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

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，低消費電力かつ高パフォーマンスなプロセッサの必要性が増している．このような要求を満たすプロセッサアーキテクチャの1つ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で効率の良い処理を行うには，アプリケーションの特性を解析し，並列性を抽出し，生成したタスクをバランス良くプロセッサに配置するなどのチップマルチプロセッサ用最適化が必要となる．また，近年のメモリウォール問題の深刻化により，プログラムの持つデータローカリティの有効利用やデータ転送オーバヘッドの削減などの最適化技術も効果的な並列処理のために必須となっている．本論文では，MPEG2 エンコードに対する，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手法からなる並列処理手法の提案を行うとともに，OSCAR チップマルチプロセッサ上での性能評価を行う．性能評価の結果，データローカリティの利用およびデータ転送オーバヘッド隠蔽手法からなる提案手法を適用した MPEG2 エンコードは，動作周波数 400 MHz 時で逐次実行に対し，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，動作周波数 2.8 GHz 時で逐次実行に対し，1 プロセッサ利用時 1.36 倍，2 プロセッサ利用時 2.61 倍，4 プロセッサ利用時 4.46 倍，8 プロセッサ利用時 6.54 倍の速度向上率の速度向上率が得られることが確認できた．

Parallel Processing of MPEG2 Encoding on a Chip Multiprocessor Architecture

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

With the popularization of multimedia applications like image and audio processing on PCs, mobile phones and PDAs, development of low cost, low power consumption and high performance processors for multimedia applications has been expected. To this end, chip multiprocessor architectures that allows us to exploit multi-grain parallelism such as coarse grain level parallelism, loop level parallelism and instruction level parallelism have been extensively researched. However, to realize efficient parallel processing on chip multiprocessor architectures, sophisticated techniques are required for decomposition of a program into adequate grain of tasks, analysis of parallelism and scheduling of the tasks onto processors considering data locality. This paper describes a parallel processing scheme for MPEG2 encoding using data localization which optimizes execution efficiency assigning coarse grain tasks accessing the same array data on the same processor consecutively on a chip multiprocessor and data transfer overlapping technique which minimize the data transfer overhead by overlapping task execution and data transfer. Performance of the proposed scheme is also evaluated. As the evaluation result on an OSCAR chip multiprocessor architecture, when the clock frequency is assumed as 400 MHz, the proposed scheme gave us 1.24 times speedup for 1 processor, 2.47 times speedup for 2 processors, 4.57 times speedup for 4 processors and 7.97 times speedup for 8 processors against sequential execution without the proposed scheme respectively. Similarly, when 2.8 GHz, the proposed scheme gave us 1.36 times speedup for 1 processor, 2.61 times speedup for 2 processors, 4.46 times speedup for 4 processors and 6.54 times speedup for 8 processors against sequential execution without the proposed scheme respectively.

† 早稲田大学コンピュータ・ネットワーク工学科
Department of Computer Science, School of Science and Engineering, Waseda University
☆ 現在，株式会社東芝
Presently with TOSHIBA Corporation

1. はじめに

近年，ディジタル TV や DVD プレーヤ，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1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では，そのアプリケーション専用に作られているため処理速度が非常に速く，消費電力も低く抑えることができる．ただし，新しい規格への対応では，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[1)~3)]．

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，パイプライン化やスーパスカラ命令発行，マルチメディア拡張命令セットなどの SIMD 命令の追加，VLIW などを用いて演算の高速化を行ってきた．しかし，これらの高速化手法は，主に命令レベル並列性を利用しているため命令レベル並列性の限界[4)]により，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が今後の課題としてあげられている[5)]．

命令レベルよりさらに大きな並列処理粒度を利用し，トランジスタ集積度向上に対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や，イタレーション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えるアーキテクチャであると期待されている．たとえば Stanford 大学で研究されている Hydra は，ほぼ同数のトランジスタ規模であるスーパスカラアーキテクチャや Simultaneous Multithreading （SMT）アーキテクチャより高いパフォーマンスが得られている[6)]．また，従来のスーパスカラやアウトオブオーダなどを利用したプロセッサでは性能向上のためには高速化するハードウェアを開発しなければならないのに対し，チップマルチプロセッサはプロセッサコア数を増やすことにより性能向上を得られるアーキテクチャのため，ハードウェア設計の再利用により開発コストが削減されるので費用対効果の高いアーキテクチャである[3)]．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[7)]．

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不可欠である．

ソフトウェア最適化では，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．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，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し，メモリアクセスを効率化する必要がある．

筆者らは実効性能が高く価格性能比およ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[8)]．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こと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[9)]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[10)~12)]．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ってきたが，マルチメディアアプリケーションにおける OSCAR CMP の性能を確認するのため，まず第 1 歩として静止画像圧縮技術として最も一般的に用いられている規格の 1 つである JPEG エンコードを用い，JPEG エンコードの特徴を利用した粗粒度タスク並列処理と近細粒度タスク並列処理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を行い，OSCAR CMP 上でその性能を確認した[13)]．

