計算機アーキテクチャ 150－6
（２００２．１１．２８）

OSCAR型シングルチップマルチプロセッサにおける動きベクトル探索処理のマルチグレイン並列処理

小高　剛[†]　鈴木貴久[†]
木村啓二[†]　笠原博徳[†]

近年の JPEG，MPEG などのマルチメディアコンテンツの増加により，マルチメディアアプリケーションを効率良く処理できる低コスト，低消費電力かつ高性能なプロセッサの開発が望まれている．このようなプロセッサとして，複数のプロセッサコアを搭載したシングルチップマルチプロセッサが命令レベル以外の並列性も自然に引き出すことができ，集積度向上に対しスケーラブルな性能向上が得られるアーキテクチャとして注目されている．本論文では，シングルチップマルチプロセッサ上でのマルチメディアアプリケーション処理の一例として MPEG2，H.263 など動画像処理で行われる動きベクトル探索処理のアルゴリズムの OSCAR 型シングルチップマルチプロセッサ上でのマルチグレイン並列処理におけるプログラムリストラクチャリング法，並列タスク生成手法を提案し，その評価を行なう．

Multigrain Parallel Processing on Motion Vector Estimation for Single Chip Multiprocessor

Takeshi Kodaka[†] , Takahisa Suzuki[†] , Keiji Kimura[†] and Hironori Kasahara [†]

With the recent increase of multimedia contents using JPEG and MPEG, low cost, low power consumption and high performance processors for multimedia application have been expected. Particularly, single chip multiprocessor architectures having simple processor cores that will be able to attain scalability and cost effectiveness are attracting much attention to develop such processors. Single chip multiprocessor architectures allow us to exploit coarse grain task level and loop level parallelism in addition to the instruction level parallelism, so parallel processing technology is indispensable to allow us scalable performance improvement. This paper describes a multigrain parallel processing scheme for motion vector estimation for a single chip multiprocessor and its performance is evaluated.

1　はじめに

JPEG，MPEG などのマルチメディア処理を行なうモバイル情報端末の需要が延びており，マルチメディアアプリケーションを効率良く処理できる低コスト，低消費電力の高性能プロセッサの開発が望まれている．このようなニーズに対応するために CPU ベンダーからは Intel PentiumIII などに搭載されている SSE[1] や，富士通 FR500[3]，日立 SH4[2] などのようにマルチメディア用命令セットを追加し処理能力を向上させる試みが行われている．これらのプロセッサは，スーパースカラアークテクチャや VLIW アーキテクチャによる命令レベル並列性の利用や SIMD 命令によるマルチメディア処理で多用される内積演算やベクトル変換演算など同一命令が連続する処理の高速化により処理能力の向上をねらっている．しかしながら，更なる性能向上を得るにはより多くの並列性を利用する必要がある．

このような状況から，プロセッサコアを複数搭載したシングルチップマルチプロセッサアーキテクチャは命令レベル並列性に加え，ループレベル，サブルーチンレベルなどより多くの並列性を利用することも可能であり，スケーラブルな性能向上が得られるアーキテクチャとして注目を集めている．

シングルチップマルチプロセッサを用いたメディア系アプリケーションの高速化として，NEC MP98[4] は，1 チップ上に 4CPU を搭載しマルチスレッド処理により高速化を行なっており，Stanford 大は，2 次キャッシュ共有型のシングルチップマルチプロセッサ Hydra[5] 上でのメディアアプリケーションの高速化が提案している [6]．また，東芝は，異種プロセッサをワンチップ上に集積した MPEG2 コーデックチップが発表している [7]．

本論文では，シングルチップマルチプロセッサ上でのマルチメディアアプリケーションの処理の一例として MPEG2，H.263 など動画像処理で行われる動きベクトル探索処理のアルゴリズムを解析し，本アルゴリズムの OSCAR 型シングルチップマルチプロセッサをターゲットとしたマルチグレイン並列処理におけるプログラムリストラクチャリング法，並列タスク生成手法を提案し，その評価を行なう．

