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
並列処理に関する研究

Study on Parallel Processing
for Multimedia Application
on a Chip Multiprocessor

2005年3月

小 高　剛

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
並列処理に関する研究

Study on Parallel Processing
for Multimedia Application
on a Chip Multiprocessor

2005年3月

早稲田大学大学院理工学研究科
電気工学専攻
アドバンスト・コンピューティング・システム研究

小 高　剛

目 次

図 目 次

表 目 次

第1章

序論

1.1 本研究の背景と目的

近年，ディジタル TV や DVD プレーヤー，ディジタルビデオカメラなどのディジタル情報機器では，マルチメディア処理が重要なアプリケーションの一つとなっている．そのため，マルチメディアアプリケーションを効率的に処理できるデバイスの開発が求められている．これらのデバイスは操作の快適性や高品質の要求などから高パフォーマンスであることに加え，低コスト，低消費電力であることも望まれている．これらの要求を満たすために従来は，ハードウェアによって処理されるマルチメディア処理専用のアクセラレータが用いられてきた．ハードウェアによるマルチメディア処理は，そのアプリケーション専用に作られいるため処理速度が非常に早く，消費電力も低く抑えることができる．ただし，新しいメディア処理の規格への対応では，ハードウェアを設計しなおす必要があるため開発コストが大きくなってしまう．近年のメディア処理は，規格が多様化しており，対応すべきアプリケーション数の増大や，開発期間もコスト削減のために短縮傾向にある．そのため，メディア処理用のデバイスは，ソフトウェアの書き換えにより柔軟な対応ができる汎用プロセッサや FPGA，DRP（Dynamic Reconfigurable Processor）など，プログラマブルなものの利用が期待されている [DD97, LS96].

従来のプロセッサアーキテクチャ上でのマルチメディア処理は，既存の汎用プロセッサに SSE[SVJ00]，VIS[TONH96] のような SIMD 命令を中心としたマルチメディア拡張命令セットの追加や，FR-V[OSS+02] のように複数命令をパックして同時に処理を行なう VLIW を用いて演算の高速化を行なっている．例えば，Sun Ultra SPARC-II ベースのプロセッサ上でのマルチメディアアプリケーションの評価では，命令レベル並列性の利用により 2.3 倍から 4.2 倍の性能向上が得られ，Sun VIS マルチメディア拡張命令セットを利用することによりさらに 1.1 倍から 4.2 倍の性能向上が得られている [RAJ99]．これらのように従来のプロセッサにマルチメディア命令セットを追加することによる高速化では，今後の課題として，1 命令

あたりの演算効率を上昇させる SIMD 命令の追加や柔軟性のある条件分岐命令の追加などの命令レベル並列性を向上させることや，プログラムに存在する命令レベルよりさらに粒度の大きな並列処理粒度を利用することなどが挙げられている [LHL02].

命令レベルよりさらに大きな並列処理粒度を利用し，スケーラブルな性能向上を得るアーキテクチャとしてマルチスレッドやチップマルチプロセッサが次世代プロセッサアーキテクチャとして注目されている．特に，複数の CPU コアを持つチップマルチプロセッサアーキテクチャは，並列性の大きいイタレーションレベルの並列性や，サブルーチン，ループブロック間の粗粒度タスクレベルの並列性に加え，より小さな並列処理粒度のステートメントレベルの近細粒度並列性の利用が可能であり，さまざまな粒度の並列性を柔軟に利用したパフォーマンス向上が行なえる．そのため，今後のチップ内半導体集積度の向上に対しスケーラブルな性能向上が行なえるアーキテクチャであると期待されている．例えば Stanford 大で研究されている Hydra は，ほぼ同数のトランジスタ規模であるスーパースカラアーキテクチャや Simultaneous Multithreading （SMT）アーキテクチャより良いパフォーマンスが得られている [LBK97]．さらに，消費電力の面においてもマイクロプロセッサアーキテクチャにおける従来の低消費電力手法である電圧，周波数のスケーリングを用いるアーキテクチャに比べてチップマルチプロセッサではパフォーマンスの低下なしに電力面でも優れた結果を得られたという報告がある [MM02].

このようにチップマルチプロセッサは低消費電力および性能のスケーラビリティ両面を確保できるアーキテクチャである．しかし，チップマルチプロセッサアーキテクチャはスーパースカラアーキテクチャのようにハードウェアによる並列性の抽出などを行なっていないためチップマルチプロセッサ上において効率的な演算を行なうためにはチップマルチプロセッサに対応したソフトウェア最適化が重

要である．

一方で，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）が提案されている [木村 01, KKOK03]．OSCAR CMP はチップ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．OSCAR CMP が前提とする並列処理方式は，プログラムの持つ階層的な並列性を利用したマルチグレイン並列処理 [HMK00] である．また，データローカリティ利用については共有データを分割し，それら共有データにアクセス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も提案している [吉田 99, IOK01, 中野 03]．ただし，これらの評価は主に科学技術計算プログラムを用いて行なってきた．

近年の産業界におけるアプリケーションとしてマルチメディアアプリケーションが重要な要素となってきているため評価アプリケーションとしてマルチメディアアプリケーションの重要性が増しており，マルチメディアアプリケーションのチップマルチプロセッサ上での処理性能を向上させる手法の研究および，その効果を検討する必要がある．

以上のような背景を踏まえて，本研究では，

(1) マルチメディアアプリケーションのチップマルチプロセッサ上での効率の良い並列処理手法として，メディアアプリケーションの特徴を考慮した複数粒度の並列性を階層的に利用しグローバルな並列性を引き出すマルチグレイン並列処理手法の提案，近年の CPU とメモリの速度差に起因するメモリウォール問題を克服するためにチップ内のローカルメモリを積極的に利用しメモリ利

用効率を向上させるデータローカライゼーション手法の適用，および，CPU と非同期にデータのロード／ストアを行なうデータ転送ユニット（DTU）を用いて CPU でタスク実行時に並行して DTU でデータ転送を行なうオーバーラップデータ転送スケジューリングの提案を行なう．

(2) メディアアプリケーションの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の効果を検証するため，メディアアプリケーションで用いられる基本的なアルゴリズムを用いている静止画像エンコードの JPEG エンコードを用い，マルチグレイン並列性の抽出法の提案を行ない，メディアアプリケーションでのマルチグレイン並列処理が有効であることを示す．

(3) メモリ利用量が多いメディアアプリケーションにおけるメモリアクセスオーバーヘッドの問題解決の有効な手法を提案するために，メモリ利用量が多い MPEG2 エンコードを用いて，並列性の抽出法の提案に加え，チップマルチプロセッサに搭載された CPU 近傍のアクセス速度の速いローカルメモリを積極的に利用しメモリ利用の高効率化を行うデータローカライズ手法の適用法の提案や，CPU と非同期にデータ転送を行うデータ転送ユニットを利用してデータ転送オーバーヘッドの隠蔽を図るプレロード・ポストストア手法を実現するスケジューリング手法の提案を行ない，チップマルチプロセッサ上でその手法が有効であることを示す．

を目的とする．

1.2 本論文の概要

本論文の第 2 章以降の概要を以下に述べる．

第 2 章「JPEG エンコードの並列処理」では，国際標準規格で定められた静止画

像の画像フォーマットで，現在ディジタルカメラや携帯電話など様々な機器で最も一般的に利用されている JPEG におけるエンコード処理のチップマルチプロセッサ上での効率よい処理手法を述べる．本章では，チップマルチプロセッサ上で効率のよい JPEG エンコード処理を行うために JPEG エンコードプログラムからマルチグレイン並列性の抽出法を提案する．マルチグレイン並列処理とは，プログラム全体から抽出可能な並列性を最大限利用する並列処理手法で，ソースプログラムをサブルーチンブロック，繰り返しブロック，基本ブロックの 3 種類の粗粒度タスクに分解し，粗粒度タスク間のデータ依存及び制御依存を考慮してタスク間の並列性を抽出するための最早実行可能条件解析により，各粗粒度タスク間の並列性を抽出し，並列性を示すマクロタスクグラフを生成する．マクロタスクグラフ生成後，各粗粒度タスクをプロセッサあるいは複数のプロセッサをグルーピングしたプロセッサグループに割り当てる．このとき，割り当てられたマクロタスクが，イタレーションレベルで並列性を抽出できるループブロックの場合，プロセッサグループ内プロセッサ間で中粒度並列処理が行われる．また，プロセッサグループに割り当てられたマクロタスクが中粒度並列処理あるいはマクロタスク内部に粗粒度並列処理が適用できない場合，粗粒度タスク中の各ステートメントを近細粒度タスクとして定義し，近細粒度タスク間のデータ依存を解析して並列性を抽出し，プロセッサグループ内プロセッサ間で近細粒度並列処理が行われる．マルチグレイン並列性の抽出では，JPEG エンコードで用いられる基本エンコードデータ単位が 8 × 8 ピクセルブロックであることに注目する．各エンコードステージが 8 × 8 ピクセルブロックを処理するサブルーチンをループで画像全体に対して適用しているため，8 × 8 ピクセルブロックを処理するエンコードステージ間での並列性を抽出し，粗粒度タスク並列処理を行う．このとき，プログラムから可能な並列性をより多く得るためにループアンローリングや粗粒度タスク融合を適用してプログラムのリストラクチャリングを行い，粗粒度タスク並列性を最大限に

引き出す．さらに，粗粒度タスクの 8 × 8 ピクセルブロック処理内部においてもステートメントレベルの近細粒度並列性を抽出し，8 × 8 ピクセルブロック処理間の粗粒度並列性と 8 × 8 ピクセルブロック処理内の近細粒度並列性を階層的に組み合わせるマルチグレイン並列性を利用し，並列性抽出の向上を図る．以上のように抽出した JPEG エンコードのマルチグレイン並列性を OSCAR チップマルチプロセッサ上で評価した結果，プロセッサ数 4 のとき逐次実行に対して，8 × 8 ピクセルブロック処理間の粗粒度並列処理のみを用いた場合 3.36 倍，8 × 8 ピクセルブロック処理内の近細粒度並列処理のみを用いた場合 2.70 倍，粗粒度並列処理と近細粒度並列処理を階層的利用したマルチグレイン並列処理の場合 3.59 倍の速度向上率が得られ，提案したマルチグレイン並列処理手法の有効性が確かめられた．

第 3 章「MPEG2 エンコードの並列処理」では，国際標準規格で定められた動画像フォーマットで，DVD や HDTV など現在のメディア配信で最も一般的に用いられている MPEG2 におけるエンコード処理のチップマルチプロセッサ上での効率よい処理手法について述べる．MPEG2 エンコードで利用するデータ量は非常に大きいため，近年のプロセッサ動作速度とメモリアクセス速度のギャップから生じるメモリウォール問題の深刻化により，アプリケーションの速度向上には並列性の抽出だけでなくメモリアクセスオーバーヘッドの軽減も重要な問題である．そのため，本章では，MPEG2 エンコードの並列性の抽出法の提案に加え，チップマルチプロセッサに搭載された CPU 近傍のアクセス速度の速いローカルメモリを積極的に利用しメモリ利用の高効率化を行うデータローカライズ手法の適用法の提案や，CPU と非同期にデータ転送を行うデータ転送ユニットを利用してデータ転送オーバーヘッドの隠蔽を図るプレロード・ポストストア手法を実現するスケジューリング手法の提案をする．並列性抽出は，MPEG2 データ構造が階層構造をしていることに注目し，データローカリティの利用を考慮しながら，マクロブロックと呼ばれる 16 × 16 ピクセルブロック単位を処理単位としたマクロブロックレベル

処理を粗粒度タスクとして定義し粗粒度並列処理を行う．このマクロブロックレベルの並列性を抽出するとき，データローカリティの利用のため，エンコードステージ全体をマクロブロックレベル処理に分割し，分割したマクロブロックレベル処理の粗粒度タスクの実行順番を同じマクロブロックを処理する粗粒度タスクを連続して実行するようにスケジューリングを行う．これにより，各粗粒度タスク間の共有データをアクセス速度の速いローカルメモリを介して授受できるためデータアクセスオーバヘッドの削減が可能となる．次に，初期データのロードや演算結果のストアなどで発生するオーバーヘッドの大きいチップ外メモリへのアクセスを削減するために，CPU 実行と非同期にデータ転送が可能なデータ転送ユニットを利用したオーバーラップデータ転送を行う．オーバーラップデータ転送とは，あるタスクに必要なデータのロードをそのタスクの実行開始前にその時点でプロセッサが実行しているタスクとオーバーラップしてデータ転送ユニットが当該データをロードしたり，あるタスクのデータストアを後続タスクの実行とオーバーラップしてデータ転送ユニットが当該データをストアすることによりデータ転送オーバーヘッドを削減する手法である．以上のように，MPEG2 エンコードに提案手法を適用し，OSCAR チップマルチプロセッサ上にて評価した結果，データローカライゼーションとオーバーラップデータ転送を含む粗粒度並列処理の提案手法は，逐次実行に対し，1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 2.46 倍，4 プロセッサ利用時 4.57 倍，8 プロセッサ利用時 7.97 倍，16 プロセッサ利用時 11.93 倍の速度向上率が得られ，提案した MPEG2 エンコードの並列処理手法はチップマルチプロセッサ上で効果的な並列処理を実現できることが確認できた．

第 4 章「結論」では，本研究により得られた成果と今後の課題について述べる．

第2章

JPEGエンコードの並列処理

2.1 まえがき

JPEG とは，静止画像の符号化標準であり，主にモノクロームやフルカラーの自然画像を符号化するように設計された画像フォーマットである．一般的には，非可逆符号化が用いられ，そのアルゴリズムは，画像を固定サイズのブロックに分割し，そのブロック単位で、離散コサイン変換（DCT: Discrete Cosine Transform）を用いて、空間領域から周波数領域へ変換する．変換されたデータは，量子化によって情報量を落としてから，ハフマン符号によるデータの生起確率の高低に応じて異なる長さの符号を割り当てデータ圧縮を行なうエントロピー符号化がなされ圧縮が行われる．現在，主に使われている MPEG2 や H.264 などのメディアアプリケーションでは，JPEG のようにブロック分割したデータへの符号化適用，DCT，量子化，ハフマン符号化アルゴリズムが用いられている．

本章では，メディアアプリケーションのマルチグレイン並列処理手法の提案およびチップマルチプロセッサ上での性能を確認するために，離散コサイン変換や量子化，エントロピー符号化といったマルチメディア処理で多用されるアルゴリズムを利用した画像圧縮処理である JPEG エンコードを対象アプリケーションとし，JPEG エンコードの特徴をとらえたマルチグレイン並列処理手法を提案を行なう．そして，提案するマルチグレイン並列処理手法を適用した JPEG エンコードを OSCAR チップマルチプロセッサ（OSCAR CMP）上で評価し，その結果について述べる．

2.2 マルチグレイン並列処理

ここでは，OSCAR チップマルチプロセッサ上で扱う並列処理技術であるマルチグレイン並列処理について説明する．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性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理）[HMK00]，ループイタレーション間の並列性を利用する中粒度並列処理（ループ並列処理），基本ブロック内部のステートメント間の並列性を利用する近細粒度並列処理 [木村 01] を階層的に組み合わせて使用し，プログラム全域に渡る並列性を有効に引き出して利用する手法である．

2.2.1 粗粒度タスク並列処理（マクロデータフロー処理）

粗粒度タスク並列処理では，はじめに，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 としたものである．

MT 生成後，コンパイラは BPA，RB，SB 等の MT 間の制御フローとデータ依存を解析し，結果を図 2.1 に示すようなマクロフローグラフ（MFG）[HMK00] として表す．図 2.1 において各ノードは MT を表し，ノード中の小円は条件分岐を表す．ノード間の点線エッジ，実線エッジはそれぞれマクロタスク間の制御フローおよびデータ依存を表す．なお，図中のエッジの矢印は省略されているが，エッジはすべて下向きを仮定している．

MFG 生成後，MFG から MT 間の並列性を最大限に抽出するためにデータ依存と制御依存を考慮し，各 MT が最も早い時点で実行可能となる条件解析である最早実行可能条件解析 [HMK00] が行なわれる．その結果は，図 2.2 に示すようなマ

