修士論文

題目

動き探索処理向け
高並列SAD演算命令に関する研究

指導教員

近藤 利夫 教授

平成 23年度

三重大学大学院 工学研究科 情報工学専攻
計算機アーキテクチャ研究室

中村 佳史 (410M520)

内容梗概

近年，動画像の高精細化が進んでいる．この高精細化によって増大する動画像のデータ量を圧縮する符号化は，方式の高度化も加わり処理量が大幅に増大している．動き探索処理は，その符号化処理の大半を占めているため高速化の研究が古くから行われてきた．しかし，汎用プロセッサに組み込まれている差分絶対値和 (SAD) 演算命令は，x86 プロセッサの MPSADBW 命令以降，進歩が止まっており，H.264/AVC 符号化に対応できていないためソフトウェア処理の高速化の障害となっている．そこで，本論文では H.264/AVC の可変ブロックサイズにも対応できる高並列 SAD 演算命令を実現することにより動き探索処理の高速化を図る．

動き探索では，符号化対象画像と参照画像との間でブロックマッチングを行っており，このブロックマッチングのために SAD 演算を行う．x86 プロセッサには，複数の SAD 演算命令として MPSADBW 命令がある．しかし，この命令は水平方向の SAD 演算に限られており，現在ソフトウェア処理の基本的な動き探索処理である追跡型の動き探索を効率良く実行できない欠点がある．当研究室でデータ再利用性の高さから多用している 3x3 のスクエアパターンを用いる追跡型探索においても，一度に 3 点分しか並列処理できない．そこで，この問題点の解決に向けて，本論文ではスクエアパターンの 9 点の SAD 演算を一度に並列処理できる高並列 SAD 演算命令セットを提案し，その有効性を評価した．さらに，この提案する命令セットを実行する回路構成について設計を行った．この提案する高並列 SAD 演算命令セットの利用により動き探索処理を高速化することが可能である．

x86 プロセッサの MPSADBW 命令と提案する高並列 SAD 演算命令セットとの間で 9 点の SAD 演算に必要なサイクル数と高速化率の評価を行った．その結果，提案する高並列 SAD 演算命令セットを使用することによって約 4.6 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 recent years, the high definition of video image has made progress. The encoding for compressing an increasing number of data volumes of video image by this high definition progresses the sophistication of method and is greatly increasing the throughput. Because the motion estimation processing occupies most of the encode processing, the speeding up was being studied since early times. But the SAD operation instruction which embedded on the general purpose processor stopped advance since the MPSADBW instruction of x86 processor and is an obstacle the speeding up of software processing to don't correspond H.264/AVC encoding. Therefore, in this paper, I speed up the motion estimation by the realization of the highly parallel SAD operation instruction that is able to correspond with any the variable block sizes of H.264/AVC.

The motion estimation does the block matching between the current picture and the reference picture, and calculates the SAD for this block matching. X86 processor has the MPSADBW instruction as the instruction of multiple SAD operations. But this instruction is limited to the horizontal SAD operations and have disadvantages that can't efficiently execute the motion estimation of tracking type that is the basic motion estimation of software processing at the moment. It can parallelize only three points at a time even if it used the estimation of tracking type which uses the square pattern of 3x3 which is using for high degree of data reuse in this laboratory. Hence, in order to solve this problem, in this paper, I proposed the highly parallel SAD operation instruction set that is able to parallelize the SAD operations of nine points for the square pattern at a time, and evaluated its effectivity. In addition, I designed the circuit structure which executes this proposed instruction set. It is able to speed up the motion estimation by using this instruction set.

I evaluated the number of cycles that required to the SAD operations of nine points and the rate of speeding up between the MPSADBW instruction of x86 processor and the proposed highly parallel SAD operation instruction set. As a result, the performance of processing speed improved about 4.6 times and it was sped up the motion estimation.

目 次

1 まえがき **1**
1.1 研究背景 1
1.2 研究目的 1

2 動き探索処理と並列 SAD 演算命令に対する要求 **2**
2.1 動き探索 2
2.2 スクエアサーチ 2
2.3 差分絶対値和演算 4
2.4 可変ブロックサイズ対応の差分絶対値和演算 5
2.5 MPSADBW 命令の問題点 7
2.6 並列 SAD 演算命令に対する要求条件 8

3 提案高並列拡張命令 **9**
3.1 拡張命令セットの仕様 9
3.1.1 高並列 SAD 演算命令 10
3.1.2 入力命令 11
3.1.3 出力命令 11
3.1.4 加算及び比較命令 11
3.1.5 可変ブロックサイズ SAD 演算コード 12
3.2 回路構成 21
3.2.1 高並列 SAD 演算器構成 21
3.2.2 加算器及び比較器複合器構成 24

4 評価 **25**

5 あとがき **31**

謝辞 **31**

参考文献 **32**

A 評価用動画作成 (mjpegtools 使用) **33**
A.1 mjpegtools のインストール 33
A.2 動画作成 33

B x264 使用方法 **34**
B.1 インストール 34
B.2 実行 35

C yasm インストール **35**

図 目 次

2.1 スクエアパターンでのブロックマッチング 3
2.2 SAD 計算例 4
2.3 可変ブロックサイズ 5
2.4 4x4 ブロックサイズの組み合わせ 6
2.5 MPSADBW 命令による SAD 演算 7
2.6 MPSADBW 命令による 8 個の SAD 演算詳細 8
3.7 レジスタ構成 13
3.8 16x16 の SAD 演算コード 14
3.9 16x8 の SAD 演算コード 15
3.10 8x16 の SAD 演算コード 17
3.11 8x8 の SAD 演算コード 17
3.12 8x4 の SAD 演算コード 18
3.13 16 画素幅における各可変ブロックサイズの SAD 値生成の流れ 19
3.14 8 画素幅における各可変ブロックサイズの SAD 値生成の流れ 20
3.15 高並列 SAD 演算器の構成 22
3.16 演算器ユニット及びサブユニットの構成 23
3.17 加算器及び比較器の複合器構成 25
4.18 スクエアパターンの SAD 演算量 26
4.19 スクエアパターンの高速化率 27
4.20 HD 画像における SAD 演算量 28
4.21 4Kx2K 画像における SAD 演算量 29
4.22 UHD 画像における SAD 演算量 29

表 目 次

2.1 MPSADBW 命令の使用による評価結果 9
4.2 スクエアパターンにおける 9 点の SAD 演算に必要なサイクル数 26
4.3 提案手法の SAD 演算量 30
4.4 提案手法の高速化率 30

1　まえがき

1.1　研究背景

現在，動画像はハイビジョン (High Definition：HD) が一般的に使用されている．しかし，次世代の方式としてハイビジョンの 16 倍の画素数を有するスーパーハイビジョン (Ultra High Definition：UHD) の試験放送が 2020 年に予定されており，近年このスーパーハイビジョンに向けて動画像の高精細化が進んでいる．これに伴い，符号化のために要求される処理量が増大している．また，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．H.264/AVC [1,2] の動き探索では，より精度の高い探索処理を行うために 7 種類の可変ブロックサイズを用いて探索をしており，これらの可変ブロックサイズの使用によってさらに動き探索の処理量が増大している．そこで，符号化処理にかかる時間を短縮するためには処理の大半を占める動き探索処理を高速化することが必要である．

1.2　研究目的

符号化処理の大半を占める動き探索処理では，ブロックマッチングが行われている．このブロックマッチングでは，２つの画像内にあるブロック間の類似度を調べるために差分絶対値和 (SAD) 演算が行われる．SAD 演算は動き探索の主処理であり，複数の SAD 演算を高効率かつ高速に処理することが動き探索処理を高速化する上で必要になる．高効率の動き探索法の 1 つである追跡型のスクエアパターン探索 (スクエアサーチ) では，3x3 点のブロックマッチング (SAD 演算) を繰り返す．x86 プロセッサには SSE4(Streaming SIMD Extensions 4) と呼ばれる SIMD(Single Instruction stream Multiple Data stream) 命令があり，現在その 1 つに SAD 演算命令として MPSADBW(Multiple Packed Sums of Absolute Difference Byte Word) 命令 [3–5] がある．この MPSADBW 命令は，水平方向の複数の SAD を並列に演算することが可能な命令である．しかし，MPSADBW 命令は水平方向の複数の SAD を並列に演算することに限られており，スクエアサーチにおいて必要な 3x3 点の SAD 演算を並列に処理することが不可能である．そのため，3x3 点の SAD 演算を並列に処理することができる命令を新たに追加することによって，動き探索処理をより高速に行うことが可能になる．そこで，3x3 点の SAD 演算を並列に処理することができる高並列 SAD 演算命令の実現を目指す．

2　動き探索処理と並列SAD演算命令に対する要求

2.1　動き探索

動き探索では，符号化対象画像と参照画像の 2 枚の画像を用いて，符号化対象画像内に写っている物体の動きについて参照画像を使用してブロック単位で探索を行う．このブロック単位での探索ではブロックマッチングが行われており，ブロックマッチングを繰り返しながら探索が進められる．ブロックマッチングは，符号化対象画像のブロックと参照画像のブロックの類似度を調べるために使用されている．2 つのブロックにおける類似度を計算する方法として差分絶対値和 (SAD) があり，SAD は動き探索処理を行うために使用される．そのため，動き探索での主な処理は SAD を計算することになる．

動き探索には，ソフトウエア処理向上の高効率探索法として追跡型の動き探索がある．追跡型の動き探索は，予測ベクトルをもとに参照画像上に開始点を決定し，その開始点から決められた探索範囲のブロックマッチングを必要に応じて探索範囲を移動させながら繰り返し行い，最も類似度の高いブロックを検出する方法である．この追跡型の動き探索法の 1 つにスクエアサーチがある．

2.2　スクエアサーチ

スクエアサーチは，追跡型の動き探索法の 1 つに含まれる手法である．このスクエアサーチでは，探索パターンとして 3x3 の正方形パターン (スクエアパターン) を使用しており，このスクエアパターンを用いて動き探索を行う．スクエアパターンにおけるブロックマッチングを図 2.1 に示す．図 2.1 でブロックマッチングを行うブロックのサイズは，4x4 である．

スクエアサーチにおけるブロックマッチングでは，符号化対象画像のブロックを参照画像の 9 個のブロックとそれぞれのブロックについてブロックマッチングを行う．参照画像での 9 個のブロックは，探索中心のブロックとその周囲の 8 個のブロックからなる 3x3 のブロックである．

スクエアパターンは，ダイヤモンドやヘキサゴンなどの他パターンの動き探索法と比較するとデータの再利用の面で最も優れており，データの再利用によって読み込み (ロード) のオーバーヘッドを低減することが

