高位合成技術による動き検出技術のハードウェア化及び改良手法の検討

永井 翔太 †　　神戸 尚志 ‡　　藤田 玄 ‡‡

† 近畿大学大学院 総合理工学研究科
‡ 近畿大学 理工学部 電気電子工学科
‡‡ 大阪電気通信大学 情報通信工学部

概要

現在の動画像は大画面化によりデータ量が膨大である．そのため動画像を圧縮する技術が年々向上してきたが，そこに反比例してエンコードに要する処理時間が増大する問題が発生している．そこでソフトウェア面での高速化や処理のハードウェア化が行われてきたが，符号化効率の低下や設計期間が問題となっている．以上の背景から本研究では高位合成技術を用いて動画像圧縮技術の中心部分である動き検出技術を対象にハードウェア化及びその改良を行った．行った改良は処理の変更，並列化，パイプライン化，さらにソフトウェア上で実現された高速化手法の実装であり，これらの結果から高位合成技術と各高速化手法の有用性を検討する．

A Hardware Implementation of Motion Estimation Technology Using High Level synthesis, and Examination of Improved Technique

Shota Nagai†　　Takashi Kambe‡　　Gen Fujita‡‡

†Graduate School of Science and Engineering Research, Kinki University
‡Department or E & E, Faculty of Science and Engineering, Kinki University
‡‡Faculty of Information and Communication Engineering, Osaka Electro-Communication University

Abstract

Recent videos have a large amount of data for large scale screen display and high quality. Therefore, video coding technology has been improving every year, but the problem is that the processing time increase has occurred. From this background, we propose hardware implementation and its improvement for motion estimation process of H.264/AVC using high level synthesis technology. And we evaluate the usefulness of each design method.

1　はじめに

大画面・高画質指向による 4K・8K テレビの登場で動画像の圧縮符号化の必要性が高まり，それにつれ処理の効率向上が求められている．要求に応じて性能を高めてきたが，対照的に処理時間を増大させている．その対策としてハードウェア実装が行われてきたが，RT レベル設計では設計に時間がかかり，ハードウェア設計・評価が間に合わないという問題がある．一方，ハードウェア記述言語よりも抽象度の高い C 言語記述による設計は，高位合成することにより設計・評価を

早期に実施することを可能としている．本研究ではこの高位合成技術の 1 つである Bach システム [1] を用いて回路設計を行い，合成された回路の検証・改善を実施する．H.264/AVC(以下 H.264) の動き検出技術を対象として多種類の回路設計を行い，処理の改良の有効性について評価する．

2　H.264 動き検出技術

動画像は複数の画像の連続再生から構成されていることを利用して，圧縮対象とする部分に対して画素値が近い箇所を前後の符号化済みピクチャから求めて圧縮が行われる [2]．これは動画像圧縮技術の中心的処理で，全処理時間の過半数を占める．動き検出技術は，16 画素 x16 ラインを 1 つのマクロブロック (MB) とし，現在の符号化対象となる MB が符号化済みの参照ピクチャ中のどの位置に相当するかを探索しその動きベクトルを求める．そのためにまず現在の符号化対象ブロックに対して隣接するブロックの動きベクトル情報は互いに相関が高いことを利用して符号化対象ブロックの左・上・右上の 3 つの動きベクトルを用いて参照ピクチャ中のどの位置の該当するかを予測する．その予測位置から全探索 (Full Search) におけるスパイラルサーチ (図 1(a)) や，ダイヤモンドサーチ (図 1(b)) 等を用いて周囲の参照ピクチャにわたって現在ブロックとの Sum of Absolute Difference(SAD) 計算を行いコスト値を算出し，最小コスト値となる位置を探索する．