本論文では，マルチメディアアプリケーションに有効な並列処理手法の提案に加え，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ヘッド問題の解決の有効な手法を提案するため MPEG2 エンコードを用い，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ともなう MPEG2 エンコードの並列処理手法を提案し，OSCAR CMP 上での性能について述べる．MPEG2 エンコードはリアルタイム処理を行う場合，1 秒間に 100 億命令のオーダ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[14)~17)]．また，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 1 つである．

以下，2 章で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の OSCAR チップマルチプロセッサアーキテクチャの説明，3 章で MPEG2 エンコードアルゴリズムの説明を行い，4 章で従来のプロセッサ上での MPEG2 エンコードの高速化の説明およびその性能について述べ，5 章で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，6 章で OSCAR チップマルチプロセッサ上での提案手法の性能評価結果について述べた後，7 章で結論を述べる．

2. OSCAR チップマルチプロセッサアーキテクチャ

本章では，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．

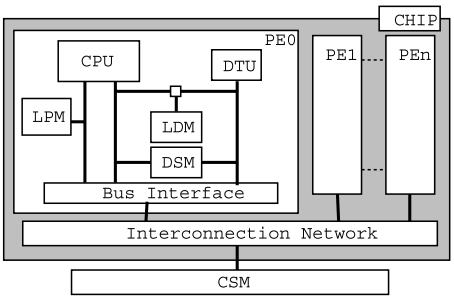


図 1 OSCAR CMP アーキテクチャ
Fig. 1 OSCAR CMP architecture.

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 の処理とオーバラップして行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），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，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 PE 外部に接続したアーキテクチャである．

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うことが効率の良い並列処理のために必要となる．

3. MPEG2 エンコードアルゴリズム

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る基本的なアルゴリズムで構成されている．

本論文では，MPEG2 エンコードアルゴリズムの参照実装として，MediaBench[18)] 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 1** に示される MediaBench で用いられているエンコード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う．

表 1 MPEG2 エンコードパラメータ
Table 1 MPEG2 encoding parameters.

# 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	112 kbit
progressive sequence	false
intra dc precision	8 bit
quantization scale type	non-linear
entropy scan type	Zig-Zag scan
search window size	P:11×11
	B1（forw.）: 3×3
	B1（backw.）: 7×7
	B2（forw.）: 7×7
	B2（backw.）: 3×3

MPEG2 ビデオのデータ構造は **図 2** 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 1 つ 1 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．本論文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，**図 3** 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．

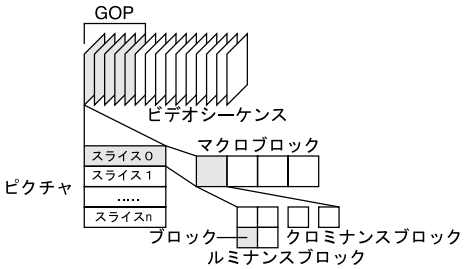


図 2 MPEG2 データ構造
Fig. 2 MPEG2 data structure.

動き推定 動きベクトルの探索を行いピクチャタイプにより符号化モードの設定を行う．

動き予測 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号化対象ピクチャを生成する．

DCT モード選択 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 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 DCT を適用したピクチャに対し量子化の適用を行い，各種ヘッダの送出，ビットストリーム出力を行う．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う．

逆量子化 後続のエンコードで参照するピクチャをすで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 逆量子化適用後のピクチャに対して逆 DCT を適用し，後続のエンコード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において各マクロブロック単位に符号化が行われ，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．

4. 従来のプロセッサ上での MPEG2 エンコードの高速化

ここでは，従来のプロセッサ上での MPEG2 エンコードの高速化について確認するために Intel Xeon プロセッサを用いた場合の MPEG2 エンコードの高速化およびその性能を Intel Xeon 搭載 PC 上で評価した結果について述べる．

Intel Xeon プロセッサでは，ストリーミング SIMD

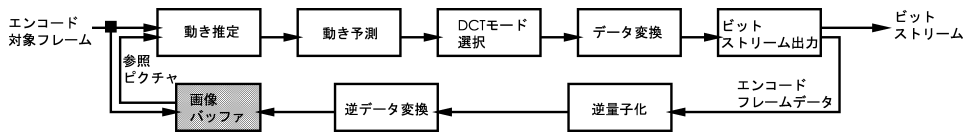


図 3 MPEG2 エンコードブロック図
Fig. 3 Block diagram of MPEG2 encoding.

