ヘテロジニアスマルチコア上での階層的粗粒度タスクスタティックスケジューリング手法

和田康孝[†] 林明宏[†] 伊能健人[†]
益浦健[†] 白子準[†] 中野啓史[†]
鹿野裕明[†] 木村啓二[†] 笠原博徳[†]

本稿では，ヘテロジニアスマルチコア上での階層的粗粒度タスクスタティックスケジューリング手法について述べる．ヘテロジニアスマルチコアは，1 チップ上に汎用プロセッサに加え，動的再構成可能プロセッサ (DRP) や信号処理用プロセッサ (DSP) などのアクセラレータを複数集積したプロセッサで，低消費電力で高い処理性能を得ることができるアーキテクチャとして情報家電等の分野で注目を集めている．本稿で提案するスタティックスケジューリング手法は，ループやサブルーチン，基本ブロック間の並列性を利用する粗粒度タスク並列処理において，各タスクの特性，チップ上の各コアの種類を考慮して処理時間を最小とするようにタスクを汎用コア及びアクセラレータに割り当て，コア間でのデータ転送は DMA を用いてタスク処理とオーバーラップして行うことにより，プログラムの階層的な並列性とチップ上のアクセラレータを最大限利用する手法である．本手法を MP3 エンコーダに適用し評価した結果，SH4A 1 コアのみを用いた場合に対して，SH4A 4 コアで 3.97 倍，SH4A 2 コアと DRP 2 コアで 12.64 倍，SH4A 4 コアと DRP 4 コアを用いたときに 24.48 倍の速度向上を得られることが確認できた．

A Hierarchical Coarse Grain Task Static Scheduling Scheme on a Heterogeneous Multicore

Yasutaka Wada,[†] Akihiro Hayashi,[†] Taketo Iyoku,[†]
Takeshi Masuura,[†] Jun Shirako,[†] Hirofumi Nakano,[†]
Hiroaki Shikano,[†] Keiji Kimura[†] and Hironori Kasahara [†]

This paper proposes a static scheduling scheme for hierarchical coarse grain task parallel processing on a heterogeneous multicore processor. A heterogeneous multicore processor integrates not only general purpose processors but also accelerators like dynamically reconfigurable processors (DRPs) or digital signal processors (DSPs). Effective usage of these accelerators allows us to get high performance and low power consumption at the same time. In the proposed scheme, the compiler extracts parallelism using coarse grain parallel processing and assigns tasks considering characteristics of each core to minimize the execution time of an application. Performance of the proposed scheme is evaluated on a heterogeneous multicore processor using MP3 encoder. Heterogeneous configurations give us 12.64 times speedup with two SH4As and two DRPs and 24.48 times speedup with four SH4As and four DRPs against sequential execution with one SH4A core.

1. はじめに

携帯電話，ゲーム機，DVD レコーダ，デジタル TV 等に代表される情報家電の市場拡大に伴い，その付加価値の源泉となる低消費電力高性能プロセッサに対する要求は年々高まっている．特に，価格性能比の向上や開発期間の短縮，低消費電力化はそのプロセッサの市場競争力を大きく左右する．これらの要求に対応するため，現在では組み込みシステムにおいても，1 つのチップ上に複数のプロセッサコアを集積したマルチコアアーキテクチャ[1)2)] が積極的に用いられるようになってきている．

さらに，情報家電システムにおいてよく用いられるメディアアプリケーションなどの処理を高速化するために，信号処理用プロセッサ (DSP) や動的再構成可能プロセッサ (DRP) などのアクセラレータをチップ上にあわせて集積するヘテロジニアスマルチコア[3)4)] も開発されている．

ヘテロジニアスマルチコアプロセッサでは，アクセラレータを利用することで高速な演算処理と低消費電力化を両立することが可能となるが，その反面，効率よくアクセラレータを利用するためのチューニングが大変困難なものとなる．そのため，実効性能の向上と開発期間の短縮を両立させるためには，チップ内の各コアの特性やデータ転送を考慮して適切に処理を割り当てる手法およびそれを実現する自動並列化コンパイラの開発が求められる．

本稿では，プログラム全域から並列性を引き出して利用することが可能な階層的マルチグレイン並列処理[5)] の主要な構成要素である粗粒度タスク並列処理において，ヘテロジニアスマルチコア上の各コアの特性を考慮してタスクの割り当てを行うスタティックスケジューリング手法について述べる．

以下，2 章で本手法が対象とするヘテロジニアスマルチコアアーキテクチャについて，3 章でヘテロジニアスマルチコア上での粗粒度タスク並列処理について，4 章でヘテロジニアスマルチコアアーキテクチャを対象とした粗粒度タスクのスタティックスケジューリングアルゴリズムについて，5 章で MP3 エンコーダを用いた性能評価についてそれぞれ述べる．

† 早稲田大学理工学術院基幹理工学部情報理工学科
Department of Computer Science and Engineering,
School of Fundamental Science and Engineering,
Waseda University

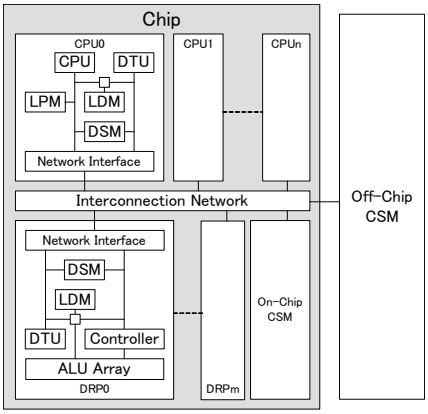
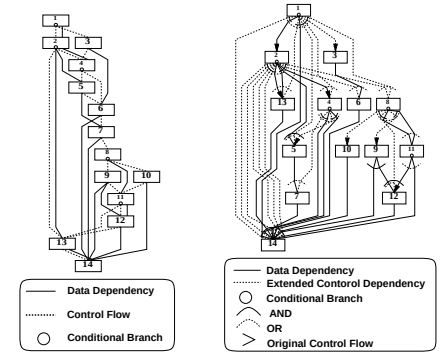


