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

2008－ARC－179 (13)
2008／8／6

OpenMPを用いた並列ベンチマークプログラムによる組込み向けマルチコアプロセッサの評価

塙 敏博[†1] 李 珍泌[†2] 今田貴之[†2]
木村英明[†2] 佐藤三久[†1,†2] 朴 泰祐[†1,†2]

近年，組込みシステムでは高性能化と低消費電力の要求に伴いマルチコアが導入されてきている．本研究では，共有メモリを持つ組込み向けマルチコアプロセッサ3種と，デスクトップ向けマルチコアプロセッサを用いて，OpenMPにより並列化したベンチマークを用いて性能評価を行なった．その結果，メモリ性能が低い場合でもmemory-intensiveでなければ並列化によりコア数に応じた性能向上が得ることができた．またOpenMPを用いて並列化することにより，低い作業コストで高い性能向上が得られることを示した．

Evaluation of Multi-core Processor for Embedded Systems by Parallel Benchmark Program using OpenMP

Toshihiro Hanawa,[†1] Jinpil Lee,[†2] Takayuki Imada,[†2]
Hideaki Kimura,[†2] Mitsuhisa Sato[†1,†2] and Taisuke Boku [†1,†2]

Recently, embedded systems introduce multicore technology to improve performance and reduce power consumption. In this study, three SMP multicore processors for embedded systems and multicore processor for desktop PC are evaluated by the parallel benchmark using OpenMP. As the result, even if the memory performance is low, the application that is not memory-intensive demonstrates large speedup by parallelization. Also, we represent the high performance improvement by the parallelization using OpenMP in spite of the low cost for parallelization.

1. はじめに

近年，テレビなどのディジタル家電やカーナビゲーションシステムなどの車載情報端末など，複雑な機能を持つ組込みシステムが多く使われるようになってきた．このような組込みシステムでは，ユーザインタフェースの高機能化や，扱う情報の大規模化により，高性能な処理が求められる．さらに，携帯端末における連続使用時間の改善や環境への配慮から消費電力の一層の削減が求められている．そのため，組込みプロセッサにおいても，処理性能当たりの消費電力の点で有利なマルチコアによる並列化が進んでいる．

従来マルチコアプロセッサで並列プログラミングを行なうにはPOSIX threadライブラリを用いたマルチスレッドプログラミングや，場合によってはアセンブリ言語を用いて同期コードを記述するなどの必要があった．しかしながら，組込みシステムにおける開発期間の短縮への要請や，組込みシステムに用いられるアプリケーションの大規模化により，習熟が必要で高度な技術を要する複雑なプログラミングに代わって，より簡便にアプリケーションの並列化を行なう手法が求められている．

OpenMP[1)]は，共有メモリ型の並列計算機において並列アプリケーションの開発を容易にする，柔軟性の高いインタフェースを提供するプログラミングモデルである．既存のプログラミング言語であるCやC++, Fortranで書かれたコードに対してディレクティブと呼ばれるヒントを与えることによって，コンパイラが並列化を行なう．近年，商用のコンパイラにもOpenMPに対応している製品が増えてきた．オープンソースのコンパイラとしてよく用いられているGCCも，バージョン4.2から正式にOpenMPに対応しており，OpenMPの開発環境が広く利用できるようになってきている．また逆に，これらのディレクティブを無視すれば逐次版のプログラムを得ることができ，開発やデバッグの際に有用である．

そこで本研究では，共有メモリを持つ組込み向けマルチコアプロセッサに対して，OpenMPを用いて並

†1 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba
†2 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

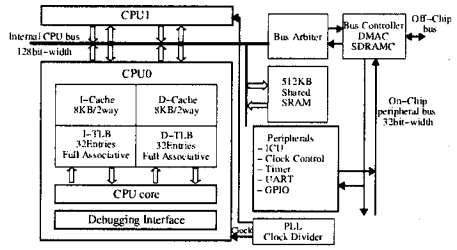

図 1 M32700 プロセッサのブロック図[2)]

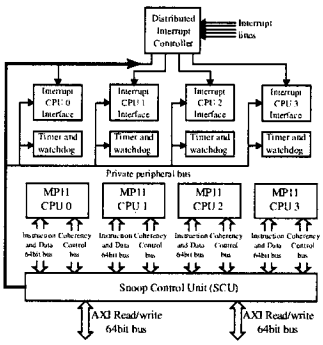

図 2 MPCore プロセッサのブロック図[3)]