拡張命令セット（SSE）と呼ばれる 1 つの命令で複数のデータの演算を同時に行うマルチメディア拡張命令セットを備えており，この SSE を利用することでアプリケーションの高速化を行うことができる．SSE とは，たとえばデータ依存のない複数の “add” 命令を 1 つの命令で行う “padd” や同様にデータ依存のない複数のシフト演算を行う “psrl” などである．これらを用いることで積和演算などのマルチメディアアプリケーションで利用される演算の高速化が行われる．

SSE の効果を確認するための評価を Intel Xeon プロセッサを搭載した PC 上で行った．評価に用いるプログラムは，後述するチップマルチプロセッサとの性能と比較が行えるように 6 章で用いるプログラムと同一のものを利用した．コンパイラは，Intel Fortran Compiler Version 8.0（Build 20040611Z）を用い，SSE を利用しない場合は，コンパイルオプションなし，SSE を利用する場合はコンパイラが自動的に Intel Xeon プロセッサ用に SSE 命令を生成する “/QxW” オプションを用いてコンパイルを行った．評価に用いた PC のスペックを **表 2** に示す．SSE の効果を評価するために物理プロセッサ 1 つ HT なしで評価を行った．また，計測には，Intel Vtune Performance Analyzer Version 7.1（Build 14038）を用いた．

評価結果を **表 3** に示し，キャッシュヒット率を **表 4** に，MPEG2 エンコードの各エンコードステージの実行時間およびその割合を **表 5** に示す．評価プログラムは，入力データサイズが小さいため，キャッシュのヒット率が非常に高くメモリアクセスによる速度低下はほとんどないと考えられる．評価結果より，ストリーミング SIMD 拡張命令セットを利用することにより各エンコードステージの時間が短縮されて速度向上が得られ，SSE 命令の利用により 2.45 倍の速度向上が確認できた．

表 2 評価マシンのスペック
Table 2 Specification of PC.

モデル名		DELL PRECISION 450
CPU		Intel Xeon プロセッサ 2.8 GHz
hoge		with HT Technology x2
L1I Cache（TC）		12 K μops
L1D Cache	Size	8 KB
	Way	4-way
	Line Size	64 B
	Latency	2 clocks
	Write Policy	Write Through
L2 Cache	Size	512 KB
	Way	8-way
	Line Size	128 B
	Latency	7 clocks
	Write Policy	Write Back
Main Memory		Dual Channel
hoge		DDR-SDRAM 333 MHz 2 GB

表 3 Intel Xeon 搭載 PC 上での評価結果
Table 3 Evaluation result on Intel Xeon PC.

	SSE なし	SSE あり
実行時間 [ms]	1,047	427

表 4 キャッシュヒット率
Table 4 Cache hit rate.

	SSE なし [%]	SSE あり [%]
L1D Cache	97.4	97.0
L2 Cache	99.8	99.8

5. 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

本章では，チップマルチプロセッサ上での MPEG2 エンコードの効率的な処理を行うために，MPEG2 エンコードの特徴を考慮した並列処理粒度の考察と並列化，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ヘッドの増加による性能低下を考慮したメモリ利用最適化手法の提案および CPU と非同期にデータ転送を行いメモリアクセスオーバヘッドをさらに削減する手法であるデータ転送オーバラップスケジューリングの提案を行う．

5.1 MPEG2 エンコードの粗粒度並列性

MPEG2 エンコードから並列性を考察するために，MPEG2 ビデオデータ構造に注目する．MPEG2 ビデオデータは 3 章で述べたように階層的な構造をしており，各レイヤのデータはランダムアクセス性の確保や

表 5 エンコードステージの実行時間と割合
Table 5 Execution time and ratio of each encoding stage.

	実行時間 [ms]（割合 [%]）	
ステージ	SSE なし	SSE あり
動き推定	816 （77.9%）	320 （74.9%）
動き予測	7 （0.7%）	5 （1.1%）
DCT モード選択	3 （0.3%）	3 （0.6%）
データ変換	86 （8.2%）	19 （4.4%）
ビットストリーム出力	48 （4.6%）	26 （6.1%）
逆量子化	7 （0.6%）	4 （0.9%）
逆データ変換	14 （1.3%）	9 （2.1%）
その他	67 （6.4%）	42 （9.9%）

エラー耐性の補償などのためにデータどうしでその独立性が確保されている部分がある．ここでは，各レイヤでのデータの独立性に注目し，並列性の考察を行う．

まず，GOP レイヤでは，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，ピクチャレイヤでは **図 4** 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．1 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う．2 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．3 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考察すると，

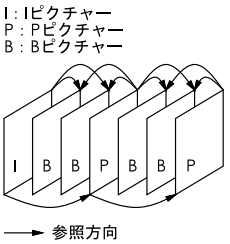


図 4 ピクチャタイプ
Fig. 4 Picture types.

