計算機アーキテクチャ　144－28
（２００１．　７．２７）

マイクロプロセッサ向け低消費電力アーキテクチャの HDL 設計とその評価

杉谷樹一 [1]　佐藤寿倫 [2,3]　有田五次郎 [2]

[1] 富士通九州ディジタル・テクノロジ(株)
〒812－0011　福岡市博多区博多駅前 3－22－8
E-mail: kiichi@qdt.fujitsu.com
[2] 九州工業大学　情報工学部　知能情報工学科
[3] 九州工業大学　マイクロ化総合技術センター
〒820-8502　飯塚市大字川津 680－4

近年のマイクロプロセッサにおいては，高性能化のためのクロック動作速度の高速化に伴い，その消費電力が増加の一途にある．しかしながら携帯端末用途の組み込みプロセッサにおいては，性能の向上のみならず消費電力の削減も重要である．消費電力の削減を目的としたプロセッサアーキテクチャに関しては，近年様々な研究が行われているが，それらの多くはアーキテクチャレベルでの評価に留まっており，回路レベルの評価まで行われているものは稀である．そこで本研究では，これらの低消費電力アーキテクチャの中からループキャッシュを選択し，実際に HDL 設計を行うことで，その消費電力削減の効果を評価する．

HDL Design and its Evaluation of the Low Power Consumption Architecture for Microprocessors

Kiichi SUGITANI [1]　Tosinori SATO [2,3]　Itsujirou ARITA [2]

[1] Fujitsu Kyushu Digital Technology Limited
22-8, Hakataekimae, 3-chome, Hakata-ku, Fukuoka, 812-0011, Japan
E-mail: kiichi@qdt.fujitsu.com
[2] Department of Artificial Intelligence, Kyushu Institute of Technology
[3] Center for Microelectronic Systems, Kyushu Institute of Technology
680-4, Kawazu, Iizuka, 820-8502, Japan

Current trend of increasing clock frequency on modern microprocessors is also increasing power consumption. In embedded applications, the low power consumption is one of the important design goals for microprocessors as well as high performance. Under this situation, there are many proposals for low power microarchitecture, most of which are evaluated only on architectural design stage. Therefore, we evaluate one of the proposals, loop cache, on circuit design stage by implementing it with hardware description language.

1. 背景

近年のマイクロプロセッサにおいては，性能向上のためのクロック周波数の上昇に伴い, その消費電力は増加の一途をたどる傾向にある[3,9]. しかしながら携帯端末用途に使用される組み込みプロセッサにおいては, プロセッサの性能のみならず消費電力を削減するための考慮も必要である．それは携帯電話やノートパソコンなどにおいて, 連続使用時間の長さとプロセッサの性能とのトレードオフを考えれば容易に推察できる. そのため, 消費電力を削減できるプロセッサアーキテクチャに関する研究が広く行なわれている[3,10]．組み込みプロセッサでは，その消費電力の大部分をキャッシュが占める[5]ことから，

なかでもキャッシュの消費電力削減を目的とした研究が盛んである[1,2,6,7,8].

多くのアプリケーションにおいては，その全実行時間の内のかなりの部分は, 小さな領域にある命令に当てられていることはよく知られている[10]. 命令キャッシュの電力削減を目的とした研究の多くはこの特徴を利用しており，プロセッサ と1 次キャッシュの間に小さな命令バッファを置くことにより, 低消費電力を実現しようというものが多い[2,6]. しかしながらこのアプローチでは, 命令バッファの中に要求された命令が見つからない場合, あらためて1次キャッシュにアクセスする必要があり，サイクルペナルティを負ってしまうことになる.

これらのアプローチに対して，近年提案されたループキャッシュ[7]は上記の問題を解決している．ループキャッシュにおいても，1次キャッシュとは別に小さな命令バッファ(ループキャッシュ)が使用されている．しかし，命令がバッファ内にあることが確実である場合に限りループキャッシュにアクセスするため，ループキャッシュミスによるサイクルペナルティを被らない．したがって効果の高い方式だと言える．

一方，これらの消費電力削減方式の多くは，アーキテクチャレベルにおける評価に留まっており，回路レベルでの評価が行われることは稀である．ループキャッシュにおいても同様で，まだ回路レベルでの評価は行われていない．そこで本研究では，実際にループキャッシュとそれを制御するキャッシュコントローラ回路をVerilog-HDL を用いて設計し，その回路規模と消費電力削減の効果を評価する．