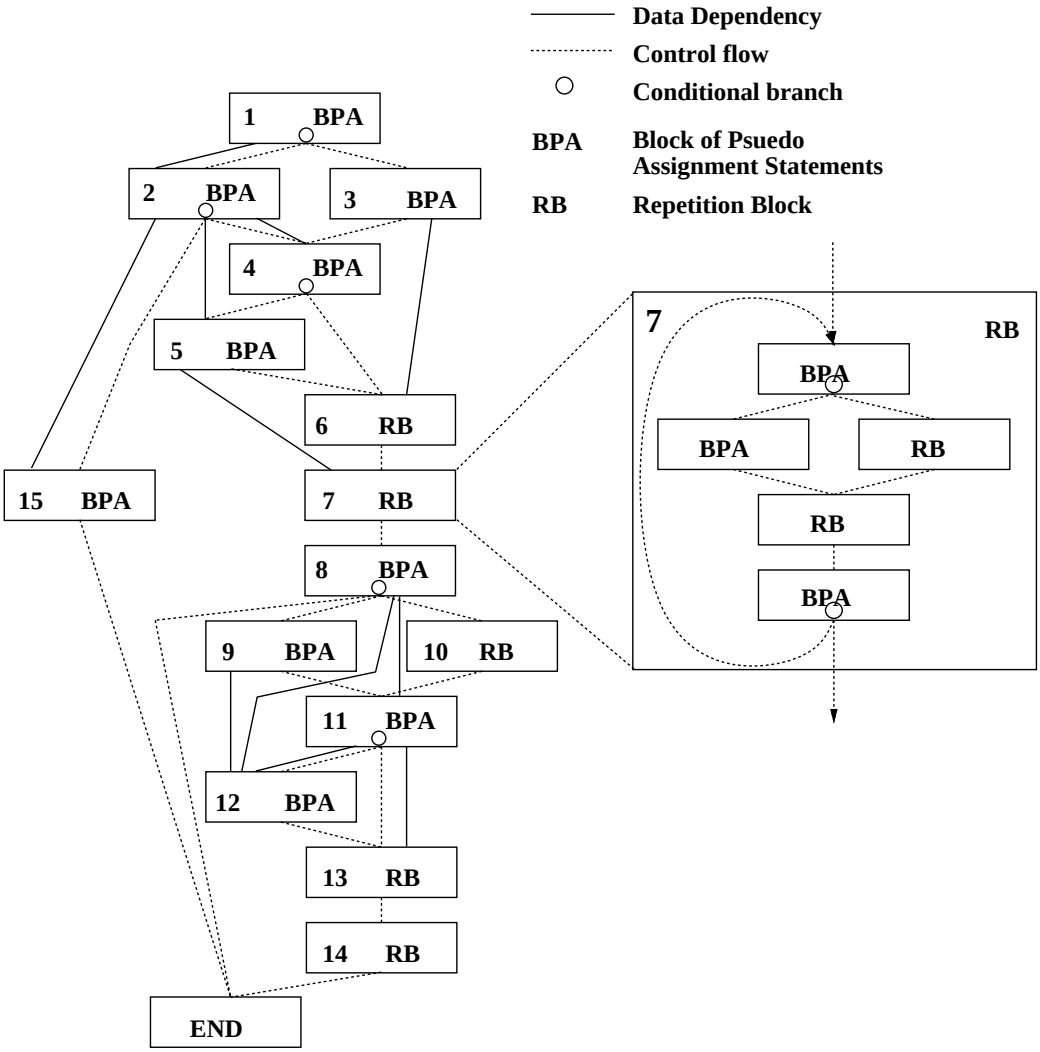


図 2.1: マクロフローグラフ（MFG）

クロタスクグラフ（MTG）[HMK00] として表現される．図 2.2 においても各ノードは MT を表し，ノード中の小円は条件分岐を表す．ノード間の点線エッジ，実線エッジはそれぞれ拡張された制御依存およびデータ依存を表す．ここで，拡張された制御依存とは，通常の制御依存だけでなく MT_i のデータ依存先行 MT が実行されない条件も含むものである．エッジを束ねる実線アークは，アークによって束ねられたエッジが AND 関係にあることを表し，点線アークは OR 関係にあることを表す．MTG においてもエッジの向きは下向きを仮定しておりほとんどの矢印は省略されている．矢印がついているエッジは，元の MTG 上での条件分岐を表すエッジである．

MTG 生成後，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）にスケジューリングする．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．

2.2.2 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．

2.2.3 近細粒度並列処理

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 により近細粒度並列処理 [木村 01] される．

近細粒度並列処理では，基本的に MT 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

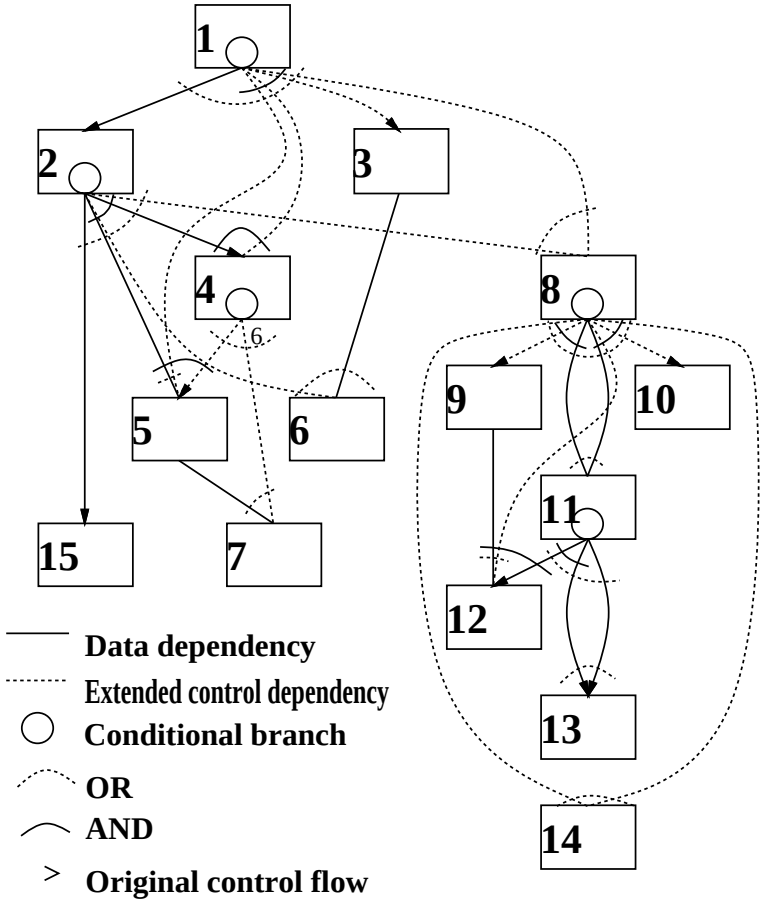


図 2.2: マクロタスクグラフ（MTG）

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各タスク間のデータ依存，すなわち先行制約を表したタスクグラフを作成する．例として，クラウト法によるスパース行列の求解をシンボリックジェネレーション法を用いてループフリーコードに展開して行なう図 2.3 のプログラムの各ステートメントを近細粒度タスクとして定義したときのタスクグラフを図 2.4 に示す．図 2.4 においてノード内の各数字はタスク番号 i を表し，ノード脇の数字は PE 上でのタスク処理時間 t_i を表す．また，ノード N_i から N_j に向けて引かれたエッジはタスク T_i がタスク T_j に先行するという半順序制約を表している．タスク間のデータ転送を考慮するとき，各々のエッジは一般に可変な重みを持つ．タスク T_i とタスク T_j が異なる PE へ割り当てられた場合，エッジの重み t_{ij} がデータ転送時間となる．逆にこれらのタスクが同一 PE に割り当てられた場合，タスク間のデータの授受はプロセッサコアのレジスタを介して行なわれるため重み t_{ij} は 0 となる．

タスクグラフ生成後，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．スケジューリングの際，データのローカルメモリ，分散共有メモリ，レジスタへの配置などデータ配置の最適化やデータ転送・同期オーバーヘッドの最小化といった各種最適化が行なわれる．

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．マシンコード生成時，スタティックスケジューリングの情報を用いたコード最適化が行なわれる．例えば，同一データを使用する異なるタスクが同一 PE に割り当てられたとき，そのデータをレジスタを介して受渡しを行なうことやタスク割り当て状況や実行順序から冗長な同期を除去することなどである．

```
<< LU Decomposition >>
1)  u12 = a12 / I11
2)  u24 = a24 / I22
3)  u34 = a34 / I33
4)  I54 = I52 * u24
5)  u45 = a45 / I44
6)  I55 = a55 / I54 * u45
<< Forward Substitution >>
7)  y1 = b1 / I11
8)  y2 = b2 / I22
9)  b5 = b5 / I52 * y2
10) y3 = b3 / I33
11) y4 = b4 / I44
12) b5 = b5 / I54 * y4
13) y5 = b5 / I55
<< Backward Substitution >>
14) x4 = y4 / u45 * y5
15) x3 = y3 / u34 * x4
16) x2 = y2 / u24 * x4
17) x1 = y1 / u12 * x2
```

図 2.3: 近細粒度タスクの例

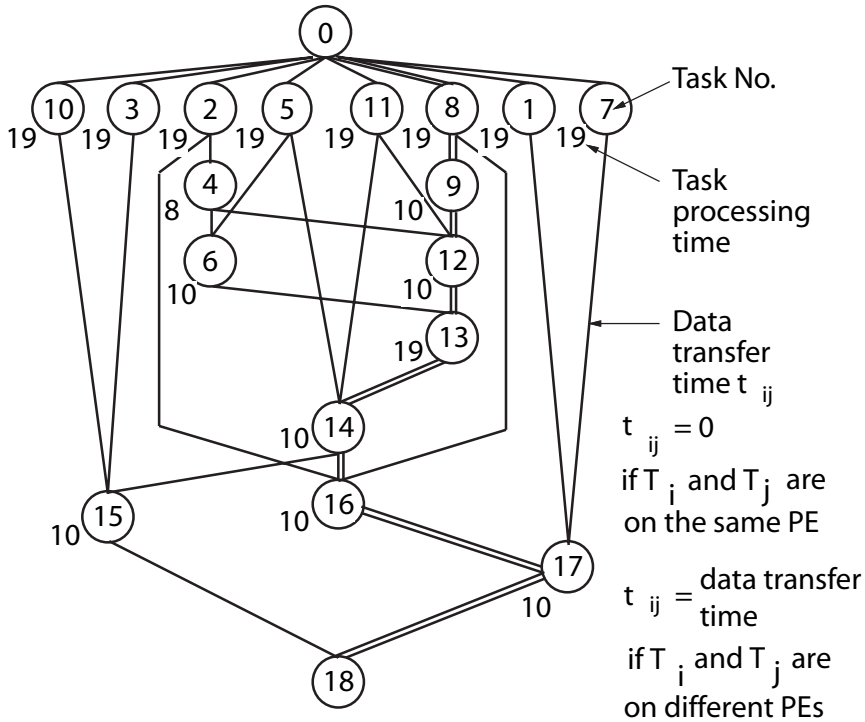


図 2.4: 近細粒度タスクのタスクグラフ

2.3 OSCAR チップマルチプロセッサアーキテクチャ

ここでは，対象アーキテクチャであるマルチグレイン並列処理と協調しスケーラブルな性能を得る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）[KKOK03] 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，既存のプロセッサのトランジスタ数を参照しながらその回路規模について概算を行ない，OSCAR CMP の回路規模を推定する．

2.3.1 メモリアーキテクチャ

OSCAR CMP のネットワークおよびメモリアーキテクチャは，図 2.5 に示すように複数のプロセッサエレメント（PE）をバスやクロスバなどの相互結合網で接続し 1 チップ上に搭載したアーキテクチャで，各 PE は，演算を行なう 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），アドレスがグローバルアドレス空間にマップされており自 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）で構成されている．また，集中共有メモリ（CSM）を PE 外部に接続し各 PE で共有するデータなどを格納する．OSCAR CMP では，以上のようなメモリアーキテクチャ構成をマルチグレイン並列処理と協調しながら利用しスケーラブルな性能を得る．

粗粒度タスク並列処理では，LDM は各 MT 内でプライベート化できるデータを格納する．DSM は，違う PE に割り当てられた MT 間での共有データの格納や MT 間同期用フラグを DSM 格納することによりビジーウェイトを PE 内部で行なえるため PE 間ネットワークに影響を与えない同期を行なうことができる．また，CSM は，DSM に格納できないような MT 間で共有するサイズの大きなデータを格納し

たり，動的な MT 割当を行なうダイナミックスケジューリング時の MT 間共有データを格納する．DTU は，MT のタスク処理とデータ転送をオーバーラップしデータ転送オーバーヘッドを隠蔽するために利用される．

ループ並列処理では，LDM はイタレーション間でプライベートなデータの格納する．DSM は，Doacross ループやリダクションループにおけるプロセッサ間で転送が必要なデータを格納することにより高速なチップ内通信による低レイテンシデータ転送を可能とする．

近細粒度並列処理では，LDM は他の並列処理と同様プロセッサ固有のデータを格納する．近細粒度タスクは，粒度が小さいためコンパイル時でのスケジューリングと実際のプロセッサの動作の誤差が性能に大きな影響を与える．そのため，動作の予測可能なシンプルなプロセッサコアを必要とする．また，各タスクの実行コストが小さいためデータ転送オーバーヘッドの影響を受けやすいので，低レイテンシデータ転送を可能とする PE 内 DSM に近細粒度タスク間での共有データを格納し，同様に同期オーバーヘッドによる影響を低く抑えるために DSM に近細粒度タスクの同期フラグを格納する．

2.3.2 プロセッサコアアーキテクチャ

本論文では各 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[Sun97] 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を仮定する．

演算ユニットは，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 本持った 9 段パイプラインのシングルイシューというシンプルな構成とした．表 2.1 にプロセッサコアの仕様を示す．なお，比較としてスーパースカラプロセッサの UltraSPARC-II 相当のプロセッサ（4-issue）

の構成の場合も示す．4-issue のプロセッサは，IEU を 2 本，LSU を 1 本，FPU を 2 本持つ 4 イシュー in-order 発行構成で，動的スケジューリング用命令バッファエントリ数は 12 である．

2.3.3 トランジスタ数の概算

ここでは，OSCAR CMP の回路規模を確認するために既存のプロセッサを基に概算を行なう．

プロセッサコアのようなランダムロジックが必要とするトランジスタ数は，論理回路の最適化の度合や高速化のために費やす回路などにより大きく変化するが，ここではプロセッサコアに要するトランジスタ数を式 2.1[BK98] で概算値を求め推定する．ただし，ここでは簡単のために I/O ドライバに関しては扱わない．また，キャッシュのトランジスタ数の概算は式 2.2[BK98] で概算を行なう．

プロセッサコアのトランジスタ数

＝ チップの総トランジスタ数

－ キャッシュのトランジスタ数－I/O ドライバのトランジスタ数 (2.1)

キャッシュのトランジスタ数

＝ data array＋tag array＋LRU array＋data MUX＋tag comparators (2.2)

ここで各 array は，SRAM セルのメモリアレイを示し 1 セル（=1bit）6 トランジスタで構成される．データマルチプレクサ（data MUX）および tag comparators のトランジスタ数は，それぞれ式 2.3，式 2.4 により求められる [BK98].

$$\text{data MUX} = \text{associativity} \times (2 + \text{assosiativity}) \quad (2.3)$$

$$\text{tag comparators} = \text{assosiativity} \times (\text{tag} + \text{status}) \times 4 \quad (2.4)$$

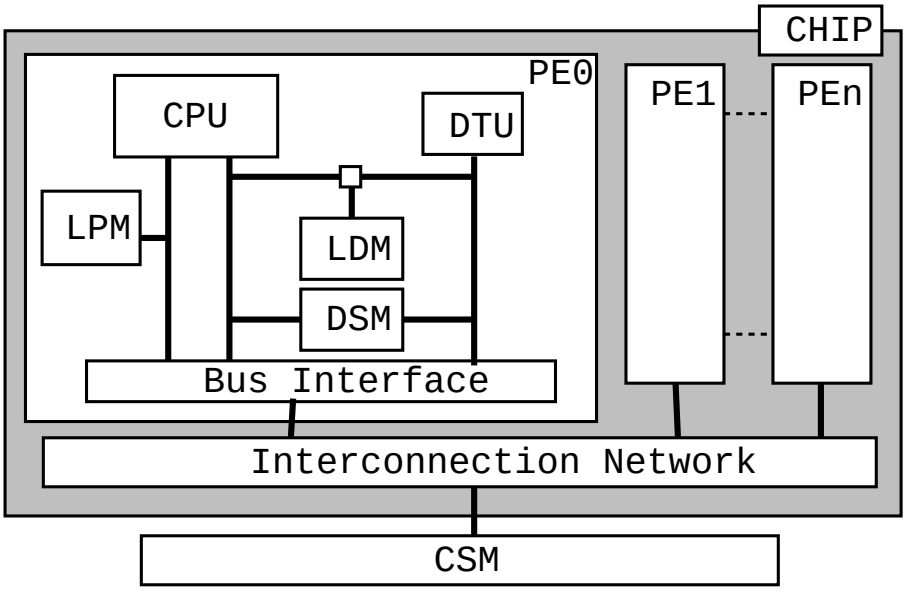


図 2.5: OSCAR チップマルチプロセッサアーキテクチャ

表 2.1: プロセッサコア仕様

	OSCAR CMP	4-issue
パイプライン段数	9	
同時命令発行数	1	4
IEU	1	2
FPU	1	2
LSU	1	1
命令発行タイプ	in-order	

