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

小高 剛[†] 内田貴之[†]
木村啓二[†] 笠原博徳[†]

近年の JPEG,MPEG などを用いたマルチメディアコンテンツの増加に伴い，これら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．特に，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は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，JPEG エンコーディングの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するとともに，その性能評価を行う．評価の結果，シンプルなシングルイシュープロセッサを 4 基搭載した OSCAR 型シングルチップマルチプロセッサアーキテクチャでは逐次実行に対して約 3.59 倍の性能向上が得られスケーラブルな性能向上が得られることが確かめられた．

JPEG Encoding using Multigrain Parallel Processing on a Shingle Chip Multiprocessor

Takeshi Kodaka,[†] Takayuki Uchida,[†] 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 attain scalability and good cost effectiveness are attracting much attention to develop such processors. Since single chip multiprocessor architectures allow us to exploit coarse grain task level and loop level parallelism in addition to the instruction level parallelism, parallel processing technology is indispensable to get us scalable performance improvement. This paper describes a multigrain parallel processing scheme for the JPEG encoding for a single chip multiprocessor and its performance. The evaluation shows an OSCAR type single chip multiprocessor having four single-issue simple processor cores gave us 3.59 times speed-up.

1. はじめに

最近のマルチメディアコンテンツの増加に従い JPEG，MPEG などのメディア系アプリケーションを効率良く処理できる低コストかつ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 の PentiumIII などに搭載されている SSE[1] や，日立の SH4[2]，富士通の FR500[3] 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，これらのような命令レベル並列性を用いるアーキテクチャでは命令レベル細粒度並列性の限界により投入されたトランジスタ資源に見合うスケーラブルな性能向上が困難になってきている．

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集積トランジスタ数の増加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を集めている．また，シングルチップマルチプロセッサアーキテクチャでは，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る．ただし，これら複数レベルの並列性を利用するためにはアプリケーションからの並列性抽出が重要となる．

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 の MP98[4] では，1 チップ上に 4CPU を搭載しマルチスレッド処

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

理により高速化を行なっており，Stanford 大では，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されている[6]．

本論文では，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 エンコーディングにおけるマルチグレイン並列処理手法を提案し，提案手法を適用した JPEG エンコーディングの OSCAR 型シングルチップマルチプロセッサ上での評価結果を述べる．

以降，2 章でマルチグレイン並列処理，3 章で提案する JPEG エンコーディングのマルチグレイン並列処理手法，4 章で本論文の評価で用いる OSCAR 型シングルチップマルチプロセッサアーキテクチャ，5 章でマルチグレイン並列処理を適用した JPEG エンコーディングの OSCAR 型シングルチップマルチプロセッサ上で行った性能評価について述べる．

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理）[7]，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[8] 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[7]（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[8]

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．

3. JPEG エンコーディングアルゴリズムのマルチグレイン並列処理手法

ここでは，JPEG エンコーディング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．本論文で扱う JPEG エンコーディングアルゴリズムは，MediaBench[9] 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．まず，JPEG エンコーディングにおける各処理について簡単に述べた後，マルチグレイン並列性を利用する手法を述べる．

3.1 JPEG エンコーディングアルゴリズム[10),11)]

JPEG エンコーディングは以下の 6 つの処理からなる．

8x8-pixel ブロック分割

入力画像データを JPEG 基本処理単位である 8x8-pixel ブロックに分割する．

YCbCr 変換（YCbCr）

分割された 8x8-pixel ブロックのデータを JPEG のデータフォーマットである 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（DCT）

YCbCr フォーマットへ変換された 8x8 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（Quantize）

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 8x8 ブロックの量子化 DC 係数と 1 つ前の 8x8 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．

上記 6 つの処理を入力画像データが終るまで繰り返して行なわれる．図 1 に JPEG エンコーディング実行ブロック図を示す．

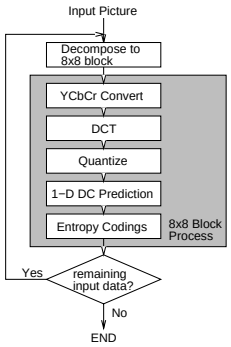


図 1 JPEG エンコーディング実行ブロック図

3.2 JPEG エンコーディングのマルチグレイン並列処理

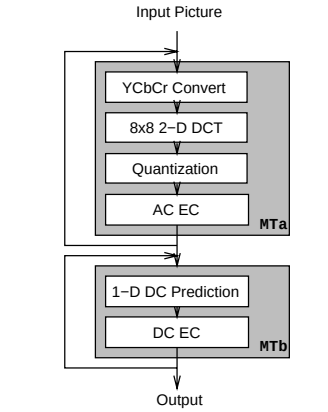
ここでは，JPEG エンコーディングのマルチグレイン並列処理手法について述べる．

