2006－ARC－166（3）
2006／1／24

ヘテロジニアスチップマルチプロセッサにおける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[†]　押山直人[†]　鈴木裕貴[†]
白子準[†]　中野啓史[†]　鹿野裕明[††]
木村啓二[†,††]　笠原博徳[†,††]

本論文では，汎用プロセッサに加え，DRP(Dynamically Recofigurable Processor) や DSP(Digital Signal Processor) などのアクセラレータを 1 チップ上に複数集積したヘテロジニアスチップマルチプロセッサ上で，アクセラレータの利用による高い実効性能と動作周波数・電圧の制御による低消費電力化を実現する，各コアの特性を考慮した粗粒度タスクスタティックスケジューリング手法を提案する．本手法は，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，チップ上の各コアの種類や実行可能なタスクおよびコストを考慮した処理時間最小を目指したスタティックスケジューリング手法であり，その性能を MP3 エンコーダに適用し評価した．今回の評価では，汎用プロセッサ 4 コアとアクセラレータとして DRP2 コアを搭載し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評価を行った結果，本手法を適用せず汎用プロセッサ 1 コアのみを用いて逐次実行した場合に対して，最大 8.8 倍の速度向上が得られることが確認できた．

A Static Scheduling Scheme for Coarse Grain Tasks on a Heterogeneous Chip Multi Processor

Yasutaka Wada,[†] Naoto Oshiyama,[†] Yuki Suzuki,[†]
Jun Shirako,[†] Hirofumi Nakano,[†] Hiroaki Shikano,[††]
Keiji Kimura[†,††] and Hironori Kasahara [†,††]

This paper proposes a static scheduling scheme for coarse grain tasks on a heterogeneous chip multi processor which integrates not only general purpose processors but also accelerators like DRP or DSP. A heterogeneous chip multi processor allows us to get high performance by using the accelerators and to save energy by frequency/voltage control by the compiler. In this scheme, the compiler aim to minimize the execution time of an application in consideration of the characteristic in each core. Performance of the proposed scheme is evaluated on a heterogeneous chip multi processor which has 4 general purpose processors and 2 accelerators using MP3 encoder and gives us 8.8 times speedup against sequencial execution without the poroposed scheme.

1. はじめに

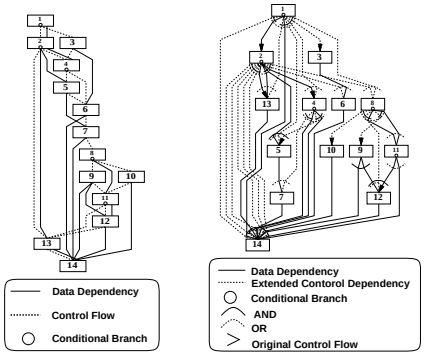
近年，携帯電話やゲーム機，DVD レコーダ等情報家電の市場が拡大しており，それに従って組み込みシステムに対する要求，特に価格性能比の向上，ソフトウェア/ハードウェア開発期間の短縮，低消費電力化といった事柄に対する要求が高まっている．これに対する解決策のひとつとして，1 つのチップ上に複数のプロセッサを集積するチップマルチプロセッサ (CMP) アーキテクチャが組み込みシステムにおいても導入され始めており，MPCore[1] や FR-V[2]，Cell[3] などが開発されている．

さらに，メディアアプリケーションにおける特定の処理を加速するために，DRP(Dynamically Recofigurable Processor) を含め各種アクセラレータを集積した MP211[4] や UniPhier[5] のようなヘテロジニアスチップマルチプロセッサも開発されている．このようなシステムでは，コアの特性を考慮してプログラム中の各処理を適切なコアにデータ転送オーバーヘッドを考慮して割り当てるチューニングが非常に難しく，自動並列化コンパイラの開発が望まれる．

本論文では，粗粒度タスク並列化，ループ並列化，近細粒

† 早稲田大学理工学部コンピュータ・ネットワーク工学科
Department of Computer Science,
School of Science and Engineering,
Waseda University

