M32R マルチプロセッサへの Linux SMP カーネルの実装

藤原　隼人[†a)]　　山本　整[††]　　高田　浩和[†]
坂本　圭[†]　　作川　守[†]　　近藤　弘郁[†]

Implementation of Linux SMP kernel for M32R multiprocessor

Hayato FUJIWARA[†a)], Hitoshi YAMAMOTO[††], Hirokazu TAKATA[†], Kei SAKAMOTO[†], Mamoru SAKUGAWA[†], and Hiroyuki KONDO[†]

あらまし
近年，組み込み機器に対する高性能化かつ低消費電力化の要求の高まりとともに，一つのチップに複数の CPU コアを内蔵した，シングルチップマルチプロセッサ技術が注目されてきている．マルチプロセッサを活用するにあたっては，プロセスの管理やリソースの競合の制御などを行なうマルチプロセッサ対応 OS が不可欠である．我々は，ルネサステクノロジの 32 ビット RISC マイクロプロセッサである M32R 用の Linux — Linux/M32R を開発し，マルチプロセッサ機能の実装を行なった．
本稿では，Linux/M32R のマルチプロセッサ対応について，また，現在の Linux/M32R の状況について述べる．

キーワード　M32R アーキテクチャ，Linux 移植，シングルチップマルチプロセッサ，SMP カーネル，組み込み用 Linux ソフトウェア・プラットホーム，クロス開発

1. はじめに

近年，携帯電話・デジタルカメラに代表される組み込み機器に対する，高性能化の要求が高まってきている．これにともない，これらの機器に搭載されるプロセッサの性能向上が求められている．しかし，組み込み用プロセッサにおいては性能のみならず，低消費電力動作も同時に要求される．高性能なプロセッサ・コアを求めクロック周波数の高速化に依存した性能向上を行なうと，消費電力が大きくなってしまう．このため，回路技術・プロセス技術のみではなくアーキテクチャ面においても何らかのブレークスルーとなる技術が不可欠となってくる．このような技術として，マルチプロセッサ技術，とりわけシングルチップマルチプロセッサ技術が注目されてきている．マルチプロセッサ技術は，複数のプロセッサ・コアを搭載する事でクロック周波数を上げずに性能向上を図ることができる．これに加えて，データ処理の機能分散を行いシステム性能を向上できる可能性を持つ事から，組み込み向けプロセッサにおいても非常に有用な技術であると考えられる．また，複数のプロセッサ・コアを一つのチップに搭載する事によって，複数のチップを用いた場合に比べ消費電力を大幅に削減する事ができ，実装面積を小さくする事が可能である．このため，シングルチップマルチプロセッサ技術は，組み込み向けプロセッサにおいてマルチプロセッサを実現する場合重要となる．

このような状況のもとルネサステクノロジでは，組み込み用 32 ビット RISC マイコン M32R のシングルチップマルチプロセッサを試作した [1]．M32R コアは，コンパクトなプロセッサ・コアであるため，このように複数のプロセッサ・コアを搭載するのに適している．

マルチプロセッサを活用するにあたっては，プロセスの管理やリソースの競合の制御などの OS のサポートは必須である．一方，Linux はフリーのマルチプロセッサ対応 OS として注目されている．

我々は 2000 年より M32R プロセッサ用の Linux — Linux/M32R の開発を続けており，カーネルの移植，GNU ツールの拡張，プラットフォーム開発，ラ

† 株式会社ルネサステクノロジ
Renesas Technology Corp.
†† 三菱電機株式会社
Mitsubishi Electric Corp.
a) E-mail: fujiwara@linux-m32r.org