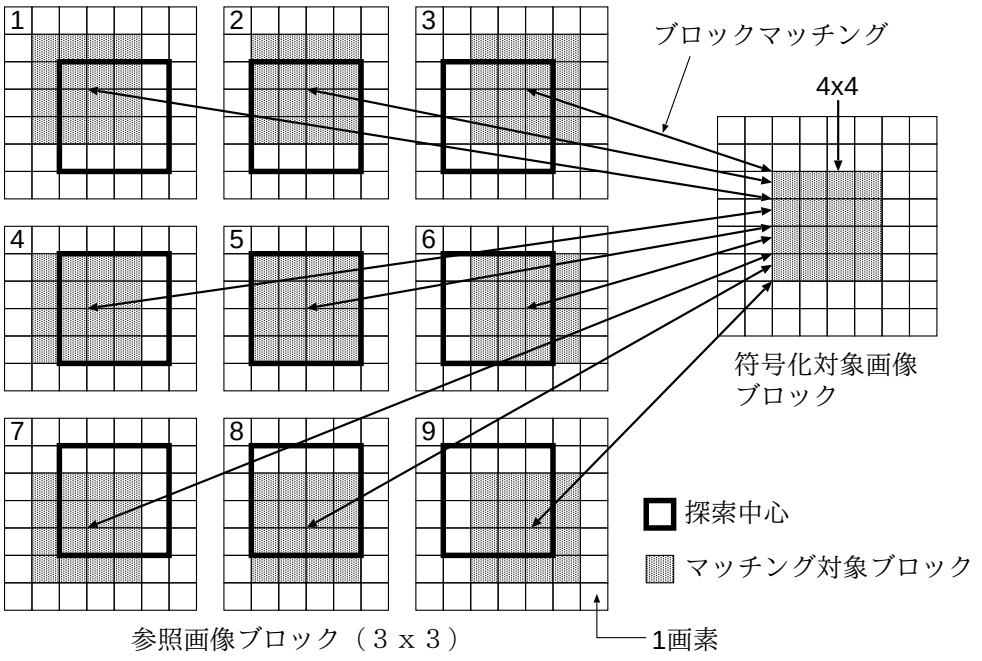


図 2.1: スクエアパターンでのブロックマッチング

可能である．また，スクエアサーチは当研究室で提案している拡張テンプレート法と組み合わせることによって参照画像の高再利用率と EPZS に匹敵する探索精度を実現することができる．従来の方法では単一のブロック単位でブロックマッチングを行っているが，拡張テンプレート法では隣接するブロックを組み合わせることによって拡張したブロック単位でブロックマッチングを行う．この隣接するブロックを組み合わせることで，拡張テンプレート法は探索の精度を向上させ，さらに演算量も低減させており，高精度かつ低演算量の動き探索処理を行うことを可能にしている．EPZS は，周囲の動きベクトルと前後のフレームから予測動きベクトルを作成することで不要な範囲の探索を減らし，効率良く探索を行うことができる方法である．しかし，探索精度を維持するために多数の候補が必要となり，再利用率が大きく低下する問題がある．そこで，スクエアサーチと拡張テンプレート法を組み合わせることによって高精度かつ高再利用率の動き探索処理を高速に行うことが可能である．

2.3 差分絶対値和演算

ブロックマッチングでは，2つのブロックについて比較を行うために類似度を調べる必要がある．そこで，2つのブロック間の類似度を計算する方法として差分絶対値和 (Sum of Absolute Differences：SAD) がある．符号化対象画像のブロックをC，参照画像のブロックをRとして，MxNのブロックサイズにおけるブロックCとブロックR間のSADを求めるための計算式を式1に示す．

$$SAD(C,R) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} |C_{xy} - R_{xy}| \tag{1}$$

また，M=4，N=4とした4x4のブロックサイズにおけるSADの計算例を図2.2に示す．

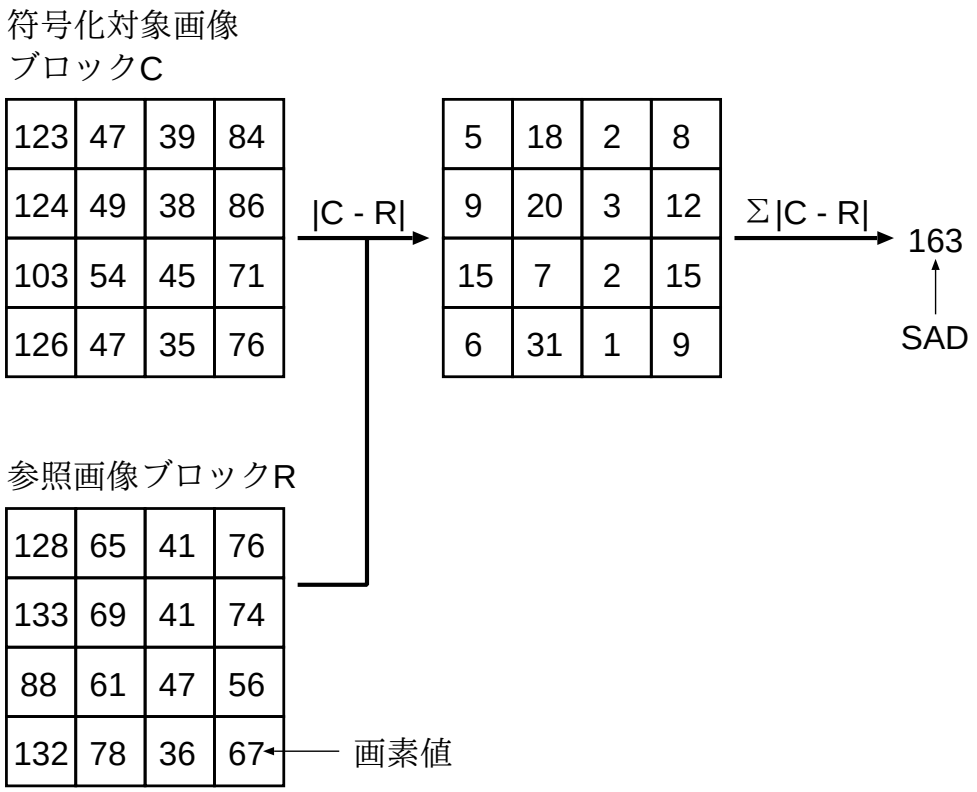


図 2.2: SAD計算例

SADを計算した結果，SADの値が小さければ小さいほど，2つのブロッ

クにおける類似度は高くなる．逆に，大きければ大きいほど，類似度は小さくなる．このSAD演算は動き探索処理の内部に存在し，符号化対象画像のブロックと参照画像のブロックの類似度を調べるために使用されている．

2.4 可変ブロックサイズ対応の差分絶対値和演算

現在主流である符号化圧縮規格のH.264/AVCでは，動き探索処理において可変ブロックサイズが使用されている．この可変ブロックサイズは，全部で7種類あり，16x16，16x8，8x16，8x8，8x4，4x8，4x4のブロックサイズとなる．7種類の可変ブロックサイズを図2.3に示す．

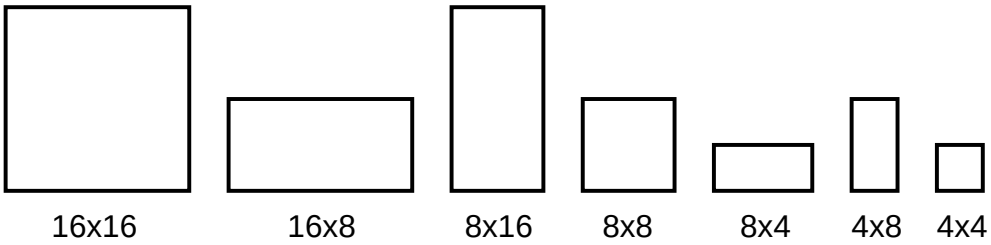


図 2.3: 可変ブロックサイズ

16x16，16x8，8x16，8x8，8x4，4x8の6種類のブロックサイズは，7種類の中で最小のブロックサイズである4x4のブロックを複数組み合わせることによって生成することが可能である．4x4のブロックサイズの組み合わせから他のブロックサイズを生成するのを表したものを図2.4に示す．

従来の動き探索処理では，7種類の可変ブロックサイズの中からブロックサイズを1個選択する．そして，このブロックサイズのSADを演算し，探索処理を行う．しかし，当研究室では，16x16，16x8の16画素幅のブロックサイズと8x16，8x8，8x4の8画素幅のブロックサイズを用いた動き探索を行うときに，そのブロックサイズが包含するブロックサイズについても同時に探索処理を行う．この方法では，他のブロックサイズのSAD演算も包含して同時に行うので，演算量を低減することができる．また，4x8，4x4の4画素幅のブロックサイズについては，個別に探索を行わず，包含された状態で動き探索処理を行う．4x4のブロックサイズを単位サイズとし，この4x4のブロックサイズのSAD値を演算する．これ