GOP レイヤ，スライスレイヤ，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ではデータ依存がなく，各レイヤで並列処理が可能であることが分かる．

5.2 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，並列性の利用だけではなく，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ヘッドの削減も重要となる．そのため，各レイヤで利用されるデータ容量を考慮した並列処理粒度の決定も重要な要素の 1 つである．ここでは，MPEG2 エンコードのメモリ利用量がどの程度になるか考察を行う．

5.2.1 各レイヤでのメモリ利用量

まず，マクロブロックレイヤ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 32 K 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1 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 $32 \times 11 = 352$ K バイト，QVGA では約 $32 \times 20 = 640$ K 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 レイヤは，複数のピクチャを持ったスライス

レイヤよりさらに大きな粒度のデータが必要となるレイヤである．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における並列性の利用が有効であると考えられる．

5.3 マクロブロックレベルの並列性の抽出

5.2 節の解析から得られた考察より，マクロブロックレベルの並列性を利用した並列処理を行う．ここでは，コンパイラによる並列性の抽出法について述べる．

5.3.1 コンパイラによる並列性の抽出

本論文では，OSCAR 自動並列化コンパイラ[9] を用いて並列性の抽出を行う．コンパイラでは，はじめに，プログラムを疑似代入文ブロック（BPA），繰返しブロック（RB），サブルーチンブロック（SB）の 3 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たりして 1 つの BPA を生成する．MT 生成後，コンパイラは BPA，RB，SB などの MT 間の制御フローとデータ依存を解析し，結果をマクロフローグラフ（MFG）[9] として表す．次に，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[9] が行われる．その結果は，各 MT の制御依存とデータ依存を表したマクロタスクグラフ（MTG）[9] として表現される．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．ここで，PG に割り当てられた繰返しブロックが，イタレーションレベルでデータ依存がなく並列処理可能な Doall ループの場合は，PG 内 PE 間でループ並列処理が行われる．

5.3.2 マクロブロックレベルの並列性

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ディング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 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

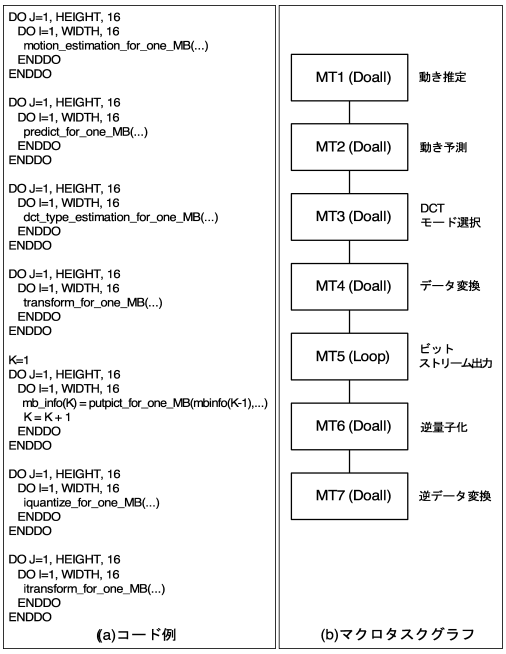


図 5 MPEG2 エンコードのコード例とマクロタスクグラフ
Fig. 5 Code fragment of MPEG2 encode and macro task graph.

演算できる．**図 5** 左は，MediaBench での実装を基にした MPEG2 エンコーディング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うと，各ステージがそれぞれ MT として定義され図 5 右のような 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 5 右より，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 1 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ヘッドが発生し全体的な実行効率が低下する．

5.4 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べ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いデータローカリティ利用の向上を行うデータローカライゼーション手法[10)~12)]により解決される．

5.4.1 データローカライゼーション手法

データローカライゼーション手法は，以下の手順で行われる．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[10)]を行う．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）[10)]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[11)]やチップマルチプロセッサ[12)]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．

5.4.2 データローカライゼーション手法の MPEG2 エンコードへの適用

本項では，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ローカルメモリ容量を考慮し，各ステージのループがマクロブロックレベルの部分ループになるようにコンパイラ指示文により指定し，コンパイラにより各ステージをマクロブロッ

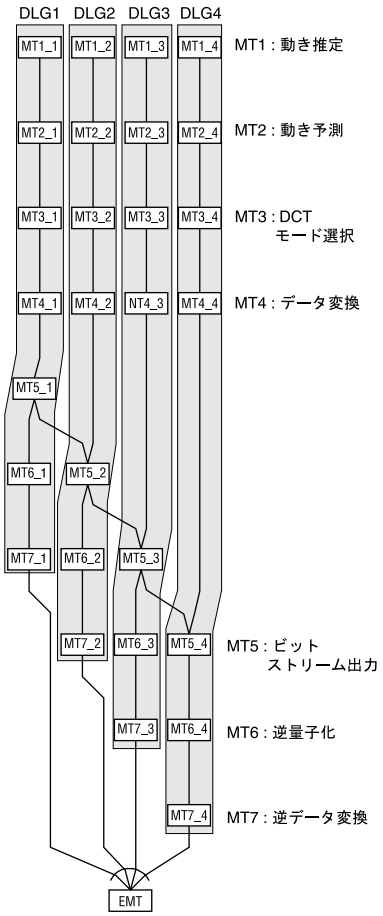


図 6 データローカライゼーション手法適用後の MPEG2 エンコードのタスクグラフ
Fig. 6 MacroTask Graph of MPEG2 encoding with data localization.

