社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ARC－160　(11)
2004／12／2

組込みチップマルチプロセッサ M32R32700 への OpenMP 処理系の実装と評価

堀田義彦† 佐藤三久†
中島佳宏† 小島好紀†

CMP(Chip Multiprocessor) は高性能化だけでなく，消費電力やエネルギー削減という観点において重要なアーキテクチャーである．CMP を有効に活用するためには，並列プログラミングが不可欠であるが，我々は，組込み向けプロセッサ M32R チップマルチプロセッサに並列プログラミング環境として OpenMP コンパイラを実装した．本論文では，M32R CMP における科学技術計算やマルチメディアアプリケーションでの OpenMP の性能の測定結果について報告する．OpenMP を用いることによってわずかな指示文の挿入で CMP においてもユーザーが適切な性能向上を実現できることがわかった．また，CMP における OpenMP のスレッドランタイムスケジューリングによる低消費電力化の可能性についても議論する．

OpenMP implementation and performance on Embedded M32R32700 Chip-Multi processor

Yoshihiko Hotta,† Mitsuhisa Sato,† Yoshihiro Nakajima†
and Yoshinori Ojima†

CMP (Chip Multiprocessor) is a promising processor architecture, not only for high performance but also for reducing power and energy consumption in embedded applications. We have implemented an OpenMP compiler for an embedded Renesas M32R chip multiprocessor as a parallel programming environment. In this paper, we report the preliminary performance of OpenMP benchmarks, including scientific and multimedia applications on the M32R CMP. We found that OpenMP allows users to easily obtain reasonable performance improvement using multiple CPUs in the CMP with just a few directives inserted. Also, we discuss the possibility of OpenMP thread run-time scheduling and some compilation techniques for power-aware computing on the CMP.

1. はじめに

近年，マイクロプロセッサの周波数の向上による高性能化にによって，消費電力・発熱量が増大し，実装・冷却において大きな問題となっている．その一方で，ムーアの法則を維持しながら進歩する微細化による集積度の向上によって，これまでのプロセッサの面積に余剰なトランジスタが生じている．その豊富なトランジスタを有効利用する手段として CMP(チップマルチプロセッサ) が注目されている．CMP は 1 チップ上にプロセッサコアを複数搭載しているプロセッサであり，低電力化と高性能化の両方が期待されている[1]．

一般にプロセッサの消費電力は，消費電力を P，C をコンデンサの静電容量，電圧を V，クロック周波数を f とすると

$$P = \alpha C V^2 f \qquad (1)$$

で示される．クロック周波数は電圧と関係があり，高周波数にするためには電圧を下げる必要がある．微細化によって C を下げることにより消費電力は削減できるが，周波数の上昇とそれにともなう電圧によってプロセッサはより消費電力が高くなっている．

一方で CMP の消費電力を非常に単純な式で示すと

$$P = n * \alpha C V^2 f \qquad (2)$$

となる．CMP では，プロセッサコアが n 個搭載されているため消費電力は n 倍となるが，クロック周波数を低くすることで電圧も下げ，低消費電力化を図っている．また，複数のタスクをそれぞれのプロセッサで並列処理することによって，低いクロック周波数を維持しながら高性能化を実現できる．

最新の情報によれば，組込み向けプロセッサメーカーである ARM は 1 チップに 4 つの CPU コアを搭載した MPCore を発表している[2]．また，インテルなど高性能プロセッサ製造メーカも上昇し続けるクロック周波数と消費電力の問題に対処するべくロードマップを変更し，プロセッサの CMP 化が予定されている．大量の投機実行のサポートを行うのと同様に CPU に

† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba

対して幅広い実行命令を備えている一方で，単純なプロセッサで構成されている CMP は効率的にスレッドレベル並列性を実現可能なアーキテクチャになる可能性がある．

このような CMP を使用するには，並列 (もしくはマルチスレッド) プログラミングが必要となる．CMP は共有メモリ型マルチプロセッサであり，そのプログラミング環境として OpenMP を用いることが考えられる．OpenMP は CMP 内の複数の CPU を使用する際，使い易く，プログラマの負担を軽減することができ，容易に性能向上を行うことができる．