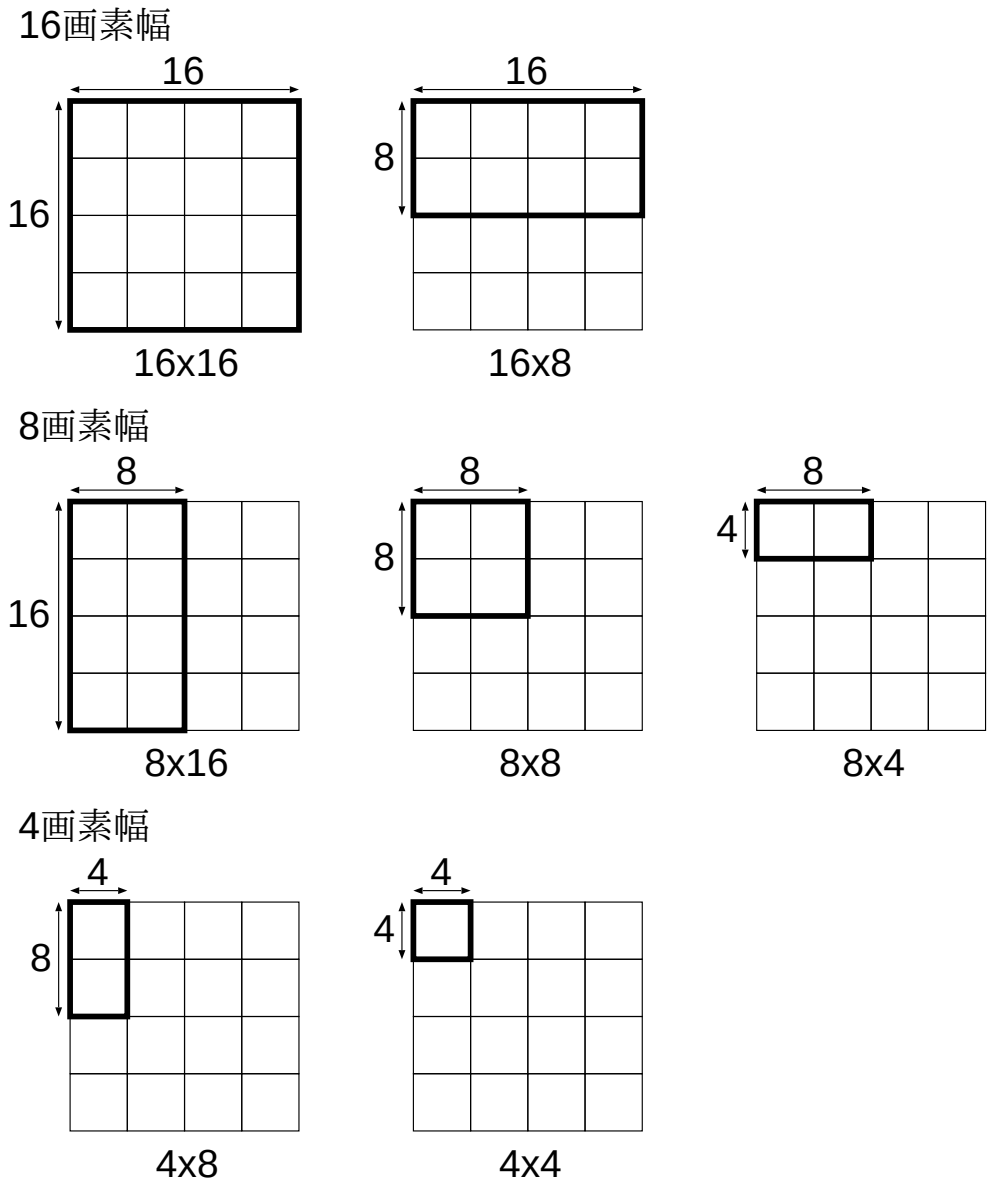


図 2.4: 4x4 ブロックサイズの組み合わせ

によって求められた 4x4 の SAD 値を足し合わせることによって必要なブロックサイズの SAD 値を計算する．最大のブロックサイズである 16x16 のブロックで動き探索を行うときには，4x4 のブロックの SAD 値を 16 個演算することが必要になる．この求められた 16 個の SAD 値を使用して 16x16 などのブロックサイズの SAD 値を求める．

4x4 のブロックサイズを単位サイズとすることで，H.264/AVC の動き探索において使用される可変ブロックサイズの SAD 演算に対応することが可能である．

2.5 MPSADBW 命令の問題点

MPSADBW 命令は，x86 プロセッサに搭載されている SIMD 型の命令である SSE4 命令の 1 つである．この MPSADBW 命令は 2 オペランド形式の命令であり，4 画素幅の SAD 演算を水平方向の 8 個分並列に処理することができる．第 1 オペランドに与えられたレジスタからは，SAD 演算に使用される 4 画素幅の画素値データが 8 個取り出される．この 8 個のデータは 1 画素分ずつずらされたデータである．第 2 オペランドに与えられたレジスタからは，SAD 演算に使用される 4 画素幅の画素値データが 1 個取り出され，同一のデータが 8 個の SAD 演算において使用される．SAD 演算によって求められた結果は，第 1 オペランドで指定されたレジスタに格納される．MPSADBW 命令による SAD 演算では，動き探索において水平方向に並んでいる探索点の SAD 演算を並列に処理することを行っている．MPSADBW 命令を使用した SAD 演算を図 2.5 に示す．また，MPSADBW 命令で行われる 8 個の SAD 演算の詳細なデータを表したものを図 2.6 に示す．

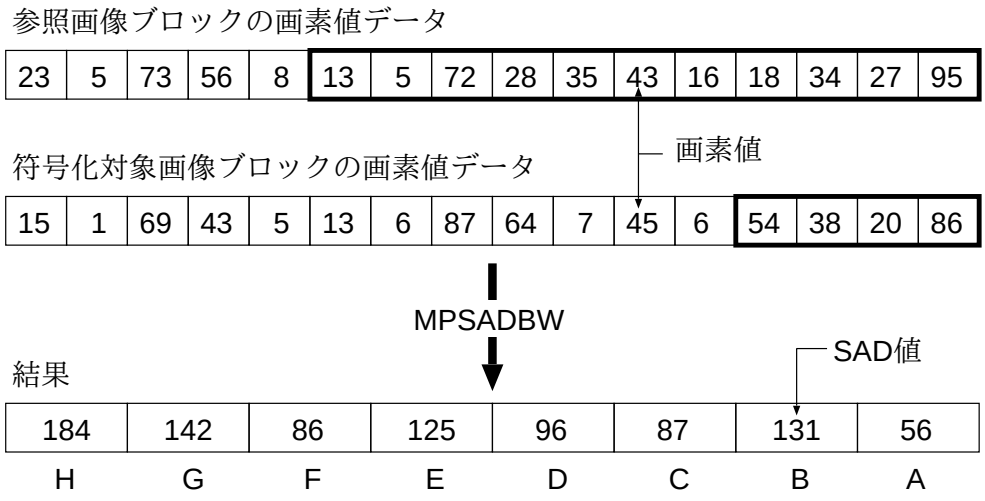


図 2.5: MPSADBW 命令による SAD 演算

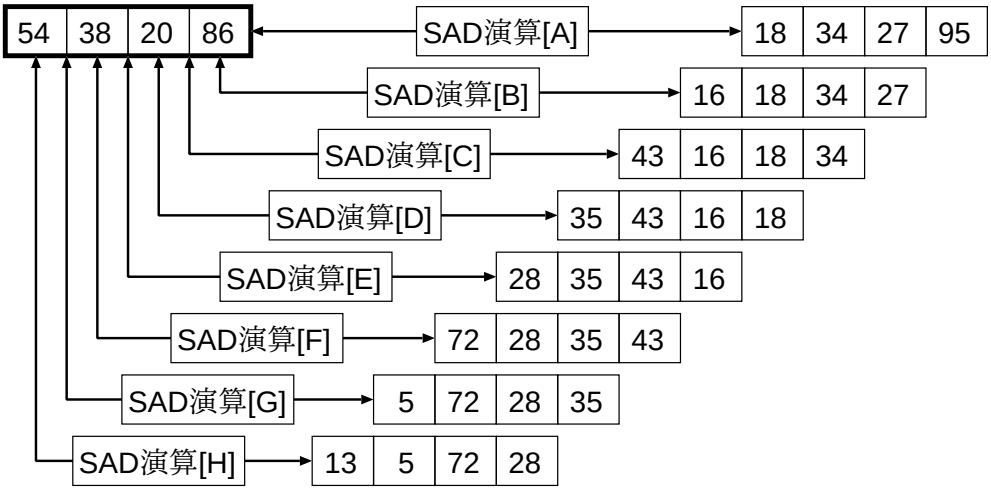


図 2.6: MPSADBW 命令による 8 個の SAD 演算詳細

MPSADBW 命令は水平方向の 8 個の SAD 演算を並列に処理する命令であるため，動き探索処理においては水平方向の SAD 演算を並列に処理することに限られる．しかし，スクエアサーチでは 3x3 の SAD 演算を並列に処理することが要求されている．そのため，MPSADBW 命令の使用による問題点はスクエアサーチで必要とされる垂直方向の SAD 演算を並列に処理することが不可能なところである．

2.6 並列 SAD 演算命令に対する要求条件

スクエアサーチに MPSADBW 命令を使用して評価を行った結果を表 2.1 に示す．この評価は，以下の 3 つの場合それぞれについて行った．

- MPSADBW 命令を使用せずにスクエアパターン (3x3) で探索を行った場合
- MPSADBW 命令を使用してスクエアパターン (3x3) で探索を行った場合
- MPSADBW 命令を使用してレクタングルパターン (5x3) で探索を行った場合

レクタングルパターン探索は，横 5x 縦 3 の長方形パターンで探索を行う方法である．高速化率は，スクエアサーチにおいて MPSADBW 命令を使用せずに動き探索処理を行った場合を 1 として正規化を行っている．

表 2.1: MPSADBW 命令の使用による評価結果

	サーチパターン [WxH]		
	スクエア [3x3]	スクエア [3x3]	レクタングル [5x3]
MPSADBW 命令使用の有無	無し	有り	有り
高速化率 [倍]	1	1.08	1.18

その結果，MPSADBW 命令の使用により約 1.18 倍の速度向上が確認されたが，動き探索処理の高速化には不十分である．そのため，さらに動き探索処理を高速化する必要がある．そこで，スクエアサーチにおいて必要となる 9 点分のブロックマッチングを行う並列化を水平方向の並列化だけではなく，垂直方向の並列化も併せて行う必要がある．

水平・垂直の両方向を合わせた 9 個の SAD 演算を並列に処理するには，現在ある命令では処理することが不可能であるので，水平・垂直の両方向に対応した新たな命令を追加することが要求される．この要求を実現するためには，水平・垂直両方向の 9 個の SAD 演算を並列に処理することができる高並列 SAD 演算命令が必要になる．

3 提案高並列拡張命令

3.1 拡張命令セットの仕様

スクエアサーチにおいて必要とされる 9 点の SAD 演算及び可変ブロックサイズの SAD 値の計算を行うことができるようにするための拡張命令セットとして以下の命令を提案する．

- 高並列 SAD 演算命令 (Highly Parallel Multiple Packed Sums of Absolute Difference Byte Word : HPMPSADBW) : 3x3 の 9 点の SAD 演算を並列に処理する命令

- 入力命令 (Move Input : MovIn) : 高並列 SAD 演算器にデータを入力する命令
- 出力命令 (Move Output : MovOut) : 高並列 SAD 演算器内の累算器に格納されているデータを出力する命令
- 加算及び比較命令 (Add and Compare : AddCom) : 高並列 SAD 演算命令によって求められた SAD 結果から各可変ブロックサイズの SAD 値を計算するための加算命令及び最小の SAD 値を検出するための比較命令を組み合わせた複合命令

3.1.1 高並列 SAD 演算命令

高並列 SAD 演算 (Highly Parallel Multiple Packed Sums of Absolute Difference Byte Word : HPMPSADBW) 命令は，スクエアサーチにおいて必要とされる 3x3 の 9 点分の SAD 演算を並列に処理することができる命令である．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．

可変ブロックサイズには，16 画素幅と 8 画素幅がある．16 画素幅の SAD 演算を並列に行う命令では，1 ライン分のデータを処理する．一方，8 画素幅の SAD 演算を並列に行う命令では，2 ライン分のデータを処理することができる．

高並列 SAD 演算命令の命令名は hpmpsadbw であり，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 オペランドには，符号化対象画像のデータが格納されたレジスタ (r1) を指定する．第 2 オペランドには，参照画像のデータが格納されたレジスタ (r2) を指定する．第 3 オペランドには，0 または 1 のどちらかの値 (i) を指定する．この値 (i) が 0 のときには，16 画素幅のブロックサイズの SAD 演算を行う．また，1 のときには，8 画素幅のブロックサイズの SAD 演算を行う．

```
hpmpsadbw r1, r2, i
```

この hpmpsadbw 命令を後に示す addcom 命令と併せて使用した場合，それぞれの命令における処理を同時に実行することが可能である．つまり，hpmpsadbw 命令は addcom 命令と同時実行可能な命令である．