2　マルチグレイン並列処理

ここでは，OSCAR 型シングルチップマルチプロセッサで扱うマルチグレイン並列処理技術について述べる．マルチグレイン並列処理とは，ループやサブルーチン等の粗粒度タスク間の並列処理を利用する粗粒度タスク並列処理 (マクロデータフロー処

† 早稲田大学電気電子情報工学科
Dept. of Electrical, Electronics and Computer Engineering, Waseda University

理)[8]，ループイタレーションレベルの並列処理である中粒度並列処理，基本ブロック内部のステートメントレベルの並列性を利用する近細粒度並列処理[9]を階層的に組み合わせてプログラム全域に渡る並列処理を行なう手法である．

2.1 粗粒度タスク並列処理[8] (マクロデータフロー処理)

マクロデータフロー処理では，ソースとなるプログラムを疑似代入文ブロック (BPA)，繰り返しブロック (RB)，サブルーチンブロック (SB) の三種類の粗粒度タスク (マクロタスク (MT)) に分割する．ここで，BPA は基本的には通常の基本ブロックであるが，並列性抽出のために単一の基本ブロックを複数に分割したり，逆に複数の基本ブロックを融合して一つの BPA を生成する．MT 生成後，コンパイラは BPA，RB，SB 等の MT 間のコントロールフローとデータ依存を解析しそれらを表したマクロフローグラフ (MFG) を生成する．さらに MFG から MT 間の並列性を最早実行可能条件解析により引きだし，その結果をマクロタスクグラフ (MTG) として表現する．その後，コンパイラは MTG 上の MT をプロセッサあるいは複数のプロセッサエレメント (PE) をひとつのグループとしたプロセッサグループ (PG) に割り当てる．なお，このグループはプログラム中の並列性に応じソフトウェア的に形成される仮想的なものでハードウェア的なグループ化とは異なる．

2.2 中粒度並列処理 (ループ並列処理)

PG に割り当てられた MT が Doall 可能な RB である場合，この RB は PG 内のプロセッサエレメント (PE) 上で，イタレーションレベル並列実行される．

2.3 近細粒度並列処理[9]

PG に割り当てられた MT が，BPA や中粒度並列処理あるいはループボディ部に粗粒度並列処理を適用できない RB である場合，それらはステートメントレベルのタスクに分割され，PG 内の PE により並列処理される．

近細粒度並列処理においては，基本的に BPA 内のステートメント，もしくは IF-THEN-ELSE 等で囲まれた複数ステートメントから構成される疑似代入文を一つの近細粒度タスクとして定義する．その後，近細粒度タスク間のデータ依存を解析してタスクグラフを作成し，このタスクグラフ上のタスクを，データ転送・同期オーバーヘッドを考慮して実行時間を最小化できるように各 PE にスタティックにスケジューリングする．スケジューリング後，PE に割り当てられたタスクに対応する命令列を順番に並べデータ転送命令や同期命令を必要な箇所に挿入し各 PE 毎に異なるマシンコードを生成する．

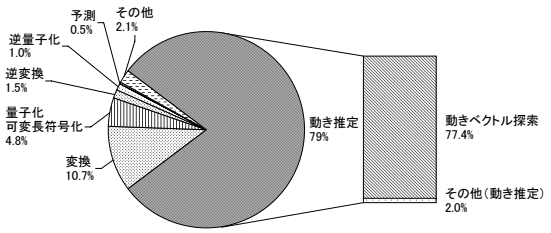

図 1: MPEG2 処理時間割合

3 動きベクトル探索処理のマルチグレイン並列処理

ここでは，本論文で対象とする動きベクトル探索処理アルゴリズム，およびこのアルゴリズムのマルチグレイン並列処理手法について述べる．

3.1 動きベクトル探索処理

本論文では動きベクトル探索処理として MediaBench[10] に収録されている “mpeg2encode” 中の動きベクトル探索処理関数 “fullsearch” を解析対象とし，この関数をもとに動きベクトル探索処理のマルチグレイン並列処理を行なう．まず，“mpeg2encode” における動きベクトル探索処理のプログラム全体の処理時間の割合を Sun Ultra80（UltraSPARC-II 450MHz ×4 ただしシングルプロセッサで処理，L2 キャッシュ4M バイト）上でコンパイルオプション “-pg -O” にてビルドしたバイナリを gprof を用いて計測した結果を図 1 に示す．図 1 より動きベクトル探索処理は処理全体の 77.4%を占めることが分かる．