図 1 OSCAR 型メモリアーキテクチャを持つヘテロジニアスマルチコアプロセッサの例



(a) Macro Flow Graph (MFG) (b) Macro Task Graph (MTG)

図 2 マクロフローグラフ (MFG)，マクロタスクグラフ (MTG) の例

2. ヘテロジニアスマルチコアアーキテクチャ

本章では，汎用コアに加えて動的再構成可能プロセッサ (DRP) や DSP などのアクセラレータを 1 チップ上に集積した，ヘテロジニアスマルチコアプロセッサアーキテクチャについて述べる．

2.1 全体の構成

本手法で対象とする OSCAR 型メモリアーキテクチャ[6)7)8)] を持つヘテロジニアスマルチコアプロセッサは，各々がローカルメモリを持つ汎用 CPU コアおよび DRP などのアクセラレータコアを複数バスやクロスバ等の結合網で集中共有メモリ (CSM) へ接続したものである．

2.2 プロセッサエレメント (PE) の構成

各プロセッサエレメント (PE) は，図 1 に示すように，汎用プロセッサコアまたはアクセラレータコア，ローカルデータメモリ (LDM)，2 ポート構成の分散共有メモリ (DSM) およびデータ転送ユニット (DTU) を持つ．

LDM はアプリケーションプログラム中においてコンパイラが検出あるいは分割配置により割り当てたプロセッサプライベートデータを格納する．DSM は上述のように 2 ポート構成となっており，他の PE からも同時にアクセスが可能である．DSM は，タスク間のデータ・同期フラグの授受に利用される．DTU は高機能 DMAC で，プロセッサの処理と非同期に PE 間，PE-CSM 間のデータ転送を行うことが可能であり，データ転送とタスク処理のオーバーラップのために利用される．ローカルプログラムメモリは各 PE が実行するプログラムコードを格納する為に用いられる．また，アクセラレータ PE は動的再構成可能プロセッサの機能書き換えやタスクのスケジューリング情報の処理，アクセラレータコアの起動，その他一定の計算処理が可能なコントローラを持つ．

3. ヘテロジニアスマルチコア上での階層的粗粒度タスク並列処理

本章では，ヘテロジニアスマルチコアアーキテクチャ上での階層的粗粒度タスク並列処理について述べる．

3.1 粗粒度タスク並列処理

粗粒度タスク並列処理では，対象とするプログラムはまず擬似代入文ブロック (BPA)，繰り返しブロック (RB)，サブルーチンブロック (SB) の 3 種類の粗粒度タスク (マクロタスク (MT)) に分割される．MT 生成後，MT 間のコントロールフローおよびデータ依存を解析し，図 2(a) に示すようなマクロフローグラフ (MFG) が生成される．さらに最早実行可能条件解析によって MFG から MT 間の並列性を抽出して図 2(b) に示すようなマクロタスクグラフ (MTG) を生成する[9)10)]．この際，RB 及び SB はネスト構造を持つ場合があり，その場合には内部ボディ部に対して階層的に MT 及び MTG 生成を行う．

その後，各階層の MT は 1 つ以上のプロセッサエレメント (PE) をグループとしたプロセッサグループ (PG) に割り当てられる．このとき，MTG 内に条件分岐等がなければ，プロセッサ間の同期やデータ通信などのオーバーヘッドを最小化するために静的に MT を割り当てるスタティックスケジューリング手法が適用される．本稿ではこのスタティックスケジューリング手法を対象としており，実行時のタスク割り当て等のオーバーヘッドを最小化することができる．また，本稿では取り扱わないが，MTG に条件分岐等による実行時不確定性が存在する場合には，MT のコードに加えて MT の最早実行可能条件を管理しつつ MT を PG に割り当てるためのスケジューラのコードも合わせて生成し，実行時に MT を PG に割り当てるようにするダイナミックスケジューリング手法が適用される．

さらに MTG 内の SB や RB 内部に粗粒度並列性が存在する場合，その SB や RB 内部で階層的に MTG を生成し，階層的に粗粒度タスク並列処理を適用する．

3.2 ヘテロジニアスマルチコアを考慮したプロセッサグルーピング

3.1 節で述べたように，階層的粗粒度タスク並列処理では PE を階層的にグルーピングした PG が MT の割り当ておよび実行単位として扱われる．しかし，ヘテロジニアスマルチコアを対象とした場合，汎用コア PE では全ての種類の MT を実行することが可能であるのに対し，アクセラレータ PE では特定の処理は非常に高速化することが可能だが，実行可能な MT が限られてしまう．そのため本手法では，汎用コア PE のみの場合と異なり，アクセラレータ PE の利用を考慮したプロセッサのグルーピングを行う．

1 つの方法として，汎用コア同様アクセラレータもプログラムの階層構造に従ってグルーピングし，各 MTG に関連付けする方法が考えられる．しかしこの場合，アクセラレータ数は通常あまり多くないことが想定されるため，粗粒度タスク並列処理を行う MTG のネストが深くなるに従ってアクセラレータの数が足りなくなってしまうことと，固定的に MTG とアクセラレータが関連づけられてしまうため，全体としてアクセラレータの利用効率が下がってしまうことが考えられる．

