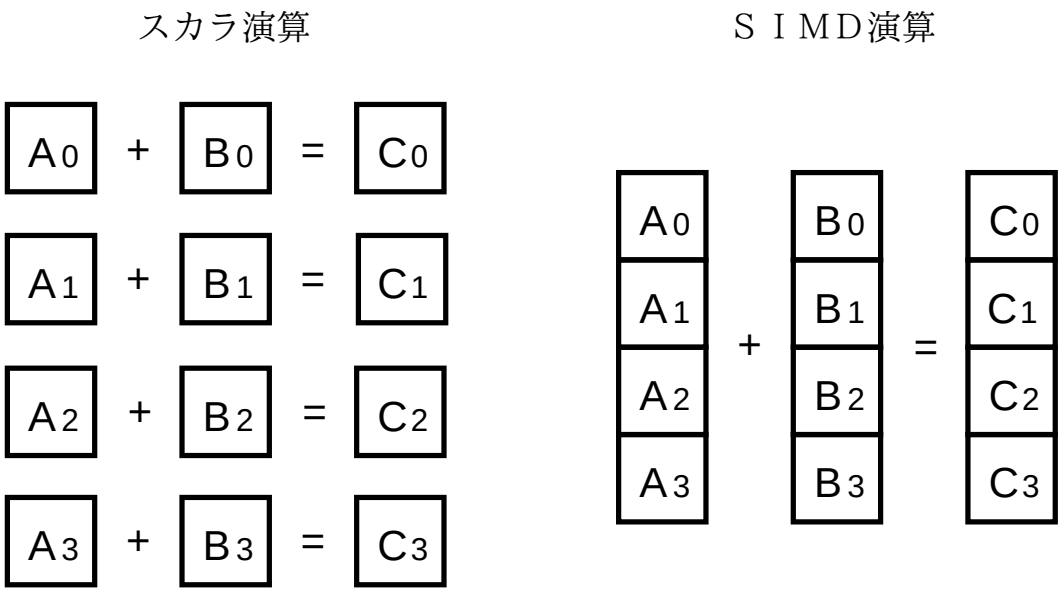

図 3.8: SIMD 演算

SIMD 命令には加算・減算・比較があるほか，動き探索処理用の並列 SAD 演算命令 MPSADBW と，最小値抽出命令 PHMINPOSUW がある.

しかし，これらの動き探索処理用の命令は水平，垂直方向の探索点が 3 と小さいスクエアパタンに適した命令ではなく，スクエアサーチを効率よく処理できない．そのため，より高速な動き探索処理用命令が必要不可欠である．

3.2 MPSADBW 命令とその問題点

MPSADBW 命令は x86 プロセッサに搭載されている 2 オペランド形式の SIMD 型の命令である．第 1 オペランドに与えられたレジスタには，4 画素幅の画素値データが 8 個格納される．これは，8 個のデータを 1 画素分ずつずらしたデータである．また，第 2 オペランドに与えられたレジスタには，4 画素分の画素値データが 1 個格納される．この 2 つのレジスタに与えられた画素値データから SAD 値演算をおこない，その結果を第 1 オペランドに指定したレジスタに格納することで，水平方向の 8 点の SAD 演算を並列に処理することが可能である．MPSADBW 命令を使用した SAD 値の結果を図 3.9 に示す．