MPEG2 エンコード処理では，図 2 に示すように階層的なデータ構造を使用している．MPEG ビデオの全シーケンスは，複数の GOP（Group of Picture）で構成され，GOP は複数のピクチャー（フレーム）で構成されている．それぞれのピクチャーは，いくつかのマクロブロックから構成されるスライスからなり，さらにマクロブロックは 6 つのブロック（4 つ

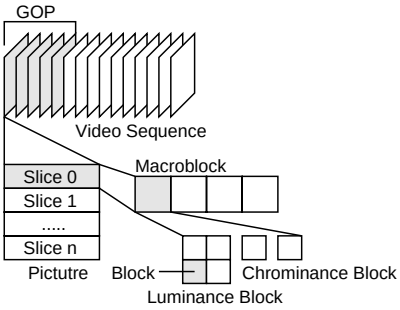

図 2: MPEG2 データ構造

のルミナンスブロックと 2 つのクロミナンスブロック）から構成される．また，1 ブロックは 8×8 ピクセルで構成される．

通常，動きベクトル探索処理は，上記データ構造のうちマクロブロックレベルで行なわれる．ただし，本論文で用いる動きベクトル探索処理関数 “fullsearch” アルゴリズム [11] では，マクロブロック中のルミナンスブロック（16×16 ピクセルブロック）に対してのみ動きベクトルの検出処理を行なう．動きベクトル探索処理は，探索対象画像と参照画像の間で行なわれるが，ここでは説明を簡単にするため直前のフレームを参照画像として動きベクトル探索を行なう場合を例に説明する．また，以下では特に記述がない場合マクロブロックとはマクロブロック中のルミナンスブロックを示す．

はじめに，Minimum Absolute Difference（MAD）法により参照画像である直前のフレームの入力画像と動きベクトル探索を行なう対象マクロブロック間の絶対誤差を検出し，絶対誤差が最小となるもとの座標からの変位を検出する．このとき，絶対誤差は，X_c を動きベクトル探索を行なうマクロブロック，X_r を参照画像とすると，1 ピクセルの精度で以下の式で求められる．

$$(i,j) = \underset{i,j}{min} \left(\sum_k \sum_l |X_c(k,l) - X_r(k+i,l+j)| \right)$$

ここで，探索はマクロブロック単位で行なわれるため $1 \leq k,l \leq 16$ である．探索範囲となる i,j は任意に設定できるが，H.263 での探索範囲は 0.5 ピクセル精度で通常 -16 から $+15.5$ であることを考慮し，本論文では $-16 \leq i,j \leq 16$ とした．この条件での探索範囲は図 3 に示す灰色の部分（Search Window）に相当し，検索対象マクロブロック（Target MB）と同位置の 1 マクロブロックと，その周辺マクロブロックが検索範囲となる．

絶対誤差が最小となる i,j が求まった後，次に，直前のフレームのエンコード値をデコードした画像

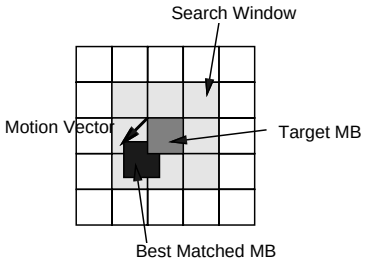

図 3: 動きベクトル探索範囲

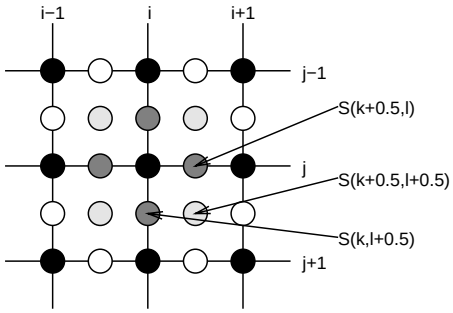

図 4: 中点補完

を参照画像とし，これと動きベクトル探索を行なう対象マクロブロックとの絶対誤差の最小値を，先に求めた i,j の 0.5 ピクセル四方周辺に対して求める．このとき，図 4 のように中点となる値を以下の式で補完する．ここで，演算子 “//” は演算結果を近い整数値に丸める除算である．