よって，本手法においては，汎用コア PE は各 MTG の粗粒度並列性と汎用コア PE の数に応じて階層的なグルーピングを行うが[11)]，アクセラレータ PE は階層的なグルーピングからは独立して扱われ，適宜必要に応じてあらゆる階層の MTG から MT が割り当てられるものとする．これにより，アクセラレータで実行可能な MT がどの階層の MTG にどのように分散しているかに関わらず，全汎用コアグループからアクセラレータを有効に利用することができる．

プロセッサのグルーピングの例を図 3 に示す．図 3 は，汎用コア 4 基，DRP および DSP をそれぞれ 2 基ずつ持つヘテロジニアスマルチコアの例である．図 3 では，汎用コアはプログラムの第 1 階層でそれぞれ 2PE からなる 2 つの PG(PG1_0 および PG1_1) にグルーピングされており，各々の PG にこの階層の MTG 中の MT

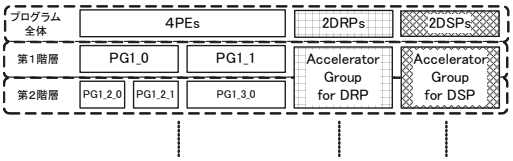


図 3 ヘテロジニアスマルチコアを考慮した階層的なプロセッサグルーピングの例

を割り当てて実行することで粗粒度タスク並列処理が行われている．ある時刻において，PG1_0 に内部に粗粒度並列性を持つ MT が割り当てられた場合，PG1_0 はさらに第 2 階層で 1 プロセッサずつの 2 つの PG(PG1_2_0 および PG1_2_1) にグルーピングされる．このように，汎用コアはプログラムの並列性に応じて階層的にグルーピングされる．それに対し，図中の DRP および DSP は汎用コアの階層グルーピングと異なり，同一機能を持つアクセラレータ毎にグループを生成しているため，あらゆる階層の MTG からのタスク処理依頼を受け付けることができる．

4. ヘテロジニアスマルチコア用タスクスケジューリングアルゴリズム

本章では，第 2 章で述べたヘテロジニアスチップマルチプロセッサを対象とした粗粒度タスクスタティックスケジューリング手法について述べる．

ヘテロジニアスマルチコアプロセッサでは，汎用コア PE では全ての種類の MT が実行可能であるが，アクセラレータ PE ではその種別により実行が可能な MT の種類が限られてしまう．また，同じ MT であっても，どの種別の PE で実行されるかによって処理にかかる時間が異なる．一般に，アクセラレータコアで実行可能な MT はそのままアクセラレータコアに割り当てて実行した方がその MT 自体の実行効率が向上するが，負荷集中時もそのままアクセラレータコアに MT を割り当ててしまっては，逆にプログラム全体の実行時間が伸びてしまう場合が考えられる．本アルゴリズムは，このような場合には各種アクセラレータ PE と汎用コア PE を効率よく利用し，プログラムの実効効率の向上を図るものである．

4.1 複数階層に渡る割り当て優先度　グローバル CP 長

3.2 節で述べたように，本手法では異なる階層の MTG 中の MT を必要に応じてアクセラレータ PE に割り当てる．そのため，実行可能タスクを各 PE に割り当てる際の割り当て優先度を比較する際，全ての階層の MTG に共通の優先度を定義する必要がある．本手法では，プログラム全体を考慮したメインプログラムの出口ノードから各 MTG 中のタスクまでの最長のパス長をグローバル CP 長として定義し，これを割り当ての優先度として採用した．グローバル CP 長の算出例を図 4 に示す．

図 4 において，MT3 の内側の MTG に含まれる MT3-3 のグローバル CP 長を求めることを考える．まず，プログラム全体の出口ノードから，内部に MTG を持つ MT3 の末尾までの最長パス長を求める．ここでは，MT4 を経由するパスが最も長く，パス長は 70 となっている．MT3 は最も上位の MTG に属するので，これが MT3 の末尾までのグローバル CP 長となる．次に，MT3-3 の MTG 内 CP 長は 40 であることがわかる．よって，MT3-3 に対するグローバル CP 長は，MTG 内での CP 長 40 に，MTG を内包する MT3 の末尾までのグローバル CP 長 70 を加えて 110 となる．また，仮に MT3 が RB であった場合には，MT3 内部の MTG の出口ノードから入り口ノードまでの CP 長に (RB の回転数-1) を乗じたものをさらに加える．そうすることで，ループの回転数を考慮した CP 長を算出することができる．

4.2 処理及びデータ転送コストを考慮したタスクスケジューリングアルゴリズム

本手法では，MT を割り当てる対象を，汎用コアの集合であるプロセッサグループ (PG) またはアクセラレータコアとする．また，3.2 節で述べたように，汎用コア PE はプログラムの階層構造に応じて階層的にグルーピングされるが，アクセラレータコアは階層的なグルーピングには含まれず，様々な階層の MTG 上の MT が割り当てられるものとする．