参照するプロセッサとしては，シングルイシュープロセッサコアの推定に microSPARC [Tex00] を用い，また，比較としてに UltraSPARC-II 相当の 4 イシュー in-order スーパースカラプロセッサコアの推定には UltraSPARC[Le95] を用いる．各プロセッサの仕様，および算出したトランジスタ数を表 2.2 に示す．

以上のプロセッサコアの推定値をもとにプロセッサコアと DSM のトランジスタ数を合計し，各アーキテクチャのトランジスタ数として表 2.3 に示す．LDM は各アーキテクチャで同容量としているのでここでは扱わない．また，バスはトライステートバッファのみ，調停回路は Bus Interface のみで構成されるとし簡単のためにここでのトランジスタ数見積りでは考慮しない．表 2.3 より，シングルイシュープロセッサを 4 基搭載した OSCAR CMP と UltraSPARC-II 相当の 4 イシュー in-order スーパースカラアーキテクチャ（4-issue）は，ほぼ同程度のトランジスタ数であると推定できる．

2.4 JPEG エンコードアルゴリズム

ここでは，JPEG エンコードアルゴリズムの説明をする．

本論文で扱う JPEG エンコードアルゴリズム [E. 92, Ald92] は，MediaBench [CMW97] に収録されている JPEG ベンチマークプログラムである “ jpeg-v6a” をベースとする．そのアルゴリズムは，図 2.6 のように，6 つのエンコードステージからなる．各ステージの処理概要は，以下である．

8×8 ピクセルブロック分割

入力画像データを JPEG 基本処理単位である 8×8 ピクセルブロックに分割する．

YCbCr 変換

分割された 8×8 ピクセルブロックのデータを JPEG のデータフォーマット

表 2.2: プロセッサコアのトランジスタ数推定

チップ	microSPARC	UltraSPARC
総トランジスタ （万）	80	520
命令キャッシュ （KByte）	4	16
データキャッシュ （KByte）	2	16
ライン （Byte）	32（I）/16（D）	32
キャッシュのトランジスタ （万）	35	244
コアのトランジスタ （万）	45	276

表 2.3: 各アーキテクチャのトランジスタ数推定

アーキテクチャ	OSCAR CMP （4PE）	4-issue
コアのトランジスタ （万）	180	276
DSM のトランジスタ （万）	104	0
総トランジスタ （万）	284	276

2.5. チップマルチプロセッサ上での JPEG エンコードのマルチグレイン並列処理

である YCbCr フォーマットへ変換する．

2 次元離散コサイン変換処理

YCbCr フォーマットへ変換された 8×8 ピクセルブロックデータに対し 2 次元離散コサイン変換により画像信号を空間周波数へ変換する．なお，変換後の 2 次元配列 S_{ij} は，S_{00} から S_{77} までの 64 要素存在し，S_{00} を直流（DC）係数，その他の 63 要素は交流（AC）係数と呼ばれる．ここで，S_{ij} は i または j が大きいほど高い周波数成分に相当する．

一様量子化

量子化テーブルを用い DCT 係数に対し 1 次元一様量子化を行なう．

量子化 DC 係数 1 次元予測

処理を行なっている 8×8 ピクセルブロックの量子化 DC 係数と 1 つ前の 8×8 ピクセルブロックの量子化 DC 係数を減算し 1 次元 DC 予測値を計算する

エントロピー符号化

1 次元 DC 予測値および量子化 AC 係数を可変長符号を用いてエントロピー符合化を行なう．

2.5 チップマルチプロセッサ上での JPEG エンコードのマルチグレイン並列処理

ここでは，JPEG エンコードの並列処理における特性を解析し，チップマルチプロセッサ上での JPEG エンコードのマルチグレイン並列処理手法について述べる．

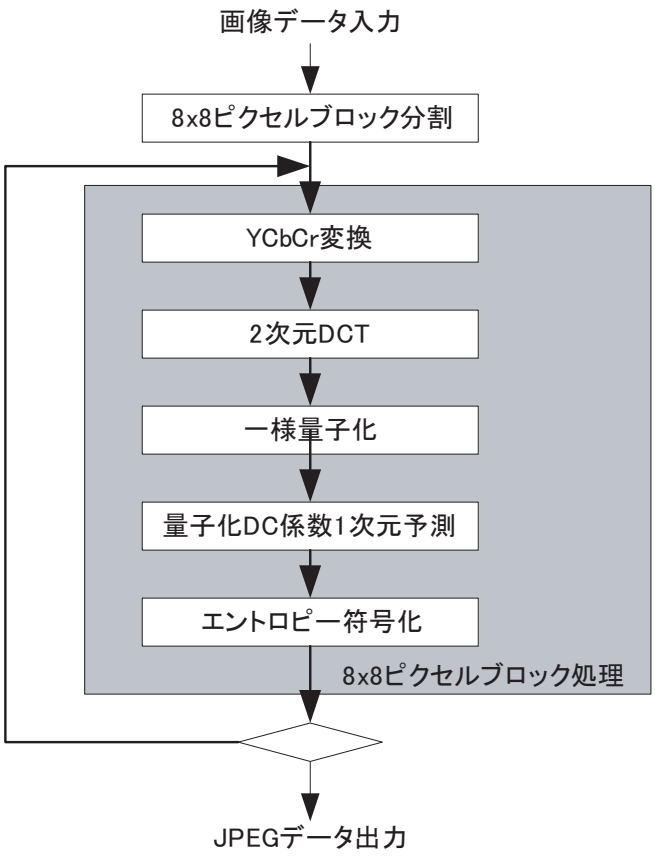


図 2.6: JPEG エンコード実行ブロック図

2.5.1 粗粒度並列性の抽出

2.4 節で述べた基本的な JPEG エンコードアルゴリズムでは，量子化 DC 係数 1 次元予測ステージで，エンコード対象の 8×8 ピクセルブロックの量子化 DC 係数と 1 つ前の 8×8 ピクセルブロックの量子化 DC 係数を用いて演算を行なうため，8×8 ピクセルブロック間でのデータ依存が存在する．しかし，その他の YCbCr 変換ステージ，DCT ステージおよびエントロピー符号化ステージでは 8×8 ピクセルブロック内のデータのみ扱うため，これらのステージでは 8×8 ピクセルブロック間のデータ依存は存在しない．これらのデータ依存を考慮し，並列処理を効果的に行なうために量子化 DC 係数 1 次元予測ステージとエントロピー符号化ステージの中で量子化 DC 係数 1 次元予測ステージの演算結果を用いる DC 係数エントロピー符号化とその他の AC 係数エントロピー符号化を各々エンコードステージとして分離しする．そして，図 2.7 のように YCbCr 変換ステージ，2 次元 DCT ステージ，一様量子化ステージおよび AC 係数エントロピー符号化ステージを一つのグループとして粗粒度タスク MT_a と定義し，量子化 DC 係数 1 次元予測ステージおよび DC 係数エントロピー符号化ステージを一つのグループとして粗粒度タスク MT_b と定義する．このように粗粒度タスクを定義することにより，AC 係数エンコードと DC 係数エンコードを分離することができ AC 係数エンコード時は MT_a の各 8×8 ピクセルブロック処理間で並列処理が行なえ，DC 係数エンコード時は MT_b の各 8×8 ピクセルブロック処理間で並列処理可能となる．

MT_a，MT_b の各 PG への割り当ては，各 PG で処理される MT_a および MT_b の数が等しくなるように割り当る．その際，データ転送の最小化を考慮し同じ 8×8 ピクセルブロックを同一 PG 内で処理できるように MT_a，MT_b のスケジューリングを行ないスタティックに各 MT を割り当てる．例えば，16x16 ピクセルの入力データを 4 つの PG で実行する場合の各 MT の割り当ては，図 2.8 に示したようになる．ここで，MT_{ai} および MT_{bi} は，入力データを 8×8 ピクセルブロックに分割したと

きシーケンシャルスキャン順での i 番目のブロックである block i を処理する MT に相当する．このときの実行イメージは，まず，入力データは block1 から block4 の 4 つの 8×8 ブロックに分割され，PG1 では block1 を処理する MT_{a1}，PG2 では block2 を処理する MT_{a2}，PG3 では block3 を処理する MT_{a3}，PG4 では block4 を処理する MT_{a4} が各々の PG で実行される．このとき，各 MT_{ai} 間では依存がないためこれらの MT_{ai} は全 PG で並列に処理される．すべての MT_{ai} が実行された後，MT_{bi} 実行前に，PG2 の MT_{b2} で必要とされる block1 の量子化 DC 係数を PG1 から PG2 へ転送しする．他の PG3，PG4 でもそれぞれ MT_{b3}，MT_{b4} で必要とされるデータが転送される．データ転送後，各 MT_{bi} は全 PG で並列に処理される．

この粗粒度タスク並列性の抽出は一般最適化により抽出可能である．まず，図 2.9（a）のようにループにより表現された JPEG エンコード処理をループアンローリングにより展開する（図 2.9（b））．ループアンローリング後，1 つの 8×8 ピクセルブロックを処理する YCbCr 変換，2 次元 DCT，一様量子化，AC 係数エントロピー符号化，量子化 DC 係数 1 次元予測，DC 係数エントロピー符号化の各ステージをそれぞれ粗粒度タスクとして定義し，粗粒度タスク間並列性抽出を行う．図図 2.9（c）に，ループアンローリング後の粗粒度タスク間並列性抽出結果を表したマクロタスクグラフを示す．各ノードは粗粒度タスクを示しノード内の番号は図 2.9（b）の各処理に対応する．また，ノードから出る実線はデータ依存を表す．この粗粒度タスク間並列性抽出結果より粗粒度タスクを網掛け部分の粗粒度タスクを 1 つの粗粒度タスクとして融合することにより図 2.9（d）のようながマクロタスクグラフが得られ，本章で提案した粗粒度並列処理が可能となる．