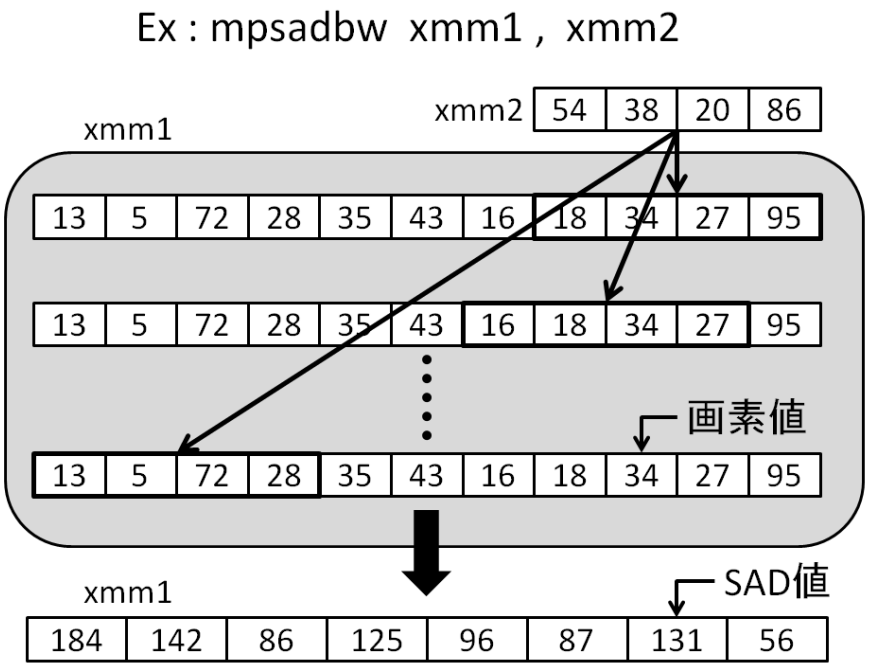

図 3.9: MPSADBW 命令による SAD 演算

しかし，水平方向の 8 点の SAD 値は，現在一定の走査範囲を左上から右下まで 1 つずつ調べる全探索などにしか使用されない．そのため，当研究室で用いられるスクエアサーチに適合性がなく，4 画素幅ブロックのスクエアサーチの構成ラインに対する 3 点分を並列に処理できるだけである．スクエアサーチと全探索において，MPSADBW 命令を使用し，水平方向の 8 点の SAD 値演算を行う例をそれぞれ図 3.10 と図 3.11 に示す．

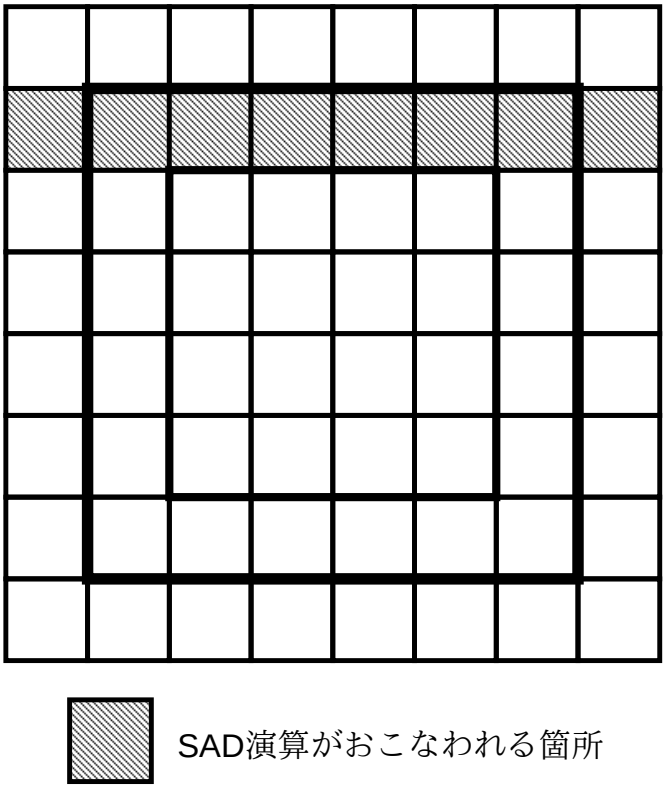

図 3.10: スクエアサーチによる動き探索

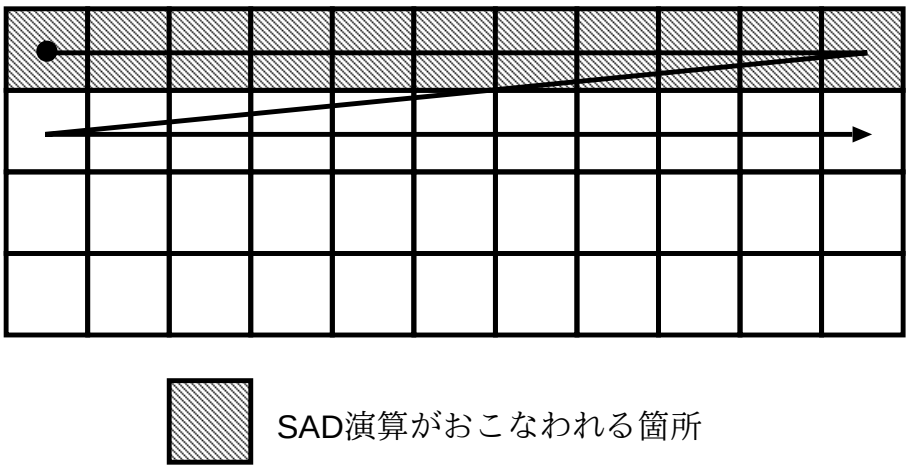

図 3.11: 全探索による動き探索

3.3 PHMINPOSUW 命令とその問題点

PHMINPOSUW 命令は x86 プロセッサに搭載されている 2 オペランド形式の SIMD 型の命令である．第 2 オペランドに与えられたレジスタに納められている 8 組の 16bit 幅の SAD 値をまとめて比較し，その最小値と格納位置 (Index) を求める．その結果は第 1 オペランドで指定されたレジスタに格納され，最小 SAD 値と次の探索点を求めることができる．PHMINPOSUW 命令を使用した最小値抽出の結果を図 3.12 に示す．