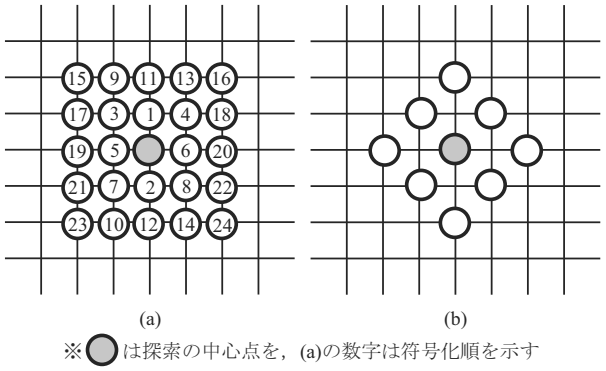

図 1: スパイラルサーチとダイヤモンドサーチの例

3　高位合成に向けた設計手法

本研究において実施した C 言語ベースによる動き検出技術の改良手法を示す．高速化と回路規模の削減を主に行い，本章では Full Search に対して適用する．

3.1　メモリアクセス削減

ハードウェアにおいて処置の高速化のネックとなるのがメモリアクセスである．アクセス時間や通信時間が必要となるため可能な限り使用回数を削減することが望ましい．Full Search では探索の中心点から外側に向かって探索をするスパイラルサーチを行うので，同じ画素値を複数回メモリから読み込む．このアクセス回数を削減するため，図 2 に示すようにサーチを行う順を中心から外へではなく左上から右下への探索に変更し，最上部から 1 ラインづつ，計 16 ライン分をメモリから配列としたレジスタに格納する．必要となるブロック分の画素値をレジスタから読み込み，SAD 計算を行い，必要のなくなったラインから次のラインへと更新させていくことでメモリアクセスの回数を最小限に抑える．メモリから一度レジスタに格納するため回路規模は増加するものの，高速な処理を実現する．

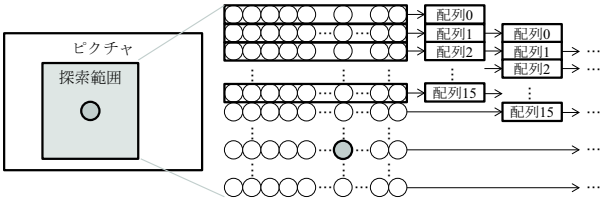

図 2: メモリアクセス削減

3.2　ループ融合

同一回数の複数ループ内で大量の値を関数やスレッドを介してやり取りする場合はメモリを用いる必要が発生する．動き検出技術にも SAD 計算を行った値をマージする処理があるが，図 3 のように逐一メモリ格納・メモリアクセス・マージを Full Search の探索範囲分繰り返すため，その総量は膨大な数である．そこで，図 4 で示す同一ループ内で処理を行うよう変更することでこの問題を解消する．これはループ融合により大容量の値を一度に送る必要がなくなるため，メモリアクセスの頻度を減らすことができる．

3.3　回路規模削減

ハードウェア設計において考えるべき事柄として，回路規模の削減も挙げられる．ビットそのものが回路の規模に影響を及ぼすため，ビット幅を可能な限り最小にする．また高位合成では動作合成を自動で行うため，細かく回路の再利用を設定することは難しく，回

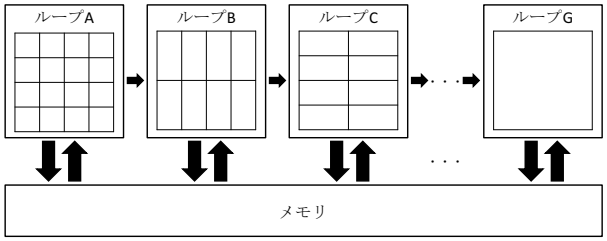

図 3: ループ融合前

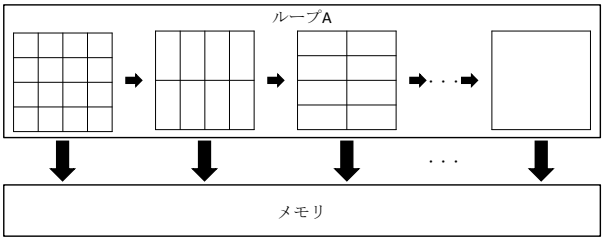

図 4: ループ融合後

路の再利用性の少ない設計を行った場合は，回路規模が増大する．そのため，分岐の有効利用やループ処理等を活用する．特に，ループ処理は第 3.5 節のパイプライン化でも利用することができるため，有効である．H.264 における動き検出技術における C 言語記述は詳細な処理が異なるが，大部分が同一の処理となるコードとなっているため，C ベース設計に適したコードへと変更する．