クレベル処理の部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う．MPEG2 エンコードでは各 MT は 1 つのマクロブロックのエンコード処理を行う粒度となる．たとえば，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 6** のマクロタスクグラフ（MTG）として表される．図 6 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，$x = 7$ が逆データ変更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 6 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分かる．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う方が効率的である．

以上のように，マクロタスク間のデータ共有量を情報としてコンパイラにより，各 MT のプロセッサへのスケジューリングを行うと，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うようにスケジュールされる．例として，総マクロブロック数が 8 個のときのスケジュール結果を**図 7** に示す．

5.5 データ転送オーバ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ラップすることでデータ転送オーバヘッドを隠蔽する．

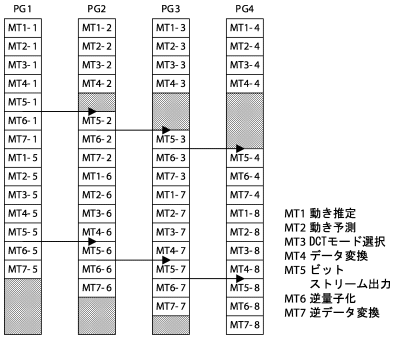


図 7 スケジューリング
Fig. 7 Scheduling result.

ここで，ロードオーバヘッド隠蔽技術は，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 の先頭で行われていた D_i のデータロードによるオーバヘッドの隠蔽を行うことを示し，プレロードと定義する．また，ストアオーバヘッド隠蔽技術は，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ヘッドの隠蔽を行うことを示し，ポストストアと定義する．

5.5.1 データ転送オーバ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ラップすることでデータ転送オーバ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う．提案する手法は，PE 内に CPU と非同期にデータ転送ができるデータ転送ユニットもしくは DMA コントローラと分散共有メモリを有するアーキテクチャで利用できる．

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，1 つの DTU では同時に 1 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 前述の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，ポストストアともに同時に行う．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．

5.5.2 MPEG2 エンコードでのデータ転送オーバ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ラップを考慮したスケジューリングに従い，コンパイラによりプレロード・ポストストアを決定すると，**図 8** のようなスケジューリング結果が得られる．

図 8 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示し，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，時間軸はコンパイラでスケジューリング時に利用されるスケジューリングクロックで表示されている．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う．そのため，図 8 における $MT1\_i$ の先頭までにデータのロードおよび，$MT7\_i$ 終了後にデータのストアが必要となる．各 PG の先頭の $MT1\_1$，$MT1\_2$ のロード $LD1$，$LD2$ は，先行するマクロタスクがないためプレロードはできず，各 PG の最後の $MT7\_3$，$MT7\_4$ のストア $ST3$，$ST4$

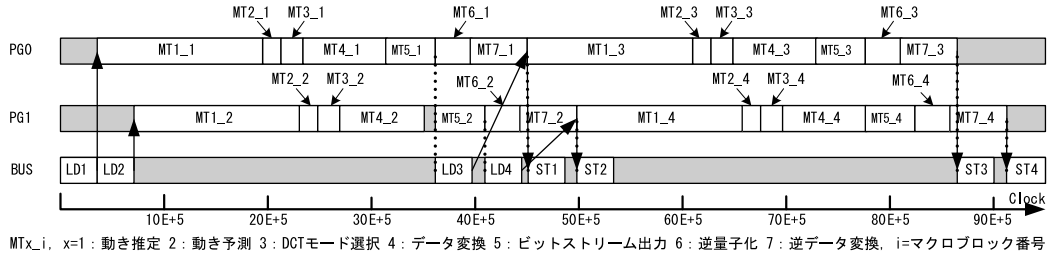


図 8 データ転送オーバラップを考慮したスケジューリングイメージ
Fig. 8 A scheduling result image considering data transfer overlapping.

は，後続するマクロタスクがないためポストストアは行えない．しかし，中間に位置する $MT1\_3$，$MT1\_4$ のロード $LD3$，$LD4$ は，先行するマクロタスクとオーバラップしてプレロードし，$MT7\_1$，$MT7\_2$ のストア $ST1$，$ST2$ は，後続するマクロタスクとオーバラップしてポストストアするスケジューリング結果が得られた．

6. 性能評価

本章では，MPEG2 エンコードに対して 3 章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．

6.1 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う．このとき，並列処理を行う階層として 5 章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手動で指定し，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ラップスケジューリングを 5.4.2 および 5.5.1 項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 の各メモリサイズは，LDM が 128 KB，DSM が 32 KB，CSM はオフチップメモリで 3 MB とし，各メモリへのメモリアクセスレイテンシは，OSCAR CMP の動作周波数を 400 MHz と 2.8 GHz を想定し **表 6** に示すものとした．表 6 は，90 nm プロセスにおけるレイテンシを CACTI 3.0 [19] を用い推定を行った[20]．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．

表 6 OSCAR CMP メモリアクセスレイテンシ
Table 6 Memory access latencies for OSCAR CMP.