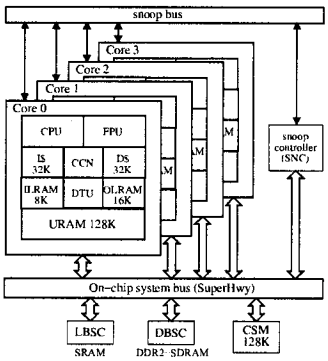

図 3 RP1 プロセッサのブロック図[5)]

列化したベンチマークプログラムを用いて評価を行なう．近年組込みの分野でマルチコアが普及してきつつある中で，

- プログラミング手段として，OpenMP がどの程度効果があるか．
- 同時に，SMP マルチコアプロセッサとして，メモリ性能，同期性能を明らかにし，どう全体性能に影響を与えるか．

について報告する．

2. 組込み向けマルチコアプロセッサ

本研究では共有メモリを持つ組込み向けマルチコアプロセッサとして，ルネサステクノロジ M32700, ARM・NEC MPCore, ルネサステクノロジ・日立・早稲田大による RP1 の 3 種を用いる．さらに比較のためにデスクトップ PC 向けの Intel Core2Quad Q6600 を加え，計 4 種のマルチコアプロセッサを対象とする．

以降，M32700, MPCore, RP1 について詳しく説明する．また，詳細については表 1 にまとめる．

2.1 ルネサステクノロジ M32700

M32700[2)] は 1 チップに M32R-II コアを 2 つ搭載している．図 1 に M32700 のブロック図を示す．M32R-II コアは 7 段パイプラインからなる．チップ内部に 512KB の共有 SRAM を内蔵している．128bit バスで 2 つのプロセッサコアに接続されており，高速にアクセスすることができる．命令セットは 32bit 命令と 16bit 命令からなり，使用頻度の高い命令を 16bit に割り当て，コード効率を高めている．16bit 命令が 2 つ連続しているときは，命令の組み合わせによっては同時発行が可能である．一方，浮動小数点演算の機能は持たない．

2.2 ARM/NEC MPCore

MPCore[3)] は 1 チップに，MP11 コアを 4 つ搭載している．分散割り込みコントローラを備えるなど，様々な構成に柔軟に対応できるように設計されている．図 2 に MPCore のブロック図を示す．MP11 コアは ARMv6 命令セットアーキテクチャに基づいた ARM11 マイクロアーキテクチャを実現したものであり，8 段パイプライン，1 命令同時発行である．32bit の ARM 命令セットの他，16bit の Thumb 命令セット，Java 仮想マシン高速化のため可変長の Jazelle 命令セットを備えている．また，DSP 拡張命令，SIMD 命令なども備えている．今回は ARM 命令セットのみを用いている．各コアは SCU を経由してチップ外部と 2 本の AXI バスで接続されている．今回用いた評価システムでは，AXI バスインタフェース，DRAM コントローラ，SRAM コントローラ等は全て 1 チップの FPGA により実装されている．そのため，外部システムバス，DRAM コントローラはいずれも 30MHz で動作する[4)]．

2.3 ルネサス/日立/早稲田大 RP1

RP1 は SH-X3 アーキテクチャに基づく 4 コア内蔵のマルチコアプロセッサである[5)]．4 つの SH-4A コアを搭載し，600MHz 動作で 4320MIPS，16.8GFlops の性能を示す．図 3 に RP1 のブロック図を示す．SH-4A コアは 8 段パイプラインであり，16bit の命令を 2 命令同時発行可能である．専用のスヌープバスを持っており，キャッシュコヒーレンス制御のためのデータ転送は SHwy のトラフィックを避けて転送することができる．チップ内部には，1 クロックでアクセス可能な命令用ローカルメモリ ILRAM，データ用ローカルメモリ OLRAM，1〜数クロックでアクセス可能な URAM の 3 種のローカルメモリと，オンチップ内共有メモリの CSM が存在する．但し，今回用いた Linux