3.2.1 粗粒度並列性の抽出

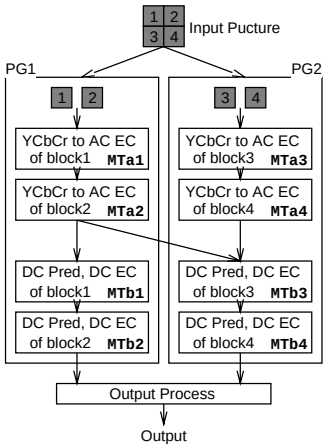
3.1 で述べた基本的な JPEG エンコーディングアルゴリズムでは，量子化 DC 係数の 1 次元予測（1-D DC Prediction）で 1 つ前の 8x8 ブロックの量子化 DC 係数を用いるため，8x8 ブロック間でのデータ依存が存在する．しかし，その他の YCbCr 変換，DCT およびエントロピー符号化処理では 8x8 ブロック内のデータのみ扱うため 8x8 ブロック間のデータ依存は存在しない．これらのデータ依存を考慮し，並列処理を効果的に行なうために 1-D DC Prediction 処理および，1-D DC Prediction の処理結果の 1 次元 DC 予測値を用いる 1 次元 DC 予測値のエントロピー符号化処理（DC EC）を図 2（a）のように YCbCr,DCT,Quantize および量子化 AC 係数のエントロピー符号化処理（AC EC）の後ろに移動する．そして，8x8 ブロック処理を JPEG エンコーディングの基本処理単位として，YCbCr,DCT,Quantize および AC EC を 1 つのマクロタスク（図 2（a）MT_a）として定義し，同様に 1-D DC Prediction および DC EC も 1 つのマクロタスク（図 2（a） MT_b）として定義する．

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 8x8 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，16x16pixel の入力データを 2 つの PG で実行する場合の各 MT の割り当ては，図 2（b）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 8x8 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 8x8 ブロックに分割され，PG1 では block1 と block2 を処理する MT_{a1} および MT_{a2} が実行され，同様に PG2 では block3 と block4 を処理する MT_{a3} および MT_{a4} が実行される．このとき，各 MT_a 間では依存がないためこれらの MT_a は全 PG で並列に処理される．すべての MT_a が実行された後，MT_b 実行前に，PG2 で処理されている block3 の 1-D DC Prediction で必要とされる block2 の量子化 DC 係数を PG1 から PG2 へ転送する．データ転送後，MT_b 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 3（a）のようにループにより表現された JPEG エンコーディング処理をループアンローリングにより展開する（図 3（b））．ループアンローリング後，1 つの 8x8 ブロックを処理する YCbCr，DCT，Quantize，AC EC，1-D DC Prediction，DC EC 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 3（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 3（b）の各



(a) JPEG encoding block diagram



(b) Execution Flow image (2PGs)

図 2 並列化 JPEG エンコーディング

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 3（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．

3.2.2 近細粒度並列性の抽出

ここでは，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 変換の変換式は入力信号により違うが，今回の評価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

$$Y_{ij} = R_{ij} * 0.29 + G_{ij} * 0.58 + B_{ij} * 0.11$$
$$Cb_{ij} = -R_{ij} * 0.16 - G_{ij} * 0.33 + B_{ij} * 0.50$$
$$Cr_{ij} = R_{ij} * 0.50 - G_{ij} * 0.41 - B_{ij} * 0.08$$
$$(0 \leq i \leq 7, 0 \leq j \leq 7)$$