クロック周波数 [MHz]	400	2,800
LDM （128 KB） [clock（s）]	1	2
DSM （32 KB） [clock（s）]	1	3
CSM （3 MB） [clock（s）]	24	105

6.2 評価結果

動作周波数 400 MHz 時の OSCAR CMP 上での性能評価結果を **図 9** に，動作周波数 2.8 GHz 時の OSCAR CMP 上での性能評価結果を **図 10** にそれぞれ示す．図 9 および 図 10 中横軸の “1PE”，“2PEs”，“4PEs” および “8PEs” 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，右側は本論文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．

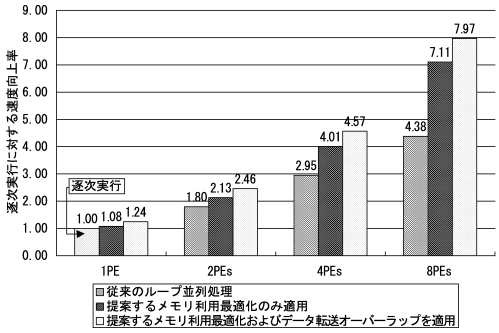


図 9 MPEG2 エンコード評価結果（OSCAR CMP@400 MHz）
Fig. 9 Evaluation result on OSCAR CMP@400 MHz.

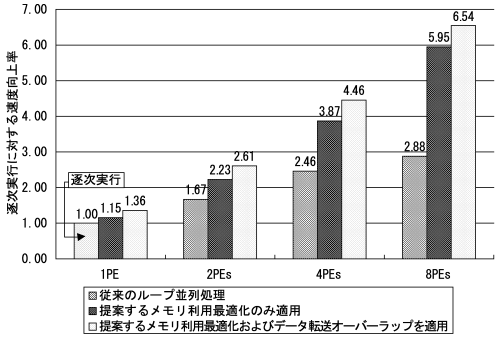


図 10 MPEG2 エンコード評価結果（OSCAR CMP@2.8 GHz）
Fig. 10 Evaluation result on OSCAR CMP@2.8 GHz.

性能評価結果より，動作周波数 400 MHz 時の OSCAR CMP では，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 倍の速度向上率がそれぞれ得られた．同様に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では，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 倍の速度向上がそれぞれ得られ，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ラップスケジューリングを適用した場合では，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 倍の速度向上率がそれぞれ得られた．動作周波数 2.8 GHz 時の OSCAR CMP では，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 1.67 倍，4 プロセッサ利用時 2.46 倍，8 プロセッサ利用時 2.88 倍の速度向上率がそれぞれ得られた．同様に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では，1 プロセッサ利用時 1.15 倍，2 プロセッサ利用時 2.23 倍，4 プロセッサ利用時 3.87 倍，8 プロセッサ利用時 5.95 倍の速度向上がそれぞれ得られ，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ラップスケジューリングを適用した場合では，1 プロセッサ利用時 1.36 倍，2 プロセッサ利用時 2.61 倍，4 プロセッサ利用時 4.46 倍，8 プロセッサ利用時 6.54 倍の速度向上率がそれぞれ得られた．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．次に，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うと，動作周波数 400 MHz 時の OSCAR CMP では，最大で，8 プロセッサ利用時 1.62 倍の性能差となり，動作周波数 2.8 GHz 時の OSCAR CMP では，最大で，8 プロセッサ利用時 2.07 倍の性能差となった．この性能差は，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．また，アクセスレイテンシの低い LDM とアクセスレイテンシの高い CSM との速度差が大きいほどデータローカライゼーション手法がより有効であることが分かる．

さらに，データローカライゼーション手法に加え，データ転送オーバラップスケジューリングを適用した場合は，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に対して，動作周波数 400 MHz 時の OSCAR CMP では，最大で 8 プロセッサ利用時 1.82 倍向上が得られ，動作周波数 2.8 GHz 時の OSCAR CMP では，最大で 8 プロセッサ利用時 2.27 倍の性能が得られ，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．

次に，一般のプロセッサで用いられているマルチメディア拡張命令セットを用いた場合と性能比較を行う．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ラップスケジューリングを適用した

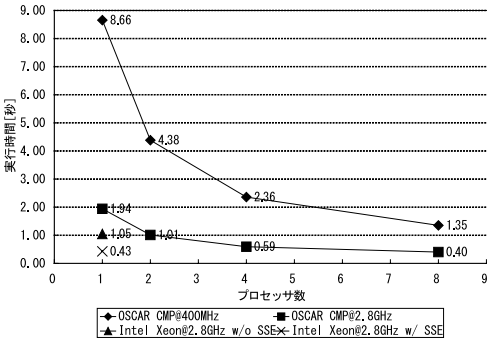


図 11 MPEG2 エンコード実行時間
Fig. 11 Execution time of MPEG2 encode.