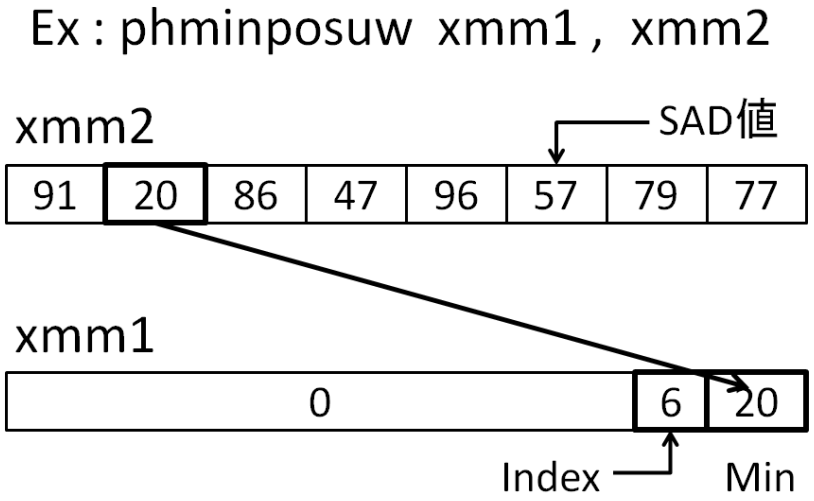

図 3.12: PHMINPOSUW 命令による最小値抽出

また，動き探索の高速化のため，SAD 演算と探索点の移動の並列処理を行う．そのため，格納された最小 SAD 値と Index を並列処理可能な汎用レジスタに移動が必要である．しかし，16bit の最小 SAD 値と 3bit の

Index を格納するため，結果の格納には 19bit 必要になる．これでは 32bit の汎用レジスタに移動させる際に，無駄な領域ができる。また，16x16 のブロックサイズの最小 SAD 値を求める際，それに含まれる全てのブロックサイズを汎用レジスタに移動させるために 41 回の移動命令が必要となり，非常に効率が悪い．そこで，レジスタの無駄な領域を減らすと同時に，高効率かつ高速な最小値抽出命令が必要となる．

4　提案命令

4.1　概要

符号化処理を高速化させるため，動き探索処理用の拡張命令セットと包含されるブロックサイズの SAD 値の計算を効率良く行うための探索法として，以下の 3 点を提案する．

- 高並列 SAD 演算命令 (Highly Parallel Multiple Packed Sum of Absolute Difference Byte Word： MPSADBW)：スクエアサーチにおいて必要とされる SAD 演算を並列に処理することができる命令 [6].
- 8 点スクエアサーチ：包含されるブロックサイズの SAD 値を効率良く処理できる探索法
- 最小値抽出命令 (Double Comparison for Square Search： DCSS)：2 つの最小 SAD 値を同時に求めることが可能な最小値抽出命令

4.2　8 点スクエアサーチ

現在，当研究室で使用しているスクエアサーチでは，9 点分の SAD 値 144bit の演算が必要となり，効率よく処理できない．例えば，前述の PHMINPOSUW 命令は 8 点分の値から最小値を抽出する命令であるため，9

点分の最小値を求めるためには処理が 2 回分必要になる．また，144bit のデータを効率良く加算する命令は現行では存在しないため，選択したブロックサイズが包含するブロックサイズの SAD 値を効率良く求めることはできない．そこで，8 点スクエアサーチを提案する．

8 点スクエアサーチでは，探索中心の周囲の 8 点の SAD 演算を，SAD 最小点に探索中心を移動しながら，SAD 最小点が探索済み領域の端に来なくなるまで処理を繰り返す．8 点の 16bit 幅の SAD 値がちょうど 128bit に収まるため，より演算幅が 2 のべき乗になる現行の汎用プロセッサのデータパス向きである．従来手法の 9 点スクエアサーチと提案手法の 8 点スクエアサーチの探索図を図 4.13 に示す．