3.1.2 入力命令

入力 (Move Input : MovIn) 命令は，高並列 SAD 演算器にデータを入力するための命令である．高並列 SAD 演算を行うためには，符号化対象画像のデータが 1 ライン分と参照画像のデータが 3 ライン分必要である．しかし，高並列 SAD 演算命令によって入力されるデータは，符号化対象画像と参照画像共に 1 ライン分のデータである．そのため，SAD 演算を行うには参照画像のデータが 2 ライン分不足することになる．そこで，この入力命令を使用して SAD 演算器に参照画像の 1 ライン分のデータを入力する．この命令によって，スクエアパターンによる 9 点の SAD 演算を行うために必要なデータの準備を行う．

入力命令の命令名は movin であり，この命令のフォーマットは 1 オペランド形式である．第 1 オペランドには，高並列 SAD 演算器へ入力するデータが格納されたレジスタ (r1) を指定する．

```
movin r1
```

3.1.3 出力命令

出力 (Move Output : MovOut) 命令は，高並列 SAD 演算命令の処理によって得られた結果が格納されている累算器内のレジスタから各可変ブロックサイズの SAD 値を計算するために使用される専用レジスタへデータの転送を行い，累算器内のレジスタを 0 に初期化する命令である．

出力命令の命令名は movout であり，この命令のフォーマットは 1 オペランド形式である．第 1 オペランドには，累算器内のレジスタに格納されたデータを転送する先の専用レジスタ (r1) を指定する．演算器ユニットにある累算器内のレジスタは，データ転送後に 0 で初期化される．

```
movout r1
```

3.1.4 加算及び比較命令

加算及び比較 (Add and Compare : AddCom) 命令では，加算処理と最小の SAD 値を検出するための比較処理の 2 つの処理が行われる複合命令である．加算の処理では，入力で与えられた 2 つのレジスタ内に格納された SAD 値データの加算をそれぞれの対応する位置どうしで行う．そして，指定されたレジスタに結果を格納する．比較の処理では，最小の

SAD 値を検出するために比較処理を行う．2 つのレジスタからデータが送られてくるので，それぞれのレジスタにおいてレジスタに格納されたデータから比較によって最小値のデータを検出する．この検出された最小値は，指定されたレジスタに格納される．

加算及び比較命令の命令名は addcom であり，この命令のフォーマットは 4 オペランド形式である．第 1 オペランドには，加算によって求められた結果を格納する専用レジスタ (r1) を指定する．第 2，3 オペランドには，加算演算及び比較処理のために必要なデータが格納された専用レジスタ (r2，r3) を指定する．第 4 オペランドには，比較処理よって検出された最小の SAD 値のデータを格納する汎用レジスタ (r4) を指定する．

```
addcom r1, r2, r3, r4
```

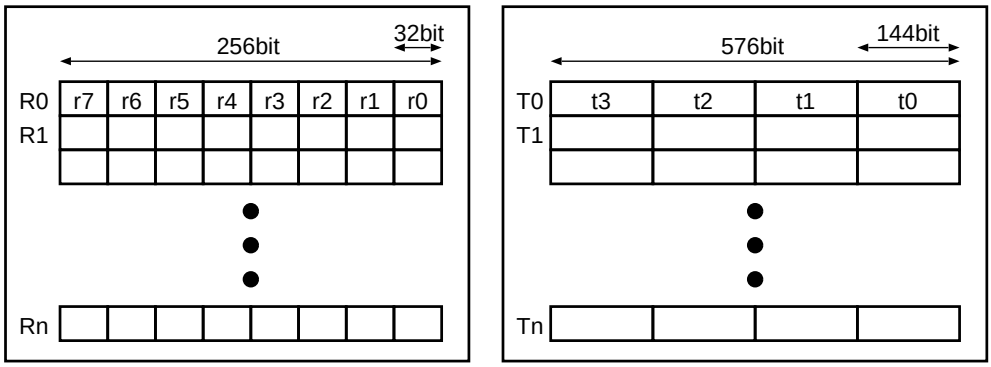
この addcom 命令を先に示した hpmpsadbw 命令と併せて使用した場合，それぞれの命令における処理を同時に実行することが可能である．つまり，addcom 命令は hpmpsadbw 命令と同時実行可能な命令である．

3.1.5 可変ブロックサイズ SAD 演算コード

提案する拡張命令セットを用いてスクエアパターンにおける 9 点の SAD 演算を行うためのコードを以下に示す．ここで示すコードは，動き探索処理での可変ブロックサイズの SAD 演算に当研究室で行っている包含されるブロックサイズの SAD 演算も同時に行う方法で SAD 値を求めるためのコードとなる．さらに，hpmpsadbw 命令と addcom 命令は同時に実行することが可能であるので，これについても考慮を行ったコードである．レジスタには，256bit 幅の汎用レジスタ R0〜R36 と 576bit 幅の専用レジスタ T0〜T5 の 2 種類がある．汎用レジスタは，32bit 幅のレジスタ 8 個として扱うことができ，r0〜r295 がある．専用レジスタは，144bit 幅のレジスタ 4 個として扱うことができ，t0〜t19 がある．レジスタの構成を図 3.7 に示す．符号化対象画像のデータは汎用レジスタ R0〜R15 に，参照画像のデータは汎用レジスタ R16〜R33 に格納されているとする．ここで示したレジスタ数は，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おり，レジスタが最適な数であるとは限らないものである．最適なレジスタ数は，できるだけ少ない数のレジスタにおいて SAD 演算のために使用される画像データの置き換えや求められた各可変ブロックサイズの SAD 値データの格納などを行うために高

効率かつ円滑に使用することができるだけの数が存在する状態でのレジスタ数である．様々な処理のために使用される汎用レジスタや専用レジスタの最適な数については，今後検討していくことが必要な課題である．

16x16 のブロックサイズにおける SAD 演算のコードを図 3.8 に，16x8 のブロックサイズにおける SAD 演算のコードを図 3.9 に，8x16 のブロックサイズにおける SAD 演算のコードを図 3.10 に，8x8 のブロックサイズにおける SAD 演算のコードを図 3.11 に，8x4 のブロックサイズにおける SAD 演算のコードを図 3.12 に示す．



汎用レジスタ　　専用レジスタ

図 3.7: レジスタ構成

```
 1 movin R16
 2 movin R17
 3 hpmpsadbw R0, R18, 0
 4 hpmpsadbw R1, R19, 0
 5 hpmpsadbw R2, R20, 0
 6 hpmpsadbw R3, R21, 0
 7 movout T0
 8 hpmpsadbw R4, R22, 0      # 8，9 行目同時実行
 9 addcom t4, t3, t2, r272
10 hpmpsadbw R5, R23, 0      # 10，11 行目同時実行
11 addcom t5, t1, t0, r272
```

```
12 hpmpsadbw R6, R24, 0
13 hpmpsadbw R7, R25, 0
14 movout T0
15 hpmpsadbw R8, R26, 0      # 15, 16 行目同時実行
16 addcom t6, t3, t2, r272
17 hpmpsadbw R9, R27, 0      # 17, 18 行目同時実行
18 addcom t7, t1, t0, r272
19 hpmpsadbw R10, R28, 0     # 19, 20 行目同時実行
20 addcom t4, t4, t6, r272
21 hpmpsadbw R11, R29, 0     # 21, 22 行目同時実行
22 addcom t5, t5, t7, r272
23 movout T0
24 hpmpsadbw R12, R30, 0     # 24, 25 行目同時実行
25 addcom t6, t3, t2, r272
26 hpmpsadbw R13, R31, 0     # 26, 27 行目同時実行
27 addcom t7, t1, t0, r272
28 hpmpsadbw R14, R32, 0     # 28, 29 行目同時実行
29 addcom t8, t4, t5, r272
30 hpmpsadbw R15, R33, 0
31 movout T0
32 addcom t3, t3, t2, r273
33 addcom t1, t1, t0, r273
34 addcom t6, t6, t3, r273
35 addcom t7, t7, t1, r273
36 addcom t9, t6, t7, r273
37 addcom t10, t4, t6, r274
38 addcom t11, t5, t7, r274
39 addcom t12, t8, t9, r274
40 addcom t13, t12, t12, r275
41 addcom t14, t10, t11, r276
```

図 3.8: 16x16 の SAD 演算コード

```
 1 movin R16
 2 movin R17
 3 hpmpsadbw R0, R18, 0
 4 hpmpsadbw R1, R19, 0
 5 hpmpsadbw R2, R20, 0
 6 hpmpsadbw R3, R21, 0
 7 movout T0
 8 hpmpsadbw R4, R22, 0      # 8, 9行目同時実行
 9 addcom t8, t3, t2, r272
10 hpmpsadbw R5, R23, 0      # 10, 11行目同時実行
11 addcom t9, t1, t0, r273
12 hpmpsadbw R6, R24, 0
13 hpmpsadbw R7, R25, 0
14 movout T1
15 addcom t10, t7, t6, r274
16 addcom t11, t5, t4, r275
17 addcom t12, t8, t10, r276
18 addcom t13, t9, t11, r277
19 addcom t14, t12, t13, r278
20 addcom t15, t3, t7, r279
21 addcom t16, t2, t6, r279
22 addcom t17, t1, t5, r279
23 addcom t18, t0, t4, r279
24 addcom t19, t14, t14, r279
25 addcom t19, t15, t16, r280
26 addcom t19, t17, t18, r281
```

図 3.9: 16x8 の SAD 演算コード

```
 1 movin R16
 2 movin R17
 3 hpmpsadbw R0, R18, 1
 4 hpmpsadbw R1, R20, 1
 5 movout T0
 6 hpmpsadbw R4, R22, 1      # 6, 7行目同時実行
 7 addcom t4, t3, t1, r272
 8 hpmpsadbw R5, R24, 1      # 8, 9行目同時実行
 9 addcom t5, t2, t0, r273
10 movout T0
11 hpmpsadbw R8, R26, 1      # 11, 12行目同時実行
12 addcom t6, t3, t1, r274
13 hpmpsadbw R9, R28, 1      # 13, 14行目同時実行
14 addcom t7, t2, t0, r275
15 movout T0
16 hpmpsadbw R12, R30, 1     # 16, 17行目同時実行
17 addcom t8, t3, t1, r276
18 hpmpsadbw R13, R32, 1     # 18, 19行目同時実行
19 addcom t9, t2, t0, r277
20 movout T0
21 addcom t3, t3, t1, r278
22 addcom t2, t2, t0, r279
23 addcom t0, t4, t5, r280
24 addcom t1, t6, t7, r281
25 addcom t10, t8, t9, r282
26 addcom t11, t3, t2, r283
27 addcom t12, t4, t6, r284
28 addcom t13, t5, t7, r284
29 addcom t14, t8, t3, r284
30 addcom t15, t9, t2, r284
31 addcom t16, t0, t1, r284
32 addcom t17, t10, t11, r285
33 addcom t18, t16, t17, r286
```

```
34 addcom t19, t18, t18, r288
35 addcom t19, t12, t13, r289
36 addcom t19, t14, t15, r290
```