本論文では，組込み向けシステムを対象として開発された Renesas 社製 M32R CMP での OpenMP による並列プログラミング環境の構築といくつかのアプリケーション (科学技術計算，マルチメディアアプリケーション) での性能評価について報告する．今回使用した M32R が搭載されている M3T-32700UT は M32R CMP を使用した組込みアプリケーション開発のために設計されたプラットホームである．

近年，プロセッサの消費電力に着目した Power-Aware Computing は PDA や携帯電話など組込みシステムにおいて活発に研究されている分野である．高性能並列システムにおいても，消費電力の削減や実装密度の向上は非常に重要である．我々の以前の研究において，低消費電力プロセッサを用いたクラスタは高性能プロセッサを用いたシステムよりも低消費電力かつ高性能を実現できることがわかった．CMP は低消費電力化と高性能化を同時に実現できる可能性のあるアーキテクチャであり，これを有効に利用することによって低消費電力かつ高密度・高性能なシステムの構築ができることを期待できる．また，組込みシステムにおいても CMP は非常に有効である．我々は組込み向けシステムで重要な，動作時間，発熱，リアルタイムアプリケーションでの必要な性能を十分に満たすことが実現できることも期待している．

Power-Aware Computing において OpenMP の重要な特徴は，OpenMP プログラムの記述が，実行するプロセッサ数に依存しないことである．システムが状況に応じて使用するプロセッサ数の変更が可能である．このことを利用し，OpenMP を使用する CMP ではアプリケーションの性能と消費電力との間でのトレードオフをすることが考えられる．例えば，システムが消費電力の削減を必要とする場合，いくつかのプロセッサでの実行をストップすることで，性能は落ちるが，低消費電力で実行することが可能になる．

我々が考える組込み CMP における OpenMP の役割を以下にまとめる．

–ポータブルな並列プログラミング環境の提供

OpenMP は組込み向けアプリケーションの使用のために共有メモリマシンへ標準的な C/C++や Fortran の API を提供している．

–使い易い並列プログラミング環境の提供

オリジナルの逐次プログラムは OpenMP ディレクティブの挿入によって容易に並列化が可能である．現在，POSIX スレッドライブラリが組込みアプリケーションの開発に広く使われている．OpenMP スタイルプログラミングモデルはスレッドを管理し同期させるためのコードを排除し，開発にかかるコストの削減が可能である．

–マルチメディアアプリケーションの並列化

マルチメディアアプリケーションは多くの並列性を保持していることが多く，並列化によって高速化が可能である．

–柔軟なランタイムスケジューリング

精巧なランタイムスレッドスケジューリングは CMP 内の CPU の状態をコントロールすることによってアプリケーションの性能と消費電力のトレードオフを可能にする．

–OpenMP ディレクティブの拡張の可能性

OpenMP ディレクティブを拡張することによって，組込み向けシステムに新しい機能を入れる可能性がある．

本論文の構成は以下のようになっている．2 章では，今回使用したシステム M3T-32700UT ならびに CMP である M32R32700 の概略を示す．3 章では，構築した OpenMP クロス環境の説明と EPCC による OpenMP プリミティブの基本性能を示す．4 章では，ベンチマークでの結果を示す．5 章では，M3T-32700UT の消費電力，組込みシステムでの OpenMP による低消費電力化手法を提案する．最後に 6 章で，まとめと今後について説明する．