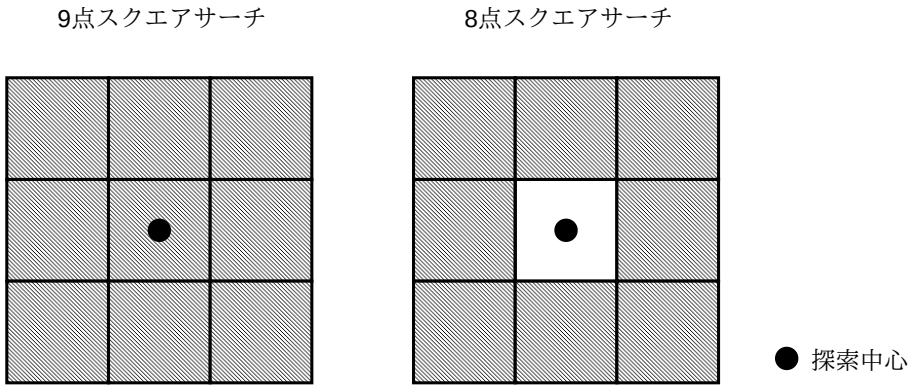

図 4.13: 8 点スクエアサーチ

この 8 点スクエアサーチを使用することで，既存の SIMD 命令を効率良く使用可能となる．これにより選択したブロックサイズが包含するブ

ロックサイズのSAD値を効率良く同時演算することで，高速化が見込める．ただし，9点スクエアサーチと比べ，探索点が1つ少なく，探索回数が増えるため，その分の処理速度が落ちる問題点もある．この問題点に対しては，性能評価を行い考察する必要がある．

4.3 DCSS命令

4.3.1 DCSS命令とその問題点

DCSS(Double Comparison for Square Search)命令は，8点スクエサーチにおいて必要とされる8点分のSAD値から最小SAD値を抽出する命令である．この命令のフォーマットは3オペランド形式である．第1オペランドと第2オペランドには，それぞれSAD値が格納されたレジスタを指定する．このレジスタから最小値を抽出し，その値と格納位置(Index)を第3オペランドに格納する．これにより，2つの最小SAD値を同時に求めることができる．しかし，SAD値が16bit，Indexが3bitが2つずつあるため，第3オペランドには38bitのデータが格納される．これは汎用レジスタの32bitに格納させることができず，レジスタ使用の効率も悪い．そのため，32bit以内に収めるために6bitの削減が必要である．そこで，SAD演算時にほぼ使用することがない，上位1bitと誤差最大3しかでない下位2bitを削減をおこなう．SAD値の削減を図4.14に，DCSS命令を

使用した 2 つの最小値抽出の結果を図 4.15 に示す.

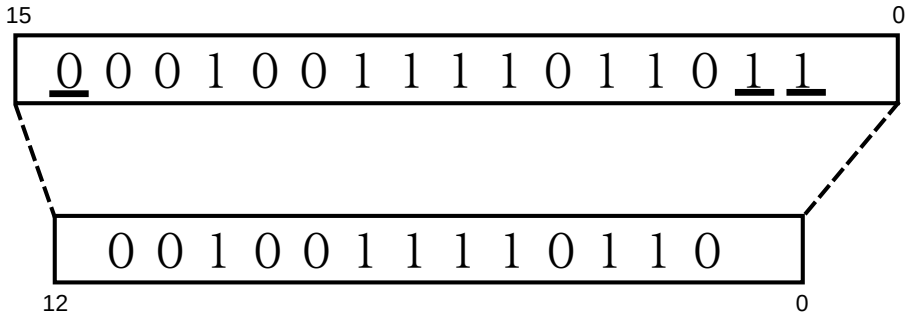

図 4.14: SAD 値の削減

この命令を使用することで，汎用レジスタへの移動回数を減らし，高速化することができる．また，一度に 2 つの最小 SAD 値を求めることができるため，PHMINPOSUW 命令と比べ，高速化が可能である．しかし，問題点として最小 SAD 値の bit 数の削減により画質劣化の可能性がある．また，今回 SAD 値の削減した箇所は，ある画像の一定のフレームで SAD 演算をおこなった際に削減しても問題ないであろうと判断したため，必ずしも最適な削減箇所とは限らない．そのため，今後様々な画像のパターンに対して評価をおこない，検討していくことが必要である．