で表される．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

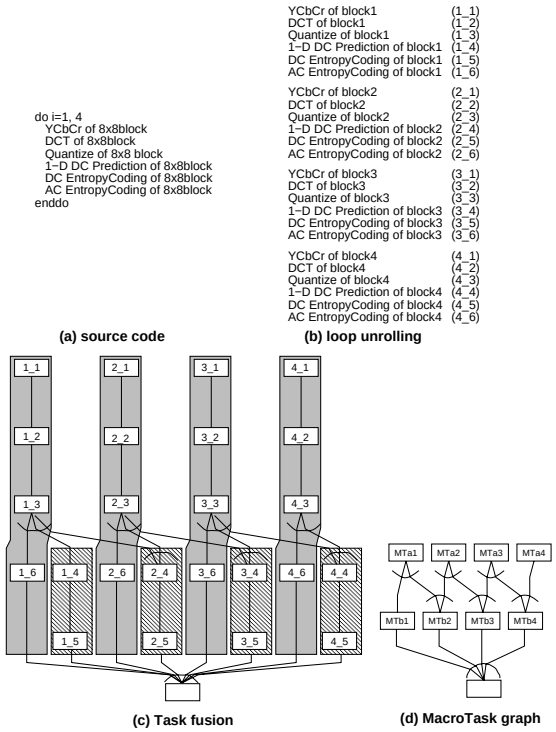


図 3 粗粒度並列性抽出

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が，近細粒度並列性を高めるため今回の評価ではループを展開した．ステートメントを近細粒度タスクとして定義し，そのときの 2 次元離散コサイン変換のタスクグラフを図 4 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している． 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．

一様量子化処理は，以下の式で示される．

$$Sq_{ij} = round\left(\frac{S_{ij}}{q_{ij}}\right)$$
$$(0 \leq i \leq 7, 0 \leq j \leq 7)$$

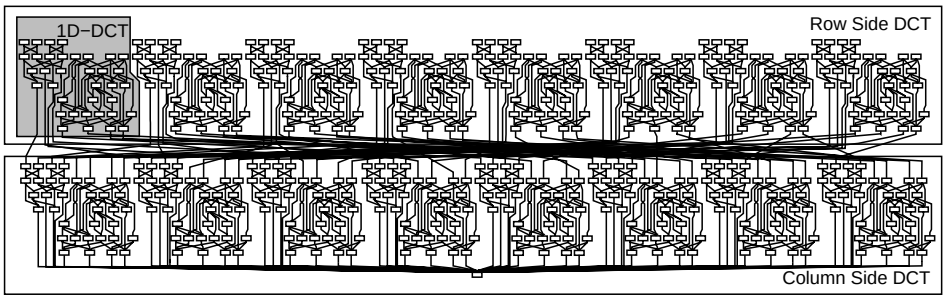


図 4 2 次元 DCT タスクグラフ

この場合も，YCbCr 変換同様，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ことがわかる．

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 5（a）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 5（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，今回の評価では図 5（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，分割数は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負荷のバランスを考慮て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．本評価では，JPEG エンコーディング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 になる傾向があるという性質を利用し，複数の入力画像のプロファイル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，

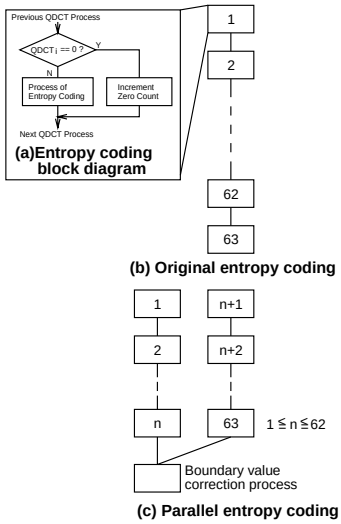


図 5 エントロピー符号化処理の近細粒度タスク

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．

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[8)]

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 のように CPU，データ転送ユニット（DTU），ローカルプログラムメモリ（LPM），ローカルデータメモリ（LDM），および分散共有メモリ（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 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，データ転送を CPU の処理とオーバーラップして行なえる 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の開発が終っていないため利用してない．また，PE 間相互結合網は 3 本バスを利用している．

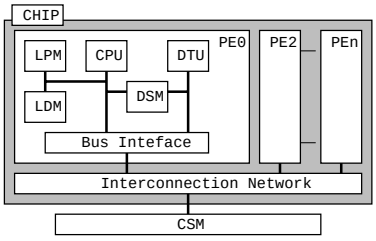


図 6 OSCAR 型 SCM アーキテクチャ

LPM は各々の CPU で実行するプログラムを格納し，1 クロックでアクセスできるものとする．同様に，LDM は PE 固有のデータを保持するために使用し，その容量は 1 チップあたりの総容量を 1M バイトとする．例えば，1 チップに 2PE 搭載する SCM では，各 PE あたり 512k バイトとして評価を行なっている．また，LDM へのアクセスレイテンシは 1 クロックとする．DSM は，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であり，近細粒度タスク間のデータ転送等に使用する．DSM の容量は 1PE あたり 16k バイトとし，自 PE からのアクセスレイテンシは 1 クロック，他 PE からのアクセスレイテンシは 4 クロックとする．さらに，チップ外部には集中共有メモリ（CSM）が接続され，各 PE で共有されるデータを格納する．この CSM のアクセスレイテンシは 20 クロックとする．