3.4 並列化

依存関係のない処理やデータを分割し，依存関係を断ち切った処理を並列化することで高速化が可能となる．本研究では 4x4 の画素単位が最小となるブロックのまとまりより，図 5 のように 16 並列化を行うことにより高速化を行う．また，それぞれの 4x4 の画素単位をさらに 16 並列化を行うことも可能である．

3.5 パイプライン化

前節におけるデータ分割による並列化は大規模なデータに対して有効な場合があるが，並列度に応じて回路規模が増大する．こうした場合にはパイプライン化が有効である．本研究では第 3.5 節でのデータ分割による並列化において，16 並列した処理にさらに 16 並列を適用した 256 並列を行っていたが，256 並列では並列前後のオーバーヘッドが大きくなり 16 並列に

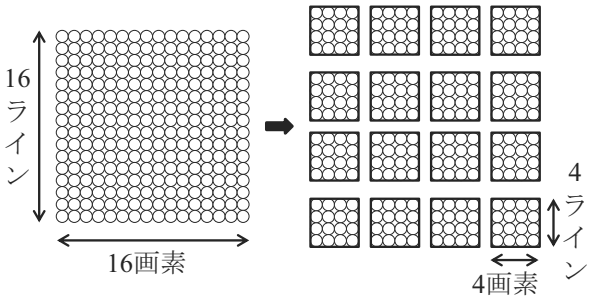

図 5: 16 並列

対して並列化の効果が薄い．そこで，処理速度と回路規模の比率が向上するよう，一方の 16 並列に変わってパイプライン化を適用する．

3.6 構造体の分解

ソフトウェア言語で用いられる構造体を Bach システムでも用いることができるが，関数の引数として使うと，構造体すべての要素が関数に引き渡され，配線が増加すること，要素数に応じて同期通信を行うため，要素数分のクロックサイクルが必要という問題がある．構造体は効率よく設計が行える半面，構造体を多用しすぎてしまうと性能が落ちる．そのため設計期間と性能とのトレードオフを考える必要がある．動き検出技術で用いられている構造体は，主に入力される動画像データや符号化方法の選択といった入力データ群と処理全般に渡って用いられる動画像パラメータ群，及び各処理部で必要とする要素や演算結果を集めた各種要素群である．これらの構造体を分解することにより高速化や回路規模の削減を図る．

4 Enhanced Predictive Zonal Search Algorithm (EPZS) の実装

ソフトウェアにおける動き検出技術の高速化手法は複数提案されている．その内の 1 つである EPZS[3] をハードウェア化する．EPZS 以前に提案された高速化手法である Predictive Motion Vector Field Adaptive Search Technique(PMVFAST)[4] や Advanced Predictive Diamond Zonal Search (APDZS)[5] と比較すると予測精度や閾値の決定方法，及び探索手法に改良が行われ，性能が向上している．

4.1 予測パターンと閾値，探索パターン

EPZS における予測は，サブセットと閾値によって決定される．予測候補とする複数の動きベクトルをサブセットとしてそれぞれサブセット A，サブセット B，サブセット C の 3 つに分けられる．サブセット A に該当するのが符号化対象ブロックの周囲に存在する左，上，右上の 3 つのブロックが持つ動きベクトルから生成した予測動きベクトルの中央値である．この動きベクトルは Full Search を含むすべての動き検出技術に含まれる．次にサブセット B となる候補は，先ほど挙げた符号化対象ブロックの周囲に存在する左，上，右上の動きベクトルに加えた (0，0) の動きベクトルである．このサブセット B までの予測については前述した PMVFAST や APDZS にも用いられている．残るサブセット C は図 6 のように符号化対象ブロックに対し，空間的同位置に存在する参照ブロックの動きベクトルを複数枚用いて定速度で移動した場合の動きベクトルを用いる．また図 7 に示すように参照ブロックの上下左右に位置するブロックの動きベクトルをサブセット C として含むことで，より正確な予測が行える．