以下，第2章でループキャッシュの詳細を述べる．第3章で使用するプロセッサの仕様と，今回設計した HDL の説明を述べる．第4章で評価環境と測定結果を述べる．最後に第5章でまとめと今後の課題について述べる．

2. ループキャッシュ

この章でループキャッシュ[7]を紹介する．まず初めに電力削減のターゲットとなる命令の説明を行う．続いて，ループキャッシュの構成と動作を説明する．最後にループキャッシュコントローラについて述べる．

2.1 キャッシュ対象となる命令の決定

ループキャッシュでは，命令バッファに保持する領域を小さなループに限定している[7]．ループキャッシュは類似した方式[2,6]とは異なり，0次キャッシュとしてではなく，1次キャッシュと同じレベルに配置されている．したがってこの方式における命令フェッチのデータパスは，1 次キャッシュとループキャッシュとの２つとなる．命令がバッファ内にあることが確実である場合に限りループキャッシュにアクセスすることにより，サイクルペナルティの問題を解消している．プログラム実行時に命令の多くがループキャッシュ内で発見できれば命令フェッチの電力を低下できる．

続いて，ループキャッシュ内に保持すべき命令を如何にして決定するかを説明する．すでに述べたように，キャッシュの対象となる領域は小さなループである．したがって，ジャンプの距離の小さな後方への分岐命令が発見できればよい．このような分岐命令はsbb (short backward branch)と呼ばれており，図1 に示すような命令フォーマットとなる．

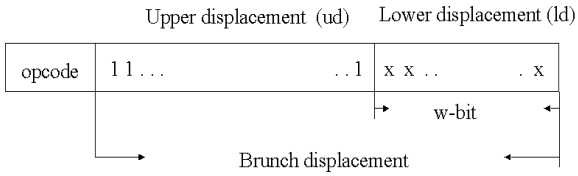

図1　sbb の命令フォーマット

一般に，分岐命令におけるジャンプの距離は2 の補数表現である．したがって sbbの分岐ディスプレースメントの上部(ud: upper displacement)は後方への分岐であるので，すべてがネガティブな分岐ディスプレースメントを指し示している．分岐ディスプレースメントの下部(ld: lower displacement)をw ビットとすると，sbb は2^w命令離れた後方までジャンプすることが許される．言い換えると，ループキャッシュのサイズは2^wワードであり，サイズがそれ以下のループがキャッシュ可能なわけである．デコーダが sbb を検出した場合には，問題のループのサイズはループキャッシュのサイズより大きくないということである．デコーダはsbbを検出するとキャッシュコントローラに通知を行い，ループキャッシュを利用するための制御が始まる．この状態遷移を引き起こした sbb をtriggering sbb と名付ける．

2.2 ループキャッシュの構成

図2 は2^w個のエントリを持ったループキャッシュの構成である．同時に，ループキャッシュがどのようにして命令アドレスA[31 :0]でアクセスされているかも示している．ジャンプ先命令アドレスの下位ビットであるindex (A) フィールドを使ってインデックスされる．ループキャッシュに保持される領域は，ループを構成する連続した命令列である．したがって，ループキャッシュにはタグアレイは必要なく，

またダイレクトマッピングでインプリメントされるべきである．言い換えると，プログラム実行中のループキャッシュのアクセスは単一で競合が無い．なぜならループ中の各命令は，常にループキャッシュの中で唯一な場所に割りつけられること．そして特定のキャッシュの位置に対し，競合を与えるような命令が一つも無いからである．

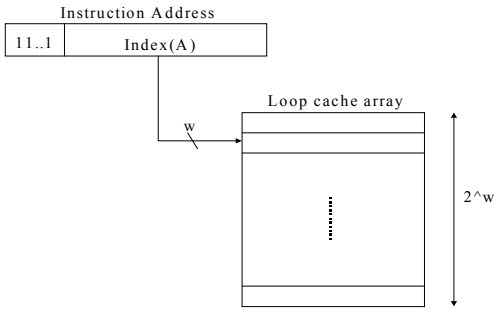

図2　ループキャッシュの構成

2.3 ループキャッシュのヒット/ミスの決定

次にフェッチされる命令がループキャッシュの中に存在するかどうかを決定するためには，サイクル毎に以下の情報が必要となる．

(a) 次の命令フェッチが逐次的なフェッチなのか，それとも命令流に変化(以後，cof: change of control flow)があるのか．
(b) もしcofがあるならば，それはtriggering sbb によるものかどうか．
(c) 1次キャッシュの代わりにループキャッシュをアクセスするための準備は完了しているかどうか．