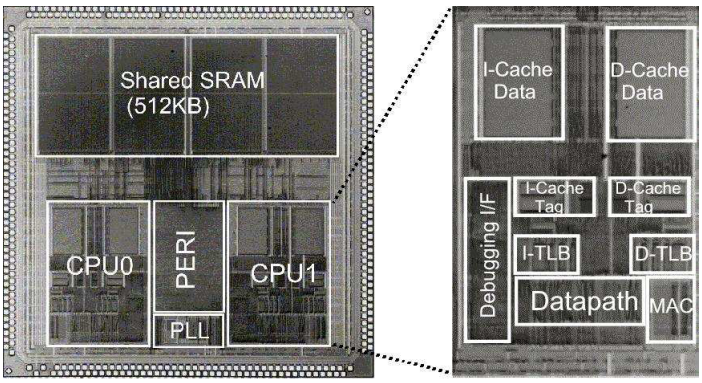

図 1　M32R Single-Chip Multiprocessor

イブラリ移植などを行なってきた [2] [3] [4] [5] ．Linux/M32R は，UP(Uni-processor)，SMP(Symmetric Multi-processor)，さらには MMU を持たない CPU でも動作する noMMU 版と幅広いプラットフォームをサポートしている．

本稿では，Linux/M32R のマルチプロセッサ対応を，ハード・ソフト双方について述べる．また，現在の Linux/M32R の状況についても述べる．

2. M32R シングルチップマルチプロセッサ

2.1 M32R アーキテクチャ

システム全体を一つの LSI に集積した SoC(System on a Chip) では，目的とするシステムに最適なチップ機能や回路構成を実現する事ができ，システムの性能とコストを最適化する事ができる．このようなシステム LSI においては，組み込まれるプロセッサ・コアが LSI の性能・コストを左右する重要なコンポーネントとなる．このため，よりコンパクトで高性能なプロセッサ・コアが求められている．M32R コアはこのようなシステム LSI 向けに開発された，ルネサステクノロジ・オリジナルの 32 ビット RISC プロセッサ・コアである．

現在，Verilog-HDL で記述されフル合成可能な M32R ソフトマクロがオープン化されて VDEC から無償公開されており [6]，テストベンチ，テストケースなどを含む開発環境を研究および教育目的で利用（改変を含む）する事が可能となっている．

2.2 マルチプロセッサ機能

前述したように，マルチプロセッサ技術は，組み込み向けプロセッサにおいても非常に有用な技術であると考えられる．

図 1 は M32R シングルチップマルチプロセッサの試作チップの写真である．このプロセッサは 2 つの CPU が共有バスを介してメモリ・I/O と接続されているバス結合型のマルチプロセッサである．このプロセッサではマルチプロセッサ機能を次のような方式を用いて実現している．

- 共有メモリと共有バス

  共有メモリは全 CPU から単一の物理アドレスによりアクセスできる．2 つの CPU および共有メモリを管理するバスアービタが共有バス上で結合されており，各 CPU 間はバスアービタによりラウンドロビン方式で調停される．

- キャッシュ

  各 CPU は，専用のプライベートキャッシュを持つ．プロセッサ間のデータのコヒーレンシを保つために，CPU バスのオペランドアドレスを監視するスヌープ機能を備えている．

  キャッシュラインの状態の管理は Modified (変更)，Exclusive (排他)，Shared (共有)，Invalid (無効) の 4 つのステートを用いる MESI プロトコルで行なう．

- I/O レジスタ

  一部のレジスタを除き，全 CPU から単一の物理アドレスによりアクセスが可能である．

- 同期機構

  同期機構として，他の CPU あるいは自 CPU に対して割り込み（IPI: InterProcessor Interrupt）を発生させるプロセッサ間割り込み機能を備える．ま

た，LOCK 命令，UNLOCK 命令を用いて共有バスを専有するバスロック機構を備える．

- 割り込みの分配

外部端子や周辺 I/O からの割り込み入力は，割り込みコントローラで調停された後，すべての CPU に分配される．

3. Linux/M32R へのマルチプロセッサ機能の実装

シングルチップマルチプロセッサ技術は，組み込み向けプロセッサにおいても近い将来普及していくものと考えられる．これらのマルチプロセッサ技術を有効に，かつ容易に活用するためには，プロセス管理やリソースへのアクセスの制御などの OS のサポートが必須であり，フリーのマルチプロセッサ対応 OS である Linux は組み込みシステムを開発するにあたって，非常に有効な解になるものと考える．

2.2 で述べたシングルチップマルチプロセッサ上で Linux SMP カーネルを動作させるべく Linux/M32R の開発を行なった．マルチプロセッサでの動作をサポートするために，以下のような機構・処理を実装した [4].