2.5.2 近細粒度並列性の抽出

2.5.1 節にて，粗粒度タスク並列性の抽出を行なったが，粗粒度タスク内においてもプログラム中のステートメント間での並列性は存在し，粗粒度タスク内でス

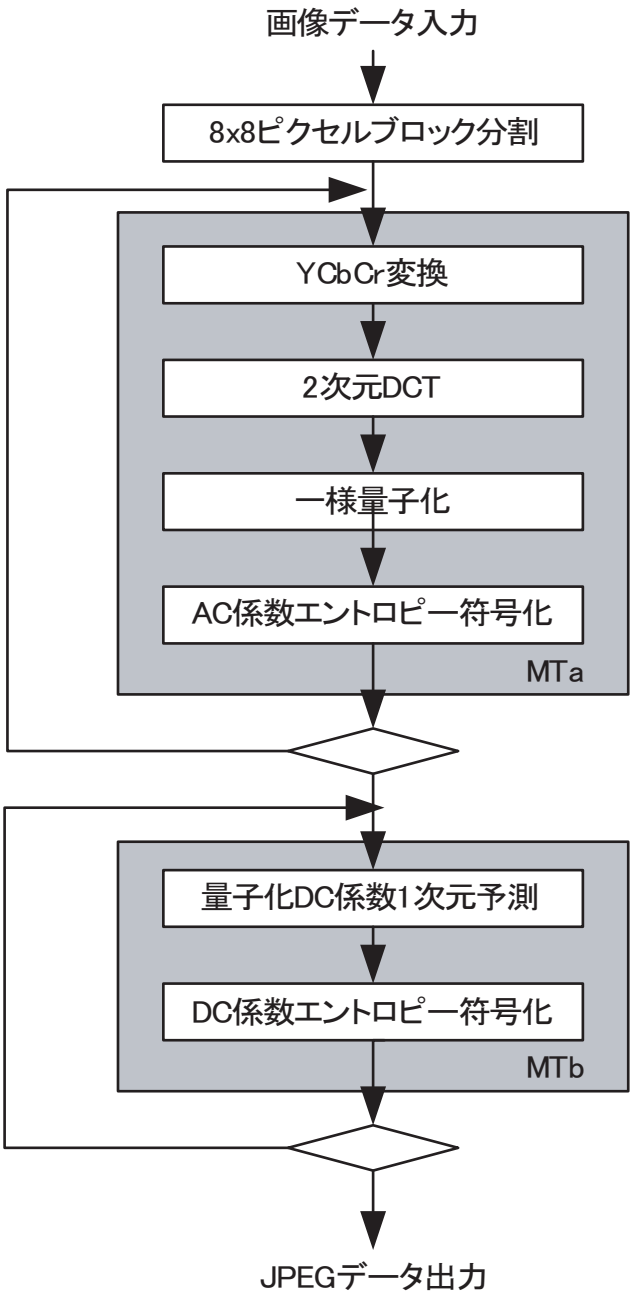


図 2.7: 並列化 JPEG エンコードブロック図

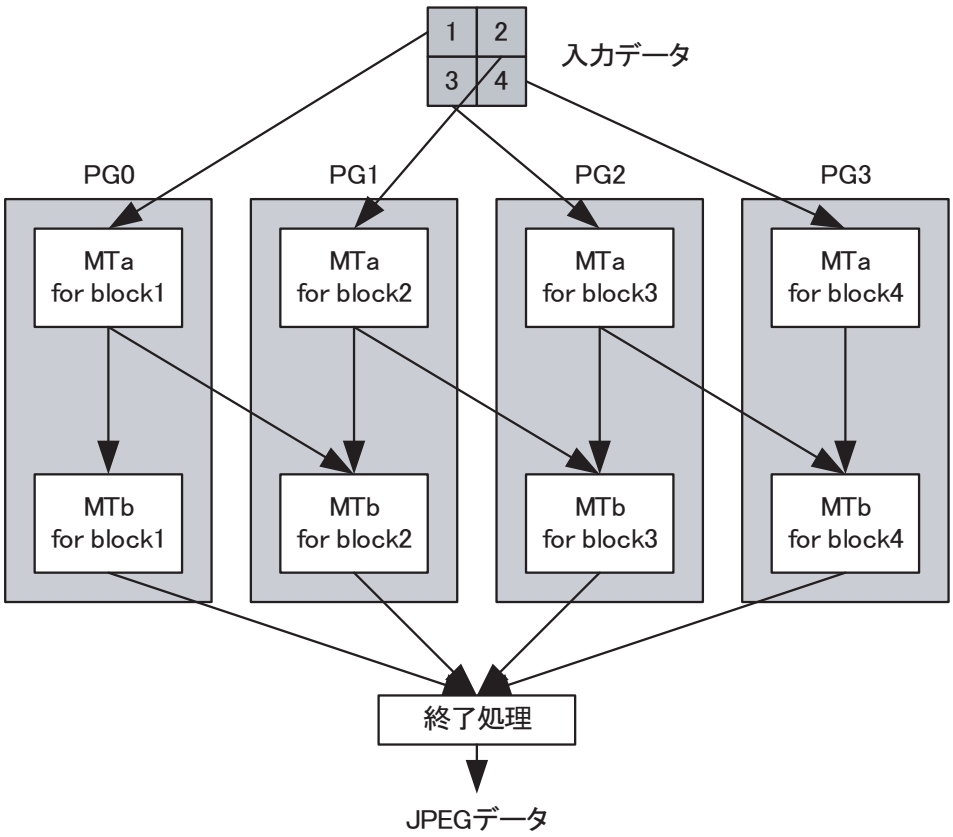


図 2.8: 並列化 JPEG エンコード MT 割り当てイメージ

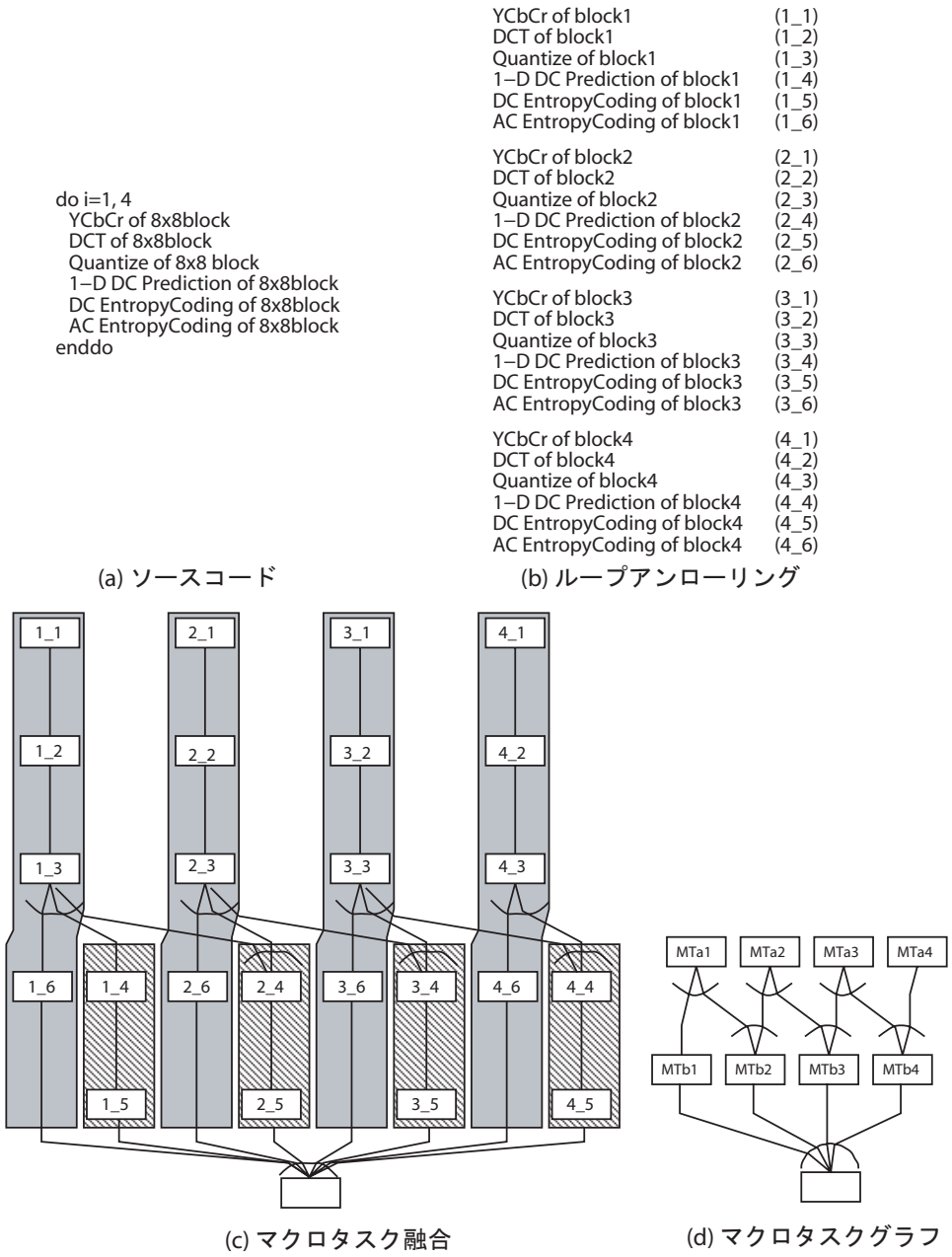


図 2.9: 粗粒度並列性抽出

テートメント間の並列性を利用する近細粒度並列処理を行なうことにより，さらに有効な並列処理が可能となる．

ここでは，2.5.1 節で抽出を行なった JPEG エンコードの粗粒度タスク MT_a，MT_b 内の近細粒度タスク定義およびスケジューリングについて述べる．

YCbCr 変換ステージの近細粒度並列処理

YCbCr 変換の変換式は，入力信号により違うが本論文では一般的に用いられる RGB 信号からの変換を使用する．RGB から YCbCr への変換は，式 2.5，2.6，2.7 で表される．ここで式中 i，j は $0 \leq i \leq 7, 0 \leq j \leq 7$ である．

$$Y_{ij} = R_{ij} * 0.29 + G_{ij} * 0.58 + B_{ij} * 0.11 \quad (2.5)$$

$$Cb_{ij} = -R_{ij} * 0.16 - G_{ij} * 0.33 + B_{ij} * 0.50 \quad (2.6)$$

$$Cr_{ij} = R_{ij} * 0.50 - G_{ij} * 0.41 - B_{ij} * 0.08 \quad (2.7)$$

jpeg-v6a では，この式をループによる繰り返しにより各要素の演算を行なっているが近細粒度並列性を高めるため今回の評価ではループの展開を行なう．各ステートメントを近細粒度タスクとして定義すると式よりそれぞれの近細粒度タスク間にはデータ依存が存在しない．そのため並列処理可能であることが分かる．

2 次元 DCT ステージの近細粒度並列処理

2 次元離散コサイン変換（DCT）は，1 次元 DCT を行方向へ 1 回，列方向へ 1 回処理することにより実現される．jpeg-v6a では，この処理を 1 次元処理をループによる繰り返し実行を行なうことにより実現している．この特徴は，行または列への DCT 演算時には各行または列間での 1 次元 DCT ではデータ依存がないことを意味する．また，潜在的に存在するステートメント間の並列性を利用するためにループを展開し基本ブロックを大きくし並列性の抽出範囲を大きくした．この

ようにしてプログラムのリストラクチャリングを行なった後，各ステートメントを近細粒度タスクとして定義し並列性の抽出を行なう．並列性抽出後の 2 次元離散コサイン変換のタスクグラフを図 2.10 に示す．各ノードは近細粒度タスクを示しノードより出る実線はデータ依存を表す．図のように，ループを展開することによって行方向 1 次元 DCT 処理中では各行方向の処理間ではデータ依存が存在しないため並列処理可能なタスクが多数存在していることが分かる．列方向 1 次元 DCT 処理中に関しても同様である．

一様量子化ステージの近細粒度並列処理

一様量子化処理は，式 2.8 で示される．ここで式中 i，j は $0 \leq i \leq 7, 0 \leq j \leq 7$ である．

$$Sq_{ij} = round\left(\frac{S_{ij}}{q_{ij}}\right) \tag{2.8}$$

式 2.8 より，YCbCr 変換同様この場合も，要素（i,j）間ではデータ依存はないためループを展開し各ステートメントを近細粒度タスクとして定義することでタスク間でのデータ依存は存在せず並列実行可能である．

エントロピー符号化ステージの近細粒度並列処理

エントロピー符号化処理は，1 つの量子化 DCT 係数を符号化する処理は図 2.11（b）のように IF-THEN-ELSE により記述された処理になる．データ転送のオーバーヘッドや同期オーバーヘッドなどを考慮し，この IF-THEN-ELSE 文で囲まれた複数のステートメントを疑似代入ステートメントとして扱い近細粒度タスクとして定義する．エントロピー符号化処理は，非 0 係数と非 0 係数が出現するまでの間の 0 の連続値を用いて符号化を行なうため，図 2.11（b）のように直前までの 0 の連続値によるデータ依存が存在し並列性の抽出が難しい．そこで，図 2.11（c）のよ

うにエントロピー符号化処理を仮分割し並列性を抽出する．この手法は，ある量子化 DCT 係数のエントロピー符号化処理のところで仮分割を行なうことにより，分割された処理はそれぞれデータ依存がなくなり並列実行可能とするものである．ただし，分割境界では 0 の連続値が分離されてしまうため処理の最後に境界部分の値を補正する処理を行なう必要がある．

仮分割の際の分割数は，各プロセッサでのロードバランスを考慮して近細粒度並列処理を行なうプロセッサエレメント数で分割し分割境界より前の一連の処理と後の一連の処理とで演算コストが均等になるように分割を行なう．しかし，量子化 DCT 係数が 0 か非 0 かにより処理内容が異なるためタスク演算コストの推定が難しい．また，JPEG エンコードに用いられる入力画像は一般に自然画像を用いるため量子化 DCT 係数は DCT による空間周波数変換により低周波数成分に非 0 係数が偏り高周波成分は 0 になる傾向があるため，複数の入力画像においてプロファイルをとり，その結果に基づき各タスクの演算コストを見積もった．

近細粒度タスクのスケジューリング

近細粒度タスクの定義後，近細粒度タスクはスケジューリングにより 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 [木村 01].

2.6 性能評価

ここでは，JPEG エンコードプログラムに 2.5 節で提案したマルチグレイン並列処理手法を適用し OSCAR チップマルチプロセッサ（OSCAR CMP）上にて評価し

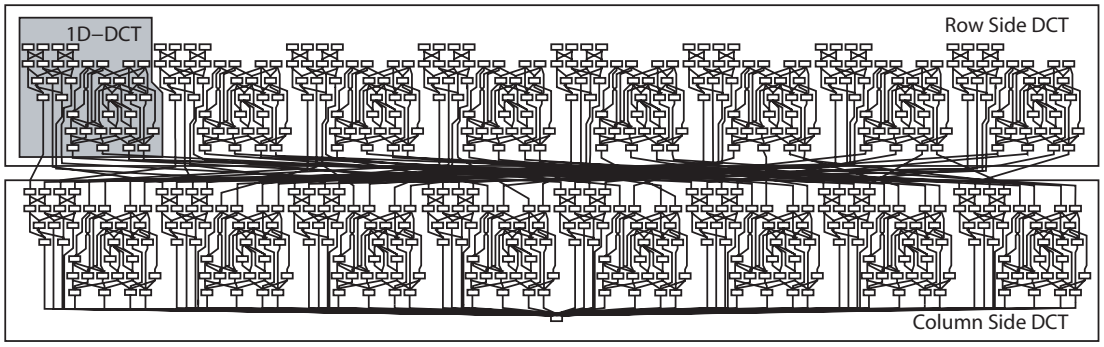


図 2.10: 2 次元 DCT タスクグラフ

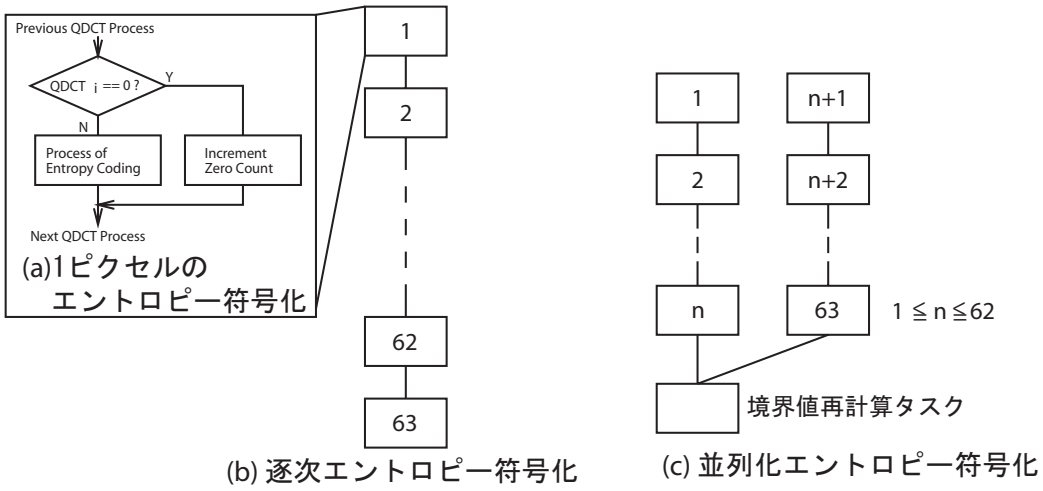


図 2.11: エントロピー符号化処理の近細粒度タスク

た結果について述べる．

2.6.1 評価条件

評価に用いるアーキテクチャは，OSCAR CMP で，実行には OSCAR CMP をクロックレベルでシミュレートする詳細なシミュレータを用いた．OSCAR CMP のパラメータとなる各メモリサイズ，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 JPEG エンコード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．また，比較参考として CPU コアを 4 イシュースーパースカラプロセッサとした場合の評価も行なう．CPU コアの仕様は，表 2.1 に示したものである．

評価に用いるプログラムは，プログラムソースレベルで 2.5 節で提案したマルチグレイン並列性抽出のためのループアンローリング，粗粒度タスク融合，エントロピーコーディングのプログラムリストラクチャリングを適用し，タスク粒度をで 2.5 節で述べた形に成形したプログラムソースを 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い OSCAR CMP 用バイナリを作成した．

入力画像は，MediaBench で用いられる入力画像（testimg.ppm:自然画像）をシミュレーション時間短縮のために 32x32 ピクセルに縮小した画像とし，CSM 上に配置した．OSCAR CMP でのデータ配置は，プログラム開始後，CSM 上に置いている入力画像データを各 PE 内の LDM へコピーし，エンコード処理は各 PE 内の LDM で行い，データ転送は DSM を用いた PE 間データ転送により行われる．処理完了後，処理結果は CSM へ格納する．

2.6.2 評価結果

OSCAR CMP 上での評価結果を逐次実行時間に対する速度向上率として図 2.12 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，OSCAR CMP 中の総 PE 数は $m \times n$ であることを示す．この mPGnPE という構成は，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．次に図中の 4-issue は，4 イシュースーパースカラプロセッサによる速度向上率である．ただし，ここでの 4 イシュースーパスカラプロセッサの評価に関しては，OSCAR コンパイラがスーパースカラプロセッサの性能を最大限に引き出すための最適化を行っていないため参考値として示す．

まず，4 イシュースーパースカラプロセッサを用いた場合，シングルイシュープロセッサの逐次実行時間に対して約 1.25 倍の速度向上率を得た．ほぼ同等のトランジスタ数であるシングルイシュープロセッサコアを 4 基使用した OSCAR CMP での 2PG2PE 実行は，逐次実行時間に対して約 3.59 倍の速度向上率が得られ，4 イシュースーパースカラプロセッサに対しても約 2.87 倍の性能向上が得られたことが分かる．これは，JPEG エンコード処理は積和演算を多用するため乗算などのマルチサイクル命令によるパイプラインストールが発生し 4 イシュースーパースカラプロセッサでは十分な性能が出せないがシングルイシュープロセッサコアを複数搭載した OSCAR CMP ではプロセッサ間負荷分散により効果的な並列処理が行なえるためである．

次に，OSCAR CMP での粗粒度並列処理単体性能と近細粒度並列処理単体性能，および粗粒度並列処理と近細粒度並列処理を階層的に組み合わせたマルチグレイン並列処理での性能を比較する．総 PE 数が 2 の時では，逐次実行時に対し粗粒度並列処理を用いた 2PG1PE では 1.50 倍，近細粒度並列処理を用いた 1PG2PE では

1.62 倍の速度向上率が得られた．同様に総 PE 数が 4 の時，8×8 ピクセルブロック間の粗粒度並列性のみを用いた 4PG1PE では 3.36 倍，8×8 ピクセルブロック内の近細粒度並列性のみを用いた 1PG4PE では 2.70 倍，PG 間で 8×8 ピクセルブロック間の並列性を用いた粗粒度並列処理を行い PG 内 PE で 8×8 ピクセルブロック内の並列性を用いた近細粒度並列処理を行うマルチグレイン並列処理を行う 2PG2PE では 3.59 倍の速度向上率が得られた．以上より使用プロセッサ数が同じ時には，階層的なマルチグレイン並列処理を使用した場合の方が高い実行効率を与えることが確かめられた．この理由を実行時間のイメージを表した図 2.13 を用いて述べる．まず，近細粒度並列処理のみを用いた場合は，使用プロセッサ数の増加により PE 間データ転送オーバーヘッドおよび同期オーバーヘッドが増加したため処理性能の低下が見られた．粗粒度並列処理のみの場合は，エントロピー符号化処理では，処理を行なう量子化 DCT 係数の値が 0 か非 0 かにより実行時間が変化するので，粗粒度タスク MT_{ai} は入力データによりタスク実行時間が異なるため粗粒度並列処理のみを利用した場合，各タスク処理時間のばらつきが生じ各プロセッサに負荷のアンバランスが生じた（図 2.13（a））．しかし，粗粒度並列処理と近細粒度並列処理を階層的に用いたマルチグレイン並列処理では，近細粒度並列処理により 1 つの MT が複数のプロセッサ上で処理されるため負荷バランスが向上し，より高い実行効率が得られたのである（図 2.13（b））．

2.7 第 2 章のまとめ

本章では，JPEG エンコード処理における，従来より行なわれていた命令レベル並列性の利用とは異なるアプローチの 8×8 ピクセルブロック処理間の粗粒度並列性と 8×8 ピクセル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