MPEG2 エンコードを動作周波数 400 MHz 時および 2.8 GHz 時のの OSCAR CMP 上で評価したときの実行時間と 4 章で評価したストリーミング SIMD 拡張命令セットを用いた MPEG2 エンコードを動作周波数 2.8 GHz の Intel Xeon 搭載 PC 上での評価したときの実行時間を **図 11** に示す．図 11 中横軸は利用プロセッサ数を示し，縦軸は実行時間を示す．

動作周波数 400 MHz 時の OSCAR CMP と動作周波数 2.8 GHz の Intel Xeon プロセッサ搭載 PC とでは，1 プロセッサ利用の OSCAR CMP は，0.04 倍もの速度低下であったものが，8 プロセッサ利用の OSCAR CMP では 0.3 倍程度の速度低下に抑えることができた．このことは，動作周波数 7 分の 1 にともなう消費電力面での有用性や，ハードウェアが単純なシングルイシューコアを集積することによる開発コスト削減への有用性などへのメリットが大きいと考えらる．また，動作周波数 2.8 GHz 時の OSCAR CMP と同一動作周波数 2.8 GHz の Intel Xeon プロセッサ搭載 PC では，1 プロセッサ利用の OSCAR CMP では，0.21 倍の速度低下であったものが 8 プロセッサ利用することで 1.05 倍の性能向上が得られた．このことより，OSCAR CMP のスケーラビリティの利用により性能を得ることができ，SSE を有効にした Intel Xeon プロセッサと同等もしくはそれ以上の性能を得ることが確認できた．

7. ま と め

本論文では，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ラップさせデータ転送オーバヘッドを最小化するデータ転送オーバラップスケジューリングからな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，その性能評価を行った．その結果，提案手法は動作周波数 400 MHz を想定した OSCAR CMP では，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 倍速度向上率が得られ，動作周波数 2.8 GHz を想定した OSCAR CMP では，逐次実行に対し，1 プロセッサ利用時 1.36 倍，2 プロセッサ利用時 2.61 倍，4 プロセッサ利用時 4.46 倍，8 プロセッサ利用時 6.54 倍の速度向上率が得らた．このことから，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．

また，マルチメディア拡張命令セットを搭載したスーパスカラ汎用プロセッサ Intel Xeon を搭載した PC との比較においても動作周波数 2.8 GHz の OSCAR CMP で 8 プロセッサ利用時で 1.05 倍の性能向上が得られ，スーパスカラやマルチメディア拡張命令セットなどの複雑なアーキテクチャを持った一般のプロセッサと比較した場合でも，単純な CPU を用いた OSCAR CMP 上で同等もしくは，それ以上の性能を得られることが確認できた．

マルチメディアアプリケーションでは，MPEG2 エンコードのように，ある単位のデータを処理単位としたデータブロックによるエンコード/デコードを行うアプリケーションが多く，MP3 や H.264 などでは，MPEG2 同様入力データをデータブロック単位で処理を行っている．そのため，本論文で提案した手法は，データブロック単位で処理を行うマルチメディアアプリケーションにおいて，データブロック単位の処理を粗粒度タスクと定義することにより，汎用的に適用することができると考えられる．

今後の課題として，MPEG2 エンコードのさらなる高速化として MPEG2 エンコードへのマクロブロック処理間の粗粒度タスク並列性とマクロブロック処理内での近細粒度並列性やマルチメディア命令セットを利用した命令レベル並列性を階層的に利用したマルチグレイン並列性の適用，およ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あげられる．

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」，NEDO「先進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），特別研究員奨励費（課題番号 1501202）

および特定課題研究助成費（課題番号 2004B-879）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），枝廣正人氏（NEC）に感謝いたします．