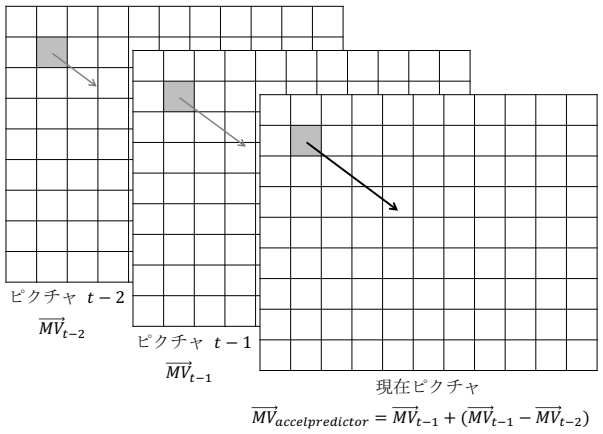

図 6: 移動度を加えた動きベクトル予測

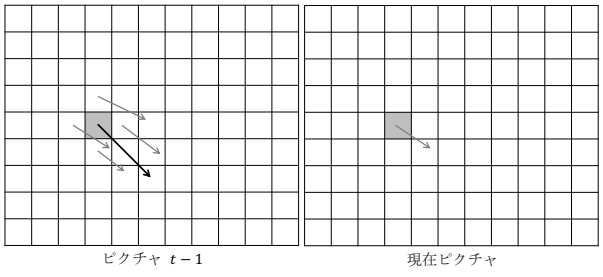

図 7: 参照フレーム周囲の動きベクトルによる予測

閾値の設定では，EPZS 以前の高速化アルゴリズムでは，3 つの隣接ブロックの最小 SAD 値を考慮して算出されていたが，不十分な予測や予測の間違いを防ぐために，それぞれのサブセットすべてにおいて SAD 値のうち最小となる候補を用いている．閾値パラメータは一般的に以下のように表される．

$$T_k = a_k \times min(MSAD_1, MSAD_2, ..., MSAD_n) + b_k \quad (1)$$

なお，ここでの a_k，b_k は定数である．

予測パターンと閾値で予測を決定した後，詳細な探索を行うための探索パターンについても工夫が施されている．複数の探索パターンを組み合わせて最終的な予測位置を決めていた処理を予測精度を向上させたことにより，EPZS ではスモールダイヤモンドサーチのみを行うことで処理時間の短縮を実現している．探索パターンは図 8 のダイヤモンド型 (a) と正方形型 (b) の 2 種類用意しており，状況に応じて使い分ける．

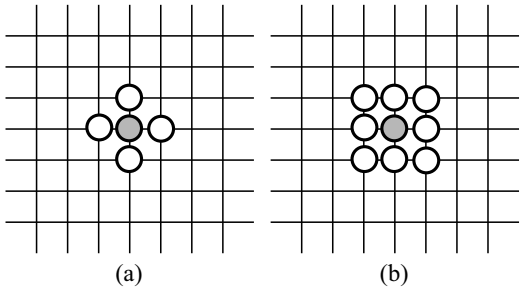

図 8: スモールダイヤモンドサーチ

4.2 EPZS のハードウェア化

EPZS 手法とそのままハードウェア実装を行うとサブセット C の予測が問題となる．第 4.1 節で述べたようにサブセット C は参照ピクチャの空間的同位置に存在する動きベクトル，及びその隣接するブロックの動きベクトルを用いるが，この情報をメモリに保持しておく必要がある．この時用いるメモリの数は参照ピクチャ枚数分で，その数分のメモリアクセスを要することになる．第 3.1 節で述べたようにハードウェア化に際し，メモリアクセスを可能な限り削減したいことから，スパイラルサーチ時の探索範囲からある定点を複数取り，それらの SAD 計算を用いることで代用する．定点の位置は，図 9 のように正方形にとる場合 (a) とダイヤモンド状にとる 2 種類の (b)・(c) の計 3 種類の場合を検討する．ソフトウェアで実装されている EPZS

と比較して予測精度は劣るものの，ハードウェアに適した処理とする．

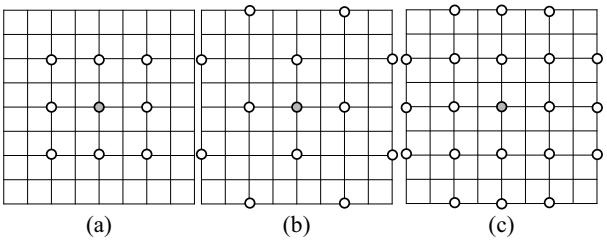

図 9: サブセット C における点の取得位置

またサブセット C における予測精度の低下を防ぐために，サブセット B において符号化対象ブロックの左上に位置するブロックが持つ動きベクトルを予測に用いる．符号化対象ブロックの左・上・右上同様に隣接するブロックであり，すでに符号化が済んでいるため予測を用いることができる．