- 同期機構

Linux では同期機構として，“スピンロック”，“セマフォ”，“アトミック操作” を使用している．このうち，MP(Multi-processor) でのみ用いられるスピンロックについては新規にコーディングし，UP(Uni-processor) でも用いられるセマフォ，アトミック操作については MP に対応するためのコード修正を行なった．スピンロックの実装では，後述するコード配置の最適化により，性能の向上をはかっている．

スピンロックは M32R の LOCK/UNLOCK 命令を用いて実装を行なった．LOCK 命令は排他制御用のロード命令，UNLOCK 命令は排他制御用のストア命令であり，LOCK 命令を実行すると UNLOCK 命令によるストアを実行するまでの間，プロセッサは CPU バスのバス権を獲得する事ができる．ただし，バス権を獲得できない場合にはプロセッサの LOCK 命令の実行はハードウェア的に待たされる．割り込みを禁止した状態で LOCK および UNLOCK 命令を用いて lock 変数にアクセスする事により，lock 変数に対するアトミックなアクセスが可能となる．Linux/M32R ではこれを利用してスピンロック操作を実現している．

この新規に追加したスピンロックのコードにおいて，当初の実装ではロックを獲得できなかった場合に実行されるビジーループのコードを，通常のコードと同一のセクションに配置していた．しかし，Linux カーネルではロック期間を短くするように設計されており，ロックを獲得しようとした場合に，獲得できずにビジーループを実行する確率は低い．このため，通常のコード中にビジーループが配置される事によってキャッシュの利用効率が落ちてしまう．そこで，従来は通常のコード中に配置されていたループ部分を別セクションへと移動した．r0 をロック変数アドレスとした場合の例について，変更前のコードをリスト 1 に，変更後のコードをリスト 2 に示す．この変更により，実行されないコードがキャッシュに入る事によるキャッシュの利用効率の低下を抑制することが出来る．この最適化を行なう事により，後述する LMbench の Local Communication latencies テストにおいて約 10% の性能向上をはかる事が出来た．

セマフォ，アトミック操作についても，MP に対応するため M32R の排他制御ロード・ストア命令

```
1:      mvfc    r5, psw
        clrpsw #0x40     # disable int.
        lock    r4, @r0  # \
        addi    r4, #-1  #  atomic op.
        unlock  r4, @r0  # /
        mvtc    r5, psw
        beqz    r4, 3f
2:      ld      r4, @r0  # busy loop
        blez    r4, 2b   # |
        bra     1b
3:
```

リスト 1 lock サブセクションなし

```
1:      mvfc    r5, psw
        clrpsw #0x40     # disable int.
        lock    r4, @r0  # \
        addi    r4, #-1  #  atomic op.
        unlock  r4, @r0  # /
        mvtc    r5, psw
        bnez    r4, 2f
        .subsection 1
        .text.lock
2:      ld      r4, @r0  # busy loop
        blez    r4, 2b   # |
        bra     1b
        .previous
```

リスト 2 lock サブセクションあり

(LOCK/UNLOCK) を用いた排他制御を使用して変数操作をするように修正を行なっている．

- CPU 間通信

ある CPU が他の CPU へ処理を要求する場合に CPU 間で通信を行なう必要がある．Linux/M32R ではこの CPU 間通信をプロセッサ間割り込み（IPI）を用いて実装した．CPU 間通信を用いて他の CPU へ送られる処理要求を以下に示す．