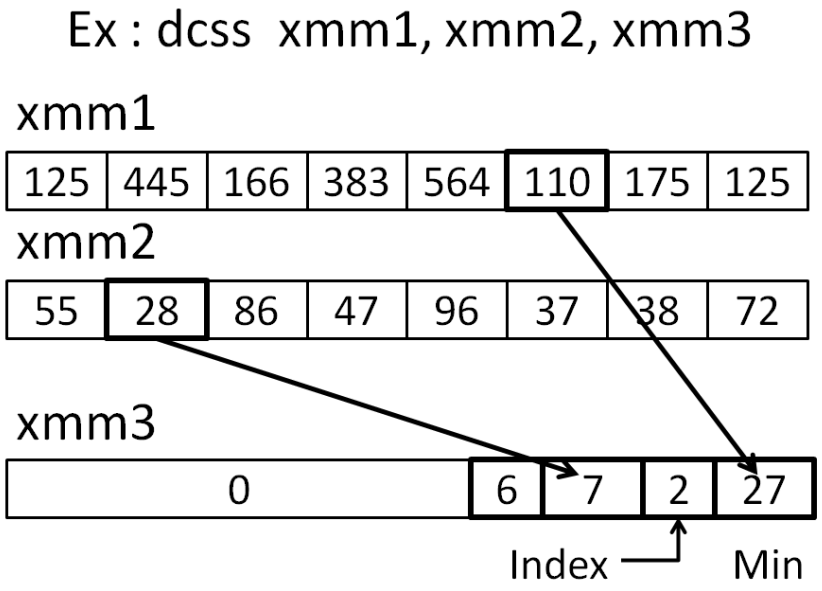

図 4.15: DCSS 命令による最小値抽出

4.3.2 回路構成

DCSS 命令による最小 SAD 値の抽出は，最小値抽出器で処理をおこなう．最小値抽出器では，スクエアサーチから得られた 8 点分の SAD 値の処理を比較器ユニットで 2 つ分並列に処理することが可能な構成になっている．また，2 つの比較器ユニットには，それぞれ 15 個の比較器と 1 個の SAD 値削減器から構成される．比較器では，8 点分の SAD 値の比較をトーナメント形式で処理することで，最小 SAD 値の抽出をおこなう．SAD 値削減器では，最小 SAD 値と Index の結合と，SAD 値の上位 1bit

と下位 2bit の削減がおこなわれる．結合した結果を指定された専用レジスタに格納する．DCSS 命令の最小値抽出器の構成を図 4.16 に示す.