$$\begin{aligned} S(k+0.5,l) &= (S(k,l) + S(k+1,l))//2 \\ S(k,l+0.5) &= (S(k,l) + S(k,l+1))//2 \\ S(k+0.5,l+0.5) & \\ &= (S(k,l) + S(k+1,l) + S(k,l+1) \\ &\quad + S(k+1,l+1))//4 \end{aligned}$$

そして，i',j' を $-0.5 \leq i',j' \leq 0.5$，増分を 0.5 とし，X_c を動きベクトル探索を行なうマクロブロック，X'_r を参照画像のエンコード値をデコードした画像として

$$(i',j') = \underset{i',j'}{min} \left(\sum_k \sum_l |X_c(k,l) - X'_r(k+\alpha,l+\beta)| \right)$$
$$\text{where } \alpha = (i+i'),\ \beta = (j+j')$$

により，絶対誤差が最小となる i',j' を求め $(i+i', i+j')$ を最終的な動きベクトルとする．

3.2 粗粒度タスクの定義

前述した動きベクトル探索処理アルゴリズムでは，1マクロブロックの動きベクトル探索処理は，図3の灰色の部分の範囲（Search Window）でのみ動きベクトルの探索を行なう．そのため，処理開始時に必要となるデータをロードしてしまうとマクロブロック間でのデータ依存は発生しないのでマクロブロック間では並列処理可能である．ただし，効率よく並列処理を行なうためには，高速なPE内ローカルメモリの利用が必要となるため，ここで各PEで必要となるデータ量を考慮する．

動きベクトル探索処理で必要となるデータ量は，図3に示すように，動きベクトルを探索したいマクロブロック1つに対し，検索範囲であるSearch Window分のデータが必要となる．まず，本論文の動きベクトル探索処理ではSearch Windowは1つの対象マクロブロックに対し9マクロブロックであるので，動きベクトルを探索する対象1マクロブロックと参照するSearch Window分の9マクロブロックのデータが必要になる．さらに0.5ピクセルの精度で検出する際に必要となる参照画像をエンコードした後，デコードを行なって復元した画像のSearch Window分の9マクロブロックも必要となるため，必要なマクロブロックは合計19マクロブロックである．よって，必要なデータ量は $(16\times16)\times19 = 4864$ ピクセルデータとなり，1ピクセルデータを4バイトで格納すると仮定すると19Kバイトのデータ量が必要となる．この値は，後述するOSCAR型SCMのLDM中に十分格納可能な値である．

以上より，本論文ではマクロブロックレベルの処理を粗粒度タスクとして定義する．各粗粒度タスクのプロセッサグループ（PG）への割り当ては，各PGで処理するマクロブロック数が均等になるように行なう．

3.3 近細粒度タスクの定義

動きベクトル探索処理では，絶対誤差の最小値を求めるため，図5のようにIF文による分岐が発生する．この分岐は入力データに依存しており静的なコスト推定は不可能である．そのため，粗粒度並列処理のみでは処理コストのアンバランスが生じてしまい効率的な並列処理ができない場合がある．また，絶対誤差の演算において演算結果のオーバーフローチェック処理によりオーバーフローが検出された時，ループ外にジャンプしてしまうためループ並列処理は適用できない．そこで，本論文では，より効率的な並列処理を行なうために粗粒度タスクとして扱う

```
      do 100 j=1, h
       do 200 i=1, w
        do 300 l=1, 16
         do 400 k=1, 16
          <絶対誤差の演算：
            （値がオーバーフロー時 200 へジャンプ）>
400       continue
300      continue
           if (<絶対誤差値> .lt. <絶対誤差最小値>) then
             <絶対誤差最小値更新処理>
           endif
200       continue
100   continue
```