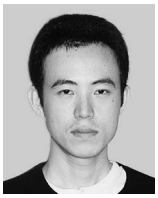
参 考 文 献

1) Diefendorff, K. and Dubey, P.K.: How Multimedia Workloads Will Change Processor Design, *Computer*, Vol.30, No.9, pp.43–45 (1997).
2) Lee, R.B. and Smith, M.D.: Guest Editors' Introduction: Media Processing: A New Design Target, *IEEE Micro*, Vol.16, No.4, pp.6–9 (1996).
3) MIPS Technologies, I.: New Degrees of Parallelism in Complex SOCs, Technical report, MIPS Technologies, Inc. (2002).
4) Wall, D.W.: Limits of Instruction-Level Parallelism, *Proc. 4th International Conference on Architectural Support for Programming Languages and Operating Systems* (*ASPLOS-IV*) (1991).
5) Lappalainen, V., Hamalainen, T.D. and Liuha, P.: Overview of Research Efforts on Media ISA Extentions and Their Usage in Video Coding, *IEEE Trans. Circuits and Systems for Video Technology*, Vol.12, No.8, pp.660–670 (2002).
6) Hammond, L., Nayfeh, B.A. and Olukotun, K.: A Single-Chip Multiprocessor, *IEEE Computer*, Vol.30, No.9, pp.79–85 (1997).
7) Nikitovic, M. and Brorsson, M.: An adaptive chip-multiprocessor architecture for future mobile terminals, *CASES '02: Proc. 2002 international conference on Compilers, architecture and synthesis for embedded systems*, New York, NY, USA, pp.43–49, ACM Press (2002).
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, pp.692–703 (2001).
9) Kasahara, H., Obata, M. and Ishizaka, K.: Automatic Coarse Grain Task Parallel Processing on SMP using OpenMP, *Proc. 12th Workshop on Languages and Compilers for Parallel Computing* (2000).
10) 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, pp.2054–2063 (1999).
11) Ishizaka, K., Obata, M. and Kasahara, H.: Coarse Grain Task Parallel Processing with Cache Optimization on Shared Memory Multiprocessor, *LCPC*, Dietz, H.G. (Ed.), Lecture Notes in Computer Science, Vol.2624, pp.352–365, Springer (2001).
12) 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 (2003).
13) 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 JPEG エンコーディングのマルチグレイン並列処理，情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，Vol.43, No.Sig.6 (HPS5), pp.153–162 (2002).
14) Zhang, N. and Wu, C.H.: Study on Adaptive Job Assignment for Multiprocessor Implementation of MPEG2 Video Encoding, *IEEE Trans. Industrial Electronics*, Vol.44, No.5, pp.726–734 (1997).
15) Liao, H. and Wolfe, A.: Available Parallelism in Video Applications, *International Symposium on Microarchitecture*, pp.321–329 (1997).
16) Iwata, E. and Olukotun, K.: Exploiting coarse-grain parallelism in the MPEG-2 Algorithm (1998).
17) Sohoni, S., Xu, Z., Min, R. and Hu, Y.: A Study of Memory System Performance of Multimedia Applications, *Proc. Joint International Conference on Measurement and Modeling of Computer Systems* (*SIGMETRICS-01/PERFORMANCE-01*), *ACM SIGMETRICS Performance Evaluation Review*, Vol.29, 1, pp.206–215, ACMPress (2001).
18) Lee, C., Potkonjak, M. and Mangione-Smith, W.H.: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, *30th International Symposium on Microarchitecture* (*MICRO-30*) (1997).
19) Shivakumar, P. and Jouppi, N.P.: CACTI 3.0:An Integrated Cache Timing, Power and Area Model, Wrl-2001-2, Compaq Computer Corporation Western Research Laboratory (2001).
20) Kimura, K., Wada, Y., Nakano, H., Kodaka, T., Shirako, J., Ishizaka, K. and Kasahara, H.: Multigrain Parallel Processing on Compiler Cooperative Chip Multiprocessor, *Proc. 9th Workshop on Interaction between Compilers and Computer Architectures* (*INTERACT-9*), pp.11–20 (2005).

(平成 16 年 9 月 24 日受付)
(平成 17 年 7 月 4 日採録)

小高　剛（正会員）

昭和 52 年生．平成 11 年早稲田大学理工学部電気電子情報工学科卒業．平成 11 年（株）ユニシアジェックス入社．平成 12 年同社退社後，早稲田大学大学院理工学研究科入学．平成 17 年早稲田大学大学院理工学研究科電気工学専攻博士課程修了．博士（工学）．平成 16 年同大学理工学部助手．平成 17 年株式会社東芝入社．現在に至る．



中野　啓史（学生会員）

昭和 52 年生．平成 13 年早稲田大学理工学部電機電子情報工学科卒業．平成 15 年同大学大学院理工学研究科電気工学専攻修士課程修了．平成 15 年同大学院博士課程進学．現在に至る．

木村　啓二（正会員）

昭和 47 年生．平成 8 年早稲田大学理工学部電気工学科卒業．平成 13 年同大学大学院理工学研究科電気工学専攻博士課程修了．博士（工学）．平成 11 年同大学同学部助手．平成 16 年同大学理工学部コンピュータ・ネットワーク工学科専任講師．平成 17 年同助教授，現在に至る．マルチグレイン並列処理用チップマルチプロセッサアーキテクチャに関する研究に従事．

笠原　博徳（正会員）

昭和 32 年生．昭和 55 年早稲田大学理工学部電気工学科卒業．昭和 60 年同大学大学院博士課程修了．博士（工学）．昭和 58〜60 年同大学助手．昭和 60 年日本学術振興会第 1 回特別研究員．昭和 61 年早稲田大学理工学部専任講師．昭和 63 年同大学助教授．平成 9 年同大学教授．現在 CS 学科教授，アドバンストチップマルチプロセッサ研究所所長．昭和 60 年カリフォルニア大学バークレー，平成元〜2 年イリノイ大学 Center for Supercomputing R & D 客員研究員．昭和 62 年 IFAC World Congress 第 1 回 Young Author Prize，平成 9 年情処坂井記念特別賞，平成 16 年 STARC 共同研究賞受賞．主な著書『並列処理技術』（コロナ社）．情報処理学会：ARC 主査，論文誌 HG 主査，会誌 HWG 主査，ACM：ICS Program Vice Chair，IEEE：CS Japan Chair，文科省：地球シミュレータ中間評価委員，経産省/NEDO：コンピュータ戦略 WG 委員長，“並列化コンパイラ”“マルチコア”プロジェクトリーダ等．