図 3.10: 8x16 の SAD 演算コード

```
 1 movin R16
 2 movin R17
 3 hpmpsadbw R0, R18, 1
 4 hpmpsadbw R1, R20, 1
 5 movout T0
 6 hpmpsadbw R4, R22, 1       # 6, 7 行目同時実行
 7 addcom t4, t3, t1, r272
 8 hpmpsadbw R5, R24, 1       # 8, 9 行目同時実行
 9 addcom t5, t2, t0, r273
10 movout T0
11 addcom t3, t3, t1, r274
12 addcom t2, t2, t0, r275
13 addcom t0, t4, t5, r276
14 addcom t1, t3, t2, r277
15 addcom t6, t4, t3, r278
16 addcom t7, t5, t2, r278
17 addcom t8, t0, t1, r278
18 addcom t9, t8, t8, r279
19 addcom t9, t6, t7, r280
```

図 3.11: 8x8 の SAD 演算コード

```
1 movin R16
2 movin R17
3 hpmpsadbw R0, R18, 1
4 hpmpsadbw R1, R20, 1
5 movout T0
6 addcom t3, t3, t1, r272
7 addcom t2, t2, t0, r273
8 addcom t1, t3, t2, r274
9 addcom t0, t1, t1, r275
```

図 3.12: 8x4 の SAD 演算コード

各可変ブロックサイズの SAD 値を生成する流れを表したものをブロックサイズが 16 画素幅の場合は図 3.13 に，ブロックサイズが 8 画素幅の場合は図 3.14 に示す．16 画素幅のブロックサイズでは 1 ラインずつ SAD 演算を行い，その結果得られた 4x4 の SAD 値から必要な SAD 値を生成していく．また，8 画素幅のブロックサイズでは 2 ライン分同時に SAD 演算を行い，その結果得られた 4x2 の SAD 値から必要な SAD 値を生成していく．小さいブロックサイズから大きいブロックサイズへと順番に SAD の値を求める．

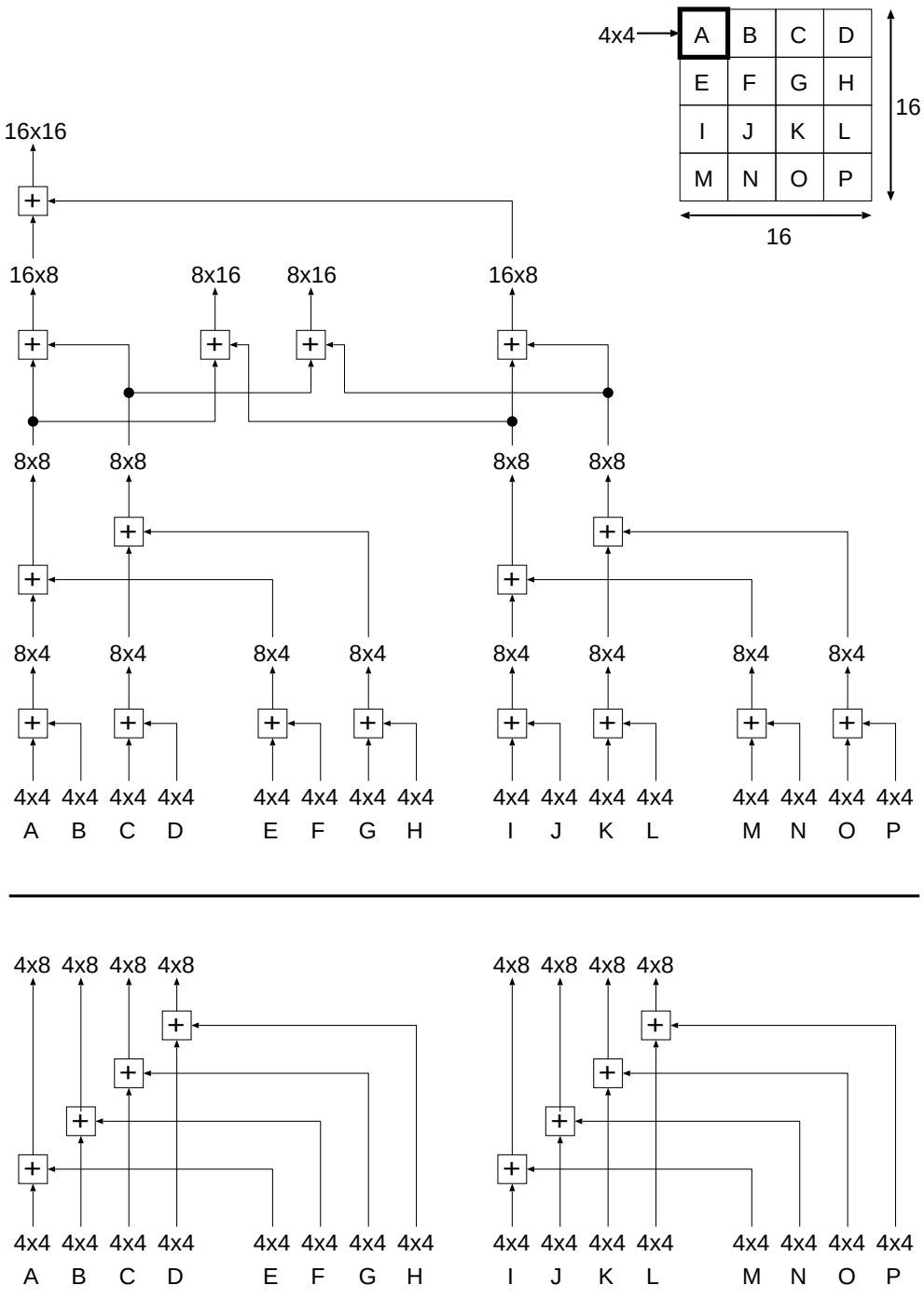


図 3.13: 16 画素幅における各可変ブロックサイズの SAD 値生成の流れ

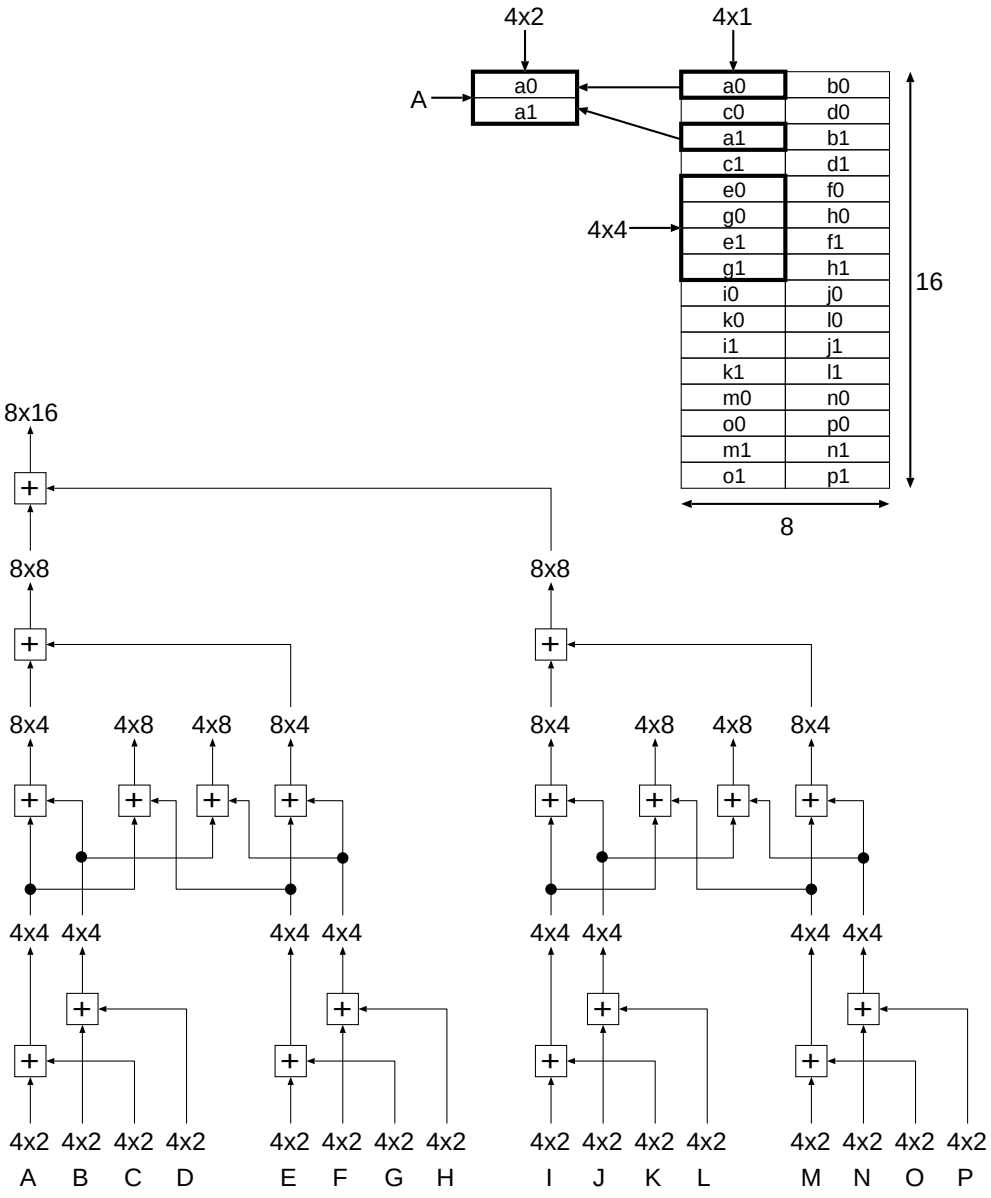


図 3.14: 8 画素幅における各可変ブロックサイズの SAD 値生成の流れ

3.2 回路構成

回路構成は，高並列SAD演算器部分と加算器及び比較器を組み合わせた複合器部分の2つから構成される．また，この2つの部分は別々であるため同時にそれぞれの処理を行うことが可能である．

3.2.1 高並列SAD演算器構成

高並列SAD演算器では，16画素幅のSAD演算を1ライン分，8画素幅のSAD演算を2ライン分並列に処理することができる構成になっている．16画素幅のときは回路全体を使用して処理を行い，8画素幅のときには左右の回路それぞれで1ライン分の処理を行う．この高並列SAD演算器は，SIMD型の回路構成であり，6個の演算器ユニット，4個のレジスタ，4個のマルチプレクサと2個の抽出器から構成されている．