2. M3T-32700UT

今回，我々がターゲットとしたシステム M3T-32700UT は CMP である M32R プロセッサや，ネットワーク，LCD などを搭載した開発プラットホームである．Renesas 社が開発した M32R32700 プロセッサ[3] は組込みシステム向けとして低消費電力化を実現した CMP である．図 2 にこのチップのブロック図を示す．M32R32700 シングルチップマルチプロセッサは 2 つの 32 ビット M32RCPU コアを搭載し，512kB の共有 SRAM を備え，組込み向けマイクロコントローラや SoC コアとしてデザインされている．このチップは 0.15μmCMOS プロセスで製造されている．2 つの CPU コアと共有 SRAM は 128 ビットバスでつながれている．パイプラインは 7 ステージであり，CPU コアは 2 ウェイセットアソシエイティブキャッシュメモリを備え，メインメモリは 32 ビットのフルアソシエイティブである．組込みシステムで様々なアプリケーションを実行する必要性が今後ますます大きくなることが予想され，性能の向上と同時に低消費電力化が必

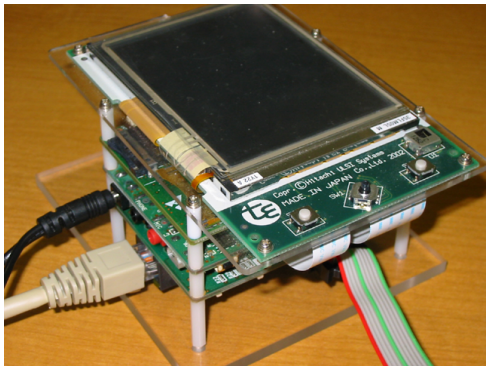

図 1 M3T-M32700UT の写真

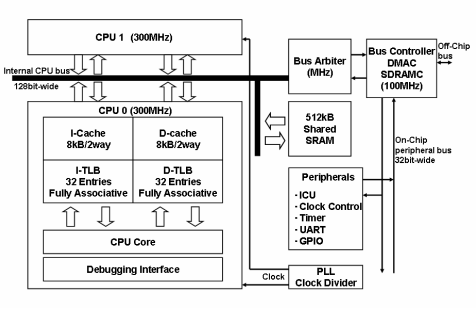

図 2 Renesas M32R32700 CMP のブロックダイアグラム

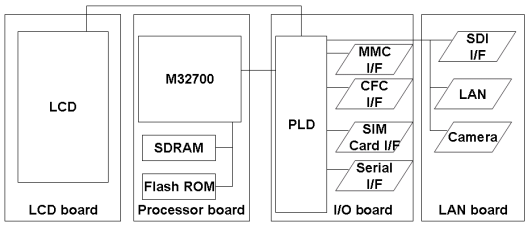

図 3 M3T-M32700UT のブロックダイアグラム

要とされている．またそれと併せて，バッテリー駆動で動くシステムにおいて低消費電力化は大きなキーポイントとなっている．消費電力は 800mW(最新の M32R：クロック周波数 400MHz，バスクロック周波数 100MHz の場合)．

図 1 に M3T-32700UT の写真を示す．表 1 に M32R32700UT の仕様を示す．このシステムはアプリケーション開発のために LCD，ネットワークインタフェース，カメラデバイスなどを備えている．オペレーティングシステムとしては，M32R/Linux が移植されている[4]．Linux/M32R は 2.4 と 2.6 両方をサポートしている．いくつかのデバイスドライバ (LAN, CF，USB，MTD など) はすでに移植されている．このシステムは組込み向けアプリケーションの開発のためにデザインされており，我々はこのシステムを用いて OpenMP でのアプリケーションの性能測定を行った．

3. OpenMP の実装

3.1 OpenMP の実装と環境

我々は M32R への OpenMP の実装に，Omni OpenMP[5] コンパイラを使用した．Omni OpenMP コンパイラは OpenMP プログラムを元にランタイムライブラリ呼出を含むマルチスレッドの C プログラムへ変換を行う．OpenMP 環境の構築にあたり，M32R32700 用のクロス環境を構築した．クロス環境

feature	spec
CPU	Renesas M32R CMP (M32700)
clock	300MHz (in this paper at 200MHz)
cache I/D	8KB/8KB 2way
Memory	16MB SDRAM / FLASH:4MB
LAN	100Base-TX
USB	Host2.0
module	CF/MMC/eTRON slot, LCD, AR camera
size	60mm * 60mm