†† 早稲田大学アドバンストチップマルチプロセッサ研究所
Advanced Chip Multi Processor Res. Inst.,
Waseda University



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

図 1 マクロフローグラフ (MFG)，マクロタスクグラフ (MTG) の例

度並列化を階層的に利用するマルチグレイン並列処理の主要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 CMP 上の各コアの特性を考慮してタスクの各コアへの割り当てを行うスタティックスケジューリング手法を述べる．

以下，2 章で粗粒度タスク並列処理について，3 章でヘテロジニアスチップマルチプロセッサアーキテクチャについて，4 章でヘテロジニアスチップマルチプロセッサを対象としたスケジューリングアルゴリズムについて，5 章でデータ転送最小化について，6 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．

2. 粗粒度タスク並列処理

粗粒度タスク並列処理では，コンパイラはまず対象とするプログラムを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)) に分割する．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，マクロフローグラフ (MFG) を作成し，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マクロタスクグラフ (MTG) を生成する[6)7)]．MFG および MTG の例を図 1 に示す．その後，コンパイラは各 MT を 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)．また 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)．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．

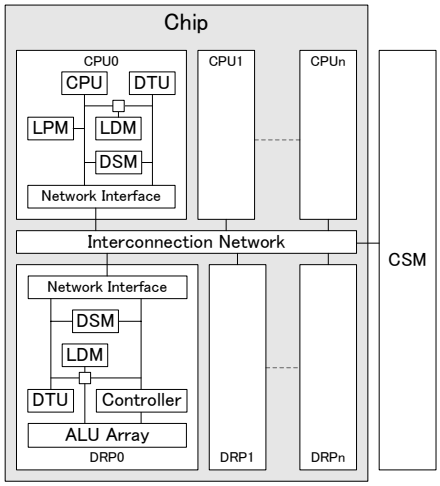


図 2 OSCAR 型メモリアーキテクチャを持つヘテロジニアスチップマルチプロセッサの例

3. ヘテロジニアスチップマルチプロセッサアーキテクチャ

本章では，汎用コアに加えて DRP や DSP などのアクセラレータを 1 チップ上に集積した，ヘテロジニアスチップマルチプロセッサアーキテクチャについて述べる．

3.1 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[8)9)10)] を持つヘテロジニアスチップマルチプロセッサは，汎用 CPU を含む汎用コアおよび DRP や DSP などのアクセラレータを含むアクセラレータコアを 1 チップ上に複数集積し，これらと集中共有メモリ (CSM) を複数バスやクロスバ等の結合網で接続したものである．

3.2 PE の構成

各 PE は，プロセッサコアまたは DRP や DSP などの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する．また，汎用コア PE は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) を，アクセラレータ PE は DRP 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などの処理を行うためのコントローラを持つ．

このアーキテクチャの例を図 2 に示す．

4. ヘテロジニアスチップマルチプロセッサを対象としたスケジューリングアルゴリズム

本章では，第 3 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．

ヘテロジニアスチップマルチプロセッサでは，汎用コアでは全ての種類の MT が実行可能であるが，アクセラレータコア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，他の MT の割り当て状況や実行状況，あるいは各 PE の動作周波数/ 電圧制御の状況によっては，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効率を下げてしまう場合が考えられる．本アルゴリズムは，このように各 PE で実行可能なタスクに制限があるような条件下において，各種アクセラレータコアと汎用コアを効率よく利用し，プログラムの実効効率の向上を図るものである．

4.1 実行可能タスクおよび処理コストを考慮した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 PG または同一機能のアクセラレータコアの集合とする．また，汎用コア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．これは，汎用コアに比べてアクセラレータコアの数は一般的に少ないこと，アクセラレータコアでの MT の実行コストが小さくなるため，あるひとつの階層の MT のみを割り当てていたのでは利用効率が下がってしまうためである．逆に，各階層から適宜アクセラレータコアを利用できるようにしておくことで，各階層にどのようにアクセラレータで実行可能な MT が分布しているかによらず，十分にアクセラレータを利用することができる．