- 再スケジューリング
- キャッシュフラッシュ
- TLB フラッシュ
- ローカルタイマ
- CPU 停止

これらは，IPI が入る事によって割り込みハンドラが起動され処理が実行される．以下にこのシーケンスを示す．

（1） 送信側プロセッサが IPI を送信する

（2） 受信側プロセッサで割り込みが発生する

（3） 送信側プロセッサは割り込みが受け付けられた事を確認して終了する

（4） 受信側プロセッサは割り込み要因に応じた割り込みハンドラを実行し，処理を行なう

- 起動シーケンス

M32R シングルチップマルチプロセッサでは，システムが起動した時に動作を開始する CPU は 1 つのみである．この CPU を BSP: BootStrap Processor という．また，BSP 以外の CPU は AP: Application Processor と呼ばれ，システム起動時はスリープモードとなり，プロセッサ間割り込みが入力されるまで動作を開始しない．

以下にマルチプロセッサとして起動する際のシーケンスを示す．

（1） システムが起動し，BSP が動作を開始する

（2） BSP がハードウェアおよび Linux の初期化を実行する

（3） BSP が各 AP に対して IPI を発行する

（4） それぞれの AP が動作を開始し，初期化後 cpu_idle に入る

4. Linux/M32R の現在のステータス

Linux の M32R への移植は，開発当初 Linux 2.2 系のカーネルに対して進めてきた．その後 2.4 系のカーネルへ移行するなどいくつかのバージョンを経て，現在[(注1)]は最新の 2.4.25 が動作している．また，2.6 系のカーネルについても，2.6.0 の正式リリース版を経て，現在では最新の 2.6.4 が動作している．2.6 系のカーネルでは Preemptible kernel などの新規に追加された拡張への対応も行なっている．さらに，2.6 系のカーネルで拡張された noMMU 版の開発も行なっており，MMU を持たない CPU でも Linux を動作させる事ができるようになっている．また，2.4 系・2.6 系ともにビッグエンディアンだけではなく，リトルエンディアンでの動作にも対応している．

M32R 用の 2.4 系・2.6 系カーネル，ツール，ユーザーランドなどは Linux/M32R プロジェクトの Web ページ[(注2)]で公開されており，自由にダウンロードする事が可能である．また，ユーザーランドとして M32R 用の deb パッケージも公開されており，“http://debian.linux-m32r.org/” からダウンロードできる．これらのパッケージは apt-get で取得する事もできる．標準カーネルへのマージを目指して開発をつづけており，今後のリリースに対しても追従していく予定である．

5. プラットフォーム

Linux を M32R 上で実行させるためのターゲットボードの開発も行なっている．また，市販されているボードでも Linux/M32R が動作可能である．現在 Linux/M32R を動作させる事の出来るプラットフォームには以下の物がある．

- Mappi
  - Maker: ルネサステクノロジ
  - CPU: M32R (M32700) ～400MHz single-chip-multiprocessor
  - SDRAM: 64MB FLASH: 4MB
  - LAN: 10Base-T
  - PCMCIA slot, Display I/F (VGA)
  - Size: 145 mm x 135 mm
  - Photo: 図 2
- Mappi-II
  - Maker: ルネサステクノロジ
  - CPU: M32R ソフトマクロ・コア
  - FPGA: CPU/MMU/キャッシュ/周辺 IO
  - SDRAM: 64MB FLASH: 4MB
  - LAN: 100Base-TX
  - USB: Host2.0
  - CF/MMC/MS slot, IDE, PC/104
  - Size: 160mm x 120mm
  - Photo: 図 3