次に閾値に固定値を用いる方法に変更する．ある一定精度までを満たしている場合，処理を打ち切り，次のダイヤモンドサーチに移行する．この時の固定値はブロックサイズごとで決定しており，本研究では各画素± 2 画素を許容範囲としている．この設定については処理速度を第一としている．

また，すべての閾値を満たさない場合についてはサブセット A〜C までの中で最もコスト値の小さい位置となる動きベクトルを用いる．最後のダイヤモンドサーチについては，処理時間・回路規模を増加させないように図 8(a) に示すスモールダイヤモンドサーチに限定する．第 3.1 節からレジスタ格納と部分メモリアクセスによりメモリの使用回数を減らし処理時間の増加を防ぐ．

5 設計結果とその評価

まず EPZS をハードウェアで実装する際に変更した手法について，BD-Bitrate(BD-BR) を用いて符号化性能を検証した．参照ソフトウェアである JM18.0[6] における Full Search を基準として，JM18.0 で EPZS を使用した場合と，本研究においてサブセット B まで (第 4.2 節での変更有り) の場合，サブセット C において図 9 で示した定点を予測に用いる各場合について測定・評価した．それらの性能評価の結果を表 1 に示す．

表 1 で示した動画像はそれぞれ，foreman.cif･動きの少ない動画，football.cif･動きの激しい動画である．本研究で行った変更についての評価だが，Full Search と比較すると foreman.cif の場合は最大 11.278[%]，football.cif では 8.188[%] の劣化が確認された．劣化問題として考えられるのは閾値の設定が挙げられる．本研究では処理速度を向上させるため閾値を各画素につき ± 2 以内を認めていたが，閾値を満たす条件を厳しくすることである一定の改善が行えると考えられる．JM18.0 の EPZS と性能差を比較するとすべての場合において劣っているものの，動きの激しい動画の場合では動きの少ない動画と比べ劣化が少なく，特にサブセット C における取得点を増加させた効果が表れている．しかし動きの少ない動画ではサブセット C における取得点数を増やすほど劣化が激しくなる問題が発生している．こちらの場合ではサブセット C での予測が無駄となっている，または処理として未確認のバグが発生している可能性がある．そのため今後原因を確認し改善を行う必要がある．

表 1: 本研究において行った変更の性能評価

内容	BD-BR[%] foreman.cif	football.cif
JM18.0 の EPZS	0.278	3.072
サブセット B まで	8.410	8.186
サブセット C 定点数 8	7.837	8.188
サブセット C 定点数 12	11.020	6.632
サブセット C 定点数 20	11.278	6.633

続いて Bach システムを用いて Bach C 言語で設計した回路の動作合成を行ない，RTL シミュレーションを行った．RTL シミュレーションは Mentor Graphics 社の Model Sim を用いてシミュレーションを行いサイクル数を測定，回路規模については VDEC 提供の Synopsys 社の Design Compiler を用いて論理合成し測定した．本設計では動作周波数 100MHz とし，日立 0.18 μ セルライブラリ，Bach version3.6，Design Compiler 2009 を使用する．

ソフトウェア記述のままシーケンシャル処理でハードウェア化した回路をベースに，第 3 章・第 4 章で述べた手法を適用した結果を表 2 に示す．各処理の適用は (g) パイプライン化，(i)EPZS を除き, 各々1 つ上段の結果に，各手法を追加的に適用している．(g) パイプライン化は (e)16 並列に対して適用しており，EPZS でのサブセット C における定点の取得点数は 8 として測定した．また動作条件はオンチップメモリとしてのメモリアクセスが時間 5ns，測定処理時間は 1MB 分の動きベクトルを求める処理を対象とした．