以下に，ヘテロジニアスチップマルチプロセッサにおける MT のスタティックスケジューリングアルゴリズムを示す．

step 1: 各 MTG 上の MT に対し，汎用コア PG あるいは当該タスクを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および階層 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の設定を行い，スケジューリング時刻を 0 にセットする

step 2: 最早実行可能条件を満たした 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 3: 現スケジューリング時刻において IDLE になっている汎用コア PG またはアクセラレータコアで実行可能なレディタスクのうち，最も優先度の高い MT を割り当て対象として選択する

step 4: 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 の終了推定時刻
+ ローカルメモリへのデータ転送コスト
+ (必要ならば) アクセラレータの起動コスト
+ 対象 PG/コア上での MT 実行コスト
+ 共有メモリへのデータ転送コスト

step 5: 終了推定時刻が最も早い汎用コア PG/アクセラレータコアに対象 MT を割り当てる

step 6: 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する

step 7: 全ての MT が割り当て済みであれば終了，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/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して step 2 へ

図 3 にスケジューリングの例を示す．なお，図 3 は 2 つの汎用コア，1 つのアクセラレータコア (DRP) を持つ CMP に対するスケジューリング例である．

図 3 において，階層的に MTG が定義されている．最上位の階層である MTG1 では，粗粒度並列性が 2 程度であると判断され，それに従って汎用コアはふたつの PG にグルーピングされる．この階層においては，MT1_1 および MT1_2 が PG0 に，MT1_3 が PG1 に割り当てられる．次に，第 2 階層である MTG1_2 および MTG1_3 に注目すると，これらの階層にはアクセラレータコアである DRP で実行可能な MT が含まれる．MT1_2 および MT1_3 の開始時点において実行可能な MT は MT1_2_1 と MT1_3_1 であるが，これらは両方とも汎用コアでのみ実行可能であるので，それぞれ PG0，PG1 に割り当てられる．次にレディタスクが発生するのは MT1_2_1 終了時であり，この時点でのレディタスクは MT1_2_2 と MT1_2_3 である．これらはともに DRP 上で実行可能であるので，PG0 上での終了推定時刻と DRP 上でのそれを比較して，終了推定時刻が早い方に割り当てられる．ここでは MT1_2_2 の方が割り当て優先度が高いとすると，MT1_2_2 は DRP 上でのコストの方が小さいためそのまま DRP に割り当てられる．MT1_2_3 に関しては，PG0 で実行した場合と MT1_2_2 の後に DRP 上で実行した場合の終了推定時刻を比較し，DRP 上のそれが早いと判断され，MT1_2_2 の後に続けて割り当てられる．次にレディタスクが発生するのは MT1_3_1 終了時点であり，レディタスクは MT1_3_2，MT1_3_3，MT1_3_4 である．このうち優先度が最も高いのは MT1_3_4 であるが，これは汎用コアでのみ実行可能なので PG1 に割り当てられる．MT1_3_2 および MT1_3_3 に関しては，MT1_3_4 の後に PG1 で実行するよりも DRP で実行した方が終了推定時刻が早いと判断され，両方とも DRP に割り当てられる．次にレディタスクが発生