(a) はオペコードにより容易に調べることができる．(b)はsbbが成立したかどうかを調べれば判る．しかしながら(c)を知るためには，新たなキャッシュコントローラが必要となってくる．

2.4 ループキャッシュコントローラ

図3にループキャッシュコントローラの状態遷移図を示す．図3 に示されている3状態は以下のとおりである．IDLEは，一次キャッシュから命令をフェッチし，ループキャッシュはオフになっている 状態である．FILLは，一次キャッシュから命令をフェッチすると同時に，その命令をループキャッシュにストアしている 状態である．最後にACTIVEは，一次キャッシュはオフにして，ループキャッシュから命令をフェッチしている 状態である．

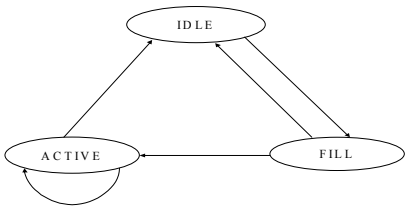

図3　ループキャッシュコントローラ

3状態の間の遷移は以下のように決定される．初期状態は IDLE である．デコードされた命令が sbb であり，しかもその分岐が成立すると，状態はFILLへ移る．このときのsbb は triggering sbbである．

FILL 状態では，ループキャッシュコントローラは，1 次キャッシュからフェッチされている命令を，同時にループキャッシュに保持する．この間は状態を移動しない．フェッチしている命令のプログラムカウンタと triggering sbb のプログラムカウンタが一致すると，ループキャッシュコントローラはループの全命令を保持できたと理解する．このとき，もし triggering sbb の分岐が成立ですればACTIVE 状態に戻る．不成立の場合には IDLE 状態へと移る．また，FILLの状態にあるときに，もし triggering sbb が原因でない cof が起こった場合は，コントローラはIDLE状態に戻る．

ACTIVE 状態のとき，命令はループキャッシュからフェッチされる．またこの状態の間に triggering sbb の分岐が不成立だったときや triggering sbb が原因でない cof があったときには，IDLE 状態へと戻る．

以上の説明にしたがって，バッファ中に命令が存在するときに限り，ループキャッシュを参照可能になる．

3. HDL 設計

この章ではまず今回使用するプロセッサを，続いてループキャッシュの実装について述べる．

3.1 プロセッサの仕様

本研究で用いたプロセッサは，三菱電気株式会社のM32R[13,14]である．M32Rはその応用分野として，PDA，CATV端末，ディジタル衛星放送端末，インターネット端末，カーナビゲーション機器などという組み込み用途を想定しており，低消費電力アーキテクチャの評価という点で適している．命令セットアーキテクチャは32ビットRISCアーキテクチャを採用している．但し，コードサイズ削減の目的で，32ビット命令の他に16ビット命令が用意されている．

本研究の対象となる分岐命令においても同様である．また，分岐命令の実行には2サイクルの遅延が存在する．

M32RのHDLは，三菱電気株式会社から提供された完全論理合成可能なVerilog-HDL記述を使用した．

3.2 ループキャッシュの設計

本節で，ループキャッシュとキャッシュコントローラの設計に関して述べる．

・ループキャッシュ

ループキャッシュには以下の入出力信号線がある．

LOOP A	アドレスの入力信号
LOOP DI	ストアするデータの入力信号
LOOP DO	ロードされるデータの出力信
LOOP WN	書き込みの制御信号
LOOP CS	ループキャッシュの制御信号
CLK	クロック

信号の意味は以下のとおりである．LOOP CS が0のときにループキャッシュはアクティブになる．そしてLOOP WN が0 のときにはLOOP A のアドレスにLOOP DI がストアされ，1 のときにはLOOP A のアドレスに格納されているデータをLOOP DO にロードされる．