(注1)：2004/04/05 時点

(注2)：http://www.linux-m32r.org/

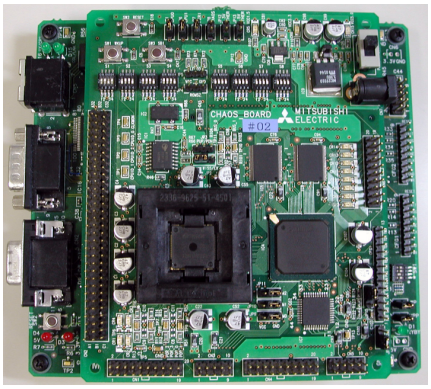

図 2　Mappi

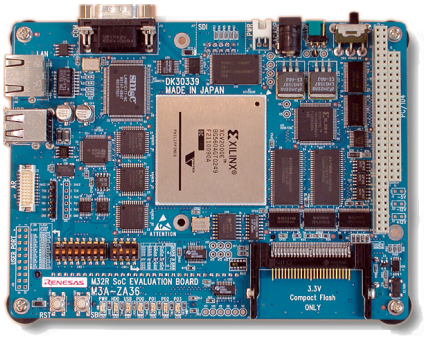

図 3　Mappi-II

- M3T-M32700UT
  - Maker: ルネサステクノロジ
  - CPU: M32R (M32700) 400MHz
    single-chip-multiprocessor
  - SDRAM: 32MB FLASH: 4MB
  - LAN: 100Base-TX
  - USB:Host2.0
  - CF/MMC/eTRON slot, LCD, AR カメラ
  - Size: 60mm x 85mm
  - Photo: 図 4
- OAKS32R
  - Maker: オークス電子
  - CPU: M32R (M32102) 66MHz w/o MMU
  - SDRAM: 8MB FLASH: 2MB
  - LAN: 10Base-T
  - Size: 60mm x 105mm
  - Photo: 図 5

以上のプラットフォームのうち Mappi, Mappi-II, M3T-M32700UT はマルチプロセッサでの動作が可能である．また，OAKS32R [7] が搭載する M32102 は，MMU を持たない CPU であるが，noMMU 版の Linux-2.6 が動作可能である．

図 4　M3T-M32700UT

図 5　OAKS32R

表 1　測定環境

	Mappi
CPU	M32R (M32700)
CPU clk.	200MHz
BUS	32bit 100Mz
Cache	I8KB+D8KB
SDRAM	64MB
File System	NFS
Tool	GCC 3.2.3

6. 性能評価

マルチプロセッサ対応 Linux/M32R の評価を行なった．評価には LMbench [8] と NPB: NAS Parallel Benchmark [9] を用いた．測定は 5. で述べた Mappi を用いて実行した．このスペックを表 1 に示す．

6.1 LMbench

LMbench の測定を，カーネル 2.4.25, 2.6.4 それぞれ UP, SMP で行なった．

LMbench は Larry McVoy 氏らによって作成された，OS・ハードウェアの基本的な性能を測定するシンプルでポータブルなベンチマークスイートである．このベンチマークスイートでは以下の項目の性能を測定することができる．