	M32700	MPCore	RP1
1	ldi r4,#1	mov r3, #1	loop:
2	loop:	loop:	movli.l @r4, r0
3	lock r5,@r0	ldrex r1, [r0]	tst r0, r0
4	unlock r4,@r0	teq r1, #0	bf loop
5	bnez r5,loop	strexeq r1, r3, [r0]	mov #1, r0
6	exit:	teqeq r1, #0	movco.l r0, @r4
7		bne loop	bf loop
8		exit:	exit:

図 4 spinlock コードの実装

では，RP1 を通常の SMP として扱い，これらの内部メモリは用いていない．

3. Omni OpenMP コンパイラ

OpenMP 処理系として我々がこれまで開発して来た Omni OpenMP コンパイラ[6] ver.2 を用いる．スピンロックを用いた実行時ライブラリの実装と，クロスコンパイル環境の実現方法について述べる．

3.1 実行時ライブラリの実装

Omni OpenMP コンパイラの実行時ライブラリでは，排他制御を行なう関数として POSIX thread の mutex_lock と，特定アーキテクチャ専用のスピンロック実装を選ぶことができる．そこで，今回用いた各マルチコアプロセッサ向けに，スピンロックを用いた実行時ライブラリを実装した．ロック変数はあらかじめ 0 に初期化されており，ロックの獲得に成功すると 0 から 1 に変更する．既にロック変数が 1 であれば，ロックは使用中なので獲得できるまでスピンウェイトする．また，false sharing を避けるため，ロック変数はキャッシュラインサイズに合わせてアラインしている．図 4 に各アセンブリ言語による実装を示す．

M32R バスロック・解放を行なう `lock/unlock` 命令を用いて実現する．バスロック中は他のプロセッサ・デバイスはバスアクセスできず性能低下の影響が大きいので，期間を最小にするため，`lock` でロック変数 `r0` の値を `r5` に読み込み，即座に `unlock` で同時にロック変数に 1 をセットする．`r5` が 0 でなければロックが使用中なので繰り返す．実際にはバスの混雑を緩和するため，2 回目以降はロックする前にテストをし，さらに `nop` を挟むなど処理を工夫している．

MPCore 排他ロード/ストア命令 `ldrex/strex` を用いる．`ldrex` でロック変数 `r0` の値を `r3` に読み込み，排他アクセスモードになる．4 行目で `r1` が 0 であれば Z(Zero) フラグをセットする．5–6 行目の命令名の後についている `eq` は，Z フラグがセットされているときのみ実行され，そうでなければ nop とみなす．5 行目では，Z フラグがセット，つまりロック獲得に成功した場合のみ `strex` を実行し，ロック変数に 1 を書き込む．他のプロセッサが書き込みをしておらず排他アクセスモードが続いていれば，書き込みに成功して `r1` に 0 が返り，他プロセッサが既に書き込むなどして失敗すれば 1 が返る．6 行目で成功したかどうかをチェックする．ロック獲得あるいは書き込みに失敗すると 7 行目から `ldrex` に戻る．

RP1 Move Linked 命令 `movli` と Move Conditional 命令 `movco` を用いる．動作は MPCore の排他ロードストアと同様である．`movli` でロック変数 `r4` の値を `r0` に読み込み，0 でなければリトライする．`movco` で 1 をロック変数に書き込み，成功すれば T(True) フラグが 1，失敗すれば 0 になる．失敗していれば 7 行目から 2 行目の `movli` に戻る．

3.2 クロスコンパイル環境

Omni OpenMP コンパイラは，C, Fortran から共通の中間表現に変換するフロントエンド部，OpenMP ディレクティブを解釈しマルチスレッドプログラムに変換する変換部と実行時ライブラリからなる．フロントエンド部と変換部はホスト PC 上 (x86) でネイティブの場合と同様に実行することができる．バックエンドとして対象アーキテクチャのクロスコンパイラを指定すると，変換後のマルチスレッドプログラムがクロスコンパイルされ，3.1 節で述べた実行時ライブラリ，Pthread ライブラリとリンクされて実行ファイルが生成される．

