データローカライゼーションを伴う MPEG2エンコーディングの並列処理

小高 剛[†] 中野啓史[†]
木村啓二[††] 笠原博徳[†]

PC，PDA，携帯電話などで静止画像，動画像，音声などを扱うマルチメディアアプリケーションを利用する機会が近年ますます増えている．このため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必要性が増してきている．このような要求を満たすアーキテクチャとして複数のプロセッサコアを 1 チップ上に搭載したチップマルチプロセッサアーキテクチャが，命令レベル以外の粗粒度タスク並列性，中粒度ループ並列性など複数レベル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プロセッサアーキテクチャとして注目されている．しかしながら，チップマルチプロセッサアーキテクチャ上で効率の良い処理を行なうには，アプリケーションの特性を解析し，その並列性とデータローカリティを考慮しながらプログラムを適切な粒度のタスクに分割し，それらのタスクをバランス良く CPU に配置する並列化技術が不可欠である．本論文では，データを共有する粗粒度タスクの連続実行によりチップ内ローカルメモリを利用したデータの授受を行ない実行効率を向上させるデータローカライゼーション手法の MPEG2 エンコーディングへの適用を提案し，OSCAR チップマルチプロセッサ上で性能評価を行なう．評価の結果，提案手法は 8 プロセッサ利用時で従来のループ並列処理に対して 1.64 倍の性能が得られ，逐次実行に対しても 6.82 倍の速度向上が得られた．

Parallel Processing for MPEG2 Encoding using Data Localization

Takeshi Kodaka[†] , Hirohumi Nakano[†] , Keiji Kimura[††] and Hironori Kasahara [†]

Recently, many people are getting to enjoy multimedia applications with image and audio processing on PCs, mobile phones and PDAs. For this situation, development of low cost, low power consumption and high performance processors for multimedia applications has been expected. To satisfy these demands, chip multiprocessor architectures which allows us to attain scalability using coarse grain level parallelism and loop level parallelism in addition to instruction level parallelism are attracting much attention. However, in order to extract much performance from chip multiprocessor architectures efficiently, highly sophisticated technique is required such as decomposing a program into adequate grain of tasks and assigning them onto processors considering parallelism and data locality of target applications. This paper describes a parallel processing scheme for MPEG2 encoding using data localization which improve execution efficiency assigning coarse grain tasks sharing same data on a same processor consecutively for a chip multiprocessor, and evaluate its performance. As the evaluation result on OSCAR CMP using 8 processors, proposed scheme gives us 1.64 times speedup against loop parallel processing, and 6.82 times speedup against sequential execution time.

1 はじめに

モバイル端末上での静止画像，動画像，音声処理などのマルチメディア処理の要求が高まっている．そのため，各種マルチメディアアプリケーションをモバイル端末上で快適に利用するために低消費電力で価格性能比の優れたプロセッサの開発が望まれている．動画像や音声処理などを扱うメディアアプリケーションでは，ユーザーが設定したビットレートなど必要な品質を確保でき，かつリアルタイム性や低消費電力を求められる場面が多い．メディア処理にかかる時間の短縮はリアルタイム性の確保だけでなく品質を優先させる場合はより高精度で時間のかかる処理の導入を可能にしたり，低消費電力を求める場合は低速，低電圧での駆動を可能するなどの応用が期待できる．

以上のような状況からマルチメディア処理を高価格性能比かつ低消費電力で実現するアプローチの 1 つとして，1 チップ上にプロセッサコアを複数搭載したチップマルチプロセッサアーキテクチャが注目を集めている．チップマルチプロセッサアーキテクチャは，従来の SIMD，VLIW などで用いられていた命令レベル並列性に加え，ループイタレーション間の中粒度並列性，基本ブロック，ループ，サブルーチン間の粗粒度タスク並列性も利用可能であり，集積度向上に対してスケーラブルな性能向上を目指す上で有望な方式と考えられている．ただし，チップマルチプロセッサ上で効率の良い処理を行なうには，アプリケーションからの並列性抽出やデータ配置の最適化などに関する高度な並列処理の知識，あるいはそれらの最適化をサポートする強力なコンパイラが必要となる．