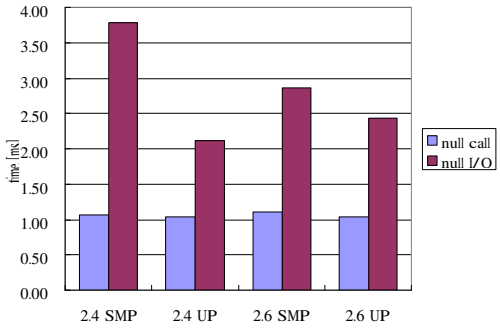

図 6　LMbench null call, null I/O

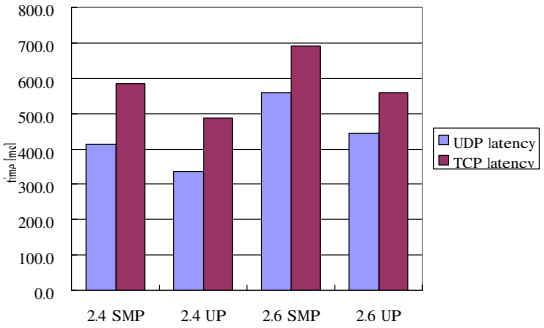

図 7　LMbench TCP,UDP latency

- Bandwidth benchmarks
  キャッシュやメモリ，通信の帯域幅を測定する．
- Latency benchmarks
  コンテキストスイッチ，ネットワーク，ファイル操作，プロセス，シグナル，システムコールおよびメモリの遅延を測定する．
- Miscellaneous
  プロセッサのクロック周波数，TLB のエントリ数，キャッシュラインサイズなどを測定する．

LMbench は安定版のバージョン 2.0.4 と開発版 (α 版) の 3.0-a3 が存在する．2.0.4 ではコンテキストスイッチのレイテンシ測定でのオーバーヘッドの算出にバグがあり，正しい結果を得る事ができないなどの不具合がある．3.0-a3 ではこのバグなどが修正されているため，今回は α 版ではあるが，3.0-a3 を用いて測定を行なった．

LMbench 3.0-a3 を測定した結果から，null call, null I/O, TCP latency, UDP latency をグラフ化したものを図 6，7 に示す．

UP と SMP の結果を比較すると，SMP での速度低下は UP と比べて 10%から 20%に抑える事ができている．これは i386 アーキテクチャにおける，SMP の速度低下とほぼ同程度の数字である．この結果より，M32R アーキテクチャに対する SMP 対応の実装が適切に行なわれていると考える．

また，2.4 カーネルと 2.6 カーネルを比較すると，2.6 カーネルでは 2.4 カーネルと比べ，速度が低下している項目が多く見られる．このため，2.6 カーネルに関しては，今後，性能低下の原因の調査・問題箇所のチューニングなどを行なうことが必要であると考える．

6.2　NAS Parallel Benchmark

NPB CLASS S の測定をカーネル 2.4.19 で行なった．測定は，Fortran90 を用いる FT を除いた EP, MG, CG, IS, LU, SP, BT について CLASS S で行なった．

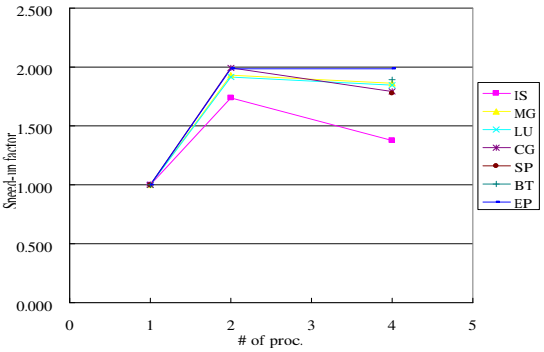

図 8　NPB Class S

NPB: NAS Parallel Benchmarks [9] は NASA Ames Research Center で開発された並列コンピュータ用ベンチマークソフトである．NPB は並列コンピュータの実効性能 (並列計算効率) を測定する代表的なベンチマークプログラムである．

NPB には以下のベンチマークが含まれている．

- EP: 乗算合同法による一様乱数、正規乱数の生成
- MG: 簡略化されたマルチグリッド法のカーネル
- CG: 正値対称な大規模疎行列の最小固有値を求めるための共役勾配法
- FT: FFT を用いた 3 次元偏微分方程式の解法
- IS: 大規模整数ソート
- LU: Symmetric SOR iteration による CFD アプリケーション
- SP: Scalar ADI iteration による CFD アプリケーション
- BT: 5x5 block size ADI iteration による CFD

表 2　NPB Class S

Time in seconds =

# of proc.	IS	MG	LU	CG	SP	BT	EP
1	0.73	25.11	222.68	263.21	391.25	858.18	6558.70
2	0.42	13.00	116.41	132.16	-	-	3305.33
4	0.53	13.51	120.71	146.89	219.89	453.31	3302.08