4. 性能評価

4.1 評価環境

本研究では，2 章で述べた 4 種のマルチコアプロセッサを用いて性能評価を行なう．これらの主な仕様および動作環境を表 1 に示す．OS には，いずれも Linux 2.6 を使用した．Intel Core2Quad を除いて，いずれも 100Mbps の Ethernet 経由で起動しており，ファイルシステムには NFS を用いている．OpenMP 処理系には，いずれの環境においても，Omni OpenMP コンパイラ ver.2 を用いた．またバックエンドコンパイラとして表 1 に示した C コンパイラをそれぞれ指定した．C ライブラリとしては glibc を用いたが，POSIX thread 実装がそれぞれ異なる．LinuxThreads は最初の Linux への POSIX thread 実装である．スレッド生成・終了を扱うマネージャスレッドが必要であり，同期関連の処理はシグナルを使って実装されている．一方，Native POSIX Threads Library (NPTL) は，linuxthreads の問題点を解決した実装であり，高速なユーザレベルロック機構である futex を利用して実装されている．RP1 の linuxthreads ライブラリについては，tas(test-and-set) 命令が使われていたが，SMP 環境で使うべきではないため，図 4 と同様に movli/movco 命令を用いて書き換え，システムの libpthread.so と入れ替えている．

4.2 ベンチマーク

MiBench suite は EEMBC ベンチマーク[7] をモデルに作られた，組込みプロセッサ向けのフリーのベンチマーク集である[8]．Automotive and Industrial Con-

表 1 評価環境

プロセッサ	ルネサス M32700	ARM/NEC MPCore	ルネサス/日立/早大 RP1	Intel Core2Quad Q6600
コア数	2	4	4	4
コア動作周波数 内部バス周波数 外部バス周波数	300MHz 75MHz 75MHz	210MHz 210MHz 30MHz	600MHz 300MHz 50MHz	2.4GHz
キャッシュ (byte) 命令+データ, ライン	2way 8K+8K 16	4way 32K+32K 32	4way 32K+32K 32	32K+32K(L1), 4M(2 コア共有)×2(L2)
メモリ 種別	32Mbyte SDRAM, 100MHz	256Mbyte DDR-SDRAM, 30MHz	128Mbyte DDR2-600, 300MHz	2Gbyte DDR2-800, 400MHz
OS (uname -m)	Linux 2.6.25 (m32r)	Linux 2.6.19 (armv61)	Linux 2.6.16 (sh4a)	Linux 2.6.18 (i686)
C コンパイラ オプション	gcc 4.1.2 20061115 -m32r2	gcc 4.1.1 -mcpu=mpcore	gcc 3.4.5 20060103 -m4a	gcc 4.1.2 20061115 -march=nocona
C ライブラリ pthread 実装	glibc 2.3.6.dsl-13 Linuxthreads 0.10	glibc 2.3.6 NPTL 2.3.6	glibc 2.3.3 Linuxthreads 0.10	glibc 2.3.6.dsl-13.etch5 NPTL 2.3.6

表 2 EPCC syncbench の結果 (上: Mutex, 下: スピンロック, 単位: μs)

	M32700	MPCore	RP1	Q6600
parallel	392.2	436.5	107.8	2.80
	376.8	434.8	107.2	2.25
for	18.5	7.46	1.66	0.364
	13.6	6.15	1.42	0.372
parallel for	399.7	436.3	108.2	3.71
	383.6	435.7	107.7	2.47
barrier	14.1	6.11	1.13	0.301
	10.7	5.98	0.867	0.316
single	50.8	3.14	295.1	4.54
	9.87	3.12	1.53	0.859
critical	273.5	0.921	128.2	1.31
	3.15	0.837	0.190	0.129
lock/unlock	273.1	1.03	121.0	1.41
	2.51	0.962	0.174	0.131
ordered	8.64	1.50	0.584	0.191
	7.08	1.33	0.598	0.168
atomic	241.0	0.894	121.0	0.474
	0.501	0.893	0.365	0.307
reduction	401.9	443.9	327.0	6.13
	387.1	443.8	109.1	3.35

trol, Consumer Devices, Office Automation, Networking, Security, Telecommunications の 6 つのカテゴリ, 計 35 のベンチマークからなる. 今回は, Automotive から画像処理アルゴリズムの Susan smoothing(SS), Security から暗号アルゴリズムの blowfish encrypt(BF), Telecommunications から FFT の 3 種を OpenMP を用いて並列化を行なった.

他に, NAS Parallel Benchmark[9] より, NPB3.3-OMP[10] の IS と, NPB の CG を C 言語と OpenMP により実装したもの, さらに MediaBench[11] より, Mpeg2encode を OpenMP 化したもの (MPEG)[12] を用いた.