その一方，近年のマイクロプロセッサでは，チップ内部の駆動速度は劇的に向上しているが，チップ外部との駆動速度の差が広がっておりメモリアクセスなどチップ外部との通信に起因するボトルネックがシステム全体の性能向上の壁になっている．そのため，チップ上のメモリをいかに効率よく有効に使うかが性能向上の鍵になっている．

筆者等は実効性能が高く価格性能比及びプログラム生産性の高いコンピュータシステムの実現を目指し，複数粒度の並列性を階層的に組み合わせて利用するマルチグレイン並列処理と協調動作する OSCAR チップマルチプロセッサアーキテクチャ（OSCAR CMP）を提案している [1]．OSCAR CMP はチップ

† 早稲田大学理工学部コンピュータ・ネットワーク工学科
Dept. of Computer Science, Waseda University
†† 早稲田大学理工学総合研究センター
Advanced Research Institute for Science and Engineering, Waseda University

内にローカルメモリと 2 ポート構成の分散共有メモリを持ちこれらのメモリをソフトウェアが適切に利用する事によりプログラムの持つ並列性とデータローカリティの両方を最大限に活用できるアーキテクチャである．また，データローカリティ利用についてはデータを共有する粗粒度タスクを連続実行し，チップ内ローカルメモリを利用したデータの授受を行ない実行効率を向上させるデータローカライゼーション手法を提案している [2),3)]．

動画像処理の規格の 1 つである MPEG2 は DVD などで広く利用されているが，そのエンコード処理は計算量が多く高速化が求められている．MPEG2 エンコードの並列性に関しては，そのデータ構造が階層構造となっているためフレーム単位，マクロブロック単位など各データ階層において並列性が抽出できることが確認されており，DSP やチップマルチプロセッサ上で並列処理の研究が行なわれている [4),5)]．筆者等もマクロブロックレベルの粗粒度並列性を用いて OSCAR CMP 上での並列性利用の確認を行なった [6)]．本論文では，OSCAR CMP 上でのさらなる性能向上を得るため，データローカライゼーション手法の MPEG2 エンコーディングへの適用を提案し，OSCAR CMP 上で評価する．また，ループ並列処理との比較を行ない提案手法の有効性を示す．

以下, 2 節で本論文で対象とする並列処理の粗粒度タスク並列処理（マクロデータフロー処理）とデータローカライゼーション，3 節で OSCAR チップマルチプロセッサアーキテクチャ，4 節で MPEG2 エンコーディングのデータローカライゼーション手法の適用の提案, 5 節で OSCAR チップマルチプロセッサ上における提案手法の評価結果について述べる．

2 粗粒度タスク並列処理とデータローカライゼーション

本節では，本論文で対象とする並列処理粒度の粗粒度タスク並列処理およびデータローカライゼーションを伴う粗粒度タスクスタティック・スケジューリングについて述べる．

2.1 粗粒度タスク並列処理 [7)]

粗粒度タスク並列処理（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) をグループ化したプロセッサグループ (PG) に割り当てる．なお，このグループ化はプログラム中の各部分の並列性に応じソフトウェア的に行なわ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 データローカライゼーション [2),3)]

データローカライゼーションでは，データを共有する各ループ (MT) のデータ使用範囲が一致するようにループ整合分割 [2)] を行ない，ループ間のデータ授受がターゲットアーキテクチャのキャッシュもしくはローカルメモリを介して行なえるようにする．

その後，マクロタスクグラフにスタティックスケジューリングを適用する場合は，粗粒度タスクのデータの生死解析情報を用いて粗粒度タスク間のデータ共有量を計算して，データを共有するタスクが同一プロセッサ上でなるべく連続して実行するように各タスクをプロセッサ上にスケジューリングする [3)]．このようにすることでデータを共有する複数の粗粒度タスク間でキャッシュもしくはローカルメモリを介した効率の良いデータの受渡しが可能となる．