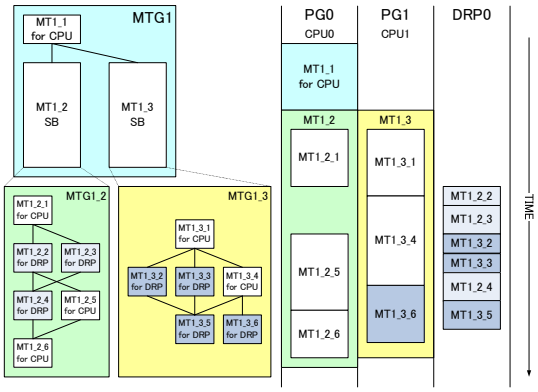


図 3 スケジューリングの例

するのは DRP に割り当てられた MT1_2_3 が終了した時点であり，レディタスクは MT1_2_4 および MT1_2_5 である．このうち優先度が高いと判断された MT1_2_5 は汎用コア上でのみ実行可能なので，PG0 に割り当てられる．MT1_2_4 は MT1_2_5 の後に PG0 で実行するよりも現在割り当て済みの MT の後に DRP で実行した方が終了推定時刻が早いと判断され，DRP 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_4 終了時である．このときのレディタスクは MT1_3_5 と MT1_3_6 であり，両者とも DRP 上で実行可能である．このうち優先度の高い MT1_3_5 は PG1 上での終了推定時刻と DRP でのそれを比較し，DRP に割り当てられるが，MT1_3_6 については，逆に PG1 上で実行した方が終了推定時刻が早いと判断され，PG1 に割り当てられる．最後に，MT1_2_5 終了時にレディタスクとなる MT1_2_6 が PG0 に割り当てられる．

以上の様にして，汎用コアとアクセラレータコアの使用状況を考慮して最も早くタスク実行が終了するコアへの割り当てが行われる．

5. データ転送の最小化

DRP などのアクセラレータを持つ PE においては，MT の処理に必要なデータがコアに近接したローカルメモリ上に配置されている必要がある場合がある．また，汎用コアにおいても，高速な LDM 上にデータを配置して処理を行うことで MT の実効効率を向上させることができる．ここでは，演算に使用されるデータをローカルメモリサイズに合うように分割し，高速なメモリ上に転送して処理を行い，ローカルメモリを介して同一の PE に割り当てられた MT 間のデータ授受を実現するデータローカライゼーション手法[11] によりローカリティ最適化が行われる．

コンパイラは，コンパイル時に各 MT が定義/使用する配列の範囲情報および生死情報の解析を行う．第 4 章で述べたスケジューリングアルゴリズムを適用して各 PG へのタスク割り当てを決定した後，必要に応じて MT の開始前および終了直後に DTU を用いたデータ転送[12] が挿入される．

本手法では，ある配列のデータをローカルメモリ上に転送した後は，同じ PG に割り当てられた MT 間でできうる限りローカルメモリ上でデータ授受を行うことを考える．そのため，ある PG で初めて配列が使用される際に共有メモリからローカルメモリに転送し，一度ローカルメモリ上に配置した後は必要になった時点で再度転送を行うようにする．ある MT の開始時に，先行タスクまでですでにローカルメモリ上に配置されている配列を再度ローカルメモリ上に転送する条件は以下のようになる．

- データ依存があり，かつ他の PE に割り当てられている MT がその配列データを定義した場合
- 対象 MT において定義され，その後共有メモリに書き戻す必要がある配列で，その際に共有メモリ上のデータとの整合性の保証が必要な場合

ただし，今回の評価においては，ローカルメモリの容量を考慮したメモリ領域管理手法[12] やプレロード/ポストストア手法[13] は適用していない．共有メモリへの書き戻しについては，配列のデータを定義した MT の末尾で速やかに共有メモリに書き戻すこととする．

図 4 にデータ転送を最小化した例を示す．なお，図 4 中で，LOAD は共有メモリからローカルメモリへのデータ転送を，STORE はローカルメモリから共有メモリへの転送を表す．また，簡単のため，図中では配列の範囲に関しては考慮していない．図 4 において，PG0 では MT_k の実行前に配列 A を共有メモリからローカルメモリに転送する．MT_k の実行後，定義した配列 B を共有メモリに書き戻すとともに，後続の MT_{k+1} で使用される配列 A および B はローカルメモリ上に配置されたまま MT_{k+1} でも使用される．最後に，MT_{k+1} で定義された配列 A が共有メモリ上に書き戻される．それに対して PG1 では，MT_{k+2} において，PG0 に割り当てられた MT_k で定義された配列 B を使用するため，MT_k による CSM への転送の終了を待ってからローカルメモリに転送を行う．MT_{k+2} では配列 C を定義するため，MT 終了後に配列 C を共有メモリに書き戻す．これら配列 B および C は後続の MT_{k+3} でも使用されるため，ローカルメモリ上に配置したまま保持される．次に後続の MT_{k+3} では，すでにローカルメモリ上に配置されている配列 B，C の他に，PG0 に割り当てられた MT_{k+1} によって定義される配列 A を使用するので，MT_{k+1} による共有メモリへの転送の終了後ローカルメモリへの転送を行う．