4.3 評価結果

4.3.1 同期操作の性能

予備評価として, EPCC マイクロベンチマーク[13] の syncbench を用いて, 各システムで同期操作の評価を行なった. 結果を表 2 に示す. M32700 は 2PU, 他は 4PU の場合を示している. Pthread 実装に Linuxthreads を用いている M32700 と RP1 では, mutex_lock を用いた single, critical, lock/unlock, atomic の結果が極端に悪い値を示している. これはスレッドの待ち合わせ処理の際, シグナルを使ってハンドリングしている影響ではないかと思われる. 一方, スピンロックの場合には, M32700 において最大で atomic の 482 倍, RP1 において最大で lock/unlock の 695 倍と, 劇的な性能向上が得られた. NPTL を用いている MPCore については, mutex_lock とスピンロックの性能差は最大で ordered の 1.12 倍程度であるが, いずれの値もスピンロックにすることで改善が見られる. そこで, 以降の評価は全てスピンロックで行なうこととする. 組込み向けマルチコアでは, parallel ディレクティブの要素を含んだ parallel, parallel for, reduction の実行時間が極めて大きい. これはスレッド生成やコピーなどのため主記憶上に頻繁にアクセスが生じるためではないかと考えられる. 実際に DRAM アクセスの最も遅い MPCore が最も時間を要しており, 次いで M32700, RP1 となっている. 従って, parallel ディレクティブ関連はループ毎に頻繁に呼び出すのではなく, できるだけ大きな単位で 1 回だけ記述した方が良い. Core2Quad に関しては, 最も高速な DDR2-800 メモリを使用している上に, 8Mbyte と大きな L2 キャッシュを持っているため, メモリアクセス時間を隠蔽できていると考えられる.

4.3.2 OpenMP による並列化

次に, NAS Parallel Benchmark IS(CLASS W) の実行時間を表 3 に, 性能向上率を図 5 に示す. なお, 表 3 には以降全てのベンチマーク結果を記載した. IS は memory intensive なプログラムであり, メモリアクセスの遅い MPCore では 3〜4 コアでも 1.6 倍, M32700 では 1.1 倍とあまり性能向上が見られない. 一方, RP1 では 4 コアで 2.9 倍の性能向上を示している. RP1 ではスヌープ専用バスを持っているためキャッシュが有効に機能しているものと考えられる.

図 6 は CG (1400×1400 疎行列) の性能向上率を示したものである. こちらは computation-intensive であり, キャッシュにもヒットしやすく MPCore を除いて同様の速度向上率を示し, 4 コアで 2.8 倍であった, MPCore においては 4 コアで 3.8 倍の性能を示し, コア辺りのデータセットが小さくなることからキャッシュのヒット率が増加しているものと考えられる.

次に, MiBench については, 様々な Workload を与えるための MiDataSets[14] から, SS には `19.pgm`, BF

表 3 性能評価の結果（単位: s）

プロセッサ	PU	IS	CG	Susan_s	FFT	Blowfish	Mpeg2encode
M32700	1	4.66	183.5	22.1	19.6	175.3	1143.8
	2	4.32	93.2	11.8	10.5	94.3 (90.6–102.4)	788.0
MPCore	1	11.3	31.0	21.7	3.19	138.9	—
	2	7.66	15.6	11.2	1.71	112.4 (69.8–140.6)	—
	3	7.09	10.8	7.80	1.31	79.4 (46.8–139.8)	—
	4	7.09	8.15	5.99	1.03	76.1 (34.9–139.3)	—
RP1	1	4.14	4.43	8.49	0.280	60.2	57.4
	2	2.26	2.44	4.51	0.147	33.5 (29.5–60.0)	46.6
	3	1.64	1.85	3.18	0.107	31.6 (19.9–59.8)	39.8
	4	1.41	1.56	2.52	0.085	23.5 (19.7–30.6)	35.9
Q6600	1	0.06	0.215	0.933	0.0076	6.12	5.47
	2	0.036	0.116	0.476	0.0048	3.14 (3.13–3.15)	3.69
	3	0.029	0.086	0.322	0.0065	2.59 (2.58–2.62)	2.88
	4	0.026	0.073	0.251	0.0063	1.70 (1.69–1.71)	2.48