3 OSCAR チップマルチプロセッサアーキテクチャ [1)]

本節では，本論文で対象とするチップマルチプロセッサアーキテクチャである OSCAR チップマルチプロセッサアーキテクチャ（OSCAR CMP）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．

OSCAR CMP のネットワークおよびメモリアーキテクチャは，図 1 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) を相互接続網 (バス結合，クロスバ結合など) で接続し，各 PE で共有するデータなどを格納する集中共有メモリ (CSM) を 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開発中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．

本論文では各メモリサイズ，アクセスレイテンシは，LDM の容量は 256K バイトとしアクセスレイ

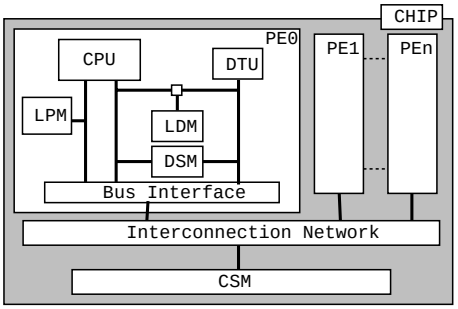

図 1: OSCAR CMP アーキテクチャ

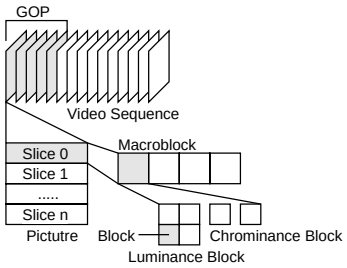

図 2: MPEG2 データ構造

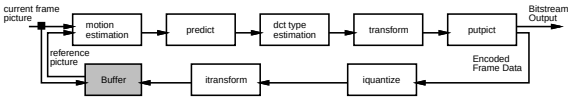

図 3: MPEG2 エンコード ブロック図

テンシは 1 クロック，同様に DSM の容量は 16K バイトで自 PE 内のローカルアクセスには 1 クロック，他 PE へのリモートアクセスには 4 クロックかかるとし，CSM は本論文で使用する MPEG2 エンコーディングで利用するメモリ容量が十分確保されているものとしてアクセスレイテンシは 20 クロックとした．OSCAR CMP 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．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 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 CMP 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．

4 MPEG2 エンコーディングの並列処理

本節では，本論文で対象とする MPEG2 エンコーディングアルゴリズムについて述べた後，MPEG2 エンコーディングの並列性抽出および提案するデータローカライゼーション手法の適用について述べる．

4.1 MPEG2 エンコーディングアルゴリズム

本論文では MPEG2 エンコーディングアルゴリズムの参照実装として MediaBench[8] に収録されている “mpeg2encode” を用いる．なお，本論文で用いるエンコーディングオプションは MediaBench で用いられているエンコーディングオプションと同様とする．

MPEG2 エンコードのデータ構造は図 2 に示すように階層的である．MPEG ビデオの全シーケンスは，複数の GOP（Group of Picture）で構成され，さらに各 GOP は複数のピクチャ（フレーム）で構成されている．それぞれのピクチャは，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．

MPEG2 エンコードのブロック図を図 3 に示す．MPEG2 エンコードは，7 つのステージからなり，エンコード対象フレーム画像が参照画像（エンコード対象画像より過去の画像または未来の画像）からどれだけ移動しているかを表す動きベクトルの探索を行ない符合化の際に前向き予測，後向き予測，双方向予測のどれを利用するかという符合化モードを決定する動き推定（motion estimation），動き推定で求めた動きベクトル，符合化モードに基づいて符合化対象ピクチャを生成する動き予測（predict），符合化対象ピクチャに対してマクロブロックレベルでフレーム構造 DCT を適用するかフィールド構造 DCT を適用するかを決定する DCT 変換構造選択（dct type estimation），符合化対象ピクチャに DCT 変換構造選択で決定した DCT 変換構造に基づき離散コサイン変換（DCT）を適用するデータ変換（transform），DCT を適用したピクチャをビットストリームとして出力しビットレート制御，量子化係数の決定，量子化，各種ヘッダ送出などを行なうビットストリーム出力（putpict），量子化適用後のピクチャを逆量子化を行なって復元する逆量子化（iquantize），逆量子化後のピクチャに対し逆 DCT を適用しピクチャを復号化する逆データ変換（itransform）からなる．MPEG2 ではこれらエンコード各ステージをピクチャ全体を構成するマクロブロックに対して適用する事によりエンコードが行なわれる．