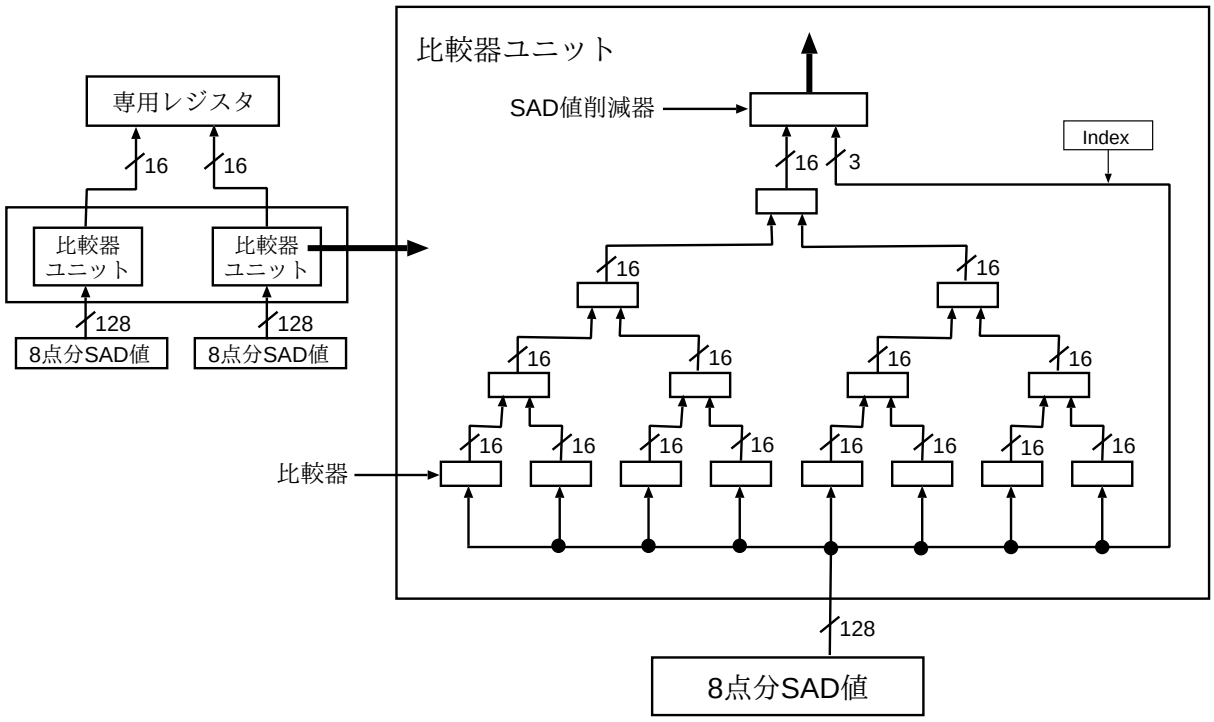

図 4.16: 最小値抽出器の構成

5　性能評価

5.1　提案手法の性能評価

提案手法の性能評価は，H.264/AVC での可変ブロックサイズを用いた動き探索処理において評価を行った．動き探索法は，追跡型のスクエアパタンを用いる．評価では，動き探索処理の演算量をどれだけ低減させることが可能かと，同時に探索精度を低下させないことが重要である．

動き探索の処理量は，従来手法と提案手法において，H.264/AVC で用いる最大の 16 画素幅のブロックサイズとそのブロックサイズが包含するブロックサイズ (16x16，16x8，8x16，8x8，8x4，4x8，4x4) の SAD 演算と最小値位置検出に必要ステップ数で評価を行った．従来手法では，動き探索法に 9 点スクエアサーチ，最小値位置検出には PHMIPOSUW 命令を使用し，また提案手法では，動き探索法に 8 点スクエアサーチ，最小値位置検出には DCSS 命令を使用する．

探索精度は，評価用の HD 画像 (1920x1056 画素)3 種類に対し，DCSS 命令を使用した場合の PSNR とビットレートを評価する．PSNR は画像の劣化度を表し，ビットレートは符号量を表す．詳細な評価条件は表 5.1 に示す．

表 5.1: 評価条件

項目	内容
入力画像	Soccer_Action
	Horse_Race
	Woman_in_Flowers
入力画像サイズ	1920x1056，30fps
フレーム数	450
エンコーダ	JM 参照ソフトウェア
シーケンス構成	M=3 (IPBBPBB$\cdots$)
ブロックサイズ	16x16，16x8，8x16，8x8，8x4 4x8 4x4

5.2 評価結果

表 5.2 は，従来手法と提案手法それぞれの動き探索処理の演算量を示す.

表 5.2: 動き探索における演算量

	ステップ数	平均探索回数 (比率)	合計ステップ数
従来手法	484	1.79	867
提案手法	274	2.12	581

また表 5.3 は，DCSS 命令を使用した場合の PSNR とビットレートを

示す.

表 5.3: 提案手法の PSNR とビットレート

	PSNR	ビットレート (倍)
20HD_Soccer_Action	Δ0.00	1.000175123
35HD_Horse_Race	Δ0.00	1.001638047
56HD_Woman_in_Flowers	Δ0.00	1.000224911

5.3 考察

評価結果より，提案手法を用いることで，動き探索に必要な演算量を32%低減し，動き探索処理が最大1.47倍の高速化が見込める．これはDCSS命令によって最小値位置検出に必要な演算量の低減されただけではなく，8点スクエアサーチを用いることで包含するブロックサイズのSAD値を効率良く加算処理できるためである．しかし，今回の評価で提案手法として使用した命令は最小値抽出命令DCSSのみである．そのため，動き探索処理の中でも多く用いられるSAD演算命令を改良し，用いることでさらなる高速化が見込めると思われる．

またPSNRとビットレートの変化がほとんど見られず，最小SAD値の上位1bitと下位2bitを削減しても問題ないと言える．

結果，探索精度を損なうことなく，動き探索処理の高速化を可能にした．

6　おわりに

本研究では，動画像の圧縮符号化処理の大半を占める動き探索処理の高速化を図るために高速かつ高効率な最小値抽出命令DCSSの提案を行った．さらに，従来の9点スクエアサーチをより汎用プロセッサのデータパス向けに改良した8点スクエアサーチを用いることで，既存のSIMD命令を効率良く使用可能とする．また，選択したブロックサイズが包含するブロックサイズのSAD値を効率良く同時演算することができる．この提案したDCSS命令と8点スクエアサーチを用いることで，x86プロセッサのPHMIPOSUW命令と9点スクエアサーチを用いた場合と比較し，理論上探索精度を損なうことなく，動き探索処理の速度を最大1.47倍向上が可能である．

しかし，この提案手法はデータパスの設計開発を行っていない．そのため，今後の課題として，データパスの設計とハードウェア規模についての考察と評価が必要である．

また，今回の研究で提案した動き探索処理向けの拡張命令セットは最小値抽出命令のみである．そこで，さらなる高速化を図るために，提案手法である8点スクエアサーチに適した高速な動き探索処理用命令の制定と評価が必要である．

謝辞

本研究を遂行するにあたり，日頃からご指導，ご助言をいただきました近藤利夫教授，大野和彦講師，佐々木敬泰助教に感謝いたします．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方々に感謝の意を表します.