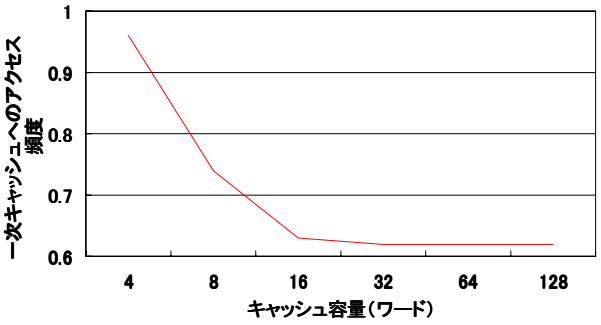

図4　一次キャッシュへのアクセス頻度

ループキャッシュの容量は以下のようにして決定した．図4は，ループキャッシュの容量を変化させた時の，1次キャッシュへのアクセス頻度を表している[7]．これは文献[7]で使用されている15本のアプリケーションの平均値である．図4からループキャッシュの容量が 32 ワードを越えると一次キャッシュへのアクセス頻度は減少していないことが見てとれ，32ワードの容量があれば十分であることが判る．しかしながら既に述べたように，パイプライン動作のため，分岐命令がフェッチされて実行されるまでには2サイクルを必要とする．その間にフェッチされた命令を破棄しないで保持しておくためには，ループキャッシュに追加の2ワードが必要となる．このためと設計を容易にするためから，ループキャッシュの容量は 64 ワードとした．

・ループキャッシュコントローラ

ループキャッシュコントローラには以下の入出力信号がある．

CODE	オペコードの入力信号
LOOP ADR	ロードするアドレス信号
LOOP WN	ループキャッシュへの書き込み制御信号
LOOP CS	ループキャッシュ制御信号
STATUS	ループキャッシュの状態信号

またレジスタには以下を定義する．

STATUS	ループキャッシュの状態
cof PC	cofが起こったときのプログラムカウンタ
LOOP WN	ループキャッシュへの書き込み制御
LOOP CS	ループキャッシュへのキャッシュ制御
LOOP ADR	ロードするアドレス信号

STATUS により，IDLE， FILL， ACTIVE の3 状態が判る．また分岐命令は3 種類に分けられ，オペコード(CODE)で識別する．その種類別にジャンプ先までの距離を決定する．今回の sbb のld フィールドの設定は5 ビットである．sbb の分岐が成立か不成立かはデコードステージでは分からず実行ステージでしか分からないので，とりあえずこの sbb は一時的なレジスタ rsbb に保持しておく．

IDLE 状態では rsbb と分岐が成立 だったときにFILL状態へと状態を移す．ここまではLOOP WN, LOOP CS ともに 0 である．この際 sbb の PC 値をcofPC として，そしてLOOP ADR を 0 にしておく．

FILL 状態ではまずその命令自身の PC がcofPC ではないときにループキャッシュへのアドレスを+1 して次のアドレスに書き込むようにする．またそうでなく更に 分岐が成立したときは，ACTIVE 状態へと移る．この際ループキャッシュへのアドレスをリセットしておく．またこの状態においては LOOP WN, LOOP CS は 1 である．上記以外は IDLE へと戻る．

ACTIVE 状態では cof でないときと，その命令自身の PC が cofPC ではないときに，ループキャッシュへのアドレスを+1 して次のアドレスを読むようにする．またそうでなく更に 分岐が成立したときはループキャッシュへのアドレスをリセットし，もう一度ループの命令流を再読み込みさせる．不成立のときにはIDLE 状態へと戻る．cof が起こった場合にも

IDLE 状態へと戻る．この状態でのLOOP WN は0, LOOP CS は1 である．

4. 評価

4.1 評価環境

ループキャッシュとループキャッシュコントローラは，Verilog-HDL を用いて記述した．動作の確認には，Cadence社のVerilog-XL を用いた．また論理合成にはSynipsys社のDesignCompiler[11,12]を，そのためのライブラリにはlsi_10_k を用いた．そしてキャッシュの電力決定には Wattchツール[4]で提供されているモデルを用いた．

4.2 結果

ループキャッシュの追加前と追加後のHDL記述を用いて論理合成を行い，DesignCompiler の report_areaコマンドを用いて回路規模を測定した．このコマンドによって得られる回路規模はゲートである．キャッシュコントロール部とプロセッサコア全体の回路規模を，それぞれ図5，図6に示す．