表 1 M3T32700UT の仕様

を使用して，M32R32700 用の OpenMP ランタイムライブラリを作成した．図 4 に構築したクロス環境を示す．生成されたプログラムはクロスコンパイラでランタイムライブラリと共にコンパイルされ，M32R で実行可能なファイルが生成される．

3.2 基本性能

表 2 に EPCC マイクロベンチマークの synchbench を用いた OpenMP プリミティブのオーバーヘッド時間の測定結果を示す．測定にあたり，ロック機構の比較を行うために，pthread_mutex_lock の場合と，M32R32700 用にアセンブルコードを作成した spin_lock での場合との比較を行った．比較のために，Pentium ベース (Pentium III 550MHz) の SMP プラットフォームでの測定結果も示す．この結果より，M32R CMP と Pentium III では性能に大きな差があることがわかる．特に，“CRITICAL” と”LOCK/UNLOCK” においては性能差が数百倍となっている．その一方で，FOR や BARRIER では，性能差は小さくなっている．しかしながら全ての場合において，性能差は両者のクロック周波数のに比べても大きくなっている．この結果は高性能プロセッサを用いる場合と比べて，M32R で OpenMP を用いる場合は同期により大きなオーバーヘッドが生じることを示している．

lock オペレーションに pthread_mutex_lock を用いていた場合はオーバーヘッドは大きい．その解決のた

processor (2 proc.)	PenIII(mutex_lock)	M32R(mutex_lock)	M32R spin_lock
PARALLEL	4.7	843.4	828.9
FOR	1.3	17.7	21.9
PARALLEL FOR	5.2	857.8	830.7
BARRIER	0.9	13.0	18.7
SINGLE	1.4	58.9	7.4
CRITICAL	0.7	313.8	4.9
LOCK/UNLOCK	0.7	313.5	3.9
ORDERED	0.4	10.3	5.7
ATOMIC	0.9	295.4	6.1
REDUCTION	6.8	855.7	845.3

表 2 EPCC マイクロベンチマークの結果 [micro sec]

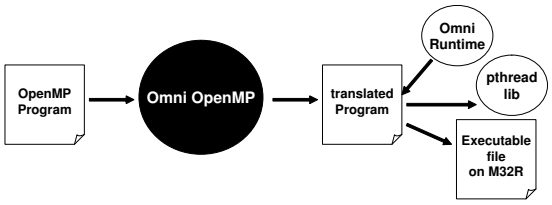

図 4 クロス環境のオーバービュー

めに M32R 用の spin-lock を実装し，性能の向上を計った．この結果のように，“LOCK/UNLOCK”，”CRITICAL” において大幅な性能向上が得られた．ただし，他のプロセスと競合する時には一般的に pthread_mutex_lock が望ましい．

4. 評　　価

性能評価のために，Laplace，NPB (NAS Parallel Benchmarks)3.1[6]，MediaBench[7] を用いた．Laplace と NPB は科学技術計算における性能評価のために，MediaBench はマルチメディアアプリケーションにおける性能評価のために用いた．コンパイラはバックエンドとして gcc3.2 ベースのクロスコンパイラである m32r-linux-gcc を使用した．

4.1 Laplace and NPB

Laplace ベンチマークはヤコビ法による 5 ステンシル計算を行う Laplace 方程式の解を求めるプログラムである．NPB IS は整数ソートを行うプログラムで，今回用いたデータサイズはクラス S である．Laplace は computation-intensive なベンチマークであり，NPB IS は memory- intensive なベンチマークである．これら 2 つの特性の異なるベンチマークによって M32R の性能特性を評価する．表 3 にこれらのベンチマークの結果を示す．2 つのプロセッサを用いることによって性能が大きく向上しているのが確認できる．Laplace ベンチマークでは，ほぼ 2 倍の性能向上が実現できている．一方で IS においては性能向上は約 1.6 倍に留まっている．したがって，computation-intensive なアプリケーションの方が，memory-intensive なアプリケーションよりも性能向上ができていることがわかった．

	single	2 proc.	speedup
Laplace	109.2	56.7	1.9
IS CLASS S	0.31	0.19	1.63

表 3 Laplace と IS の実行時間とスピードアップ [sec]