6個の各演算器ユニットには，4x1のSAD演算及びそのSAD値結果の累算を行うための回路としてサブユニットが6個含まれている．

4個のレジスタは，SAD演算のために入力された参照画像の画素値データを格納するために使用される．従来の演算器構成では，SAD演算を行うために毎回データの読み込みが必要であった．さらに，SAD演算によっては前回の演算と同一のデータを使用する場合があり，その場合にも再度データを読み込む必要があった．しかし，演算器構成内にレジスタを組み込むことによって参照画像の画素値データを順送りにすることが可能となり，データの再利用を行うことができる．参照画像のデータを再利用することで同一の画素値データをSAD演算の度に毎回読み込む必要がなくなり，読み込み(ロード)のためのオーバーヘッドを低減することができる．

4個のマルチプレクサは，16画素幅と8画素幅の切り換えを行うために使用される．このマルチプレクサに入力された値が0のときは，16画素幅のブロックサイズの処理を行う．また，入力値が1のときには8画素幅のブロックサイズの処理を行う．

2個の抽出器は，入力されたデータから必要なデータの抽出を行う．この抽出されたデータを使用してSAD演算の処理を行う．

高並列SAD演算器の構成を図3.15に示す．

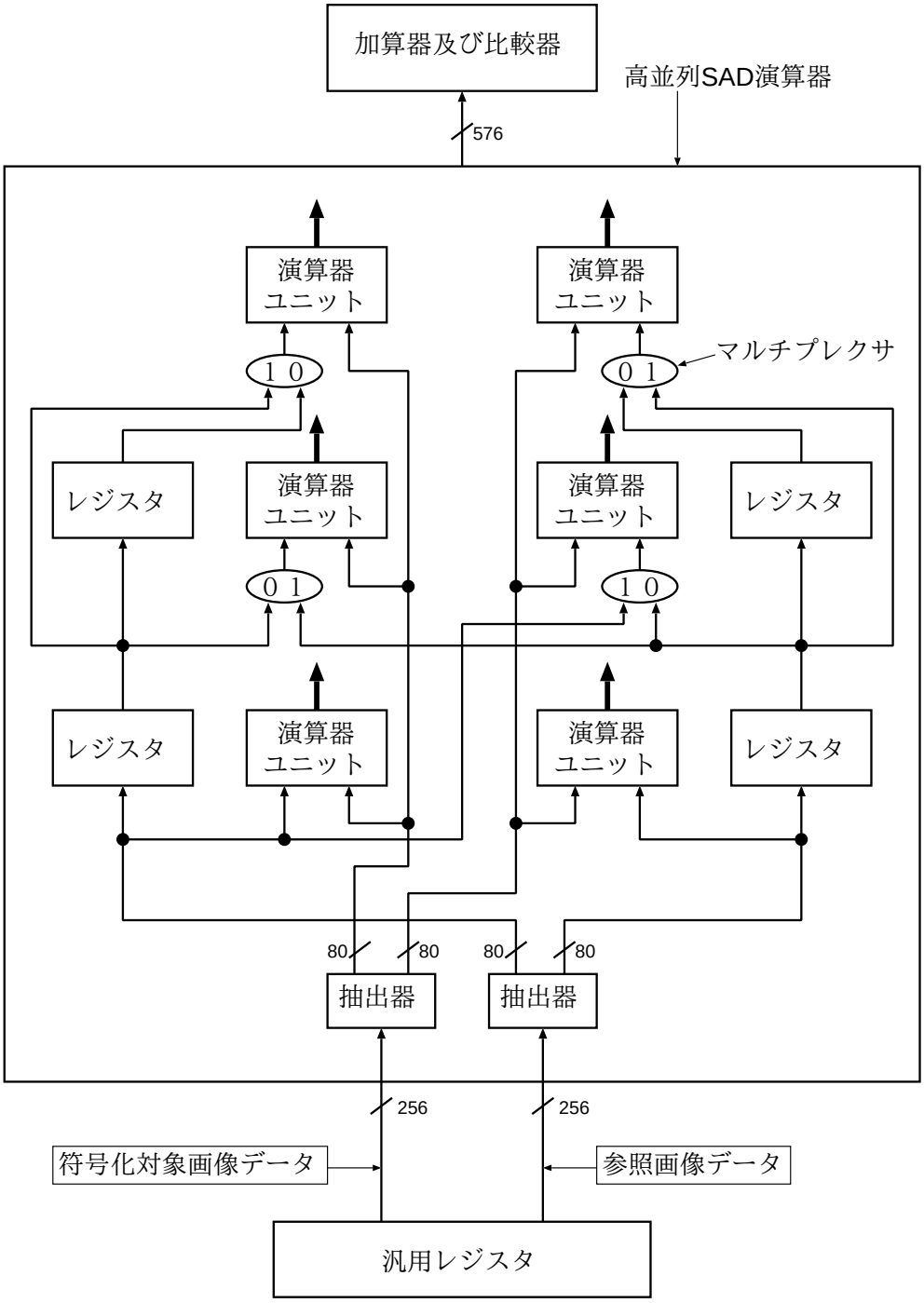


図 3.15: 高並列 SAD 演算器の構成

演算器ユニット内に含まれているサブユニットは，差分絶対値器，加算器及び累算器から構成されている．差分絶対値器と加算器において 4x1 のブロックサイズの SAD を演算する．また，累算器は，加算器とレジスタから構成された回路である．この回路の処理では，回路内にあるレジスタのデータと入力されたデータとの間で累算を行う．累算器では，4x1 の SAD 演算により得られた結果の累算を行い，累算器内のレジスタに累算した SAD 値の結果を格納する．演算器ユニット及びサブユニットの構成を図 3.16 示す.

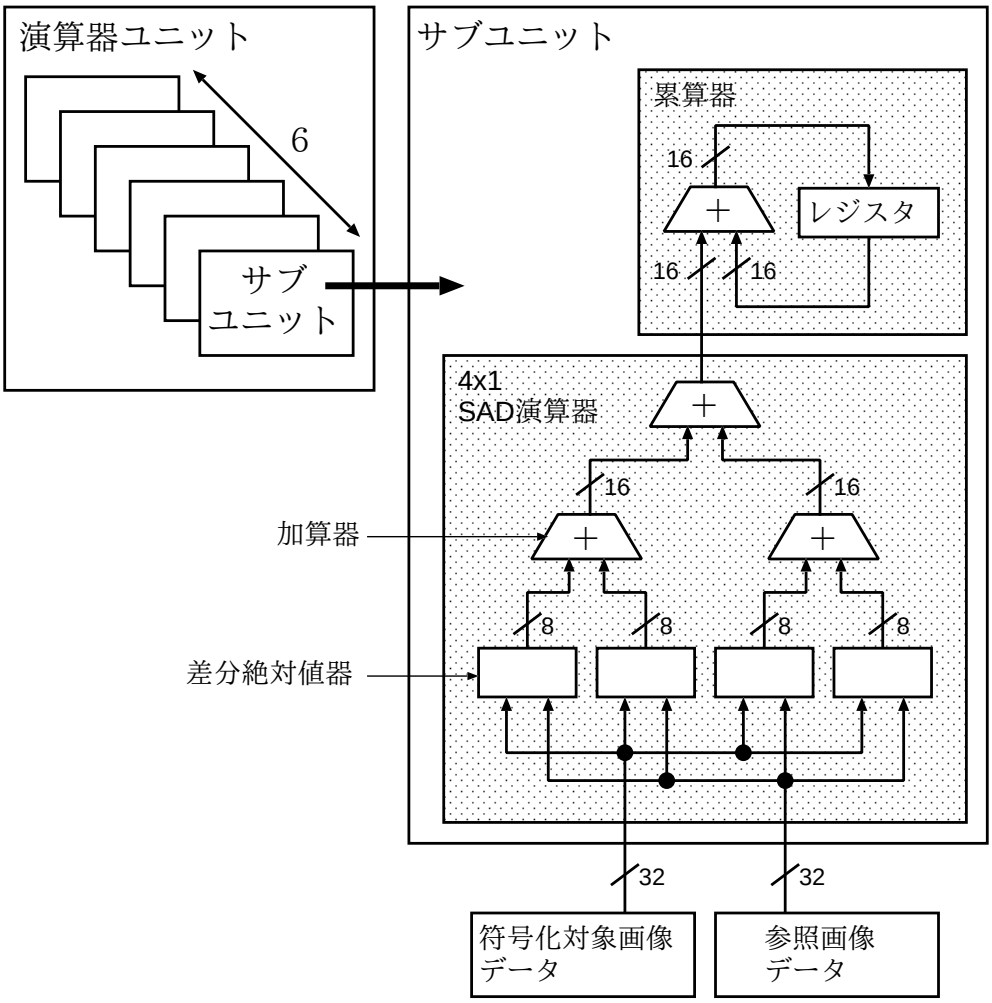


図 3.16: 演算器ユニット及びサブユニットの構成

3.2.2　加算器及び比較器複合器構成

加算器及び比較器の複合器構成は，専用レジスタ，加算器，比較器，結合器とマルチプレクサから構成される．

専用レジスタは，高並列SAD演算器において求められたSAD値のデータやこのSAD値データを使用して求められる可変ブロックサイズのSAD値データを格納するために使用される．この専用レジスタの構成は，図3.7のように576ビット幅のレジスタn本からなり，それぞれは144ビット幅のレジスタとしても使用することが可能である．

加算器は，可変ブロックサイズのSAD値を計算するために使用されるSIMD型の演算器であり，複数の加算を並列に処理することができる．

比較器は，スクエアパターンにおける9点分のSAD値から最小のSAD値を検出するために比較処理を行う．この比較器と先の加算器には，144ビットレジスタから2個のデータが入力され，加算と比較の処理が同時に行われる．

結合器は，比較器において検出された16ビット長の最小SAD値データ2個を繋ぎ合わせて32ビット長にするためのものである．そして，その32ビット長のデータは汎用レジスタの指定された場所に格納される．