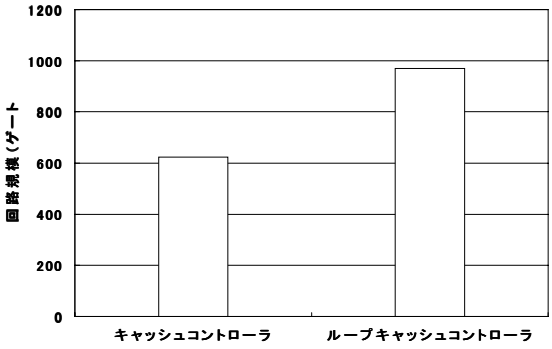

図5　コントローラ部の回路規模比較

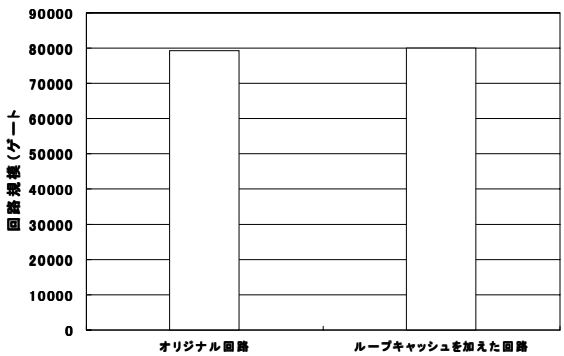

図6　プロセッサコア全体の回路規模比較

図5の結果からループキャッシュに付加した回路の増加量は約300 ゲートであることが判る．一方でプロセッサコア全体では，図6 から回路規模の増加は約800 ゲートに達していることが判る．この差は，ループキャッシュのデータパスの制御のため，制御パス部に手を加えたことによるものであると考えられる．これらの図に示される結果から，ループキャッシュ追加前後のプロセッサコア回路規模は，せいぜい約1%の増加に過ぎないことが判明した．

続いて Wattch [4]で提供されているモデルを用いて，一次キャッシュとループキャッシュで消費される電力を求めた．この結果と文献[13]に開示されている1次キャッシュの消費電力を用い，スケーリング則を施して決定された消費電力を図7に示す．

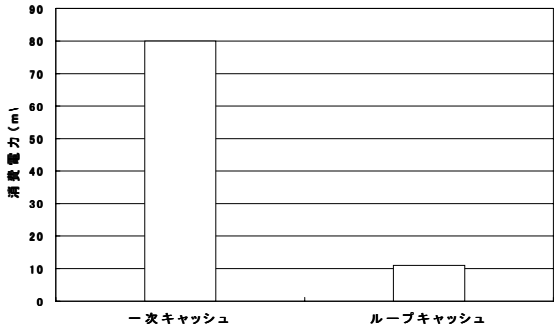

図7　キャッシュの消費電力

一次キャッシュの消費電力80mW[13]に対して，ループキャッシュの消費電力は11mWである．図4よりループキャッシュの容量が 64 ワードのときは一次キャッシュへのアクセス頻度は 62.1%であるので，一次キャッシュとループキャッシュを併用した際の消費電力を

(80mW * 62.1%) + (11mW *37.9%) = 53.8mW

と見積もることができる．ループキャッシュの追加前後で，アプリケーションの実行サイクル時間には変化は無いことに注意されたい．この結果からキャッシュ部ではループキャッシュを使った場合は使わなかった場合に比べ

80mW － 53.8mW = 26.2mW

の消費電力削減となっている．これは約30%の電力削減に相当する．

ここで プロセッサコア部の回路規模の増加は約1%に過ぎなかった先ほどの結果より，その回路規模が変化していないことから，消費電力も120mW[13]のままで変化しないとみなすことができる．よってキャッシュ部のみの消費電力の比較でループキャッ

シュのアーキテクチャが有効であるかどうかを評価できる．そして今回，キャッシュ部が約30%電力削減されているという結果より，ループキャッシュとそのコントローラの回路の追加を考慮してもそのアーキテクチャが有効なものであると言える．このときプロセッサ全体では13.1%の電力削減になる．命令キャッシュの消費電力削減が，プロセッサ全体に対しても有効であることが確認できる．