には `4.txt` の入力を用いて評価を行なった．また，FFT は MiBench の large データセットより，nwave=6, nsample=65536 を用いた．なお，BF では，MiBench で用いられていたアルゴリズムを CFB64 モードから ECB モードに変更している．SS, BF, FFT の性能向上率を図 7, 8, 9 に示す．SS については並列性が高く，3.4 倍〜3.7 倍の性能が得られている．一方で OpenMP 化の際 6ヶ所にディレクティブを挿入するのみであった．BF については，ファイルから 40byte 読み込み，100 回暗号化をして 40byte ファイルに書き込むという操作を，各コアがパイプライン処理をしているが，実行時間に大きなばらつきがあるため，最低値の場合と最高値の場合を errorbar で示した．Q6600 はほとんどばらつきを示さなかったため NFS の影響だと思われる．ECB モードでは 8byte 間にしかデータ依存がないため，高い並列性を示した．OpenMP 化には，ディレクティブを 9ヶ所，ファイルの読み書き順序を制御するためにコードを十数行追加した．FFT も並列性が高く RP1 で 4 コア時に 3.3 倍の性能を得た．Core2Quad に関しては，3〜4 コアで性能が低下しているが，非常に短時間で処理が終了しており，同期コストの方が大きいと考えられる．OpenMP 化にはディレクティブを 4ヶ所に追加した．

最後に，MPEG の性能向上率を図 10 に示す．MPCore についてはエラーで実行できなかった．Q6600 で 4 コア時に 2.2 倍であるが，RP1 では 1.6 倍であった．MPEG ではファイルの入出力が実行時間に含まれているが，NFS の影響で性能が低下したものと考えられる．M32700 についてはソフトウェア浮動小数点演算のため実行時間が長く，性能低下が起こらなかったものと考えられる．

5. 結　　論

組込み向けマルチコアプロセッサである，M32700, MPCore, RP1 と，デスクトップ向けマルチコアプロセッサ Core2Quad Q6600 の 4 種について，OpenMP を用いて並列化した様々なベンチマークを用いて性能評価を行なった．

その結果，組込み向けマルチコアプロセッサについて，同期のコストはデスクトップ PC 向けプロセッサに比べると大きく，メモリ性能も低いものの，コア数に見合った速度向上率を得ることができた．また，OpenMP を用いた並列化のコストが低く，少ない労力で大きな性能向上が得られた．

今後の課題として，マルチコアプロセッサが内蔵しているオンチップメモリを同期操作に有効に使用することや，各マルチコアプロセッサのシステム上で，用いたコア数による消費電力の変化を測定することを検討している．

謝辞 本研究で用いた M3T-32700UT を貸与していただいた（株）ルネサステクノロジに感謝致します．本論文における RP1 マルチコアチップ及び実行環境は，NEDO リアルタイム情報家電用マルチコアプロジェクトにて早稲田大学（笠原・木村研究室），（株）ルネサステクノロジ，（株）日立製作所により開発されたものを利用している．本研究の一部は，科学技術振興機構 戦略的創造研究推進事業 (CREST) 研究領域「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」，研究課題「省電力高信頼組込み並列プラットフォーム」による．

参 考 文 献

1) The OpenMP Architecture Review Board: *The OpenMP API specification for parallel programming*. http://openmp.org/wp/.
2) Kaneko, S. et al.: A 600MHz Single-Chip Multiprocessor with 4.8GB/s Internal Shared Pipelined Bus and 512kB Internal Memory, *International Solid-State Circuits Conference (ISSCC) 2003*, Vol.1, pp.254–255 (2003).
3) ARM Limited: *ARM11 MPCore Processor Technical Reference Manual* (2006).
4) ARM Limited: *Using a CT11MPCore with the RealView Emulation Baseboard* (2006).
5) Yoshida, Y. et al.: A 4320MIPS Four-Processor Core SMP/AMP with Individually Managed Clock Frequency for Low Power Consumption, *International Solid-State Circuits Conference (ISSCC) 2007*, pp.100 – 590 (2007).
6) 草野和寛，佐藤茂久，佐藤三久：Omni OpenMP コンパイラの性能評価，情報処理学会論文誌，