マルチプレクサは，高並列SAD演算器から転送されてきたデータと加算器での演算によって得られたデータを専用レジスタへ入力するための切り換えを行う．

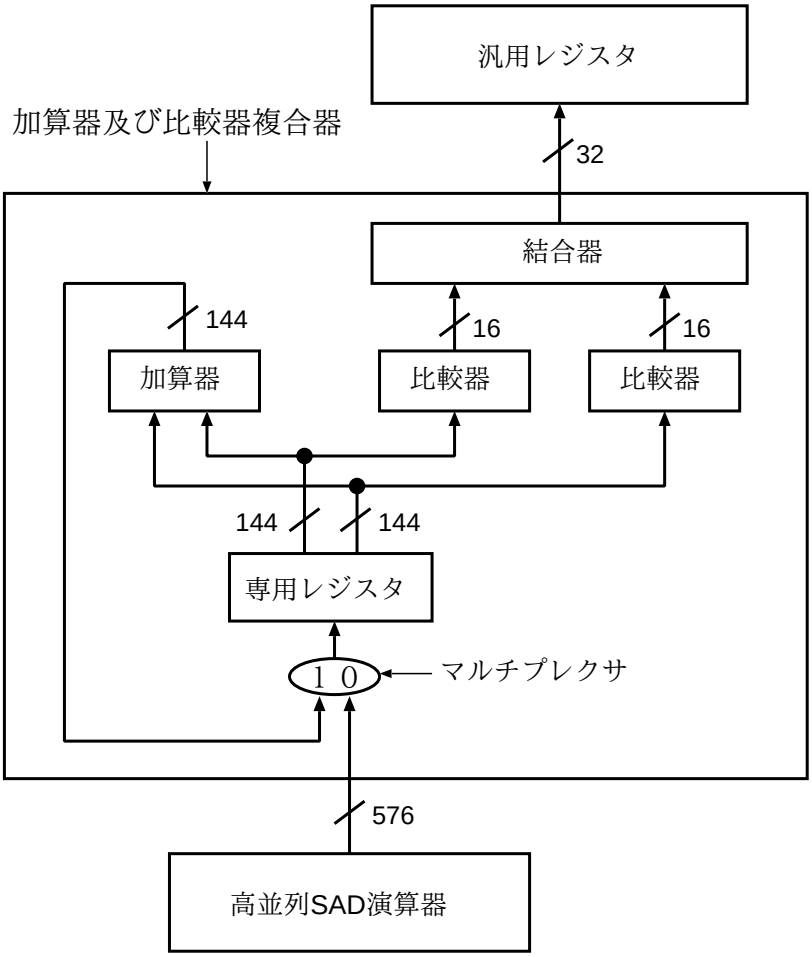


図 3.17: 加算器及び比較器の複合器構成

4 評価

H.264/AVC での可変ブロックサイズを用いた動き探索において評価を行った．動き探索法は，スクエアパターンを用いた追跡型の動き探索である．動き探索処理において使用する可変ブロックサイズの SAD 演算では，当研究室で行っている 16 画素幅と 8 画素幅のブロックサイズにおいて，そのブロックサイズが包含するブロックサイズについても併せて SAD 演算を行う方法で行った．

x86 プロセッサの MPSADBW 命令を使用した場合と提案する拡張命令セットを使用した場合について，スクエアパターンにおける 9 点の SAD

演算に必要なサイクル数を表 4.2 に示す．このサイクル数は，パイプライン処理を考慮して行ったときの値である．MPSADBW は，SAD 演算に MPSADBW 命令を使用して行った場合である．提案手法は，SAD 演算に提案する拡張命令セットを使用して行った場合である．

表 4.2: スクエアパターンにおける 9 点の SAD 演算に必要なサイクル数

	ブロックサイズ [WxH]				
	16x16	16x8	8x16	8x8	8x4
MPSADBW	219	123	123	60	27
提案手法	35	24	32	18	10

提案手法である拡張命令セットを使用したときの 9 点の SAD 演算量を MPSADBW と比較したものを図 4.18 に示す．SAD 演算量は，MPSADBW 命令を使用したときの SAD 演算量を 1 として正規化を行ったときの値である．

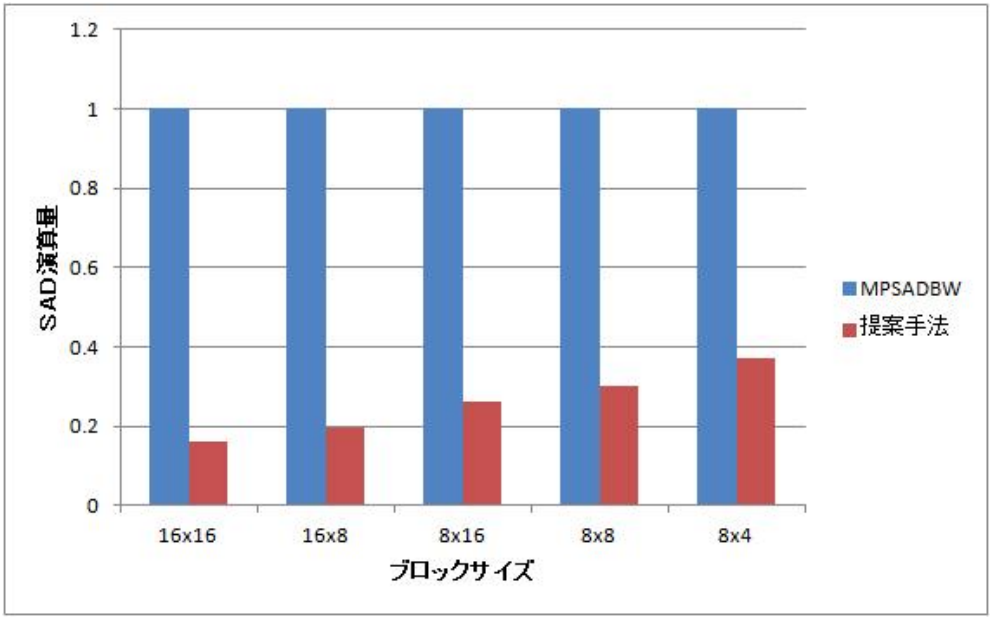


図 4.18: スクエアパターンの SAD 演算量

次に，MPSADBW に対する提案手法の高速化率を表したものを図 4.19

に示す．高速化率は，MPSADBW を 1 として正規化を行ったときの値である．

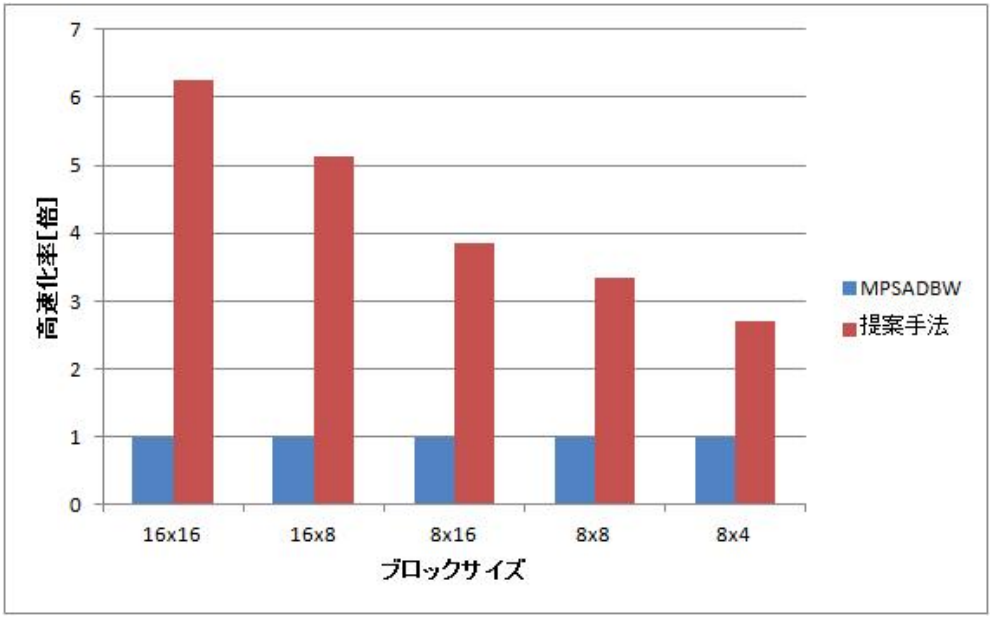


図 4.19: スクエアパターンの高速化率

図 4.18 より，提案する拡張命令セットを使用した場合の 9 点の SAD 演算量は，MPSADBW 命令を使用した場合の演算量に比べ約 26%の演算量となった．図 4.19 より，9 点の SAD 演算の処理速度が提案する拡張命令セットを使用することで，MPSADBW 命令を使用した場合に比べ約 4.3 倍高速化された．

9 点の SAD 演算の高速化率は，小さいブロックサイズほど効果が低いという結果となった．これは，SAD を演算する処理 (hpmpsadbw 命令による処理) と可変ブロックサイズの SAD 値を計算する処理 (addcom 命令による処理) が同時に実行される部分が少なかったためであると考えられる．提案する高並列 SAD 演算器命令の使用頻度が高いほど，可変ブロックサイズの SAD 値を計算する処理をより多く，同時に処理することができるため高速化率が高くなったと考えられる．高速化率の結果より，少なくとも 2.7 倍の高速化率を得ることが可能である．

H.264/AVC の符号化処理を行うためのソフトウェアエンコーダとして x264 がある．x264 は，オープンソースソフトウェアであるため誰でも自由に使用することができるソフトウェアとなっている．本評価では，こ

の x264 ソフトウェアエンコーダを使用して，スクエアパターンを用いた動き探索処理の SAD 演算に必要なサイクル数の評価を行った．この評価に使用した画像は，CrowdRun，DucksTakeOff，OldTownCross の 3 種類であり，フレーム数は 500 フレームである．画像の画質は，ハイビジョン (HD : 1920x1080)，4Kx2K(4Kx2K : 3840x2160)，スーパーハイビジョン (UHD : 7680x4320) の 3 種類である．スーパーハイビジョンに関しては，評価用の画像が無いため 4Kx2K を拡大して作成した擬似的な画像となっている．

MPSADBW と提案手法について，16x16，16x8，8x16，8x8，8x4 のそれぞれの SAD 演算にかかるサイクル数を積み上げたものを HD 画像は図 4.20 に，4Kx2K 画像は図 4.21 に，UHD 画像は図 4.22 に示す．