図5: 動きベクトル探索処理コード例

マクロブロックレベル処理の内部において，マクロブロックレベル処理内の各ステートメントを近細粒度タスクと定義し複数のPE間で並列処理を行なうことにより処理効率を高める．

近細粒度タスクの生成後，近細粒度タスクはスケジューリングルーチンでPEへ割り当てられる．このとき，データ転送オーバーヘッドを考慮し実行時間を最小化するヒューリスティックスケジューリングアルゴリズムであるCP/DT/MISF法，CP/ETF/MISF法，ETF/CP法およびDT/CP法の4手法を同時に適用し最良のスケジュールを選びスタティックに近細粒度タスクを割り当てる．

以上のように，本論文ではマクロブロック間の並列性を利用する粗粒度並列性とマクロブロック内の並列性を利用する近細粒度並列性を階層的に組み合わせマルチグレイン並列処理を行なう．

4 OSCAR型シングルチップマルチプロセッサアーキテクチャ[9)]

ここでは，OSCAR型シングルチップマルチプロセッサ(SCM)アーキテクチャおよびそのプロセッサコアアーキテクチャについて述べる．

4.1 メモリアーキテクチャ

OSCAR型SCMのネットワークおよびメモリアーキテクチャは，図6に示すようにCPU，データ転送をCPUの処理とオーバーラップして行なえるデータ転送ユニット(DTU)，各々のCPUで実行するプログラムを格納するローカルプログラムメモリ(LPM)，PE固有のデータを保持するローカルデータメモリ(LDM)，自PEと他PEの双方から同時にアクセス可能なマルチポートメモリの分散共有メモリ(DSM)を持つプロセッサエレメント(PE)を相

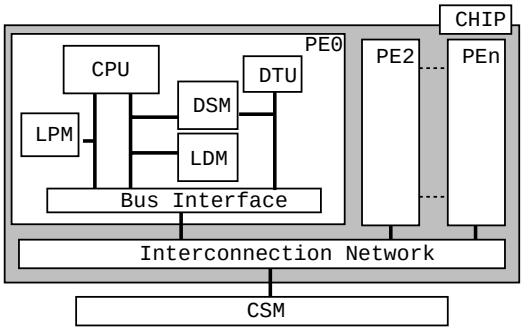

図 6: OSCAR 型 SCM アーキテクチャ

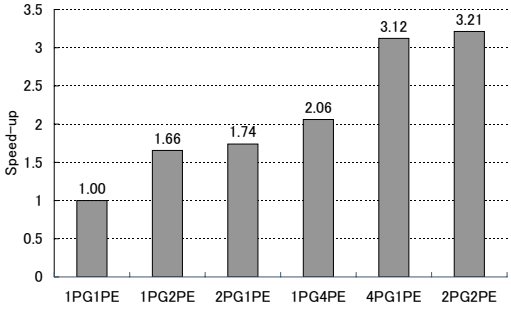

図 7: 動きベクトル検出処理評価結果

表 1: メモリ容量，アクセスレイテンシ

メモリ	アクセスレイテンシ (clock)	容量 (バイト)
LPM	1	-
LDM	1	512K/chip
DSM	1 (internal), 4 (remote)	16K
CSM	20	-

互接続網 (バス結合，クロスバ結合など) で接続し 1 チップ上に搭載したアーキテクチャである．今回の評価では，DTU についてはオーバーラップデータ転送スケジューリングアルゴリズムが未実装のため利用してない．また，本評価では PE 間相互結合網として 3 本バスを利用する．

今回の評価では各メモリサイズ，アクセスレイテンシは表 1 に示す値を仮定する．ここで LPM は，本論文では対象プログラムは数十 K バイト程度の容量であるためプログラムすべてが LPM 上に格納可能なサイズとし，CSM は，処理に必要なすべてのデータが格納出来るサイズとした．