5. まとめ

近年，低消費電力の研究が広く行われている．それらの研究の大半は CPU と一次キャッシュの間に小さな命令バッファを置くことにより，消費電力の低下を実現しようというものが多い．近年提案されたループキャッシュは，やはり小さな命令バッファを使ったものであるが，バッファに保持する領域を小さなループに限定している．ループキャッシュはタグを必要としないダイレクトマッピングで実現しているので，さらに一次キャッシュより低電力化が可能となる．残念ながらループキャッシュは，まだ回路レベルでの検証が行なわれていない．そこで本研究では，実際にループキャッシュとそれを制御するコントローラの回路をHDL を用いて設計し，その回路規模の調査と消費電力削減の効果を評価した．その結果，約30%の電力削減がループキャッシュの追加とプロセッサコアの回路規模を約 1% 増加することにより実現できることが分かった．これによりループキャッシュを用いることは低消費電力に有効な手段であると言える．

本研究に関係する将来の課題としては，今回評価できなかった他のアプローチ[1,2,6,8]を実装する必要があるかも知れない．さらにこれらの経験を元にして，独自の消費電力削減アーキテクチャの提案を行っていきたいと考えている．

謝辞

本研究は，筆頭著者が九州工業大学知能情報工学科在籍時に実施されたものです．

本研究を進めるにあたりお世話になった以下の方々に感謝致します．三菱電機株式会社の清水徹博士，奥田亮輔博士，安藤智子氏は，M32 プロセッサの設計データを提供して下さりました．九州工業大学マイクロ化総合技術センターの田中康一郎助手，九州工業大学大学院情報工学研究科(当時)の大濱智宏氏は，設計・評価環境に関して多くの助言を下さりました．九州工業大学知能情報工学科の有田・佐藤研のメンバーには，多くの励ましを戴きました．

参考文献

[1] G.Albera，R.Iris Bahar，“Power/Performance Advantages of Victim Buffer in High-Performance Processors ”IEEE Volta International Workshop on Low Power Design，1999.

[2] N.Bellas，I.Hajj, C.Polychronopoulos，“A New Scheme for I-Cache Energy Reduction in High-Performance Processors”，Power Driven Micro-architecture Workshop held in conjunction with 25th International Symposium on Computer Architecture, 1998.

[3] D.Brooks, P.Bose, S.E.Schster, H.Jacobson, P.N.Kudva, A.Buyuktosunoglu, J-D.Wellman, V.Zyuban, M.Gupta, P.W.Cook, “Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors”, IEEE Micro, vol.20, no.6, 2000.

[4] D.Brooks，V.Tiwari，M.Martonosi，“Wattch: A Framewark for Architectural-Level Power Analysis and Optimizations”，27th International Symposium on Computer Architecture, 2000.

[5] T.Ishihara, K.Asada, “A System Level Power Optimization Technique Using Multiple Supply and Threshold Voltages”, Asia and South Pacific Design Automation Conference, 2001.

[6] J.Kin，M.Gupta, W.H.Mangione-Smith，“The Filter Cache: An Energy Efficient Memory Structure”，30th International Symposium on Microarchitecture, 1997.

[7] L.H.Lee，B.Moyer，J.Arends，“Instruction Fetch Energy Reduction Using Loop Caches for Embedded Application with Small Tight Loops”，International Symposium on Low Power Electronics and Design，1999.

[8] L.H.Lee，J.Scott，B.Moyer, J.Arends，“Low-Cost Branch Folding for Embedded Applications with Small Tight Loops”，32nd International Symposium on Micro-architecture, 1999.

[9] T.Mudge, “Power: A First-Class Architectural Design Constraint”, IEEE Computer, vol.34, no.4, 2001.

[10] K.Murakami, H.Magoshi, “Trends in High-Performance, Low-Power Processor Architectures”, IEICE Transactions on Electronics, vol.E84-C, no.2, 2001.

[11] Synopsys Inc., Design Compiler チュートリアル日本語版, 1998.

[12] Synopsys Inc., DesignCompiler Fundamentals日本語版, 1998.

[13] 澤井克典, 佐藤貢, 岩田俊一, 奥村直人, 那須隆, “16Mと32ビット・プロセッサを集積，コストを上げずに性能は1.6倍に”, 日経マイクロデバイス, no.129, 1996.

[14] 三菱電機株式会社，“M32R ファミリソフトウェアマニュアル”，2000.