以上のようにして，同一 PE に割り当てられた MT 間でローカルメモリを介したデータの授受が行われる．

6. 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．

6.1 評価環境

本評価では，1 チップ上に汎用コアを 4 つまで，アクセラレータコアを 2 つまで搭載するシステムを想定する．今回の評価では，アクセラレータコアのハード構成は汎用コアと同様であるが，動作周波数を汎用コアの 8 倍に設定することにより擬似的に DRP コアによる実行をシミュレーショ

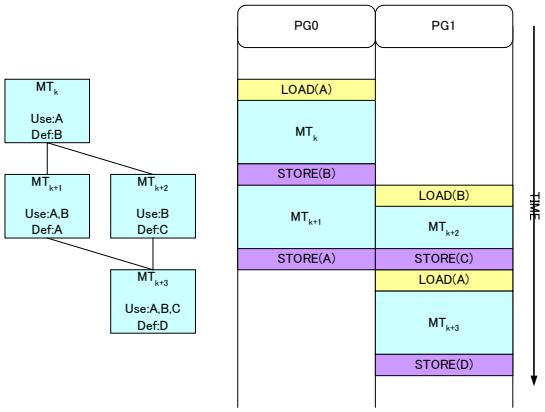


図 4 データ転送最小化の例

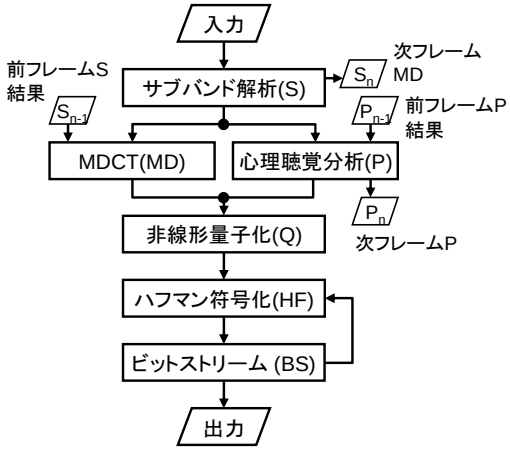


図 5 MP3 の処理フロー

ンした．またこれらのアクセラレータコアでは，DRP が扱える演算のみからなる MT だけが実行可能であるものとした．本評価における PE 間結合網は 3 本バス，CSM の構成は 4 バンク構成とした．PE 内の各種メモリのアクセスコスト，PE 間結合網の遅延等については表 1 に示すとおりである．ただし，表 1 において，分散共有メモリおよびローカルメモリは各 PE の動作周波数を基準とし，集中共有メモリおよびネットワーク遅延については汎用コアの動作周波数を基準としたアクセスコストとなっている．

本評価では，各 PE の持つプロセッサコアは，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，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．

評価には上記の構成を持つ CMP を精密に再現するシミュレータを用いた．なお，第 5 章で述べたデータ転送を最小化する手法は LDM のサイズを考慮したメモリ領域の管理を含まないため，各 PE の LDM のサイズに関しては十分に大きいものとして評価を行った．

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
ローカルメモリ	1 クロック
集中共有メモリ	100 クロック
ネットワーク遅延 (調停を含む)	3 クロック

6.2 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[14] 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および第 5 章で述べたデータ転送最小化手法を実装した OSCAR 自動並列化コンパイラ[15] 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たり，粗粒度並列性の抽出が容易になるよう，あらかじめサブルーチンのインライン展開やループのアローリングを適用した．また，DRP 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．DRP で実行可能と判断される処理[16] は，サブバンド分析の一部，心理聴覚分析の一部，MDCT 変換，非線型量子化の一部である．MP3 の処理の流れを図 5 に示す．