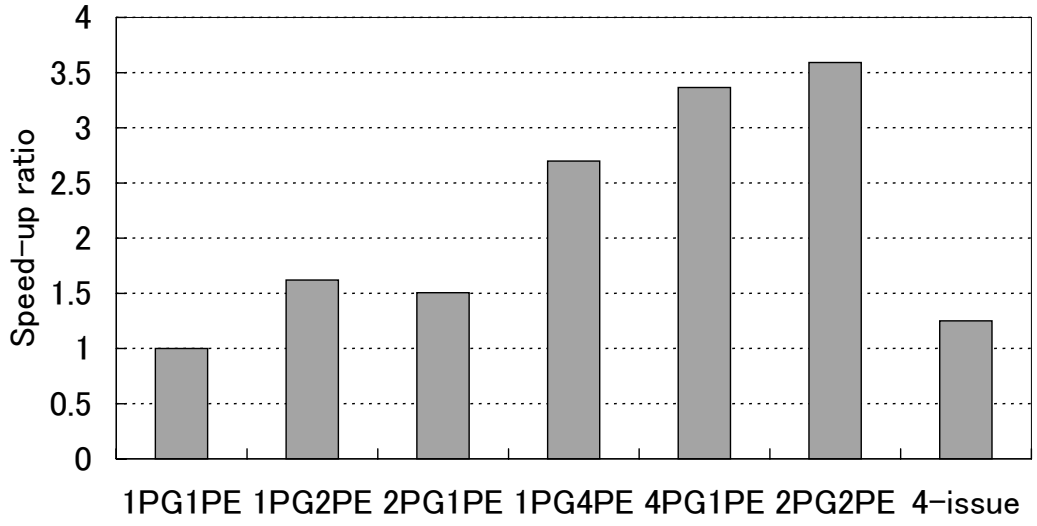


図 2.12: JPEG エンコード評価結果

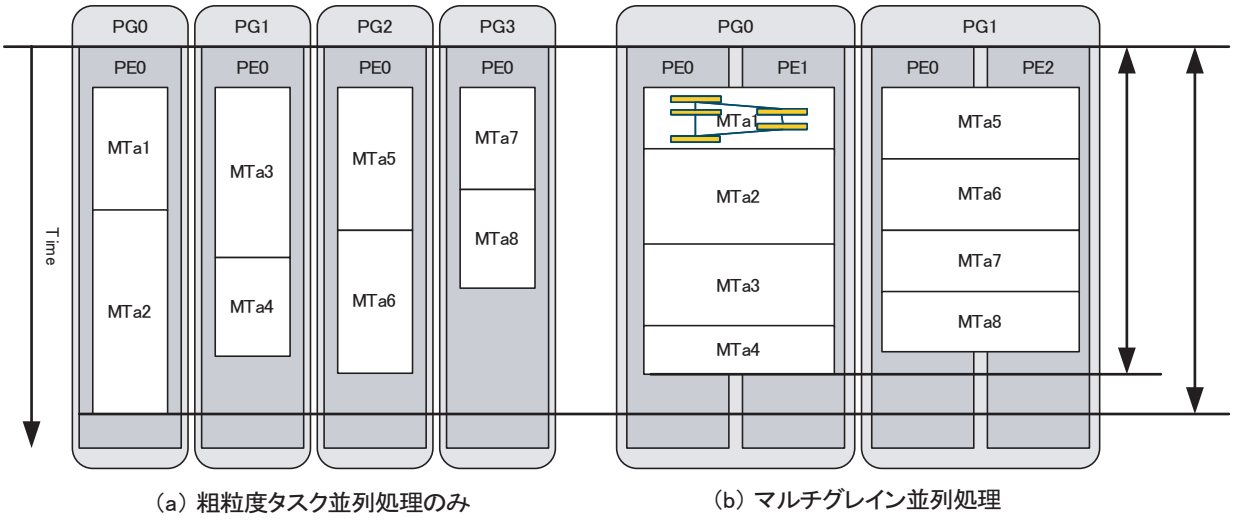


図 2.13: JPEG エンコード実行時間イメージ

た．シングルイシュープロセッサコアを 4 基搭載した OSCAR チップマルチプロセッサ（OSCAR CMP）上では，逐次処理時間に対し，8×8 ピクセルブロック処理間の粗粒度並列性のみを用いた場合 3.36 倍，8×8 ピクセルブロック処理内の近細粒度並列性のみを用いた場合 2.79 倍の速度向上率がそれぞれ得られ，提案手法の 8×8 ピクセルブロック処理間の粗粒度並列性と 8×8 ピクセルブロック処理内の近細粒度並列性を階層的に利用した場合 3.59 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 CMP 上での JPEG エンコーディングのマルチグレイン並列処理は集積度向上に対しスケーラブルな性能向上を可能とするということが確認できた．

第3章

MPEG2エンコードの並列処理

3.1 まえがき

MPEG2 は国際標準規格で定められた動画像フォーマットで，DVD や HDTV など現在のメディア配信で最も一般的に用いられている規格である．そのアルゴリズムは，動きベクトルを用いた空間冗長度の削減や離散コサイン変換による軸変換，可変長符号化による符号化などマルチメディア処理で用いられる基本的なアルゴリズムで構成されている．

MPEG2 エンコードの処理量は，リアルタイム処理を行なう場合は，1 秒間に 100 億命令のオーダーの演算が必要となり，ソフトウェアによるリアルタイムエンコードでは対象アーキテクチャにあわせた効果的な高速化技術が不可欠である．そのため，様々な高速化手法が研究されている [NC97, LW97, IO98, SXMH01]．これらの研究では主に MPEG2 の並列性を抽出し演算の効率化にのみに注目している．

また，近年のメモリウォール問題によるメモリアクセスオーバーヘッドの増加により，メモリ利用に関する最適化もソフトウェアの性能向上の重要課題になっている．MPEG2 エンコードは，動画像の冗長度削減などのために使用データ量が非常に多いため，MPEG2 エンコードの高速化には並列化による演算の高速化だけでなくメモリ利用の効率化も重要な要素の一つである．特に，チップマルチプロセッサでは，チップ内にローカルメモリを持つアーキテクチャであれば非常に高速なメモリアクセスが可能でありローカルメモリを有効に利用できるならば大きな速度向上を期待できる．そのため，メモリアクセスを効率化のためにアプリケーションに存在するデータローカリティを積極的に利用する必要がある．

本章では，動画像処理として広く用いられており，非常に処理が重い MPEG2 エンコード処理に対し，並列処理粒度としてベーシックブロック間など粒度の大きい粗粒度タスク並列性を利用した，チップマルチプロセッサ上でのメモリ利用最適化およびデータ転送最適化を伴う並列処理を提案し，OSCAR CMP 上での性能について述べる．

3.2 MPEG2 エンコードの並列性

ここでは，まず本章で参照する MPEG2 エンコードアルゴリズムについて述べた後，対象アルゴリズムから抽出可能な並列性について述べる．

3.2.1 MPEG2 エンコードアルゴリズム

本章では，MPEG2 エンコードアルゴリズムの参照実装として，MediaBench [CMW97] に収録されている MPEG2 エンコードプログラムである “mpeg2encode” を用いる．なお，以降説明するアルゴリズムは表 3.1 に示される MediaBench で用いられているエンコーディングオプションでの動作を仮定する．

まず，MPEG2 ビデオのデータ構造の説明を行なう．MPEG2 ビデオのデータ構造は図 3.1 に示すように階層的な構造をしている．ビデオシーケンスは MPEG2 ビデオデータすべてのことであり，シーケンスヘッダ，1 つまたは複数のグループオブピクチャ（GOP），およびシーケンス終了コードで構成される．MPEG2 でのランダムアクセス性の確保のために，いくつかの画像フレームをグルーピングして GOP が構成される．ピクチャは画像フレーム一つ一つのことでありその内部はいくつかのスライスからなっている．スライスはピクチャの左上から始まり右下へとスキャン順に続く任意個のマクロブロックの集合である．スライスデータの先頭には同期符号が割り当てられておりエラー低減に用いられる．本章でのエンコードパラメータである 4:2:0 カラーフォーマットでは，マクロブロックは，16 × 16 ピクセルブロックのルミナンスブロックと 2 つの 8 × 8 ピクセルブロックのクロミナンスブロックから構成されており，エンコードの単位として用いられる．最後に最も小さいのが 8 × 8 ピクセルブロックのブロックレイヤーであり，DCT 処理の単位として用いられる．

次に，MPEG2 エンコードの処理内容について説明する．MPEG2 エンコードは，

表 3.1: MPEG2 エンコードパラメータ

# of frames in GOP	15
I/P frame distance	3 （I B1 B2 P...）
Picture type	frame picture
Aspect ratio information	4:3
Frame rate	30 frames/sec.
Bit rate	5000000.0 bits/sec.
Profile	Main
Level	Main
chroma format	4:2:0
video format	NTSC
vbv buffer size	112kbit
progressive sequence	false
intra dc precision	8bit
quantization scale type	non-linear
entropy scan type	Zig-Zag scan
search window size	P:11 × 11
	B1:3 × 3（forw.）,7 × 7（back）
	B2:7 × 7（forw.）,3 × 3（back）

図 3.2 に示すように大きく分けて以下の 7 つのステージからなる．

動き推定（motion estimation）動きベクトルの探索を行ないピクチャタイプにより符号化モードの設定を行なう．

動き予測（predict）動き推定で求めた動きベクトル，符号化モードに基づいて符合化対象ピクチャを生成する．

DCT モード選択（dct type estimation）符号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離散コサイン変換（Discrete Cosine Transform: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．

データ変換（transform）DCT 変換モードに基づき符号化対象ピクチャに DCT を適用する．

ビットストリーム出力（putpict）DCT を適用したピクチャに対し量子化の適用を行ない，各種ヘッダの送出，ビットストリーム出力を行なう．また，ビットレート補償のためのビットレート制御および量子化係数の算出も行なう．

逆量子化（iquantize）後続のエンコーディングで参照するピクチャを既にエンコードした画像から生成するために，量子化適用後のピクチャに対して逆量子化を適用する．

逆データ変換（itransform）逆量子化適用後のピクチャに対して逆 DCT を適用し，後続のエンコーディングで参照するピクチャを生成し，画像バッファに格納する．

なお，MPEG2 エンコードは主にマクロブロックレイヤーにおいて各マクロブロック単位に符号化が行なわれ，画像フレームの左上から右下へと順番にスキャンされエンコードされる．

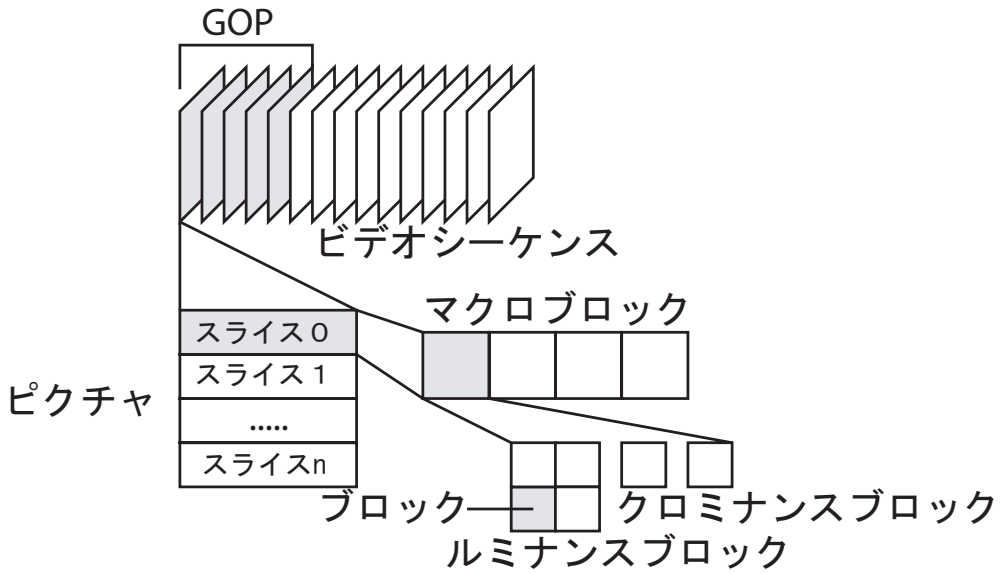


図 3.1: MPEG2 データ構造

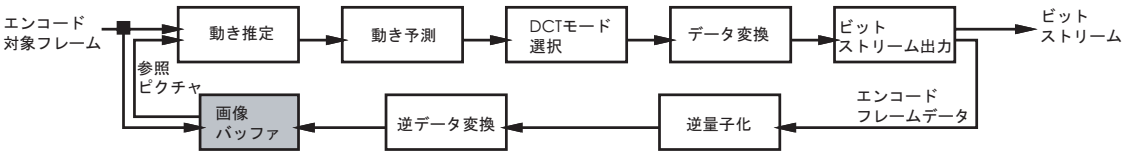


図 3.2: MPEG2 エンコード ブロック図

3.2.2 抽出可能な並列性

MPEG2 エンコードから並列性を抽出するために，MPEG2 ビデオデータ構造に注目する．MPEG2 ビデオデータは 3.2.1 節で述べたように階層的な構造をしており，各レイヤーのデータはランダムアクセス性の確保やエラー耐性の補償などのためにデータ同士でその独立性が確保されている部分がある．ここでは，各レイヤーでのデータの独立性に注目し，並列性の抽出を行なう．

まず，GOP レイヤーでは，GOP 単位でのランダムアクセス性の確保のために各 GOP 間にはデータ依存がない．一方，ピクチャレイヤーでは図 3.3 のように，空間冗長度削減によるデータ圧縮のために各ピクチャは参照関係，すなわちデータ依存関係にある．この空間冗長度削減方法は 3 つのタイプが定義されており，それぞれピクチャタイプとして定義されている．一つ目のピクチャタイプはイントラピクチャまたは I ピクチャと呼ばれ，各ピクチャでの基準となるピクチャである．I ピクチャは，他のピクチャの情報を使用せず JPEG のように自身のピクチャの情報のみで符号化を行なう．二つ目のピクチャタイプは，予測符号化ピクチャまたは P ピクチャと呼ばれる．P ピクチャは，過去の I ピクチャまたは P ピクチャを参照ピクチャとし，時間軸上で前向き動き予測で符号化される．三つ目のピクチャタイプは双方向予測符号化ピクチャまたは B ピクチャと呼ばる．B ピクチャは，過去と将来の I ピクチャまたは P ピクチャを参照ピクチャとして時間軸上で前向きおよび後向き予測符号化される．スライスレイヤーは，エラー補償のために各スライス間は独立して符号化される．マクロブロックレイヤーでは，マクロブロックが予測符号化の基本単位となっているために符号化処理では各マクロブロック間でのデータの依存はない．しかし，ビットレート補償やビットストリーム出力を行なうビットストリーム出力ステージでは，量子化係数の演算やビットストリームの出力順番はピクチャの右上から左下へのスキャン順に行なう必要がある．そのため，ビットストリーム出力ステージのみマクロブロック間でのデータ依存が

存在する．

以上のように，データ構造から並列性を抽出すると，GOP レイヤー，スライスレイヤー，およびビットストリーム出力ステージ以外のマクロブロックレイヤーではデータ依存がなく，各レイヤーで並列処理が可能である．

3.3 チップマルチプロセッサ上での MPEG2 エンコードの並列処理

ここでは，MPEG2 エンコードのメモリ利用量を考慮し，チップマルチプロセッサ上での効率的な並列処理のための並列処理粒度を解析し，メモリ利用の効率化およびデータ転送オーバーヘッドを削減する手法を提案する．

3.3.1 メモリ利用量を考慮した並列性粒度の解析

チップマルチプロセッサで高い実効効率を得るためには，並列性の抽出だけではなく，データローカリティを利用したチップ内メモリの利用によるメモリアクセスオーバーヘッドの削減も重要となる．そのため，各レイヤーで利用されるデータ容量を考慮し，チップマルチプロセッサでデータ利用効率が高くなる並列処理粒度の決定も重要な要素の一つである．

各レイヤーでのメモリ利用量

まず，マクロブロックレイヤーにおけるデータ利用量をソースコードを基に解析した．その結果，1 つのマクロブロックのエンコード処理に約 32K バイトのデータを利用することが解析できた．このマクロブロックレイヤーで利用するデータは，エンコード対象のマクロブロックと動き予測や動き推定で利用する参照画像

が主なデータである．

次に，さらに大きな粒度のスライスレイヤーに注目する．1 つのスライスに含まれるマクロブロックの数はエンコードソフトウェアの実装依存となっている．今回参照とした “mpeg2encode” では，1 スライス中のマクロブロックはピクチャ一列分が必要である．すなわち，携帯電話などで一般的なサイズである QCIF（176×144）では 1 スライスで 11 マクロブロック，QVGA（320×240）では 20 マクロブロック必要となり最小でも QCIF では約 32×11＝352K バイト，QVGA では約 32×20＝640K バイトそれぞれ必要となる．また，スライス単位のエンコードで必要となるメモリ容量は，画像サイズに依存するため, 高精細な大画面でのエンコードでは利用メモリサイズがかなり大きくなると予想される．

GOP レイヤーは，複数のピクチャを持ったスライスレイヤーよりさらに大きな粒度のデータが必要となるレイヤーである．複数ピクチャ単位になると数 M バイトの容量が必要となり，チップマルチプロセッサのチップ内メモリ容量をオーバーする可能性がある．