OSCAR 型 SCM では，これら 4 種類のメモリに対し最適なデータ配置を行なうことにより効率の良い並列処理を行なうことができる．

4.2 プロセッサコアアーキテクチャ

各 PE が持つ CPU は，SPARC V9 規格に準拠したプロセッサである Sun Microsystems 社の UltraSPARC II[12] のパイプライン構成をベースとし，バリア同期機構等用の特殊レジスタや特殊レジスタを操作するための命令を付加したプロセッサである．

今回の評価で用いる OSCAR 型 SCM のプロセッサコアは，整数演算ユニット (IEU) を 1 本，ロードストアユニット (LSU) を 1 本，浮動小数点ユニット (FPU) を 1 本持つシングルイシューのシンプルな構成とした．

5 性能評価

ここでは，動きベクトル探索処理にマルチグレイン並列処理を適用し OSCAR 型シングルチップマルチプロセッサ（SCM）上で評価した結果について述べる．なお，性能評価にはクロックレベルシミュレータを用いた．

評価に用いるプログラムは，“mpeg2encode” 中の動きベクトル探索処理関数 “fullsearch” 部を抜粋し，第 3 章で提案した粗粒度および近細粒度タスク生成を行なえるように逐次プログラムをリストラクチャリングしたプログラムである．このプログラムを OSCAR マルチグレイン自動並列化コンパイラを用いてマルチグレイン並列性の抽出およびタスクスケジューリングを行ない SCM 用バイナリコードを生成した．プログラムは，動きベクトル探索処理の並列処理効果のみを計るために，動きベクトル探索処理を行なう画像と参照する画像（原画像）および参照する画像をエンコードした後，デコードを行なった画像（デコード画像）の 3 画像を入力として動きベクトルを探索する．入力画像は，MediaBench で用いられる入力画像（compo.tar.gz 中の rec*.[YUV]）をシミュレーション時間短縮のために 176 × 128 ピクセルに縮小した画像の Frame1 を動きベクトル検索対象画像，直前のフレームの Frame0 を参照画像として用いる．ただし，MediaBench では本論文で行なう動きベクトル探索処理の入力データは用意されていないため “mpeg2encode” を改編し関数 “fullsearch” の入力より動きベクトル探索対象画像，参照画像，および参照画像のエンコード値をデコードした画像を抽出し，それを入力データとした．なお，入力データは，CSM 上に初期配置する．

OSCAR 型 SCM 上での評価結果を逐次実行時間すなわち 1PG1PE 時の処理時間に対する速度向上率として図 7 に示す．図中横軸の mPGnPE は n 個

のプロセッサエレメント (PE) をグループとした m 個のプロセッサグループ (PG) による処理を表し，SCM 中の総 PE 数は $m \times n$ であることを示す．この mPGnPE という SCM 構成は，コンパイラから見たタスク割り当て単位の仮想的な構成であり，PG 間では粗粒度タスク並列処理，PG 内 PE 間では近細粒度並列処理を行なうことを前提とし，ハードウェアとしては同一である．

ステートメント単位の並列性を用いた近細粒度並列処理のみの場合，総 PE 数 2 の 1PG2PE で約 1.66 倍，総 PE 数 4 の 1PG4PE で約 2.06 倍の速度向上率が得られ，総 PE 数が増加しているにもかかわらず速度向上の伸びが悪くなった．これは，動きベクトル検索対象とするマクロブロックはすべての PE で同じなため必要な初期データのロード時間の比率が多くなったことや，PE 間のデータ転送回数，同期回数が増加したためである．マクロブロック間の並列性を用いた粗粒度並列処理のみの場合，総 PE 数 2 の 2PG1P で約 1.74 倍，総 PE 数 4 の 4PG1PE で約 3.12 倍となり総 PE 数増加にともないスケーラブルな速度向上が得られた．これは，マクロブロック間処理においてデータ依存がないため効率の良い並列処理が行なえたためである．マクロブロック間の粗粒度並列性とマクロブロック内ステートメント間の近細粒度並列性を階層的に用いたマルチグレイン並列処理の場合，総 PE 数 4 の 2PG2PE で約 3.21 倍の速度向上率が得られ総 PE 数が同じときの粗粒度並列処理のみを用いた 4PG1PE に比べ若干の速度向上が得られた．これは，マクロブロック内処理における入力データに依存する各マクロブロックの処理コストのアンバランスが近細粒度並列処理を用いることにより負荷バランスが向上しより高い処理効率が得られたためである．