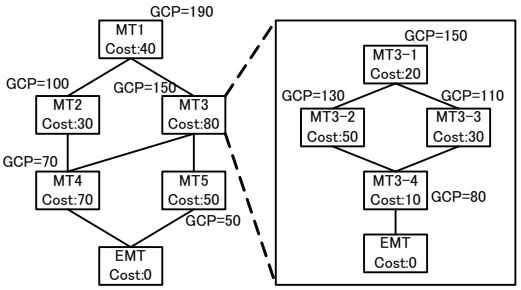


図 4 グローバル CP 長

以下に，ヘテロジニアスマルチコアにおけるスタティックスケジューリングアルゴリズムの基本フローを示す．

step 1: 各 MTG 上の MT に対し，汎用コア PG あるいはアクセラレータで実行可能な MT の場合には実行可能なアクセラレータコアにおけるタスク処理コストの計算を行う．また階層 MTG の出口ノードから各ノードまでの最長パス長に基づく割り当て優先度グローバル CP 長を計算する．

step 2: スケジューリング時刻を 0 にセットする．また，最上位 MTG をスケジューリング中の MTG 一覧に追加する．

step 3: スケジューリング中の MTG に含まれる MT の中から最早実行可能条件を満たした実行可能 MT(レディタスク) を検出し，レディキューに投入して優先度順にソートする．

step 4: 現スケジューリング時刻において，レディタスクと割り当て可能なコアの組み合わせが存在すれば，そのうち最も優先度の高い MT を割り当て対象として選択する．存在しなければ，step 8 へ．ここで,「割り当て可能なコア」とは，汎用コアに関しては現在時刻で IDLE になっている PG を指し，アクセラレータに関しては先行割り当てを行うため当該 MT を実行可能なコアを指す．

step 5: 対象 MT を割り当て可能な各コアに対して，対象 MT の終了時刻を以下の式により推定する．

終了推定時刻
= 先行 MT および CSM からの
DTU 等を用いたデータ転送終了時刻
+ 対象 PG あるいはコア上での MT 実行コスト
+ CSM への DTU 等を用いたデータ転送コスト

割り当て対象 MT が内部にスケジューリング対象の MTG を内包する場合は，内部 MTG のスケジューリングを考慮した終了予測時刻を推定する．

step 6: 割り当て可能なコアのうち，最も終了推定時刻の早い汎用コア PG あるいはアクセラレータコアに対象 MT を割り当てる．

step 7: MT を割り当てた汎用コア PG またはアクセラレータコアの情報をスケジューラに追加し，step 4 へ．なお，割り当てた MT が内部に MTG を内包していれば，その MTG をスケジューリング中の MTG 一覧に追加する．

step 8: 全ての MT が割り当ておよび実行終了済みであれば終了，そうでなければ，次に最も早く MT 実行を終了する汎用コア PG あるいはアクセラレータコアの MT 実行終了時刻を次のスケジューリング時刻とする．

step 9: 当該時刻において実行を終了する MT があれば終了した MT の集合に加え，step 3 へ．また，内部にスケジューリング対象の MTG を含む MT に関して，内部 MTG に含まれる MT の実行が全て終了していれば，これに関しても終了時刻を確定する．

4.3 データ転送コストおよび転送タイミングの決定

本手法では，配列データは DTU を用いて転送を行う．DTU の

駆動は MT の開始あるいは終了のタイミングで行うものとし，タイミングの決定は PE 間ネットワークの状態を考慮して決定される．今回の実装では，MTG をまたいだデータ転送は行わないものとした．

ある MT を割り当て実行する際に必要となるデータの DTU を用いたデータ転送タイミングの決定[12] においては，

1. 転送範囲の解析結果が不定となった場合，整合性確保に必要な CSM からの読み出しおよび CSM への書き戻し
2. MTG 外部から生きて入る配列で，当該 MT が前方露出参照する範囲
3. 異なるコアに割り当てられた先行タスクとの間のフロー依存範囲

の 3 種類のデータ転送パターン各々に対し，1. に関しては，MT の実行直前および直後に転送するものとし，2. に関しては，MTG の開始時刻から現在時刻までの間で転送可能なタイミングを探し，見つかれば転送を挿入し，転送の終了時刻を計算する．もし現在時刻までに転送のタイミングが見つからなければ，現在時刻以降に転送を開始するものとして転送の挿入と終了時刻の計算を行う．3. に関しては，先行タスクの終了時刻から現在時刻までの間で同様にして転送可能なタイミングを探索する．これに関しても，現在時刻までに見つからなければ現在時刻以降を開始時刻とするデータ転送がスケジューリングされる．なお，今回の実装では，この他の CSM への書き戻しに関しては MT の直後に転送を行うこととした．

図 5 にデータ転送の挿入例を示す．なお，図 5 は汎用コアおよびアクセラレータ (DRP) をそれぞれ 1 基，PE 間相互接続網としてバスが 1 本をもつマルチコアを想定したものである．また，便宜上下記の記述は MT の割り当て先の決定後に転送を挿入する形になっている．実際の割り当て処理においては，4.2 節のアルゴリズムの通り，MT の割り当て先決定前に割り当て可能な組み合わせ毎に転送の挿入の状態および MT の終了時刻を考慮し，最も MT の終了時刻が早いものを MT の割り当て先および転送の挿入タイミングとして採用する．