Mop/s total =

# of proc.	IS	MG	LU	CG	SP	BT	EP
1	0.90	0.30	0.46	0.25	0.25	0.27	0.01
2	1.56	0.58	0.88	0.50	-	-	0.01
4	1.23	0.56	0.85	0.45	0.44	0.50	0.01

Speed-up factor =

# of proc.	IS	MG	LU	CG	SP	BT	EP
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.738	1.932	1.913	1.992	-	-	1.984
4	1.377	1.859	1.845	1.792	1.779	1.893	1.986

アプリケーション

また，それぞれのベンチマークは問題のサイズによって，小さい順に CLASS S, W, A, B, C などに分けられている．

CLASS S について測定した結果を表 2 に，Speed-up Factor をグラフ化したものを図 8 に示す．横軸は並列度 (実行プロセス数)．縦軸が Speed-up Factor(1 プロセスに対する速度向上度) である．

並列度を 2 とした場合，Speed-up Factor がほぼ 2 となっている．これは実装されているプロセッサ数である 2 と同じである．つまり，実装されているプロセッサの処理能力をほぼすべて使って処理がなされているということである．このことから，大きなオーバーヘッド無く，高い効率で並列処理が行なわれていることが分かる．

7. おわりに

本稿では，Linux/M32R のマルチプロセッサ対応と現在の状況について述べた．

M32R 用の Linux に対し同期機構などの SMP をサポートするための機能の実装を行ない，複数のプラットフォームにおいて Linux を SMP で動作させた．また，LMbench，NPB を用いて性能評価を行ない，SMP 動作時でもカーネルの速度低下は少ないこと，大きなオーバーヘッド無く高い効率で並列処理を行なえることを示した．

Linux/M32R はオープンなプロジェクトであり，開発者の自由な参加で開発が行なわれている．今後はさらなる性能の向上を図るとともに，最新バージョンへの対応などのカーネルの開発や，ツールから応用に至るまで幅広い開発を行なっていく予定である．

謝辞　Linux/M32R の開発にあたって，御協力いただいた関係各位に深くお礼を申し上げます．また，独立行政法人 新エネルギー・産業技術総合開発機構 (NEDO) の支援をいただきました．ここに感謝いたします．

文　献

[1] Satoshi Kaneko, et al.: “A 600 MHz single-chip multiprocessor with 4.8 GB/s internal shared pipelined bus and 512 kB internal memory”, ISSCC Digest of Technical Papers, pp. 254–255 (2003).

[2] 高田浩和, 作川守, 坂本圭, 山本整, 稲岡一弘, 近藤弘郁, 清水徹：“Linux を搭載した M32R アーキテクチャ研究開発用プラットフォーム”, Linux Conference 2002 (2002).

[3] “Linux/M32R Home Page”. http://www.linux-m32r.org/.

[4] 山本整, 高田浩和：“シングルチップマルチプロセッサへの Linux の適用”, 情報処理学会第 66 回全国大会講演論文集, 第 1 巻, pp. 7–8 (2004).

[5] Hirokazu Takata, Naoto Sugai, Hitoshi Yamamoto：“Porting Linux to the M32R processor”, Ottawa Linux Symposium (2003).

[6] 東京大学大規模集積システム設計教育研究センタ (VDEC)：“(株) ルネサステクノロジ「M32R ソフトコア」提供プログラム ”. http://www.vdec.u-tokyo.ac.jp/CHIP/M32R/M32R.html.

[7] オークス電子：“OAKS32R”. http://www.oaks-ele.com/oaks32r/oaks32r.htm.

[8] Larry McVoy and Carl Staelin: “LMbench - Tools for Performance Analysis”. http://www.bitmover.com/lmbench/.

[9] NASA Ames Research Center: “The NAS Parallel Benchmarks”. http://www.nas.nasa.gov/Software/NPB/.