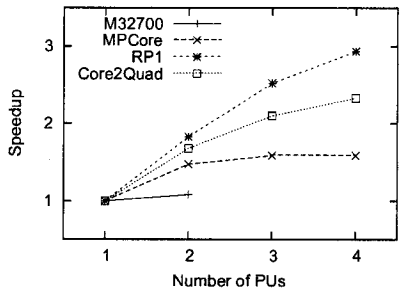

図 5 NPB IS (CLASS=W) の速度向上率

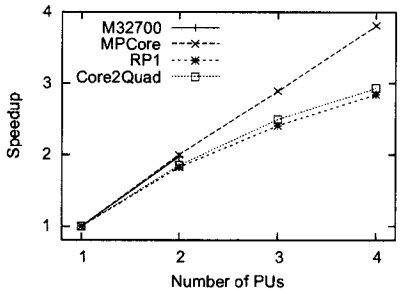

図 6 CG の速度向上率

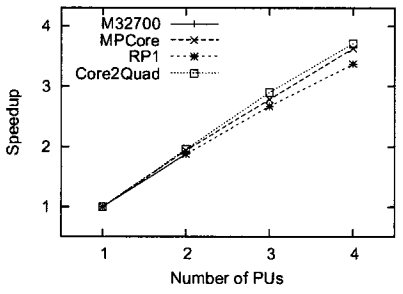

図 7 Susan (smoothing) の速度向上率

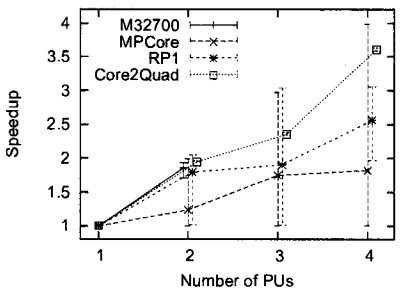

図 8 Blowfish の速度向上率

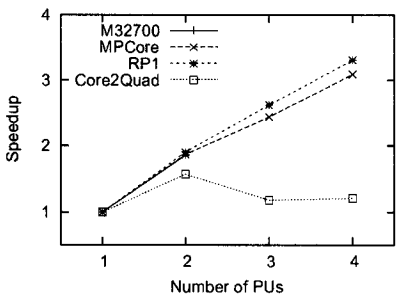

図 9 FFT の速度向上率

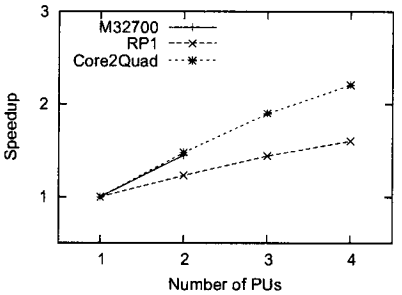

図 10 Mpeg2encode の速度向上率

Vol.42, No.4, pp.802–811 (2001).

7) The Embedded Microprocessor Benchmark Consortium: *EEMBC — The Embedded Microprocessor Benchmark Consortium.* http://www.eembc.org/.

8) Guthaus, M. et al.: MiBench: A free, commercially representative embeded benchmark suite, *IEEE 4th Annual Workshop on Workload Characterization* (2001).

9) Bailey, D. et al.: The NAS Parallel Benchmarks, *RNR Technical Report RNR-94-007*, NASA Ames Research Center (1994).

10) Jin, H., Frumkin, M. and Yan, J.: The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance, *NAS Technical Report NAS-99-011*, NAS System Division NASA Ames Research Center (1999).

11) Lee, C., Potkonjak, M. and Mangione-Smith, H.: MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, *Micro-30* (1997).

12) 堀田義彦, 佐藤三久, 中島佳宏, 小島好紀：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, 情報処理学会研究報告 2004-ARC-160, pp.59–64 (2004).

13) Bull, J. M.: Measuring Synchronisation and Scheduling Overheads in OpenMP, *European Workshop on OpenMP (EWOMP '99)*, pp.99–105 (1999).

14) Fursin, G., Cavazos, J., O'Boyle, M. and Temam, O.: MiDataSets: Creating The Conditions For A More Realistic Evaluation of Iterative Optimization, *Proceedings of the International Conference on High Performance Embedded Architectures & Compilers (HiPEAC 2007)* (2007).