図 5 において，スケジューリング対象の MTG が定義されている．まず，MTG 開始時点でレディタスクとなっている MT1 が CPU0 に割り当てられる．MT1 に対しては，開始時に CSM からのデータ読み出しが，終了時に CSM へのデータの書き戻しが挿入される．次にレディタスクが発生するのは MT1 の終了時であり，レディタスクは MT2，MT3 および MT4 である．まず，割り当て優先度の最も高い MT2 が割り当て対象として選択される．ここで，CSM からの読み出しは MT1 開始時の転送後に続けて挿入することが可能なので，MT1 の開始時の転送と連続して行われるようにスケジューリングされる．このとき，MT1 は MT1 の実行に必要なデータの転送が終わった時点で実行が開始され，MT1 の実行とオーバーラップして MT2 に対する転送が行われる．また，MT2 は MT1 と同じ CPU0 に割り当てられるので MT1 からの転送は必要なく，MT2 の終了時に CSM への書き戻しの転送が挿入される．次に，優先度を比較した結果 MT3 が割り当て対象の MT として選択される．MT3 に関しては，これより先に DRP0 で実行される MT が存在しないため，直前に CSM からの読み出しの転送が行われる．また，異なるコアに割り当てられている MT1 からの転送は，MT1 の終了時刻以降でタイミングを探すことになるが，バスの使用状況を考慮した結果，CPU0 側から DRP0 に対する転送を行うタイミングがないため，DRP0 が MT3 の開始時に CPU0 の DSM からデータを読み出す形で転送が挿入される．また，MT3 の終了時に CSM への書き戻しの転送が挿入される．次に MT4 が DRP0 に先行割り当てされる．このとき，CSM からの読み出しに関しては，MT3 開始時の転送後に連続して行うことが可能であるため，MT3 開始時に転送が挿入される．また，MT1 からのフロー依存範囲のデータは，MT3 の時と同様転送のタイミングが見つからないため，MT4 開始時に直前で転送が行われる．次にレディタスクが発生するのは MT2 の終了時であり，レディタスクは MT5 である．MT5 は汎用コア向けのタスクであるので，CPU0 に割り当てられる．MT5 に関しては，CSM からの読み出しの転送は MT2 と同様 MT1 開始時の転送と連続した形で挿入される．次にレディタスクが発生するのは MT4 の終了時であり，レディタスクは MT6 および MT7 である．ただし，この時点では CPU が IDLE ではないため MT7 は割り当て対象としては選択されないので，MT6 が割り当て対象となる．この時点で DRP0 が IDLE であるので，MT6 は DRP0 に割り当てられる．MT6 に対して，CSM からの読み出しは MT3 開始時の転送と連続した形で挿入される．別のコアに割り当てられている先行タスクである MT2 からのフロー依存範囲の転送は，MT2 終了時の CSM への書き戻しの後続けて転送が可能なので，MT2 終了時に CPU0 側が DRP0 に送信する形で挿入される．次に割り当て可能なコアと MT の組み合わせが発生するのは MT5 の終了時であり，このときのレディタスクは MT7 と MT8 である．このうち優先度の高い MT7 が割り当て対象として選択され，CPU0 に割り当てられる．MT7 に対する CSM からの読み出しは，バスの状態を考慮して MT1 の開始時の転送と連続した形で挿入される．異なるコアに割り当てられている先行タスクである MT3 および MT4 からのデータ転送は，ネットワークの状態を考慮すると MT3 の終了時の転送に加えることができないため，両者とも MT4 の終了時にスケジューリングされる．最後に，MT7 終了時に MT8 が割り当て可能となり，これは汎用コア向けのタスクであるため，CPU0 に割り当てられる．MT8 に対する CSM からの読み出しの転送は，バスの状態を考慮すると MT1 の開始時のタイミングで行われる転送には加えられないため，その次に転送可能となる MT5 の開始時の転送と連続した形で行われる．また，先行タスクである MT6 からの転送は MT6 の終了時に行われる．以上のようにして，MT の実行とネットワークの状態を考慮して転送の挿入が行われる．

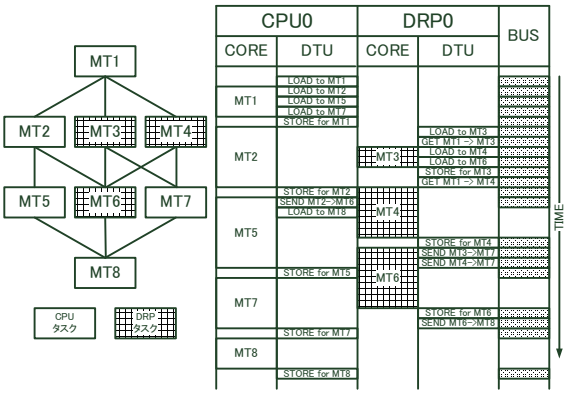


図 5 データ転送挿入例

また，スカラ変数に関しては，DTU を用いず，CPU が直接 DSM あるいは CSM にストアすることによって実現した．

4.4 内部にスケジューリング対象 MTG をもつ MT の割り当て

4.2 節のスケジューリングアルゴリズムにおいて，割り当て対象の MT が内部にスケジューリング対象の MTG を内包していた場合，その内部の MTG のスケジューリングを考慮した上で割り当て先を決定する必要がある．そのため，本手法では．割り当て対象 MT が内部にスケジューリング対象の MTG を内包する場合，内部 MTG に対して再帰的に上記のスケジューリングアルゴリズムを適用し，終了予測時刻を算出する．

また，当該 MT が RB であった場合，繰り返し構造のため他の MTG と同期してスケジューリングしていくことはできない．よって本手法では，RB 内の MTG に関しては他の MTG からは独立して扱う．

具体的には，内部にスケジューリング対象の MTG を内包する RB を割り当てる際，当該時刻において IDLE 状態となっているアクセラレータの中から必要なアクセラレータを確保し，RB の実行が終了するまでの間占有して利用する．このとき，RB が複数ネストされている場合には，外側の RB によって占有されているアクセラレータの中から占有するアクセラレータを探すものとした．ただし，すべての汎用コアが同時に参加する RB の場合は，例外として全てのアクセラレータを占有して利用する．また，データ転送に関しても他の MTG とは切り離してタイミングを決定する．