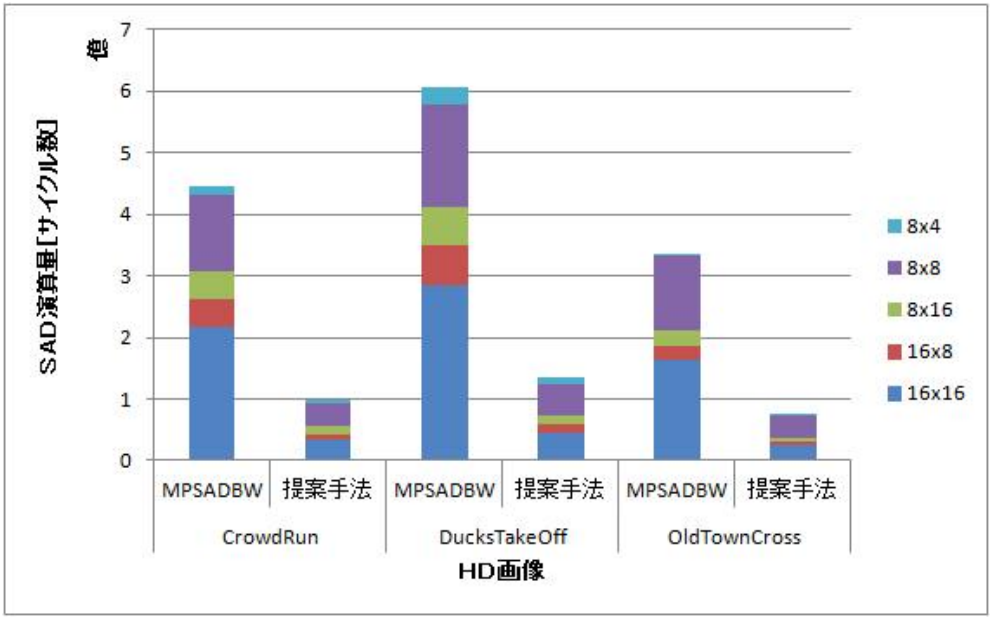


図 4.20: HD 画像における SAD 演算量

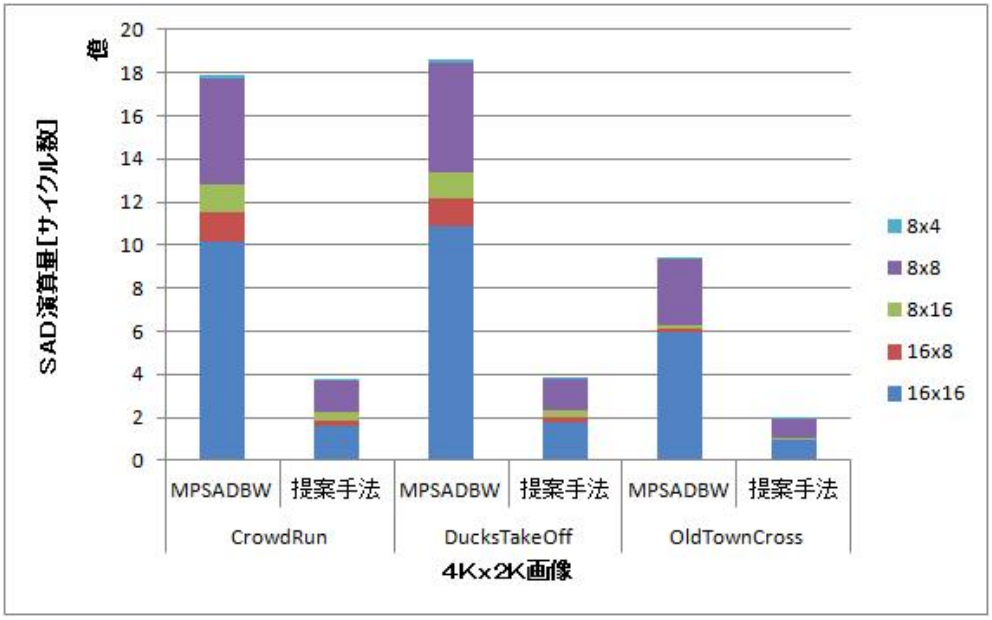


図 4.21: 4Kx2K 画像における SAD 演算量

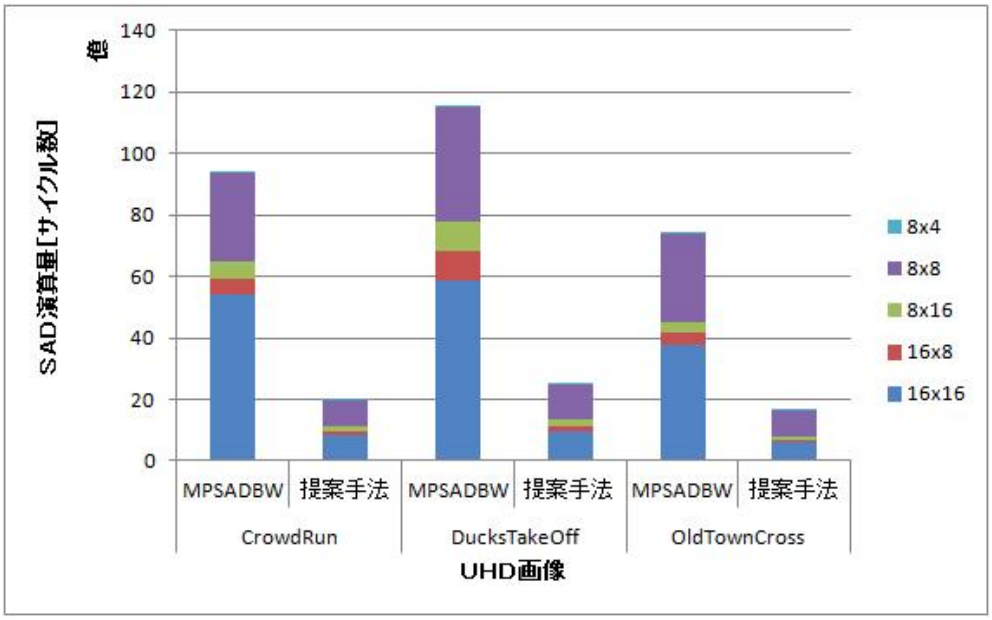


図 4.22: UHD 画像における SAD 演算量

x264 を用いた評価によって得られた結果より，提案手法の SAD 演算量と高速化率をそれぞれ表 4.3 と表 4.4 に示す．表 4.3 と表 4.4 の値は，MPSADBW を 1 として正規化を行ったときの値である．

表 4.3: 提案手法の SAD 演算量

	HD 画像	4Kx2K 画像	UHD 画像	全画像
MPSADBW	1	1	1	1
提案手法	0.22	0.21	0.22	0.22

表 4.4: 提案手法の高速化率

	HD 画像	4Kx2K 画像	UHD 画像	全画像
MPSADBW	1	1	1	1
提案手法	4.52	4.78	4.62	4.64

x264 による評価結果より，提案する拡張命令セットの使用によって SAD 演算量を約 78%低減し，動き探索処理が約 4.6 倍高速化された．動き探索において使用される各可変ブロックサイズの使用量の割合は，画像の画質や種類によって異なり，高速化率の高いブロックサイズがより多く使用されるものほど高速化率が上がる傾向にある．9 点の SAD 演算と可変ブロックサイズの SAD 値計算の処理を同時に処理することができる部分をより多くすることで動き探索処理の高速化をより高めることが可能である．

5　あとがき

本論文では，動画像の符号化処理において最もボトルネックとなっている動き探索処理の高速化を図るために新規に拡張命令セットを提案した．さらに，この提案する拡張命令セットを実行するための SIMD 型の回路構成について設計開発を行った．提案する拡張命令セットは，高いデータ再利用性を有する 3x3 のスクエアパターンを用いた追跡型の動き探索において必要とされる 9 点の SAD 演算を並列に処理することを可能にした．また，H.264/AVC の動き探索によって使用される可変ブロックサイズの SAD 演算を同時に行えるようにした．この提案する拡張命令セットを使用することにより，x86 プロセッサの MPSADBW 命令と比較すると約 4.6 倍処理速度が向上し，動き探索処理の高速化を行うことができた．

今後の課題として，様々な処理を行うために使用される汎用レジスタや専用レジスタの最適なレジスタ数や設計した回路構成のハードウエア規模について検証を行う必要がある．また，SAD 演算の処理が高並列化されたことでメモリから汎用レジスタへのデータ転送速度も重要となり，このデータ転送が SAD 演算処理の妨げにならないようにする必要がある．提案する拡張命令セットでは 8 画素幅の SAD 演算を行うときに 2 ライン分の処理を並列に行うことが可能であるので，メモリやレジスタへのデータのマッピング方法やメモリからのデータ転送の仕方を検討する必要がある．

謝辞

本研究を行うに当たり日頃ご指導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致します．また，日頃お世話になりました計算機アーキテクチャ研究室の皆様に感謝致します．

参考文献

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

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

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

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

[5] Intel,「インテル 64 アーキテクチャーおよび 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，2011/4

A　評価用動画作成 (mjpegtools 使用)

A.1　mjpegtools のインストール

mjpegtools は，x264 で使用する Y4M 形式の動画を作成するために使用するツールである．

- mjpegtools のプログラムファイル (`mjpegtools-[バージョンナンバー].tar.gz`) を以下のサイトよりダウンロードする．
  - http://sourceforge.net/projects/mjpeg/files/mjpegtools/
- ダウンロードした mjpegtools の圧縮ファイルを解凍する．
- 解凍したフォルダへ移動し，インストールのために以下のコマンドを実行する．(x86 プロセッサ搭載マシン [moule など] で実行することを推奨)
  - `./configure --prefix=/home/username/mjpegtools` (オプションでインストール先のフォルダを指定する)
  - `make install` (実行ファイルが指定先のフォルダに作成される)
- /home/username/mjpegtools/bin/内に ppmtoy4m という実行ファイルがあればインストール完了．

A.2　動画作成

- `cat *.ppm > input.txt` (PPM 形式ファイルを 1 つのファイルにまとめる)
- `./ppmtoy4m -o 0 -n 500 -I p -F 25:1 -S 420mpeg2 < input.txt > output.y4m`
- ppmtoy4m のオプションについて (他のオプションについては各自で調べて使用すること)
  - `-o` : スキップするフレーム数
  - `-n` : フレーム数を指定
  - `-F` : フレームレート (fps) を指定

B　x264 使用方法

B.1　インストール

- x264 のプログラムファイル (`last_x264.tar.bz2`) を以下のサイトよりダウンロードする.
  - http://www.videolan.org/developers/x264.html
- ダウンロードした x264 の圧縮ファイルを解凍する.
- 解凍したフォルダへ移動し，インストールのために以下のコマンドを実行する.
  - `./configure --prefix=/home/username/x264` (オプションでインストール先のフォルダを指定することが可能である)
  - `make` (x264 の実行ファイルが作成される)
  - `make install` (x264 の実行ファイルがインストール先のフォルダに作成される)
- /home/username/x264/bin 内に x264 という実行ファイルがあればインストール完了.
- ./configure を実行したときに以下のコメントが出た場合には，最新の yasm をインストールする必要がある．但し，アセンブリコードを使用しない場合には，オプション (`--disable-asm`) を付けて再度 `./configure` コマンドを実行すれば良い.

```
Found yasm 0.7.2.2153  <- 現在のバージョン
Minimum version is yasm-1.0.0  <- 最低限必要なバージョン
If you really want to compile without asm, configure
with --disable-asm.
```

- アセンブリコードを使用する場合には，最新の yasm をインストールし，configure ファイルの一部を変更する．そして，再度 `./configure` コマンドを実行する.
  - 変更前：AS=”yasm”
  - 変更後：AS=” /home/username/yasm/bin/yasm”

B.2 実行

以下のコマンドを実行する．(実行時のオプションについては各自で調べて使用すること)

- `./x264 --psnr --partitions p8x8,p4x4,b8x8,i8x8,i4x4 -o output.264 input.y4m`

C yasmインストール

- yasm のプログラムファイル (`yasm-[バージョンナンバー].tar.gz`) を以下のサイトよりダウンロードする．
  - http://yasm.tortall.net/Download.html 内の [Source .tar.gz]
- ダウンロードした yasm の圧縮ファイルを解凍する．
- 解凍したフォルダへ移動し，インストールのために以下のコマンドを実行する．
  - `./configure --prefix=/home/username/yasm` (オプションでインストール先のフォルダを指定する)
  - `make install` (実行ファイルがインストール先のフォルダに作成される)
- /home/username/yasm/bin/内に yasm という実行ファイルがあればインストール完了．