参考文献

[1] ITU-T(2011/6) 「H.264」，http://www.itu.int/rec/T-REC-H.264/e

[2] 大久保榮・角野眞也・菊池義浩・鈴木輝彦 (2006)「改訂版 H.264/AVC 教科書」

[3] Intel(2011/12) 「Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture」，253665-041US

[4] Intel(2011/12) 「Intel 64 and IA-32 Architectures Software Developer's Manual Combined Volume 2A,2B, and 2C: Instruciton Set Reference, A-Z」，325383-041US

[5] Intel(2011/4) 「インテル 64 アーキテクチャおよび IA-32 アーキテクチャー最適化リファレンス・マニュアル」,248966-024JA

[6] 中村佳史 (2011) 「動き探索処理向け高並列 SAD 演算命令に関する研究」

A　高並列SAD演算命令

高並列SAD演算命令は当研究室が提案した命令で，スクエアサーチにおける3x3の9点分のSAD値演算を並列に処理することができる．この命令は4x1のSAD演算を36個並列に処理を行い，36個の4x1のブロックサイズのSAD演算結果を累算器で累算する．その結果得られたブロックサイズ4x4のSAD値から包含するブロックサイズのSAD値を生成する．

可変ブロックサイズには，16画素幅と8画素幅がある．この命令は16画素幅のSAD演算を並列に行う場合，1ライン分のデータを処理する．一方，8画素幅のSAD演算のSAD演算を並列に行う場合，2ライン分のデータを処理することができる．そのため，576bit幅のレジスタが必要となる．これは2のべき乗である現行の汎用プロセッサのデータパス向けではない．そこで当研究では，この高並列SAD演算命令をより効率良く使用できるようレジスタの削減を行った．

当研究では，16画素幅のデータに対し，8画素幅のデータ2回分として処理することで，576bit幅のレジスタを288bit幅のレジスタに低減した．また，従来の9点スクエアサーチの代わりに提案した8点スクエアサーチを使用することで，2のべき乗である256bit幅のレジスタにすることができる．16画素幅と8画素幅に対しての処理とレジスタを図1.17

に示す.

高並列 SAD 演算命令には 256bit 幅のレジスタを 37 個，576bit 幅のレジスタを 6 個必要である．しかし，この値は各可変ブロックサイズの SAD 演算コードを分かりやすく示すための値となっていた．そのため，レジスタの再利用は考慮されていない．そこで，レジスタの再利用性を高めるために可変ブロックサイズを生成する際、加算に使用したレジスタの SAD 値を加算により生成された SAD 値，または HPMPSADW 命令で生成された SAD 値で置き換えを行う．レジスタの再利用を表した図を図 1.18 に示す.

これにより，128bit 幅のレジスタを 3 個と，256bit 幅のレジスタを 7 個にまで削減可能であることを示した．しかし，この値も SAD 演算を高効率かつ円滑に行うための理想の値であり，様々な処理のために使用されるレジスタは考慮されていないため，最適なレジスタの数については今後さらに検討していくことが必要である.