RB によるアクセラレータ占有の例を図 6 に示す．図 6 は汎用コア 2 基及びアクセラレータ (DRP) を 1 基搭載するヘテロジニアスマルチコアに対するスケジューリング例である．図中に定義される階

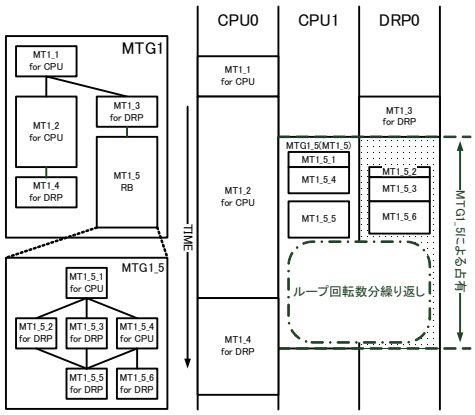


図 6 RB によるアクセラレータの占有

層的な MTG のうち，最上位の MTG である MTG1 は粗粒度タスク並列性が 2 程度あると判断されるので，汎用コアは 2 つの PG にグルーピングされる．この階層においては，まず MT1_1 が CPU0 に割り当てられる．次にレディタスクが発生するのは MT1_1 の終了時であり，レディタスクは MT1_2 と MT1_3 である．ここで，MT1_2 は汎用コアのみで実行可能なタスクであるので，CPU0 に割り当てられ，MT1_3 は DRP 上で実行可能なので DRP0 に割り当てられる．次にレディタスクが発生するのは MT1_3 の終了時であり，レディタスクは MT1_5 である．MT1_5 はこのとき IDLE となっている CPU1 に割り当てられるが，MT1_5 は内部にスケジューリング対象の MTG(MTG1_5) を含み，MTG1_5 はアクセラレータ向け MT を持つ．ここで，MT1_5 は RB であるため，RB の開始から終了までの間，アクセラレータを占有利用することを考える．ここでは，この時点で IDLE である DRP0 が MT1_5 によって占有され，同時に MTG1_5 は他の階層とは独立してスケジューリングされる．MTG1_5 開始時点でレディタスクとなっているのは MT1_5_1 であり，これは CPU1 に割り当てられる．次に MTG1_5 でレディタスクとなるのは MT1_5_2，MT1_5_3，MT1_5_4 である．まず，このうち優先度の最も高い MT1_5_4 は汎用コアのみで実行可能なタスクであるので，CPU1 に割り当てられる．次に優先度の高い MT1_5_2 は，DRP0 が IDLE であるので DRP0 に割り当てられ，MT1_5_3 が先行的に DPR0 に割り当てられる．次に MTG1_5 中でレディタスクが発生するのは MT1_5_4 の終了時であり，レディタスクは MT1_5_6 である．MT1_5_6 は CPU1 と DRP0 との間で終了予測時刻の比較を行った結果，DRP0 に割り当てられる．最後にレディタスクが発生するのは MT1_5_3 の終了時であり，レディタスクは MT1_5_5 である．MT1_5_5 は CPU1 と DRP0 との間で終了予測時刻の比較を行った結果，CPU1 に割り当てられる．ここで 1 イタレーション分のスケジューリングが完了するので，MT1_5 の回転数を考慮して MT1_5 の終了時刻が算出される．このようにして，MTG1_5 は他の MTG とは独立してスケジューリングが行われる．MTG1_5 のスケジューリングおよび MT1_5 の終了時刻の決定後，再び MTG1 のスケジューリングに戻る．MTG1 において次にレディタスクが発生するのは MT1_2 の終了時であり，レディタスクは MT1_4 である．MT1_4 はアクセラレータ向けの MT であるが，DRP0 とこの時刻で IDLE となっている CPU0 の間で終了予測時刻の比較を行った結果，CPU0 に割り当てられる．

以上のようにして，RB 内部に MTG を持つような階層的 MTG に対するスケジューリングが行われる．

5. 性能評価

本章では，性能評価の方法およびその結果について述べる．

5.1 評価環境

本評価では，1 チップ上に汎用コア PE またはアクセラレータ PE を合計 8 つまで搭載するシステムを想定した．今回の評価では，アクセラレータコアとして動的再構成可能プロセッサ (DRP)[13] を用いるものとした．本評価における PE 間結合網は 3 本バスとし，CSM の構成は 4 バンク構成とした．各種メモリのアクセスコスト等については表 1 に示すように，自 PE 内部の分散共有メモリおよびローカルメモリへのアクセスに 1 クロック，他 PE のもつ分散共有メモリへのアクセスに 4 クロック，集中共有メモリへのアクセスにチップ外を想定したときに 16 クロック，チップ内を想定したときに 4 クロックかかるものとした．なお，チップの動作周波数は 300MHz を想定している．

本評価では，汎用プロセッサコアを株式会社ルネサステクノロジの SH4A プロセッサ[14] とし，動的再構成可能プロセッサコアとしては株式会社日立製作所の FE-GA[13] を想定している．動的再構成可能プロセッサの扱うことのできる MT に含まれる処理のうち，FE-GA により速度向上が得られる処理に関しては，実際に演算セルアレイに処理をマッピングした結果から求められた実行コストで実行されるものとした．

表 1 各メモリのアクセスコスト

分散共有メモリ (2Port)	1 クロック
分散共有メモリ (Remote)	4 クロック
ローカルメモリ	1 クロック
集中共有メモリ	16 クロック (オフチップ想定時) 4 クロック (オンチップ想定時)