以上より，チップマルチプロセッサのチップ内メモリ容量を考慮すると，チップ内メモリ容量より小さいメモリを利用するマクロブロックレイヤーにおける並列性の利用が有効である．

マクロブロックレベルの並列性の抽出

3.3.1 節の解析より，ここではマクロブロックレベルの並列性の抽出について述べる．

MPEG2 エンコードで，マクロブロックレベルの並列性を抽出して利用する場合, ビットストリーム出力順やビットレート補償のための係数など直前のマクロブロックのエンコード情報を必要とするビットストリーム出力ステージ以外は，図 3.2 で示した各ステージにおいてそれぞれのマクロブロックは独立して演算できる．図

3.4（a）は，MediaBench での実装を基にした MPEG2 エンコードのコード例を示している．このコードからコンパイラにより並列性の抽出を行なうと，各ステージがそれぞれ MT として定義され図 3.4（b）のような MTG が生成される．MTG 中，各ノードは MT を表し，各エッジはデータ依存を表している．図 3.4（b）より，ビットストリーム出力ステージのみ，イタレーション間でループキャリー依存の存在により，シーケンシャルループであるが，他の各ステージからはマクロブロックレベルの並列性により 1 イタレーションが 1 つのマクロブロックを処理するループ並列性が抽出される．

しかし，このように抽出したループ並列処理を適用した場合，チップ内メモリの利用に関して以下のような問題が発生する．MPEG2 のエンコード処理は一つのステージをピクチャ全体に対して適用した後に次のステージへと進む構造をしている．そのため，各ステージの処理開始時のイタレーションで演算されたエンコードデータは後続イタレーションを演算する際，チップ内ローカルメモリの容量はピクチャ全体のデータを保持できるほど大きくないためチップ内ローカルメモリに保持しておけない．そのため，高速アクセスが可能な CPU 近傍のチップ内ローカルメモリから，低速アクセスとなってしまうチップ外メモリへエンコードデータのストアが必要となる．また，後続のステージを演算する際にも，チップ外メモリにストアされている前ステージのエンコードデータをチップ内ローカルメモリへ転送する必要がある．そのため，チップ内ローカルメモリ–チップ外メモリ間のロード・ストアによる転送オーバーヘッドが発生し全体的な実行効率が低下する．

3.3.2 データローカリティの利用

ループ並列処理を利用した場合のチップ内メモリ容量の制限によって発生するデータ転送の問題は，タスクの分割とタスク実行順序の並び替えによってプロセッサローカルメモリを利用した共有データの授受を行ないデータローカリティ利用

3.3. チップマルチプロセッサ上での MPEG2 エンコードの並列処理

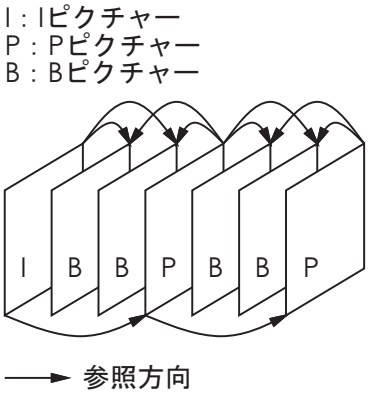
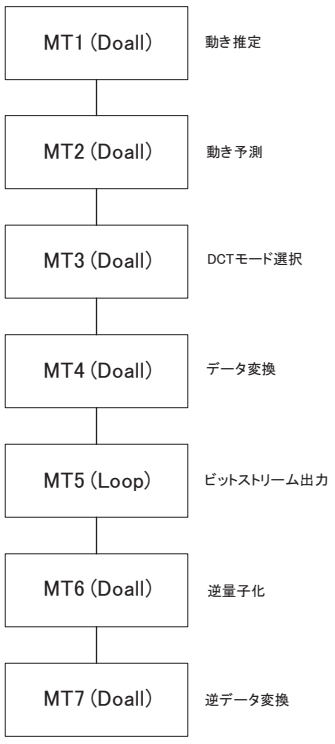


図 3.3: ピクチャータイプ

```
DO J=1, HEIGHT, 16
  DO I=1, WIDTH, 16
    motion_estimation_for_one_MB(...)
  ENDDO
ENDDO

DO J=1, HEIGHT, 16
  DO I=1, WIDTH, 16
    predict_for_one_MB(...)
  ENDDO
ENDDO

DO J=1, HEIGHT, 16
  DO I=1, WIDTH, 16
    dct_type_estimation_for_one_MB(...)
  ENDDO
ENDDO

DO J=1, HEIGHT, 16
  DO I=1, WIDTH, 16
    transform_for_one_MB(...)
  ENDDO
ENDDO

K=1
DO J=1, HEIGHT, 16
  DO I=1, WIDTH, 16
    mb_info(K) = putpict_for_one_MB(mbinfo(K-1),...)
  ENDDO
ENDDO

DO J=1, HEIGHT, 16
  DO I=1, WIDTH, 16
    iquantize_for_one_MB(...)
  ENDDO
ENDDO

DO J=1, HEIGHT, 16
  DO I=1, WIDTH, 16
    itransform_for_one_MB(...)
  ENDDO
ENDDO
```



(a)コード例　　(b)マクロタスクグラフ

図 3.4: MPEG2 エンコードのコード例とマクロタスクグラフ

の向上を行なうデータローカライゼーション手法 [吉田 99, IOK01, 中野 03] により解決でき，以下の手順で行なわれる．

データローカライゼーション手法

まず，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考慮しながら，そのデータ使用容量がプロセッサ内ローカルメモリ以下となるようにループ整合分割 [吉田 99] を行なう．次に，分割したそれぞれの小ループを粗粒度タスクとして定義し，最早実行可能条件解析を適用し並列性を抽出しマクロタスクグラフを生成する．そして，マクロタスクグラフ上でマクロタスク間のデータ共有量を計算し共有量の多いマクロタスク群をデータローカライゼーショングループ（DLG）[吉田 99] として定義する．その後，同一 DLG 内マクロタスクを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．以上により，データを共有する複数の粗粒度タスク間でプロセッサ内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り，データローカリティを利用したメモリ利用効率の最適化が行なわれる．

なお，データローカライゼーション手法は，SMP マシン [IOK01] やチップマルチプロセッサ [中野 03] のようにキャッシュアーキテクチャやローカルメモリアーキテクチャのように階層的なメモリアーキテクチャマシン上で汎用的に用いることができる手法である．

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

本節では，MPEG2 エンコードにおけるデータローカライゼーション手法の適用手法を提案する．まず，チップマルチプロセッサのチップ内メモリ容量を指定し，コンパイラがその容量を考慮しながら，マクロブロックレベルで各ステージを部

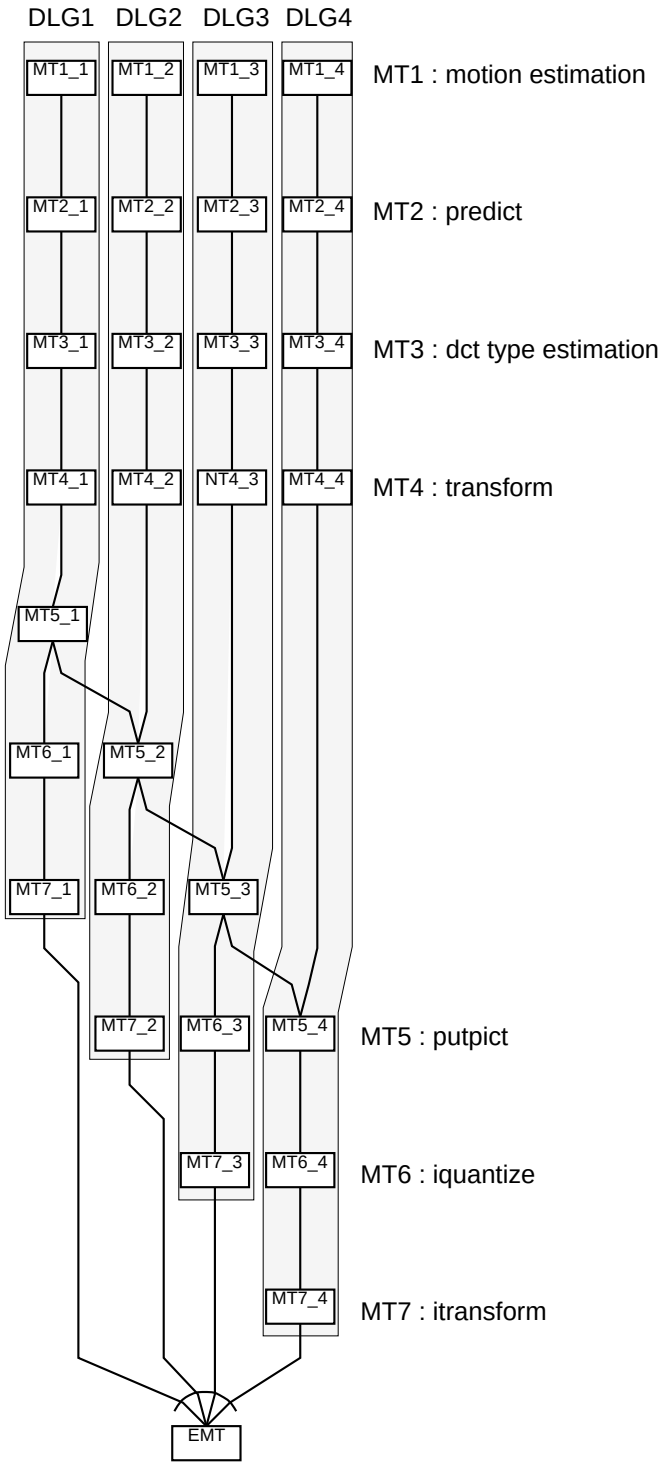


図 3.5: データローカライゼーション手法適用後の MTG

分ループに整合分割する．このとき，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ではあるが，基本的にマクロブロックレベルで処理を行なうループとなっているため，他のステージと同様にマクロブロックレベルの部分ループに分割する．ループ整合分割後，コンパイラは，分割された部分ループを粗粒度タスク（MT）として定義し並列性の抽出を行なう．MPEG2 エンコードでは各 MT は一つのマクロブロックのエンコード処理を行なう粒度となる．例えば，ピクチャの大きさが 4 つのマクロブロックである場合の並列性を抽出した結果は，図 3.5 のマクロタスクグラフ（MTG）として表される．図 3.5 では，各ノードは MT を表し，各エッジはデータ依存を表している．ノード中の $MTx\_i, (x = 1, \ldots, 7, i = 1, \ldots, N$：$N$ はピクチャ中の総マクロブロック数）は，$x = 1$ が動き推定ステージ，$x = 2$ が動き予測ステージ，$x = 3$ が DCT モード選択ステージ，$x = 4$ がデータ変換ステージ，$x = 5$ がビットストリーム出力ステージ，$x = 6$ が逆量子化ステージ，$x = 7$ が逆データ変換ステージをそれぞれ示し，i はエンコード対象とするマクロブロックのスキャン番号（ピクチャの右上から左下へスキャンされる）を示す．図 3.5 より，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レベルにおいてデータ依存が存在しないことがわかる．ここで，ビットストリーム出力ステージにおけるデータ依存エッジに関してエッジが接続されているマクロタスク間でのデータ共有量について考える．$MT5\_1$ から $MT6\_1$ および $MT5\_2$ へ接続しているデータ依存エッジを例とすると，$MT5\_1$ から $MT6\_1$ へのデータ依存エッジにおけるデータ共有量は $MT5\_1$ で量子化されたマクロブロックの演算データおよび量子化係数などである．対して $MT5\_1$ から $MT5\_2$ へのデータ依存エッジは，ループ整合分割を行なう前で発生していたループキャリー依存に関するデータ依存であり，ビットレート補償のための量子化係数，ビットレート係数やビットストリーム出力位置情報などである．これらのデータ共有量を比較すると

（$MT5\_1$ から $MT6\_1$ 間でのデータ共有量）

＞（$MT5\_1$ から $MT5\_2$ 間でのデータ共有量）

となりデータ転送の効率化の観点から $MT5\_1$ 実行後，連続して $MT6\_1$ を実行しチップ内メモリを介したデータの授受を行なう方が効率的である．以上より，各 MT のプロセッサへのスケジューリングは，同じマクロブロックをエンコードする各ステージを１つのデータローカライゼーショングループ（DLG）として定義し，１つのマクロブロックのエンコードは同一プロセッサで連続して行なうようにスケジュールする．例として，総マクロブロック数が 8 個の時のスケジュール結果を図 3.6 に示す.

3.3.3 データ転送オーバーラップを考慮したスケジューリング

データローカライゼーション手法により，データローカリティを最大限に活用してもローカルメモリへの初期データのロードや演算結果のストアなどのデータ転送が発生するため，データ転送オーバーヘッドによる速度低下が発生する．そこで，CPU と非同期にデータ転送を行なうデータ転送ユニット（DTU）を利用し，CPU でのタスク実行と DTU を利用したデータ転送をオーバーラップすることでデータ転送オーバーヘッドを隠蔽する.

ここで，ロードオーバーヘッド隠蔽技術は，あるタスク T_i の実行で必要なデータ D_i を外部メモリからプロセッサローカルメモリにロードするとき n 番先行に実行されるタスク T_{i-n} 実行中に DTU を利用して D_i をロードすることにより，本来は T_i の先頭で行なわれていた D_i のデータロードによるオーバーヘッドの隠蔽を行なうことを示し，プレロードと定義する．また，ストアオーバーヘッド隠蔽技術は，あるタスク T_j において T_j の演算結果である D_j をプロセッサローカルメモリから外部メモリへストアする必要がある場合，T_j 終了時にストアはせず，m 番後続に実行されるタスク T_{j+m} の実行中に DTU を利用し D_j をストアすることにより

D_j のデータストアによるオーバーヘッドの隠蔽を行なうことを示し，ポストストアと定義する．

データ転送オーバーラップスケジューリング

ここでは，プログラム実行とデータ転送をオーバーラップすることでデータ転送オーバーヘッドを隠蔽するプレロード・ポストストア手法を考慮したスケジューリングの提案を行なう．なお，提案手法は，各 PE で CPU，DTU，ローカルメモリを持ち，PE 外部に大容量共有メモリがあり，各 PE と PE 外部共有メモリが相互結合網によって接続されているアーキテクチャを仮定する．

まず，前提条件として，プレロードおよびポストストアは MT の先頭または末尾においてのみ開始できるものとし，複数のプレロード，ポストストアの開始が登録できるものとする．ただし，一つの DTU では同時に一つのプレロード，ポストストアのみ実行できるものとし，登録順にプレロード，ポストストアを実行する．以下，プレロード，ポストストアを考慮したスケジューリング手法を提案する．

仮プレロード開始時刻の定義 まず，プレロードスケジューリングの初期開始時刻となる，プレロードの仮開始時刻の定義を行なう．プレロード対象データは，あるマクロタスク MT_{pl} に生きて入り，かつ，MT_{pl} 内で前方露出参照されるデータであり MT_{pl} 実行開始時にプロセッサローカルメモリへデータ転送が必要なデータ D_{pl} で定義される．

ここで，MT_{pl} がプロセッサ PG_l にスケジュールが決定されたとする．このとき，プレロード開始時刻を決めるために，まず，MT_{pl} 開始時刻から部分スケジュールを時間軸上前方向へスキャンし D_{pl} が最後に生産されるマクロタスクを探索し，そのマクロタスクの末尾を仮プレロード開始時刻と定義する．このとき，D_{pl} のデータロードに必要なコストを推定し $Cost_{pl}$ とする．

仮ポストストア開始時刻の定義 次に，ポストストアスケジューリングの初期開始時刻となる，ポストストアの仮開始時刻の定義を行なう．ポストストア対象データは，PG_m 上の MT_{ps} で生産されるデータ D_{ps} が後続の PG_n に割り当てられた MT'_{ps} で使用される場合の D_{ps} である．そのため，MT'_{ps} がスケジューリングされるまでポストストアが必要か判断できないため MT'_{ps} のスケジューリングが確定した時点でポストストア開始時刻を決定する．

まず，MT'_{ps} のスケジューリングが決定したとき，MT_{ps} の終了時刻を仮ポストストア開始時刻と定義する．このとき，D_{ps} のデータストアに必要なコストを推定し $Cost_{ps}$ とする