6 まとめ

本論文では，動きベクトル探索処理におけるマクロブロック間の粗粒度並列性とマクロブロック内の近細粒度並列性を階層的に用いるシングルチップマルチプロセッサ用マルチグレイン並列処理手法を提案しその性能評価を行なった．その結果シングルイシュープロセッサコアを 4 基搭載した OSCAR 型 SCM 上では，逐次実行時間に対し 2PG で粗粒度並列処理，各 PG 内部の 2PE で近細粒度並列処理を階層的に行なうことによりを約 3.21 倍の速度向上率が得られた．以上よりシンプルなプロセッサコアを集積した OSCAR 型 SCM 上での動きベクトル検出処理のマルチグレイン並列処理により効果的な並列処理が可能であることが確認できた．

今後，様々なパターンの入力画像による処理効率の評価や MPEG2 処理全体でのマルチグレイン並列化手法の検討およびマルチグレイン並列処理を行なっていく予定である．

謝辞

本研究の一部は，STARC「自動並列化コンパイラ協調型シングルチップマルチプロセッサの研究」及び経済産業省/NEDO ミレニアムプロジェクト「アドバンスト並列化コンパイラ」により行われた．本論文作成にあたり，有益なコメントをいただいた富士通 高橋宏政氏，松下 高山秀一氏，東芝 安川英樹氏，ソニー 倉田隆弘氏，STARC 宮本俊介氏に感謝致します．

参考文献

[1] S. K. Raman, V. Pentkovki and J.Keshava: Implementing Streaming SIMD Extensions on the Pentium III Processor, *IEEE MICRO*, Vol. 20, No. 4 (2000).

[2] F. Arakawa, O. Nishi, K. Uchiyama and N. Nakagawa: SH4 RISC Multimedia Microprocessor, *IEEE MICRO*, Vol. 18, No. 2 (1998).

[3] A. Suga and K. Matsunami: Introducting the FR500 Embedded Microprocessor, *IEEE MICRO*, Vol. 20, No. 4 (2000).

[4] M. Edahiro, S. Matsushita, M. Yamashita and N. Nishi: A Single-Chip Multiprocessor for Smart Terminals, *IEEE MICRO*, Vol. 20, No. 4 (2000).

[5] L. Hammond, B. Hubbert, M. Siu, M. K. Prabhu, M. Chen and K. Olukotun: The Stanford HYDRA CMP, *IEEE MICRO*, Vol. 19, No. 2 (1999).

[6] E. Iwata and K. Olukotun: Exploiting Coarse-Grain Parallelism in the MPEG-2 Algorithm, CSL-TR-98-771, Stanford University Computer System Lab. (1998).

[7] 道中秀治, ほか: A single-Chip MPEG-2 Codec Based on Customizable Media Microprocessor, ICD2002-20 (2002).

[8] H. Kasahara, M. Obata and K. Ishizaka: Automatic Coarse Grain Task Parallel Processing on SMP using OpenMP, *Proc. 12th Workshop on Languages and Compilers for Parallel Computing* (2000).

[9] 木村啓二, 加藤孝幸, 笠原博徳: 近細粒度並列処理用シングルチップマルチプロセッサにおけるプロセッサコアの評価, 情報処理学会論文誌, Vol. 42, No. 4 (2001).

[10] C. Lee, M. Potkonjak and W. H. Mangione-Smith: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, *30th International Symposium on Microarchitecture (MICRO-30)* (1997).

[11] MPEG Software Simulation Group: Test Model 5, *http://www.mpeg.org/MPEG/MSSG/tm5/*.

[12] Sun Microelectronics: *UltraSPARC^{TM} User's Manual* (1997).