5.2 評価アプリケーション

本評価に用いるプログラムは，UZURA MPEG1 / LayerIII encoder in FORTRAN90[15] を参照実装し，Fortran77 で実装されたプログラムである．参照実装したシーケンシャルプログラムを，第 4 章で述べたスケジューリングアルゴリズムを実装した OSCAR 自動並列化コンパイラ[16] を用いてコンパイルすることで並列性の抽出および粗粒度タスクのスケジューリングを行った．ただし，通常オプションとしてあたえられるパラメータを定数として表記し，フレーム間の並列性[17]，粗粒度並列性の抽出が容易になるよう，あらかじめループのアンローリング等を適用した．また，動的再構成可能プロセッサで実行可能な MT に関しては，プログラムソース上で指示文を用いて指定されている．動的再構成可能プロセッサで実行可能と判断される処理[18] は，サブバンド分析の一部，心理聴覚分析，MDCT 変換，非線型量子化である．MP3 の処理の流れおよびフレーム間並列性が利用可能となるようにリストラクチャリングを施した場合のプログラム構造を図 7 に示す．

図 7(a) に示す通り，MP3 エンコード処理は，サブバンド解析，MDCT，心理聴覚分析，非線形量子化，ハフマン符号化からなり，MDCT は前フレームのサブバンド解析の結果を利用し，心理聴覚分析では，前フレームの心理聴覚分析の結果を用いる．MDCT および心理聴覚分析においてフレーム間でデータの受け渡しを行う箇所以外では，複数のフレームに対する処理を同時に行うことができ，フレーム間の並列性を利用して並列処理を行う手法が効果的である．よって本評価では，図 7(b) に示すような形でプログラム中のエンコード部分を記述した．図 7(b) では，複数のフレームのデータを一度に読み込み，各フレームに対して同時に図 7(a) の各処理を適用する形になっている．なお，本評価では，エンコードを行うメインループにおいて，1 イタレーションあたり 16 フレームに対する処理を並列に行う形のプログラムを用いた．

今回の評価では，エンコーディングの入力データはステレオの PCM データ 32 フレーム，エンコーディングのパラメータはサンプルレート 44.1[KHz]，ビットレート 128[kbps] である．本評価では，入力の PCM データが全て CSM 上に格納されている状態から符号化後のデータが CSM に書き込まれるまでを評価対象とし，初期値計算やエンコード結果確認のための出力部分は評価対象から除外した．また，前半の 16 フレームは無音部分が多くフレーム間のばらつきが大きいため，後半の 16 フレームに対する処理時間を比較した．

5.3 評価結果

MP3 エンコーダを用いた評価結果を図 8 に示す．図 8 において，横軸はプロセッサおよび想定する CSM の構成を表し，縦軸はそれ

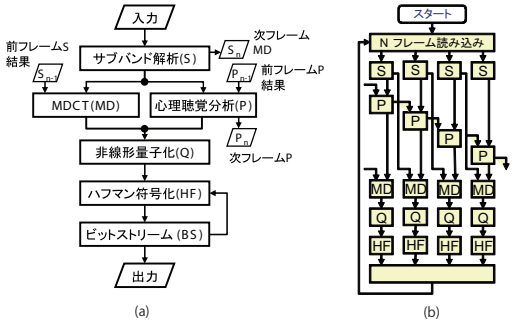


図 7 MP3 エンコードの処理フローとフレーム間並列性利用を考慮したプログラム構造

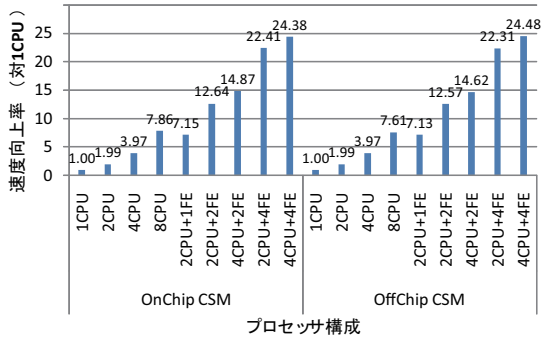


図 8 評価結果

ぞれの CSM 構成において汎用コアのみを用いて逐次処理を行った結果に対する速度向上率を表している．また，図中の nCPU+mFE とは，n 個の汎用コアと m 個のアクセラレータコアを使用しているということを示しており，OnChip CSM は CSM がチップ内にあると仮定した（レイテンシ 4 クロック）場合を，OffChip CSM は CSM がチップ外にあると仮定した（レイテンシ 16 クロック）場合を示している．

図 8 より，CSM がチップ内にあることを想定した場合，1CPU の場合と比較して，2CPU の構成で 1.99 倍，4CPU で 3.97 倍，8CPU で 7.86 倍とスケーラブルな速度向上が得られており，本手法がホモジニアスな構成の場合にも有効であることがわかる．また，アクセラレータもあわせて利用した場合には，2CPU+1FE の構成で同じく 7.15 倍，2CPU+2FE で 12.64 倍，4CPU+2FE で 14.87 倍，2CPU+4FE で 22.41 倍，4CPU+4FE で 24.38 倍の速度向上が得られており，アクセラレータを有効に利用することで，汎用コアのみの処理に比べ大幅な性能向上を得ることができた．また，CSM がチップ外にあることを想定した場合でも，2CPU の構成で 1.99 倍，4CPU で 3.97 倍，8CPU で 7.61 倍，2CPU+1FE で 7.13 倍，2CPU+2FE で 12.57 倍，4CPU+2FE で 14.62 倍，2CPU+4FE で 22.31 倍，4CPU+4FE で 24.48 倍の速度向上を得ることができた．