プレロード・ポストストアスケジューリング スケジューリング対象となる MT の仮プレロード開始時刻または仮ポストストア開始時刻の定義ができたら，次に，プロセッサローカルメモリ–外部メモリ間のネットワーク利用状況，および，データ転送コストを考慮し，以下のようにプレロード・ポストストア開始時刻を決定する．プレロードおよびポストストアの開始時刻の評価は，プレロード，ポストストア共に同時に行なう．スケジューリングに必要なパラメータ以下のように再定義する．プレロード対象データ D_{pl} およびポストストア対象データ D_{ps} をプレロード・ポストストア対象データ D_{plps}，プレロード・ポストストア対象データ D_{plps} のロード・ストアに必要なコスト $Cost_{pl}$ および $Cost_{ps}$ を $Cost_{plps}$，プレロード対象データを利用するマクロタスクの MT_{pl} およびポストストア対象データを利用するマクロタスク MT'_{ps} を MT_{plps} とする．

(1) 仮プレロード・ポストストア開始時刻から $Cost_{plps}$ の時間分ネットワークが空いている場合，評価している仮プレロード・ポストストア開始時刻をプレロード・ポストストア時刻として決定する．

(2) 評価しているプレロード・ポストストアが仮開始時刻において，他のプレロード・ポストストアと重複し $Cost_{plps}$ の時間分ネットワークが空いていない場合は，重複しているプレロード・ポストストアと
((*MT*$_{plps}$の開始時刻) – (仮プレロードポストストア開始時刻))
の値を比較し，値が小さいプレロード・ポストストア時刻を優先して，(1) に従いプレロード・ポストストアを決定する．

(3) 評価している時刻では，プレロード・ポストストアが決定できない場合は，時間軸で直後の MT の終了時刻を仮プレロード・ポストストア開始時刻と再定義し，(1) からプレロード・ポストストアの時刻を決定する．

MPEG2 エンコードでのデータ転送オーバーラップスケジューリング

4 つのマクロブロックをエンコードする MPEG2 エンコードを例とし，提案したデータ転送オーバーラップを考慮したスケジューリングに従いプレロード・ポストストアを決定すると，図 3.7 (b) のようなスケジューリング結果が得られる．

図 3.7 は，1 バス構成のアーキテクチャ上で 2 プロセッサ用いた場合のスケジューリング状態のイメージをを示し，上段の (a) は，提案手法のデータ転送オーバーラップピングスケジューリングを適用しない場合を表し，下段 (b) は，提案手法適用時を表す．時間軸は左から右へ時間が進行し，各スケジュールチャートは，上から PG0 の CPU 上でのタスクの実行状況，PG1 の CPU 上でのタスクの実行状況，バスの利用状況を示す．なお，時間軸は相対イメージで表示されている．各 PG のスケジュールチャートにおいて，*MTx_i* はマクロタスクの実行を示す．バスの LD_j は j 番目のマクロブロックのエンコード実行に必要なデータのロードを実行中を示し，ST_k は k 番目のマクロブロックでの必要なストアを実行中であることを示す．また，グレーのチャートは，アイドル状態であることを示す．

MPEG2 エンコードでは，動き推定ステージ開始時で対象マクロブロックのエ

3.3. チップマルチプロセッサ上での MPEG2 エンコードの並列処理

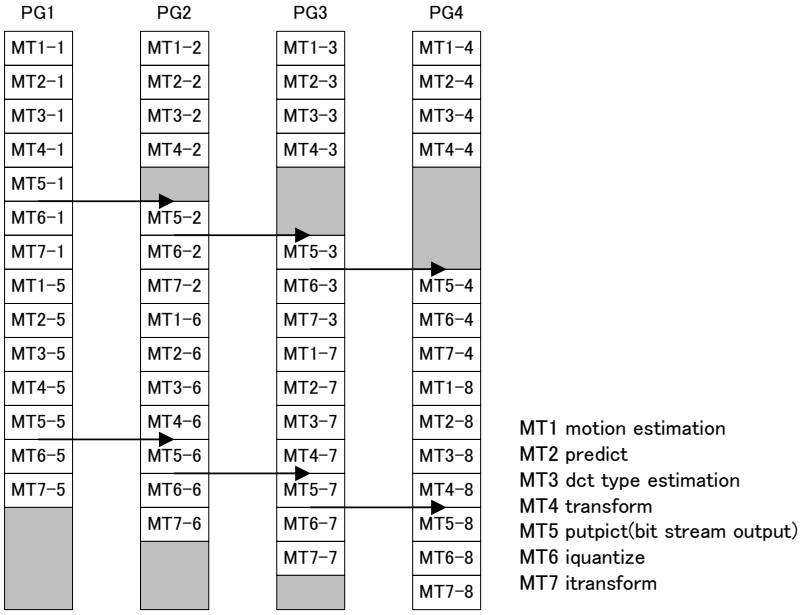


図 3.6: データローカライゼーション適用後のスケジューリング

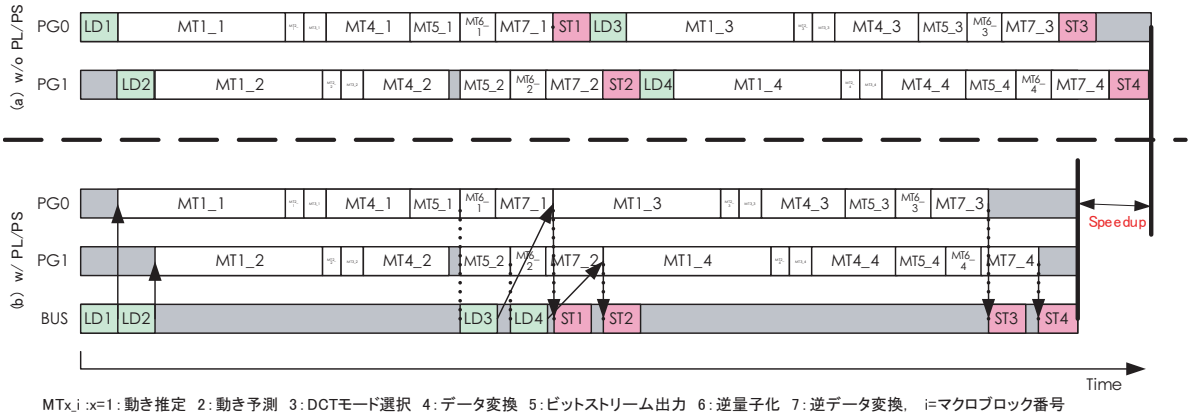


図 3.7: データ転送オーバーラップを考慮したスケジューリングイメージ

ンコードに必要なデータのロード，逆変換ステージ終了時にデータのストアを行なう．そのため，提案手法を適用しない図 3.7（a）では動き推定ステージの先頭，すなわち *MT*1_*i* の先頭でデータのロードが行なわれ，逆変換ステージ終了後，すなわち *MT*7_*i* 終了後にデータストアが行なわれるため，データのロード／ストアによるオーバーヘッド時間を必要とする．対して，提案手法を適用した図 3.7（b）では，タスク実行の中間に位置する *MT*1_3，*MT*1_4 のロード *LD*3，*LD*4 は，先行するマクロタスクとオーバーラップしてプレロードし，*MT*7_1，*MT*7_2 のストア *ST*1，*ST*2 は，後続するマクロタスクとオーバーラップしてポストストアするため，*LD*3，*LD*4 および *ST*1，*ST*2 のデータ転送時間が削減削減された．ただし，各 PG の実行開始の *MT*1_1，*MT*1_2 のロード *LD*1，*LD*2 は，先行するマクロタスクがないため通常のロードとなり，各 PG の最後の *MT*7_3，*MT*7_4 のストア *ST*3，*ST*4 は，後続するマクロタスクがないため通常のストアが行なわれる．

3.4 性能評価

本章では，MPEG2 エンコードに対して 3.2 節で提案した並列処理手法を適用し，OSCAR チップマルチプロセッサ（OSCAR CMP）上で評価した結果について述べる．

3.4.1 評価条件

評価に用いるプログラムは，MediaBench に収録されている MPEG2 エンコードプログラム “mpeg2encode” を参照実装し，OSCAR 自動並列化コンパイラを利用するために Fortran で実装されたプログラムである．並列性の抽出は参照実装したシーケンシャルプログラムを OSCAR 自動並列化コンパイラを用いて並列性の抽出を行なう．このとき，並列処理を行なう階層として 3.3 節で述べたマクロブロッ

クレベルの並列性を利用する階層をコンパイラ指示文で指定し，コンパイラによりマクロブロックレベルの並列性を自動的に抽出する．その後，データローカライゼーションおよびデータ転送オーバーラップスケジューリングを 3.3.2 節および 3.3.3 節のアルゴリズムで適用し，コンパイラにより OSCAR CMP 用バイナリーコードを生成した．

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 256 × 256 ピクセルに縮小した画像を用い 4 フレームのエンコードを行ない，エンコードオプションは MediaBench で用いられているものと同一とする．

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詳細なシミュレータを用いた．OSCAR CMP のパラメータは，各メモリサイズ，アクセスレイテンシは，LDM の容量は 128K バイトとしアクセスレイテンシは 1 クロック，同様に DSM の容量は 128K 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 は本章で使用する MPEG2 エンコード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 24 クロックとした．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り，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．

3.4.2 評価結果

OSCAR CMP 上での性能評価結果を図 3.8 に示す．図 3.8 中横軸の “1PE”，“2PEs”，“4PEs”，“8PEs” および “16PEs” は，使用プロセッサ数をそれぞれ示し，縦軸は逐

次実行時間に対する速度向上率を示す．3 本の棒グラフのうち左側は，従来のマルチプロセッサシステム用並列化手法であるループ並列処理を適用した場合の性能，中央は提案手法のうちメモリ利用最適化技術であるデータローカライゼーション手法のみを適用した場合の性能，右側は本章で提案したデータローカライゼーション手法およびデータ転送オーバーラップスケジューリングを適用した場合の性能をそれぞれ示す．

性能評価結果より，従来の並列化手法であるループ並列処理適用時は逐次実行に対して，2 プロセッサ利用時 1.80 倍，4 プロセッサ利用時 2.95 倍，8 プロセッサ利用時 4.38 倍，16 プロセッサ利用時 5.37 倍の速度向上率がそれぞれ得られた．提案手法のデータローカライゼーション手法のみを適用した場合，1 プロセッサ利用時 1.08 倍，2 プロセッサ利用時 2.13 倍，4 プロセッサ利用時 4.01 倍，8 プロセッサ利用時 7.11 倍，16 プロセッサ利用時 11.28 倍の速度向上がそれぞれ得られた．さらに，データローカライゼーション手法に加えデータ転送オーバーラップスケジューリングを適用した場合，1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 2.46 倍，4 プロセッサ利用時 4.57 倍，8 プロセッサ利用時 7.97 倍，16 プロセッサ利用時 11.93 倍の速度向上率が得られた．以上より，データローカライゼーション手法およびデータ転送オーバーラップスケジューリングを利用した提案手法は，従来手法であるループ並列化による並列処理に比べ，よりスケーラブルな性能を得ることが確かめられた．以下，データローカライゼーション手法による有効性とデータ転送オーバーラップスケジューリングによる有効性について個別に性能を確認する．

まず，データローカライゼーション手法の有効性を見るためデータローカライゼーション手法のみを適用した場合と提案する最適化手法を用いない逐次実行時と従来の並列化手法であるループ並列処理適用時を同数のプロセッサ数を利用した場合で比較を行なうと，1 プロセッサ利用時 1.08 倍，2 プロセッサ利用時 1.19 倍，4 プロセッサ利用時 1.36 倍，8 プロセッサ利用時 1.62 倍，16 プロセッサ利用

時 2.10 倍の性能が得られた．この性能向上は，データローカリティの利用によってデータアクセスオーバーヘッドの少ない CPU 近傍のメモリを利用することによるものによるものだけでなく，タスクスケジューリングにおいて，データローカライゼーション手法を利用することにより，MT の部分ループへの分割を行なったことによって，良い負荷バランスを得ることが可能となったことも貢献している．

さらに，データローカライゼーション手法に加え，データ転送オーバーラップスケジューリングを適用時の有効性は，逐次実行時および従来の並列化手法であるループ並列処理を適用した場合と同一のプロセッサ数を利用した場合でにおいて 1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 1.37 倍，4 プロセッサ利用時 1.55 倍，8 プロセッサ利用時 1.82 倍，16 プロセッサ利用時 2.22 倍の性能向上が得られた．実行プロファイルの結果より，データロードおよびデータストアの割合が約 16%存在するため，このロード・ストア時間削減分の速度向上が得られたことが確認できた．

3.5 第 3 章のまとめ

本章では，複数ループにまたがるグローバルデータローカリティ最適化を行なうデータローカライゼーション手法と粗粒度タスクの実行とデータ転送をオーバーラップさせデータ転送オーバーヘッドを最小化するデータ転送オーバーラップスケジューリングから成るチップマルチプロセッサ上での MPEG2 エンコードの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，OSCAR チップマルチプロセッサ上にて提案手法は，逐次実行に対し，1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 2.46 倍，4 プロセッサ利用時 4.57 倍，8 プロセッサ利用時 7.97 倍，16 プロセッサ利用時 11.93 倍の速度向上率がそれぞれ得られ，従来より用いられてきたのループ並列処理手法と利用プロセッサ

数が同一の場合で比較を行なっても，2 プロセッサ利用時 1.37 倍，4 プロセッサ利用時 1.55 倍，8 プロセッサ利用時 1.82 倍，16 プロセッサ利用時 2.22 倍の性能がそれぞれ得られた．

以上より，提案した MPEG2 エンコードの並列処理手法はチップマルチプロセッサ上で，スケーラブルかつ効果的な並列処理を実現できることが確認できた．

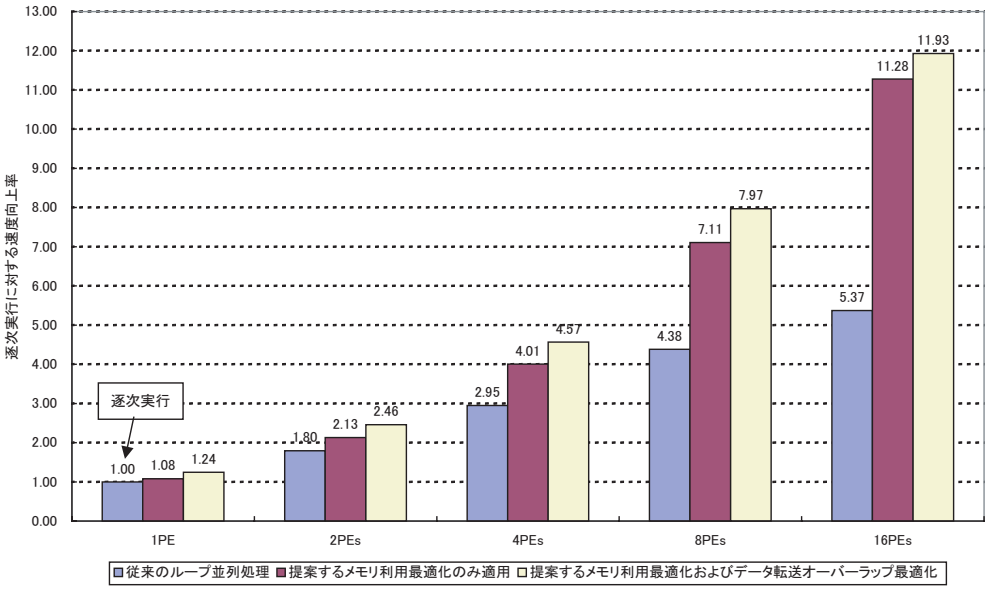


図 3.8: MPEG2 エンコード評価結果

第4章

結論

4.1 本研究により得られた成果

本論文では，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並列処理について述べた．マルチメディアアプリケーションとして，マルチメディアアプリケーションで広く用いられているアルゴリズムを採用している静止画像フォーマットである JPEG エンコードを用いて，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手法の提案および性能の確認を行ない，データ利用容量が多くデータ転送オーバーヘッドが大きい動画像フォーマットである MPEG2 エンコードを用いて，複数ループにまたがるグローバルデータローカリティ最適化を行なうデータローカライゼーション手法と粗粒度タスクの実行とデータ転送をオーバーラップさせデータ転送オーバーヘッドを最小化するデータ転送オーバーラップスケジューリングから成るチップマルチプロセッサ上でのメモリ利用最適化を考慮した粗粒度並列処理について提案し，その性能を評価した．

以下に，本研究により得られた成果を統括する．

(1) マルチメディアアプリケーションのチップマルチプロセッサ上での効率の良い並列処理手法として，ループブロックやベーシックブロック間の並列性を利用する粗粒度タスク並列処理と粗粒度タスク内のステートメント間の並列性を階層的に利用するマルチグレイン並列処理手法の提案や，近年の CPU とメモリの速度差に起因するメモリウォール問題を克服するためにチップ内のローカルメモリを積極的に利用しメモリ利用効率を向上させるデータローカライゼーション手法の適用，および，CPU と非同期にデータロード／ストアを行なうデータ転送ユニット（DTU）を用いて CPU でタスク実行時に DTU でデータ転送を並行に行なうオーバーラップデータ転送スケジューリングの提案を行ない，チップマルチプロセッサ上でその有用性の検証を行なった．

(2) 離散コサイン変換，ハフマン符号化などメディア処理で一般的なアルゴリズ

ムを用いている国際標準規格の静止画像フォーマットである JPEG エンコーディングにおいて，JPEG エンコーディングを構成するステージを 8×8 ピクセルブロックをエンコード基本単位とする粗粒度タスクに分割し粗粒度タスク並列性を抽出し，抽出した粗粒度タスク内からもプログラムのステートメントをタスクとする近細粒度並列性を抽出し，プログラム全体から有効な並列性を階層的に利用し並列処理効率を高めるマルチグレイン並列性の抽出方法を提案した．また，提案したマルチグレイン並列処理を適用した JPEG エンコードを OSCAR チップマルチプロセッサ上にて評価を行ない，利用プロセッサ数 4 のとき逐次実行に対して，粗粒度並列処理のみを用いた場合 3.36 倍，近細粒度並列処理のみを用いた場合 2.70 倍，粗粒度並列処理と近細粒度並列処理を階層的利用したマルチグレイン並列処理の場合 3.59 倍の速度向上率が得られ，提案した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手法の有効性が確かめられた．

(3) 国際標準規格の動画像フォーマットである MPEG2 エンコードにおいて，チップマルチプロセッサ上での効率の良い粗粒度タスク並列性の抽出法の提案に加え，チップマルチプロセッサに搭載された CPU 近傍のアクセス速度の速いローカルメモリを積極的に利用しメモリ利用の高効率化を行うデータローカライズ手法の適用法の提案，および，CPU と非同期にデータ転送を行うデータ転送ユニットを利用してデータ転送オーバーヘッドの隠蔽を図るプレロード・ポストストア手法を実現するスケジューリング手法を提案した．また，MPEG2 エンコードに提案手法を適用し OSCAR チップマルチプロセッサ上にて評価した結果，データローカライゼーションとオーバーラップデータ転送を含む粗粒度並列処理の提案手法は，逐次実行に対し，1 プロセッサ利用時 1.24 倍，2 プロセッサ利用時 2.46 倍，4 プロセッサ利用時 4.57 倍，8 プロセッサ利用時 7.97 倍，16 プロセッサ利用時 11.93 倍の速度向上率が得られ，

提案手法はチップマルチプロセッサ上で効果的な並列処理を実現できることが確認できた.