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 4 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．また，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エンコード後のデータが CSM に書き込まれるまでを評価対象とした．

6.3 評価結果

MP3 エンコーダを用いた評価結果を図 6 に示す．図 6 において，横軸は CMP の構成および提案手法の適用の有無を表し，縦軸は汎用コアのみを用い，本手法 (スケジューリング手法およびデータ転送最小化) を適用せずに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+mDRP 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いる．図 6 より，本手法を適用しない場合の逐次実行と比較して，最大 8.8 倍の速度向上を得ることができている．

次に，汎用コア 2 基・アクセラレータコア 2 基を用いたときの実行トレースを図 7 に示す．図 7 より，処理の前半部分は効率よくアクセラレータコアが利用されているのに対して，処理の後半部分ではまったく処理が割り当てられていないことがわかる．これは，後半の処理のほとんどを占めている非線型量子化内の階層には条件分岐が存在するためスタティックスケジューリングが適用できず，アクセラレータコ

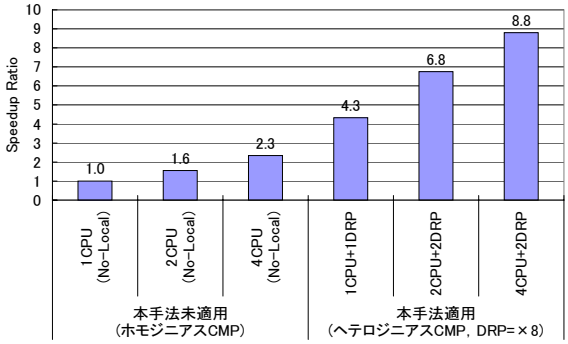


図 6 評価結果

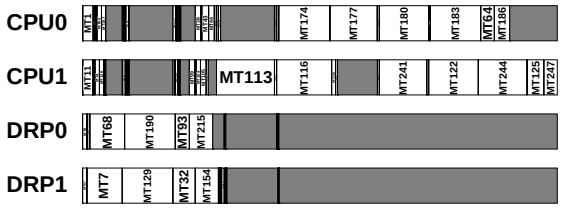


図 7 2CPU+2DRP 時の実行トレース

アへのタスク割り当てが行われなかったためである．この階層内の MT を適切にアクセラレータコアに割り当てることができれば，さらなる性能向上を得ることが可能である．

7. ま と め

本論文では，汎用 CPU に加え，DRP 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チップマルチ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，提案手法 (スケジューリング手法およびデータ転送最小化) を適用せず汎用 CPU 上で逐次処理を行った場合に対して，汎用コア 4 基と DRP に相当するアクセラレータ 2 基を集積したヘテロジニアスチップマルチプロセッサ上で最大 8.8 倍の性能向上を得られることが確かめられた．今後の課題としては，条件分岐等のためスタティックスケジューリングの対象にならない階層へのダイナミックスケジューリングの適用，低消費電力化への対応などが挙げられる．

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．

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) 木村, 藤井, 西道, 清原: デジタル家電統合プラットフォーム UniPhier におけるメディアプロセッサ, DA シンポジウム (2005).
6) 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).
7) 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).
8) 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).
9) 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).
10) 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).
11) 吉田, 前田, 尾形, 笠原: Fortran マクロデータフロー処理における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. 35, No. 9, pp. 1848–1860 (1994).
12) 中野, 内藤, 鈴木, 小高, 石坂, 木村, 笠原: 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, 情報処理学会研究報告 ARC, Vol. 159, No. 20 (2004).
13) 田中, 舟山, 飛田, 笠原: メモリ容量を考慮したプレロード・ポストストアスケジューリングアルゴリズムの評価, 情報処理学会第 62 回全国大会, No. 4R-03 (2001).
14) UZURA3: MPEG1/LayerIII encoder in FORTRAN90, `http://members.at.infoseek.co.jp/kitaurawa/index_e.html`.
15) 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).
16) 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).