6. ま と め

本稿では，汎用 CPU に加え，動的再構成可能プロセッサなどのアクセラレータを複数 1 チップ上に集積したヘテロジニアスマルチコアプロセッサ上での粗粒度タスクスタティックスケジューリング手法について述べた．MP3 エンコーダを用いた性能評価では，汎用 CPU として SH4A コアを，アクセラレータとして DRP(FE-GA) を用いた場合，SH4A 1 コアを用いた逐次処理に対して，集中共有メモリがチップ内にあることを想定した場合に SH4A 4 コアの構成で 3.97 倍，SH4A 2 コアと DRP 2 コアの構成で 12.64 倍，SH4A 4 コアと DRP 4 コアの構成で 24.38 倍の性能向上を得られることが確かめられた．また，集中共有メモリがチップ外にあることを想定した場合に SH4A 4 コアの構成で 3.97 倍，SH4A 2 コアと DRP 2 コアの構成で 12.57 倍，SH4A 4 コアと DRP 4 コアの構成で 24.48 倍の性能向上を得られることが確かめられた．

謝辞

本研究の一部は NEDO “先進ヘテロジニアスマルチプロセッサ研究開発” の支援により行われた．本研究の遂行にあたり有益な御議論をいただいた，株式会社日立製作所の小高俊彦フェロー，内山邦男氏，伊藤雅樹氏，佐藤真琴氏に感謝致します．

参 考 文 献

1) ARM: *ARM11 MPCore Processor Technical Reference Manual* (2005).
2) Suga, A. and Matsunami, K.: Introducing the FR 500 embedded microprocessor, *IEEE MICRO*, Vol. 20, pp. 21–27 (2000).
3) Pham, D., Asano, S. and et al., M. B.: The Design and Implementation of a First-Generation CELL Processor (2005).
4) Torii, S., Suzuki, S., Tomonaga, H., Tokue, T., Sakai, J., Suzuki, N., Murakami, K., Hiraga, T., Shigemoto, K., Tatebe, Y., Obuchi, E., Kayama, N., Edahiro, M., Kusano, T. and Nishi, N.: A 600MIPS 120mW 70 μ A Leakage Triple-CPU Mobile Application Processor Chip, *ISSCC* (2005).
5) Kimura, K., Kodaka, T., Obata, M. and Kasahara, H.: Multigrain Parallel Processing on OSCAR CMP, *Proc. of International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'03)* (2003).
6) 笠原, 成田, 橋本: OSCAR(Optimally Scheduled Advanced Multiprocessor) のアーキテクチャ, 電子情報通信学会論文誌 D 分冊, Vol. J71-D, No. 8 (1988).
7) 木村, 尾形, 岡本, 笠原: シングルチップマルチプロセッサ上での近細粒度並列, 情報処理学会論文誌, Vol. 40, No. 5 (1999).
8) Kimura, K., Wada, Y., Nakano, H., Kodaka, T., Shirako, J., Ishizaka, K. and Kasahara, H.: Multigrain Parallel Processing on Compiler Cooperative Chip Multiprocessor, *Proc. of 9th Workshop on Interaction between Compilers and Computer Architectures (INTERACT-9)* (2005).
9) 本多, 岩田, 笠原: Fortran プログラム粗粒度タスク間の並列性検出法, 信学論 (D-I), Vol. J73-D-I, No. 12, pp. 951–960 (1990).
10) 笠原, 合田, 吉田, 岡本, 本多: Fortran マクロデータフロー処理のマクロタスク生成手法, 信学論, Vol. J75-D-I, No. 8, pp. 511–525 (1992).
11) Obata, M., Shirako, J., Kaminaga, H., Ishizaka, K. and Kasahara, H.: Hierarchical Parallelism Control for Multigrain Parallel Processing, *Proc. of 15th International Workshop on Languages and Compilers for Parallel Computing (LCPC2002)* (2002).
12) 宮本, 中川, 浅野, 内藤, 仁藤, 中野, 木村, 笠原: マルチコアプロセッサ上での粗粒度タスク並列処理におけるデータ転送オーバラップ方式, 情報処理学会研究報告 ARC, Vol. 167, No. 10 (2006).
13) 津野田, 高田, 秋田, 田中, 佐藤, 伊藤: ディジタルメディア向け再構成型プロセッサ FE-GA の概要, 信学技報 RECONF2005-65 (2005).
14) Yoshida, Y., Kamei, T., Hayase, K., Shibahara, S., Nishii, O., Hattori, T., Hasegawa, A., Takada, M., Irie, N., Uchiyama, K., Odaka, T., Takada, K., Kimura, K. and Kasahara, H.: A 4320MIPS Four-Processor Core SMP/AMP with Individually Managed Clock Frequency for Low Power Consumption (2007).
15) UZURA3: MPEG1/LayerIII encoder in FORTRAN90, `http://members.at.infoseek.co.jp/kitaurawa/index_e.html`.
16) 岡本, 合田, 宮沢, 本多, 笠原: OSCAR マルチグレインコンパイラにおける階層型マクロデータフロー処理, 情報処理学会論文誌, Vol. 35, No. 4, pp. 513–521 (1994).
17) 鹿野, 鈴木, 和田, 白子, 木村, 笠原: MP3 エンコーダを用いた OSCAR ヘテロジニアスチップマルチプロセッサの性能評価, 情報処理学会論文誌コンピューティングシステム, Vol. 48, No. SIG8 (2007).
18) 田中, 津野田, 秋田, 高田, 伊藤, 佐藤: 再構成型プロセッサ FE-GA のオーディオ処理への応用, 信学技報 RECONF2005-67 (2005).