4.2 MPEG2 エンコーディングの並列性の抽出およびデータローカライゼーション手法の適用

MPEG2 エンコードの並列性抽出に際し各ステージにおける利用データの流れおよびデータ使用範囲に注目する．まず，動き推定，動き予測，DCT 変換構造選択，データ変換，逆量子化，及び逆データ変換では処理対象がマクロブロックであり，扱うデータ範囲がマクロブロックもしくは処理対象のマクロブロックから求められた演算結果に限られているため，各マクロブロックレベル処理の間ではデータ依存が発生しない．そのため，これらのステージではループ並列処理が適用可能である．一方，ビットストリーム出力ステージでは，ビットレート制御を行なうため直前のマクロブロックで生成されたマクロブロック量子化情報を用いて処理対象とするマクロブロックの量子化情報を生成している．また，MPEG2 エンコード出力データであるビットストリームは画像左上のマクロブロックから右へと順番に出力される必要があるため，処理対象とするマクロブロックのビットストリーム出力は直前のマクロブロックのビットストリーム出力処理が終るまで開始できない．このためビットストリーム出力ステージではループイタレーション間でデータ依存が存在するため並列処理の適用が難しい．

MPEG2 エンコーディングの場合，あるステージで利用または演算したデータが後のステージで利用される場合が多い．例えば，動き推定での参照画像が DCT 変換構造選択で利用されたり，データ変換で演算したエンコード画像を逆量子化，逆データ変換で利用する．そのためデータを共有しているステージ間のデータ授受を高速なチップ内 LDM を用いて行なえれば実行時間の短縮が可能である．ここで，本論文のターゲットアーキテクチャである OSCAR CMP の LDM 容量を考慮に入れると，QCIF，QVGA など一般的に用いられる大きさの画像の 1 フレームをエンコードするのに必要なデータすべてを LDM 内に格納するには容量が不十分である．そのため，単純に各ステージ毎にループ並列処理を適用したエンコーディングを行なうと始めのイタレーションで処理を行なったマクロブロックの演算結果はループの最後にはメモリ容量不足のため LDM 上には置いておけない．よって，単純なループ並列処理では MPEG2 エンコーディングの各ステージ間で LDM を用いたデータの授受が行なえず，各ステージ毎に LDM-CSM 間のデータ転送が発生してしまう．そこで，MPEG2 エンコーディングの各ステージを LDM にデータが格納できるように分割し，利用データ範囲が同じステージを連続的に実行するようにすれば，LDM を介した各ステージ間のデータ授受が可能となる．ループ並列処理可能な動き推定，動き予測，DCT 変換構造選択，データ変換，逆量子化，及び逆データ変換の各ステージでは，処理単位がマクロブロックレベルとなるようにループ分割し，それぞれを粗粒度タスクとして定義する．ビットストリーム出力ステージでは各ループイタレーション間でデータ依存が発生しているが，他のステージと同様にループによりマクロブロック単位の処理を行なっているため，処理単位がマクロブロックレベルとなるようにループ分割を行なう．このようにループ分割を行なうと図 4 のような粗粒度タスク間の並列性を表したマクロタスクグラフが得られる．なお，図 4 では説明のために各ステージを 4 分割している．図中各ノードが粗粒度タスクを示し，各エッジはデータ依存を示している．図 4 よりマクロブロックレベルでのループ分割によりビットストリーム出力ステージ以外では各タスクはマクロブロックをまたいでのデータ共有はしておらず，ビットストリーム出力ステージでのみマクロブロック間でのデータ依存が存在している事が分かる．ここで，ビットストリーム出力ステージのデータ依存エッジのデータ共有量に注目する．図中 MT5_1 から MT6_1 へのエッジは処理対象としているマクロブロックのビットストリーム出力ステージから逆量子化ステージへのデータ依存を表している．一方，図中 MT5_1 から MT5_2 へのエッジは処理対象の前後のマクロブロックのビットストリーム出力ステージとの間のデータ依存を表している．前者ではエンコードを行なったマクロブロックのエンコードデータ，後者ではビットレート制御で用いるマクロブロック量子化情報とビットストリーム出力開始位置情報のデータがそれぞれ受渡しされている．そのため，データ共有量を考慮しデータ共有量の多い前者のデータ授受をチップ内 LDM を用いて行なえばデータローカリティを利用できメモリアクセス時間の短縮が見込まれ処理時間の短縮が行なえる．以上をふまえ，各タスクのプロセッサへのスケジューリングは処理対象とするマクロブロックのエンコードが連続して同じプロセッサへ割り当てられるように行なう．このようにすることで各タスク間での共有するデータが LDM を介して行なうことが可能となり実行時間の短縮が可能となる．