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．

また，スーパスカラとの比較のため 4 イシュー in-order 発行スーパースカラプロセッサについても評価を行なう．このプロセッサは，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．なお，メモリアーキテクチャは OSCAR 型 SCM と同様である．

4.3 トランジスタ数の概算

ここでは，シングルイシュープロセッサを搭載した OSCAR 型 SCM の回路規模を既存のプロセッサを基に概算する．

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以下の式[13] 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．

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

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[14] を用い，また，今回比較に用いる 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[15] 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 1 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，SCM のトランジスタ数として表 2 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．表より，シングルイシュープロセッサを 4 基搭載した OSCAR 型 SCM アーキテクチャと 4 イシュー in-order スーパースカラアーキテクチャは，ほぼ同程度のトランジスタ数であると推定できる．

5. 性能評価

ここでは，JPEG エンコーディングプログラムにマルチグレイン並列処理を適用し OSCAR 型メモリアーキテクチャシングルチップマルチプロセッサ（SCM）上にて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，プログラムソースレベ

表 1　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　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	SCM（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

ルで 3 章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 3 章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 SCM 用バイナリを作成した．なお，コンパイラにより生成されたタスクスケジューリング結果は本論文 3 章で提案した結果と同一である．

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 pixel に縮小した画像とし，CSM 上に配置した．シングルチップマルチプロセッサでのデータ配置は，プログラム開始後，まず CSM の入力画像データを各 PE 内の LDM へコピーする．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．4 イシュースーパースカラプロセッサの場合も同様に，プログラム開始後 LDM へ入力画像をコピーし，LDM を用いてエンコード処理を行い，処理結果を CSM へ格納している．

OSCAR 型 SCM 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 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．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

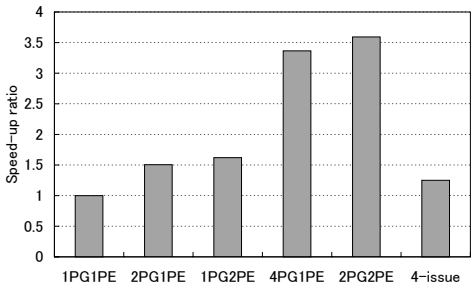


図 7　JPEG エンコーディング評価結果

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 SCM 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 エンコーディング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 SCM 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 型 SCM アーキテクチャでの粗粒度並列処理単体性能と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．図 7 に示すように，総 PE 数が 2 の時では，逐次実行時に対し 2PG1PE では約 1.50 倍，

1PG2PE では約 1.62 倍の速度向上率が得られた．同様に総 PE 数が 4 の時，粗粒度並列処理単体の 4PG1PE では約 3.36 倍，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 2PG2PE では約 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．この理由は，エントロピー符号化処理では，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，マクロタスク（MT）は入力データによりタスク実行時間が異なる．そのため粗粒度並列処理のみを利用した場合，各 MT 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てしまう．しかし，近細粒度並列処理を用いた場合，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良くなる．そのため，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，負荷バランスの向上により高い実行効率が得られた．

6. ま と め

本論文では，JPEG エンコーディング処理における，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 8x8 ブロック間の粗粒度並列性と 8x8 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約 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 型 SCM 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．

今後の課題として，MPEG-2 や JPEG2000 などのマルチメディアアプリケーションを含めたアプリケーション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サポートするシングルチップマルチプロセッサアーキテクチャの開発が挙げられる．

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」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 STARC 平田雅規氏，宮田操氏，東芝 浅野滋徳氏，安川英樹氏，富士通 高橋宏政氏，ソニー 倉田隆弘氏，松下 高山秀一氏に感謝致します．

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) 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) 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).
9) 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).
10) E. Hamilton: *JPEG File Interchange Format Version 1.02* (1992).
11) Aldus Developers Desk: *TIFF^{TM} Revison 6.0* (1992).
12) Sun Microelectronics: *UltraSPARC^{TM} User's Manual* (1997).
13) B. Geuskens and K. Rose: *MODELING MICROPROCESSOR PERFORMANCE* (1998).
14) Texas Instruments: *TMS390S10 Microprocessor* (2000).
15) L. A. Lev and et al.: A 64-b microprocessor with multimedia support., *IEEE Journal of SOLID-STATE CIRCUITS*, Vol. 30, No. 11 (1995).