4.2 今後の課題

本研究に関する今後の課題を以下にまとめる.

(1) 本論文では，メディアアプリケーションの一例として，静止画像の JPEG，動画像の MPEG2 のエンコードにおけるチップマルチプロセッサ上での並列処理手法の提案を行なった．今後は，音声圧縮の MP3 や将来の採用が有望視されている静止画像の JPEG2000，動画像圧縮の H.264 などについての並列性抽出の研究を行なって評価メディアアプリケーション数を増やし，メディアアプリケーションのより一般的な特徴をとらえたチップマルチプロセッサ上での並列処理手法の提案は課題である.

(2) 近年のプロセッサアーキテクチャでは，従来から用いられているスーパースカラアーキテクチャの CPU にマルチメディア用拡張命令セットを追加し効率の良いマルチメディア処理を行なっている．このマルチメディア用拡張命令セットは主に SIMD 処理によるもので命令レベル並列性を利用している．本論文で提案したマルチグレイン並列処理では，ステートメントレベルを近細粒度として最小タスク粒度にした．そのため，本論文で提案したエンコードブロック間の粗粒度並列性とエンコードブロック内の近細粒度並列性を利用したマルチグレイン並列性に加え，さらに小さな粒度レベルになるマルチメディア用拡張命令セットによる命令レベル並列性を用いることによりさらなる高速化が可能であると考えられる．このマルチメディア命令セットを用いたチップマルチプロセッサ上でのメディアアプリケーションの並列処理手法の提案および評価は今後の課題である.

(3) 本論文では，OSCAR チップマルチプロセッサを対象アーキテクチャとして評価を行なっていたが，マルチメディアアプリケーションを効率良く処理するアーキテクチャとして，CPU にメディア処理用命令セットを追加したアーキテクチャやローカルメモリではなくキャッシュメモリを搭載したアーキテクチャ，グローバルレジスタを搭載したアーキテクチャなど様々なアーキテクチャ上での評価を行ない，マルチメディア処理での有効なアーキテクチャ構成の実現は今後の課題である．

参考文献

[Ald92] Aldus Developers Desk. *TIFF^{TM} Revison 6.0*, Jun. 1992.

[BK98] B. Geuskens and K. Rose. *MODELING MICROPROCESSOR PERFORMANCE*, 1998.

[CMW97] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: A tool for evaluating and synthesizing multimedia and communications systems. In *30th International Symposium on Microarchitecture (MICRO-30)*, Nov. 1997.

[DD97] Keith Diefendorff and Pradeep K. Dubey. How multimedia workloads will change processor design. *Computer*, Vol. 30, No. 9, pp. 43–45, 1997.

[E. 92] E. Hamilton. *JPEG File Interchange Format Version 1.02*, Sep. 1992.

[HMK00] H. Kasahara, M. Obata, and K. Ishizaka. Automatic coarse grain task parallel processing on smp using openmp. In *Proc. 12th Workshop on Languages and Compilers for Parallel Computing*, Aug. 2000.

[IO98] E. Iwata and K. Olukotun. Exploiting coarse-grain parallelism in the mpeg-2 algorithm, 1998.

[IOK01] Kazuhisa Ishizaka, Motoki Obata, and Hironori Kasahara. Coarse grain task parallel processing with cache optimization on shared memory multiprocessor. *Proc. of 14th International Workshop on Languages and Compilers for Parallel Computing (LCPC2001)*, Aug. 2001.

[KKOK03] Keiji Kimura, Takeshi Kodaka, Motoki Obata, and Hironori Kasahara. Multigrain parallel processing on compiler cooperative oscar chip multiprocessor architecture. *The IEICE Transactions on Electronics, Special Issue on High-Performance and Low-Power System LSIs and Related Technologies*, Vol. E86-C, No. 4, pp. 570–579, 4 2003.

[LBK97] L. Hammond, B. A. Nayfeh, and K. Olukotun. A single-chip multiprocessor. *IEEE Computer*, Sep. 1997.

[Le95] L. A. Lev and et al. A 64-b microprocessor with multimedia support. *IEEE Journal of SOLID-STATE CIRCUITS*, Vol. 30, No. 11, 1995.

[LHL02] Ville Lappalainen, Timo D. Hamalainen, and Petri Liuha. Overview of research efforts on media isa extentions and their usage in video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 8, pp. 660–670, Aug. 2002.

[LS96] Ruby B. Lee and Michael D. Smith. Guest editors' introduction: Media processing: A new design target. *IEEE Micro*, Vol. 16, No. 4, pp. 6–9, 1996.

[LW97] Heng Liao and Andrew Wolfe. Available parallelism in video applications. In *International Symposium on Microarchitecture*, pp. 321–329, 1997.

[MM02] M. Nikitovic and M. Brorsson. An adaptive chip-multiprocessor architecture for future mobile terminals. *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, CASES 2002*, Oct. 2002.

[NC97] N. Zhang and C. H. Wu. Study on adaptive job assignment for multiprocessor implementation of mpeg2 video encoding. *IEEE Transactions on Industrial Electronics*, Vol. 44, No. 5, Oct. 1997.

[OSS+02] Hiroshi Okano, Atsuhiro Suga, Tetsuyoshi Shiota, Yoshimasa Takebe, Yasuki Nakamura, Naoshi Higaki, Haruo Kimura, Hideo Miyake, Tomio Satoh, Ken ichi Kawasaki, Ryuhei Sasagawa, Wataru Shibamoto, Mitsuru Sasaki, Naruyoshi Ando, Tomohiro Yamana, lsao Fukushi, Shin ichirou Tago, Fumihiko Hayakawa, Teruhiko Kamigata, Satoshi Imai, Atsushi Satoh, Yasuaki Hatta2, Noboru Nishimura3, Yoshimi Asada, Taizo Satoh', Takao Sukemura', Satoshi Ando, and Hiromasa Takahashi. An 8-way vliw embedded multimedia processor built in 7-layer metal 0.11 um cmos technology. *Solid-State Circuits Conference, 2002. Digest of Technical Papers. ISSCC. 2002 IEEE International*, Vol. 1, pp. 374–375, 2 2002.

[RAJ99] Parthasarathy Ranganathan, Sarita Adve, and Norman P. Jouppi. Performance of image and video processing with general-purpose processors and media isa extensions. In *ISCA '99: Proceedings of the 26th annual international symposium on Computer architecture*, pp. 124–135. IEEE Computer Society, 1999.

[Sun97] Sun Microelectronics. *UltraSPARC^{TM} User's Manual*, Jul. 1997.

[SVJ00] S. K. Raman, V. Pentkovki, and J.Keshava. Implementing streaming SIMD extensions on the pentium III processor. *IEEE MICRO*, Vol. 20, No. 4, Jul. 2000.

[SXMH01] Sohum Sohoni, Zhiyong Xu, Rui Min, and Yiming Hu. A study of memory system performance of multimedia applications. In Sanjeev Setia, editor, *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS-01/PERFORMANCE-01)*, Vol. 29,1 of *ACM SIGMETRICS Performance Evaluation Review*, pp. 206–215. ACMPress, Jun. 2001.

[Tex00] Texas Instruments. Tms390s10 microprocessor. *http://www.ti.com/corp/docs/company/history/microsparc.shtml*, 2000.

[TONH96] Marc Tremblay, J. Michael O'Connor, Venkatesh Narayanan, and Liang He. VIS speeds new media processing. *Micro, IEEE*, Vol. 16, No. 4, pp. 10–20, 8 1996.

[吉田 99] 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, May. 1999.

[中野 03] 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, 情報処理学会, Aug. 2003.

[木村 01] 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, Apr. 2001.

謝辞

本研究は，著者が早稲田大学大学院博士課程在学中になされたものである．本研究を進めるにあたり，終始あたたかいご指導を賜わりました笠原博徳 教授に心から感謝の意を表します．また，本論文をまとめるにあたり，有益なご助言とご指導をいただきました成田誠之助 教授，松山泰男 教授，小林哲則 教授，木村啓二 専任講師に深く感謝致します．

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」，NEDO「先端ヘテロジニアスチップマルチプロセッサ研究開発事業」，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），枝廣正人氏（NEC）に深く感謝致します．

本研究の各段階では，尾形航氏，小幡元樹氏（現日立），飛田高雄氏（現ソニー），内田貴之氏（現三菱電機），鈴木貴久氏（現富士通），宮下直久氏（現トヨタ），石坂一久氏，中野啓史氏，白子準氏，宮本孝道氏，和田康孝氏をはじめとする笠原研究室の数多くの方々のご助言，ご協力をいただきました．ありがとうございました．

最後に，今日までの著者の研究生活をいろいろな側面で支えていただいた家族

をはじめとするすべての人々に感謝いたします.

著者研究業績

種類別	題名	発表掲載誌名	発表年月	著者名
論文 ○	Parallel Processing using Data Localization for MPEG2 Encoding on OSCAR Chip Multiprocessor	Proc. of International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04), IEEE Computer Society Press, pp.119-127	2004 年 1 月	T. Kodaka H. Nakano K. Kimura H. Kasahara
	Memory Management for Data Localization on OSCAR Chip Multiprocessor	Proc. of International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04), IEEE Computer Society Press, pp.82-88	2004 年 1 月	H. Nakano T. Kodaka K. Kimura H. Kasahara
	Multigrain Parallel Processing on Compiler Cooperative OSCAR Chip Multiprocessor Architecture	The IEICE Transactions on Electronics, Special Issue on High-Performance and Low-Power System LSIs and Related Technologies, Vol.E86-C, No.4, pp.570-579	2003 年 4 月	K. Kimura T. Kodaka M. Obata H. Kasahara

種類別	題名	発表掲載誌名	発表年月	著者名
論文	Multigrain Parallel Processing on OSCAR CMP	Proc. of International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'03), IEEE Computer Society Press, pp.56-65	2003 年 1 月	K. Kimura T. Kodaka M. Obata H. Kasahara
○	シングルチップマルチプロセッサにおける JPEG エンコーディングのマルチグレイン並列処理	情報処理学会ハイパフォーマンスコンピューティングシステム論文誌, Vol. 43, No. Sig. 6 (HPS5), pp. 153-162	2002 年 9 月	小高 剛 内田 貴之 木村 啓二 笠原博徳
○	Multigrain Parallel Processing for JPEG Encoding on a Single Chip Multiprocessor	Proc. of International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'02), IEEE Computer Society Press, pp.57-63	2002 年 1 月	T. Kodaka K. Kimura H. Kasahara

種類別	題名	発表掲載誌名	発表年月	著者名
査読付きシンポジウム ○	シングルチップマルチプロセッサにおける JPEG エンコーディングのマルチグレイン並列処理	情報処理学会並列処理シンポジウム (JSPP2002) 論文集, pp.305-312	2002 年 5 月	小高 剛 内田 貴之 木村 啓二 笠原 博徳
講演（研究会）	OSCAR チップマルチプロセッサ上での MPEG2 エンコードの並列処理	情報処理学会研究会報告 2004-ARC-160-7	2004 年 12 月	小高 剛 中野 啓史 木村 啓二 笠原 博徳
	OSCAR チップマルチプロセッサ上でのデータ転送ユニットを用いたデータローカライゼーション	情報処理学会研究会報告 2004-ARC-159-20	2004 年 7 月	中野 啓史 内藤 陽介 鈴木 貴久 小高 剛 石坂 一久 木村 啓二 笠原 博徳
	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	情報処理学会研究会報告 2004-ARC-156-3	2004 年 2 月	小高 剛 中野 啓史 木村 啓二 笠原 博徳
	OSCAR 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	情報処理学会研究会報告 2003-ARC-154-10	2003 年 8 月	小高 剛 中野 啓史 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	情報処理学会研究会報告 2003-ARC-154-14	2003 年 8 月	中野 啓史 小高 剛 木村 啓二 笠原 博徳
	チップマルチプロセッサ上での粗粒度タスク並列処理によるデータローカライゼーション	情報処理学会研究会報告 ARC2003-151-3 (SHINING2003)	2003 年 1 月	中野 啓史 小高 剛 木村 啓二 笠原 博徳
	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	情報処理学会研究報告 ARC2002-150-6	2002 年 11 月	小高 剛 鈴木 貴久 木村 啓二 笠原 博徳

種類別	題名	発表掲載誌名	発表年月	著者名
	OSCAR 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	情報処理学会研究報告 ARC2002-150-7	2002 年 11 月	木村 啓二 小高 剛 小幡 元樹 笠原 博徳
	OSCAR 型シングルチップマルチプロセッサ上での JPEG エンコーディングプログラムのマルチグレイン並列処理	情報処理学会研究報告 ARC2002-146-4	2002 年 2 月	小高 剛 内田 貴之 木村 啓二 笠原 博徳
	シングルチップマルチプロセッサにおけるマルチグレイン並列処理	情報処理学会研究報告 ARC2002-146-5	2002 年 2 月	内田 貴之 木村 啓二 小高 剛 笠原 博徳
	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近細粒度並列処理	情報処理学会研究報告 ARC2001-140-11	2001 年 8 月	小高 剛 宮下 直久 木村 啓二 笠原 博徳
講演（全国大会）	マルチメディアアプリケーションのシングルチップマルチプロセッサ上での近細粒度並列処理	情報処理学会第 62 回全国大会 3P-08	2001 年 3 月	小高 剛 木村 啓二 宮下 直久 笠原 博徳
	マルチグレイン並列処理用シングルチップマルチプロセッサにおけるデータ転送ユニットの検討	情報処理学会第 62 回全国大会 4P-02	2001 年 3 月	宮下 直久 木村 啓二 小高 剛 笠原 博徳
その他（査読なしシンポジウム）	OSCAR チップマルチプロセッサ上でのデータローカリティを考慮した MPEG2 エンコード	STARC シンポジウム 2004	2004 年 9 月	小高 剛
	シングルチップマルチプロセッサにおける JPEG エンコーディングのマルチグレイン並列処理	STARC シンポジウム 2002	2002 年 9 月	小高 剛
その他（ポスター発表）	ソフトウェア・ハードウェア協調型チップマルチプロセッサ OSCAR CMP の組み込み開発環境への適用	第 5 回組込みシステム技術に関するサマーワークショップ	2003 年 7 月	木村 啓二 和田 康孝 中野 啓史 小高 剛 笠原 博徳