5 性能評価

ここでは，MPEG2 エンコーディングに対して 4 節で提案した手法を適用し OSCAR CMP 上で評価した結果について述べる．また，従来のループ並列処理の場合との比較も行ない提案手法の有効性を確認する．性能評価には OSCAR CMP をクロックレベルでシミュレートするシミュレータを用いた．評価に用いるプログラムは，MediaBench に収録されている MPEG2 エンコーディングプログラム “mpeg2encode” を参照実装したプログラムを用い，MPEG2 エンコーディング処理の中心部分である動き推定，動き予測，DCT 変換構造選択，データ変換，逆量子化，逆データ変換という一連のエンコード処理を評価対象とする．並列性の抽出はシーケンシャルソースプログラムを作成し，粗粒度タスク並

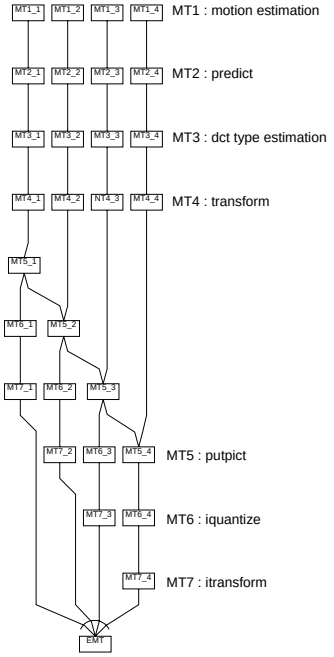

図 4: マクロタスクグラフ

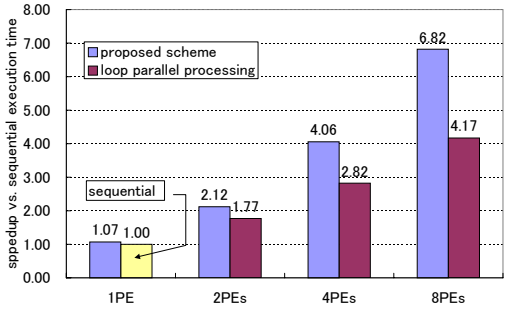

図 5: MPEG2 エンコーディング評価結果

列性抽出およびループ並列性抽出を OSCAR マルチグレイン自動並列化コンパイラを用いて行ない，手動で 4.2 節で述べたマクロタスクのスタティックタスクスケジューリングを適用し OSCAR CMP 用バイナリコードを生成した．ただし，データ転送ユニットによるデータ転送オーバーヘッド隠蔽技術は現時点でマルチメディアアプリケーションへの適用ができていないため本評価では利用しない．なお，逐次実行およびループ並列処理適用時の LDM の利用は各イタレーションの実行開始時にそのイタレーションで利用されるデータを CSM から LDM にロードし，イタレーション実行中は LDM を利用して演算を行ない，イタレーション終了時に他のステージで利用される共有データを LDM から CSM へストアしている．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 QCIF(176 × 144 ピクセル) に縮小した画像を用い，4 フレームのエンコーディングを行なう．なお，エンコーディング時のピクチャタイプの順番は I☆，P☆☆，B☆☆☆，B☆☆☆の順で行なわれる．