4.2 MediaBench

組込みシステムのメインターゲットであるマルチメディアアプリケーションにおける性能を評価するために MediaBench ベンチマークを用いた．MediaBench は組込みマルチメディア向けのベンチマークであり，画像処理や通信，DSP アプリケーションを集めたものである．MediaBench には MPEG，JPG，GSM，G.721，PGP などが納められている．MediaBench の全てのベンチマークが OpenMP によって並列化が可能であるが，今回我々は MPEG2 のエンコードを選んだ．その理由は，MPEG2 エンコードは M3T-32700UT 上での OpenMP の有効性を評価するのにもっとも適していると考えられる．

MPEG は現在の標準的な高画質映像変換規格である．MPEG の変換の主な流れはデコードやエンコードの DC(discrete cosine) 変換である．MPEG にはエンコード用に mpeg2encode，デコード用に mpeg2decode のふたつのプログラムが用意されている．本論文では，mpeg2encode の性能を評価した．図に MPEG2 エンコードの流れを示す．エンコード処理は，motion_estimation，predict，dct_type_estimation，transform，putpict，iquantize，itransform，calcSNR の 8 つの関数から成り立っている．これらの関数は putpict 以外は OpenMP による並列化が用意に可能である．

参考までに，図 5 に並列化したコードを示す．

MPEG2 エンコード処理はブロック単位で計算が行われる．そこで今回，そのブロックを高さ方向に並列化を行った．並列化に必要なコードはわずか 2 行のディレクティブを挿入することで可能であり，スレッドプログラミングを行うのに比べて非常に容易である．

4.2.1 予 備 評 価

M32R での並列化の前にどの関数を並列化すればよいのかを見極めるために，EPCC ベンチマークのときに用いた Pentium III ベースの SMP プラットホームにて予備評価を行った．表 4 にこのシステム

function	single	2 proc.	speedup
motion_estimation	1.823	1.009	1.81
predict	0.001	0.001	0.95
dct_type_estimation	0.018	0.019	0.97
transform	0.374	0.018	1.99
putpict	0.231	0.223	1.04
iquant	0.018	0.020	0.92
itransform	0.038	0.050	0.76
calcSNR	0.025	0.026	0.95
Total	2.762	1.55	1.65

表 4 mpeg2encode の Pentium III SMP (550MHz) での実行時間 [sec]

function	single	2 proc.	speedup
motion_estimation	9.03	4.94	1.83
predict	0.12	0.12	1.01
dct_type_estimation	0.46	0.53	0.87
transform	102.62	73.06	1.40
putpict	2.32	3.81	0.62
iquant	0.13	0.13	1.00
itransform	0.25	0.26	0.99
calcSNR	18.61	11.59	1.60
Total	131.25	94.48	1.41

表 5 mpeg2encode の M3T ー 32700UT での実行時間 [sec]

```
/*loop through all macro-blocks of the picture*/
#pragma omp parallel private(i,j,myk)
{
 #pragma omp for
 for (j=0; j<height2; j+=16)
 {
  for (i=0; i<width; i+=16)
  {
        ... loop body ...
  }
 }
}
```

図 5 mpeg2encode ソースプログラムの並列化例

上での実行結果を示す．シングルプロセッサでの結果から motion_estimation が実行時間の大半を占めており，次に transform が長い時間を費している．そこでこの 2 つの関数を並列化した．その結果 2 プロセッサの場合，2 つの関数に関してほぼ 2 倍の性能向上が得られ，全体として 1.6 倍の性能向上が得られた．また，他の関数についても並列化を行ったが，残念ながら性能向上は得られなかった．