(a) シーケンシャル処理の対して,(h) 構造体分解ま

表 2: 処理速度・回路規模測定結果

	処理時間 [ms]	回路規模 [gates]
(a) シーケンシャル	179.388	1,282,345
(b) メモリアクセス削減	111.288	1,535,288
(c) ループ融合	108.032	1,562,476
(d) 回路規模削減	151.963	763,208
(e) 16 並列	44.184	816,998
(f) 256 並列	38.325	1,908,852
(g) パイプライン化	39.270	790,866
(h) 構造体分解	37.328	705,673
(i) EPZS	6.253	502,069

でを比較すると, 処理時間は約 4.8 倍の高速化，回路規模では約 45.0 %の削減となり，一方の (i)EPZS と (a) シーケンシャルとの比較では処理時間は約 28.7 倍の高速化，回路規模は約 60.8 %削減の結果が得られた．

個々の処理として大きなメリットが得られたのは (a) メモリアクセス削減や (d) 回路規模削減，(e)16 並列である．処理速度に関してはメモリアクセスや多回数のループが，回路規模の面では似た処理に対し複数の回路を生成していたことがネックとなっていた．それらの原因を取り除くように C ベース設計することで改善を行えた．

一方のデメリットとしては (d) 回路規模削減時に見られる処理速度の増加や，(f)256 並列における回路規模の増大である．前者で処理速度が増加したのはループ変数において同一ビット幅の変数とレジスタを共有することで遅延が発生していたと考えられる．後者では回路規模の増加に対して処理の高速化が行われていない．これは並列内の処理量が前処理・後処理等に見合っていないことが挙げられる．よって，これを改善するために (f)256 並列を (g) パイプライン化へと変更することで処理速度は劣るものの，回路面積に対する処理速度は向上している．また (e)16 並列よりも (g) パイプライン化の回路規模が小さいのは，パイプラインを効率的に行うために処理順を変更したためである．

(i)EPZS については，(h) 構造体分解と比較して処理速度・回路規模ともに勝っている．理由としては予測を正確に行うことでメモリからデータを読み込む回数を削減でき，データを大量に保持し続ける必要がなくなったためである．また，本研究における EPZS は，ループ回数を変数となっているため，並列化やパイプライン化を実装しており，更なる高速化を行える余地を残している．

6 まとめと今後の課題

本研究で用いた高速化，回路規模削減双方の手法は一定の効果が得られ，高位合成における有用性を確認できた．また EPZS をハードウェアに適した手法を用いることにより，画質の劣化があるものの高速化を確認した．

今後，画質を維持した状態での高速化を中心により大きな回路規模の削減，さらには H.265[7] への応用を進める．

謝辞

Bach システムを用いたハードウェア設計に当たり，御指導を頂いたシャープ株式会社山田晃久様をはじめ BACH 開発グループの皆様に心から御礼申し上げます．また本研究は，東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社の協力で行われたものである．

参考文献

[1] k.okada, A.Yamada, T.kambe, “Hardware Algorithm Optimization Using Bach C,” IEICE Trans.Fundamentals vol.E85-A No.4 (2002).

[2] 大久保榮, 角野眞也, 菊池義浩, 鈴木輝彦, 改定三版 H.264/AVC 教科書, インプレス R&D，東京，2009.

[3] A.M.Tourapis, ” Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation,” in proceedings of Visual Communications and Image Processing, pp.1069-1079 (2002).

[4] A.M. Tourapis, O.C. Au, and M.L. Liou, ”Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) -Enhancing Block Based Motion Estimation,” in proceedings of Visual Communications and Image Processing 2001 (VCIP-2001), San Jose, CA, pp.883-892 (2001).

[5] A.M. Tourapis, O.C. Au, and M.L. Liou ”New Results on Zonal Based Motion Estimation Algorithms - Advanced Predictive Diamond Zonal Search,” in proceedings of 2001 IEEE International Symposium on Circuits and Systems (ISCAS-2001), Sydney, Australia, vol.5, pp.183-186 (2001).

[6] H264/AVC 参照ソフトウェア jm18.0, `http://iphome.hhi.de/suehring/tml/download/old_jm/`

[7] 村上篤道, 浅井光太郎, 関口俊一, 高効率映像符号化技術 HEVC/H.265 とその応用, オーム社，東京，2013 年 2 月.