OSCAR CMP 上での 4 フレーム MPEG2 エンコードを実行した評価結果を図 5 に示す．横軸は使用プロセッサ数を示し，1 プロセッサ利用時の “1PE”，2 プロセッサ利用時の “2PEs”，4 プロセッサ利用時の “4PEs”，8 プロセッサ利用時の “8PEs” である．棒グラフは逐次実行時間に対する速度向上率を示し，1PE 右側がベースとなる MediaBench から参照実装したプログラムの逐次実行時間 “sequential” をであり，2PEs，4PEs 及び 8PEs の右側がループ並列処理適用時，1PE，2PEs，4PEs 及び 8PEs の左側が提案手法適用時の速度向上率である．

評価結果よりループ並列処理適用時は，逐次実行時に対して 2 プロセッサ利用時 1.77 倍，4 プロセッサ利用時 2.82 倍，8 プロセッサ利用時 4.17 倍の速度向上率であった．これに対して，提案手法適用時は逐次実行時に対して 1 プロセッサ利用時 1.07 倍，2 プロセッサ利用時 2.12 倍，4 プロセッサ利用時 4.06 倍，8 プロセッサ利用時 6.82 倍の速度向上率が得られた．

提案手法適用時では，1 プロセッサ利用時で逐次実行時に対し 7%の速度向上がが得られた．これは，データを共有するタスクの連続実行により LDM を介して各ステージ間でデータの受け渡しが行われ，メモリアクセスの効率化が行なわれたためである．また，使用プロセッサ数を増やした場合でループ並列処理適用時と提案手法適用時を比較するとループ並列処理適用時は提案手法適用時に比べ速度向上の伸びが緩やかになっている．これは，提案手法適用による LDM を利用したステージ間のデータ受け渡しによるメモリアクセスの効率化に加え，提案手法ではプロセッサの負荷バランスの向上が得られ提案手法がより良い速度向上を得られたためである．ループ並列処理適用時はビットストリーム出力ステージはシーケンシャルループなため並列処理できないので使用プロセッサ数が増えるにつれてビットストリーム出力ステージの実行時間割合が大きくなってしまう．これに対して，提案手法ではすべてのステージをマクロブロックレベルに分割したため以下に説明するように負荷バランスが向上した．この理由を提案手法適用時の実行イメージを表す図 6 を用いて説明する．図は各 PG におけるマクロタスクの割り当てを示しており，縦が時間軸，各 MT がマクロタスク，網かけがプロセッサアイドル，矢印がプロセッサ間データ転送を示している．ビットストリーム出力ステージを含めマクロブロックレベルにループ分割を行なったためプロセッサ間の同期をとる必要がある場所はビットストリーム出力ステージのデータ依存が発生している部分，すなわち直前のマクロブロックのビットストリーム出力が終了し