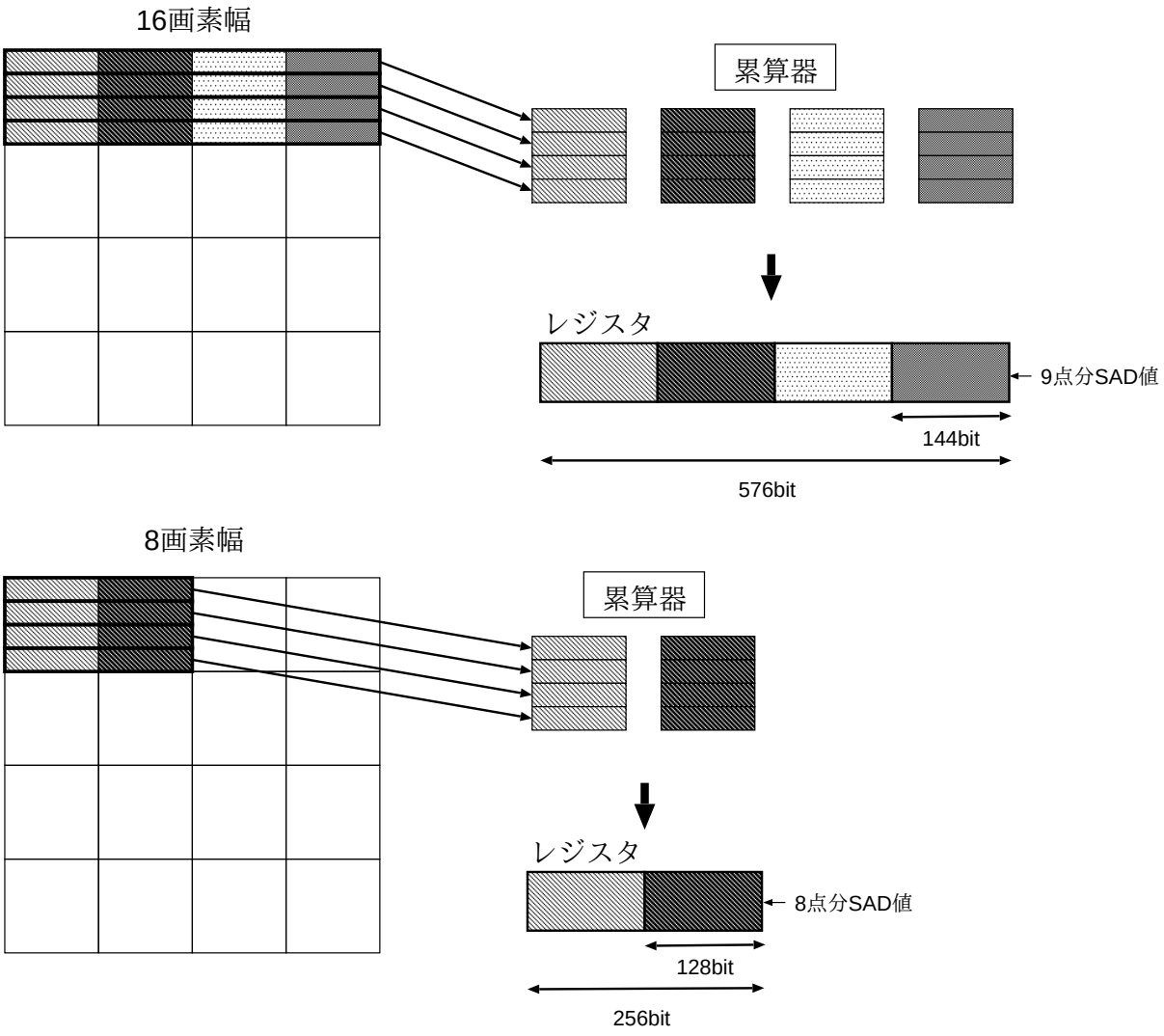

図 1.17: 高並列 SAD 演算命令による処理の詳細

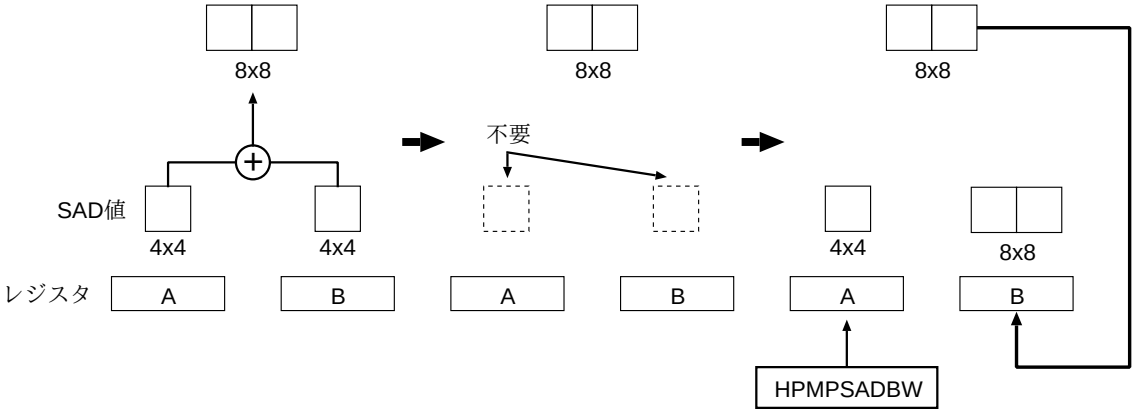

図 1.18: レジスタの再利用