4.2.2 M32R での評価

表 5 に M32R での実行結果を示す．Pentium III の場合と同様に，最初にシングルプロセッサでの評価を行った．シングルプロセッサの結果において，Pentium III ベースのプラットホームと異なる特性を示している．motion_estimation の実行時間の差は約 5 倍であるのに対して predict，transform，calcSNR の実行時間が数百倍になっている．これは M32R に FPU(浮動小数点演算装置) が備わっていないことが原因である．これらの関数は非常に多くの浮動小数点演算を含んでいる．浮動小数点演算の処理に非常に多くの時間が必要となるため，実行時間が急激に増えている．特に transform は実行時間の 99 パーセント以上が浮動小数点演算に費されていることがわかった．2 プロセッサでの実行に際して，性能向上のために calcSNR も並列化を行った．

2 プロセッサでの結果において，我々は全体で約 1.4 倍の性能向上を得ることができた．motion_estimation，calcSNR はおおよそ 2 倍の性能向上であり，motion_estimation は Pentium III ベースでの結果と同じ特性となった．その一方で，transform は Pentium III での結果と異なり 1.4 倍に留まっている．我々はこの原因は M32R が非常に小さな L1 キャッシュのみしか備えていないことであると推測している．

これらの結果より，OpenMP はわずかな量の指示文を挿入することによって用意に性能向上を実現できることが示された。我々は OpenMP が組込み向け CMP にとって非常に容易に組込みアプリケーションを並列化できる手助けになると確信している．

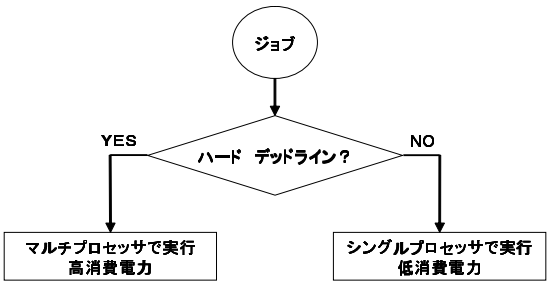

図 6 スケジューリング

5. 低消費電力化

この章では，Power-Aware Computing 実現のために OpenMP と CMP の観点から考察する．

今回，CMP での評価を行うに際に，消費電力の評価も行った．このシステム全体の消費電力は常時約 3W である (測定は DC パワーを測定した．これには CPU，LCD，ネットワークなどの消費電力が含まれている)．

5.1 リアルタイムアプリケーションでのプロセッサのランタイムスケジューリング

図に 6 提案するスケジューリングの概要を示す．リアルタイムアプリケーションでは決められた期限内に予約されたジョブを実行することが必要となる．並列での実行はプログラムの実行を逐次実行よりも早くするが，低電力化の観点からすると並列実行は必要ない．プログラムのデッドラインまで時間が十分にある場合，プログラムはひとつの CPU で実行し他の CPU をスタンドバイ状態にすることによって消費電力を抑えることができる．

このことは排熱やパッケージングにおいて非常に重要なことである．このようなランタイムスレッドスケジューリングのために，CMP においてカーネルと OpenMP スレッドスケジューリングの協調スケジューリングが必要となる．加えて，現在のシステムにおいて CPU をスタンドバイ状態にするカーネルサポートは実装されていない．

実際に消費電力を計測してみると残念なことに，OpenMP での並列実行と逐次実行の間に消費電力の差を確認することができなかった．これは，CPU での消費に比べ LCD やネットワークの消費電力が大きく，片方のわずか数 100mw で動作する CPU が動作していないときの消費電力の減少が隠れてしまったためである．