☆ フレーム内（イントラ）符合化（静止画モード）
☆☆ 前向き（順方向）動き予測符合化
☆☆☆ 前向きおよび後向き（双方向）動き予測符合化

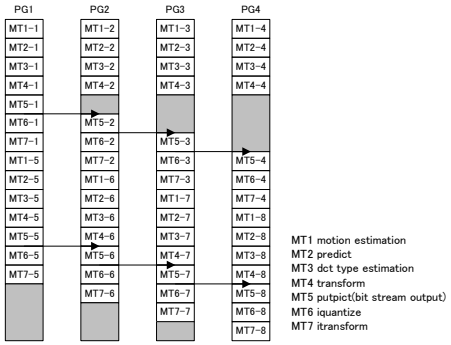

図 6: 提案手法適用時の実行イメージ

必要なデータが転送されてくるのを待つ部分である．この場合，同期にかかる待ち時間は最大で（1 マクロブロックのビットストリーム出力ステージの実行時間）×（使用プロセッサ数 − 1）であり実行開始時はこのアイドル時間が発生する．しかし，その後はビットストリーム出力ステージ以外ではマクロブロック間でのデータ依存は存在しないため，あるプロセッサが実行している n 番目のマクロブロックのビットストリーム出力ステージからのデータ転送を待っている間に，他の PE では n+i 番目のマクロブロックの動き推定からデータ変換ステージが実行可能である．そのため図のように負荷バランスが向上しプロセッサの利用効率が向上した．しかし，利用プロセッサ数が増えるに従い若干速度向上率が低下してきている．これは，MPEG2 エンコード処理中でベースとなるピクチャを生成する I ピクチャエンコードでは JPEG のように自身のフレームデータのみからエンコードを行なうため，動き推定は演算しないことに起因している．動き推定処理を行なわないためビットストリーム出力ステージより前の動き推定からデータ変換ステージまでの実行時間は短く（動き推定からデータ変換ステージの実行時間）＜（（1 マクロブロックのビットストリーム出力ステージでの実行時間）×（使用プロセッサ数 − 1））となり前のマクロブロックのビットストリーム出力処理が終了しデータが転送されてくるまでのプロセッサアイドル時間が発生しプロセッサ利用効率が低下したのである．しかし，全体のエンコード時間で見ると MPEG2 エンコードの実行時間の大半は動き推定であるため動き推定演算を行なう前向き動き予測符合化の P ピクチャエンコード，前向きおよび後向き動き予測符合化の B ピクチャエンコードにおける速度向上によりスケーラブルな結果を得ることができた．

6 まとめ

本論文では，データを共有する粗粒度タスクを連続実行することにより CPU 近接のローカルメモリを利用したデータ授受を行ない実行効率を向上させるローカライゼーション手法の MPEG2 エンコーディングへの適用について提案し OSCAR チップマルチプロセッサ上で性能評価を行なった．その結果，提案手法は従来のループ並列処理を適用した場合と比較した場合，本手法はループ並列処理に対し 2 プロセッサ利用時 1.20 倍，4 プロセッサ利用時 1.44 倍，8 プロセッサ利用時 1.64 倍の速度向上が得られ，逐次実行時間に対しても 1 プロセッサ利用時 1.07 倍，2 プロセッサ利用時 2.12 倍，4 プロセッサ利用時 4.06 倍，8 プロセッサ利用時 6.82 倍の速度向上が得られ本提案手法の有効性が確認できた．

今後の課題として，MPEG2 エンコードへのマクロブロック処理間の粗粒度並列性とマクロブロック処理内での近細粒度並列性を利用したマルチグレイン並列性の適用，及びキャッシュ共有型など他のチップマルチプロセッサアーキテクチャとの比較が挙げられる．

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」，早稲田大学理工総研プロジェクト研究「自動並列化コンパイラ協調型チップマルチプロセッサ」，文部科学省科学研究費補助金若手研究（B）（課題番号 15700074）及び特別研究員奨励費（課題番号 1501202）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宮本俊介氏（STARC），高橋宏政氏（富士通研），高山秀一氏（松下），安川英樹氏（東芝），倉田隆弘氏（ソニー）に感謝致します．

参考文献

[1] 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).

[2] 吉田明正, 越塚健一, 岡本雅巳, 笠原博徳: 階層型粗粒度並列処理における同一階層内ループ間データローカライゼーション手法, 情報処理学会論文誌, Vol. 40, No. 5 (1999).

[3] 中野啓文, 小高剛, 木村啓二, 笠原博徳: OSCAR CMP 上でのスタティックスケジューリングを用いたデータローカライゼーション手法, ARC2003-154-14 (2003).

[4] Iwata, E. and Olukotun, K.: Exploiting coarse-grain parallelism in the MPEG-2 Algorithm (1998).

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

[6] 小高剛, 中野啓文, 木村啓二, 笠原博徳: OSCAR チップマルチプロセッサ上での MPEG2 エンコーディングの並列処理, ARC2003-154-10 (2003).

[7] 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).

[8] 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).