今後カーネルサポートを実装し Power-Aware Computing 実現のために OpenMP スレッドスケジューリングの開発を行う．

5.2 低消費電力化手法

我々の過去の研究[8] において，低消費電力プロセッサを用いての消費電力，消費エネルギーの評価を行った．低消費電力プロセッサの場合，キャッシュの有効利用が非常に重要であるということがわかっている．キャッシュ最適化として，キャッシュブロッキングを施すことによって低消費電力プロセッサでは 9 倍，組込み向けプロセッサでは約 30 倍もの性能向上が得られ，消費エネルギーの大幅な削減が可能であることがわかった．高性能プロセッサと異なり，組込み向けプロセッサは L1 キャッシュしか備わっていない場合が多く，小さなキャッシュの有効利用がエネルギーの削減のために非常に重要になっている．

OpenMP は，コード生成時にこのような最適化を解析し，施すことが比較的容易であるので，キャッシュ最適化を備えることによってユーザに意識させることなく低電力化が実現できる．

6. まとめと今後

我々は組み込む向け CMP である M32R を搭載したシステムに Omni OpenMP を用いて，OpenMP のクロスコンパイル環境を実装し科学技術計算やマルチメディアアプリケーションベンチマークでの性能の評価を行った．OpenMP のわずかなディレクティブの挿入によって容易に性能向上が得られることがわかった．また，科学技術計算アプリケーションと同様に組み込みアプリケーションにおいても非常に有効なものであることがわかった．

Power-Aware Computing において，組込みアプリケーションの性能と消費電力のトレードオフを可能にするために OpenMP ランタイムスケジューリングの可能性を提案した．しかしながら，最新の OpenMP コンパイラやランタイムシステムは高性能な共有メモリマシン用に設計されている．将来 CMP が低レイテンシの同期を実現する機構を備えれば，OpenMP がリアルタイムシステムやコンパイラおいてスレッド間でのそのような低レイテンシの同期や通信を行うように再設計し，より組込みシステムにとって有効な環境になるはずである．柔軟なランタイムスケジューリングは低レイテンシの同期を使用して動的なスレッド分散スケジューリングを実現する OpenMP の新しい試みとなる．

消費電力削減という点においても，CMP は有望なアーキテクチャである．OpenMP はその利点を容易に利用するため，既存のプログラム財産をそのまま使用でき，かつアプリケーション開発者からスレッド管理など大きな負担となる作業を取り除き，視覚的にもわかりやすいプログラミングモデルを提供している．また OpenMP のランタイムスケジューリングを使用した電力効率の高いスケジューリングを用いることで消費電力やエネルギーを大幅に削減できる可能性がある．今後は，このスケジューリングを実現させるための開発と組込みシステムにおける様々な低消費電力化手法に関して研究を行っていく予定である．

謝辞 本研究で用いたシステム M3T-32700UT を提供していただいたルネサステクノロジーに感謝します．本研究の一部は科学技術研究費補助，課題番号 14208026 による．

参 考 文 献

1) Sasanka, R., V.Adve, S., Chen, Y.-K. and Debes, E.: The Energy Efficiency of CMP vs SMT for Multimedia Workloads, *ISC'04* (2004).
2) ARM MPCore: http://www.arm.com/.
3) Kaneko, S. et al.: A 600MHz Single-Chip Multiprocessor with 4.8GB/s Internal Shared Pipelined Bus and 512kB Internal Memory, *2003 International Solid-State Circuits Conference*, Vol. 1, pp. 254–255 (2003).
4) Linux/M32R Project: . http://www.linux-m32r.org/.
5) Omni OpenMP Project: . http://www.hpcc.jp/Omni/.
6) NAS Parallel Benchmarks: . http://www.nas.nasa.org/NAS/NPB.
7) Lee, C., Potkonjak, M. and Mangione-Smith, W. H.: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communicatons Systems, *International Symposium on Microarchitecture*, pp. 330–335 (1997).
8) Hotta, Y., Sato, M., Boku, T., Takahashi, D. and Takahashi, C.: Measurement and Chracterization of Power Consumption of Microprocessors for Power-Aware Computing, *CoolChips7* (2004).