Vol.2009-ARC-185 No.5
2009/10/26

ループアドレスレジスタを用いた命令キャッシュ機構

伊藤剛[†]　多田十兵衛[†]　後藤源助[†]

本研究では、ループに着目したキャッシュ機構について述べる。プロセッサとメインメモリのアクセス速度の差を解消するためにキャッシュが用いられている。キャッシュはサイズの増加に伴い性能も向上するが、同時に消費電力も増加してしまう。そこでループのみを専用キャッシュに格納して、キャッシュサイズを削減させる。ループを動的に検出する方法としてループアドレスレジスタ（LAR）がある。LAR は分岐命令、およびジャンプ命令のターゲットアドレスを格納する。本研究では LAR に格納できるターゲットアドレス数と同数のループキャッシュを導入する手法を提案する。アーキテクチャレベルのシミュレータによる検証の結果、ループキャッシュを分割した場合、1 つのループキャッシュを用いた従来手法よりもキャッシュサイズを 1KB 少なくしたうえで、IPC が約 50%向上した事が示された。

Instruction Cache Mechanism with Loop Address Register

Takeshi Ito[†]　Jubee Tada[†]　and　Gensuke Goto[†]

In this study, we speak the cache mechanism that paid its attention to a loop. Cache is used to cancel a processor and a difference of the access speed of the main memory. As for the cache, the performance improves with increase of the size, too, but the power increases at the same time, too. Therefore we store away only a loop to exclusive cache and let you reduce a cache size. There is loop address register (LAR) as a method to detect a loop for motion. LAR stores away a divergence order and the target address of the jump order. we suggest technique to introduce the loop cache of the number of the target addresses and the same number that LAR can store away into in this study. It was shown that about 50% IPC improved after having reduced a cache size 1 kbyte than the conventional technique how we used one loop cache for when we divided loop cache as a result of inspection by the simulator of the architecture level.

1. はじめに

近年、半導体技術の向上によりプロセッサの速度は急速に向上しているが、メモリアクセス速度の改善速度は遅い。よって、プロセッサの処理速度とメインメモリのアクセス速度の差が増大しており、プロセッサの性能を制約する要因として記憶システムの比重が高まっている。そこで、プロセッサとメインメモリの速度差を埋めるために高速小容量メモリをキャッシュとして使用する。キャッシュのヒット率が高くなると、プロセッサはメモリアクセスの時間が大幅に短縮され、性能が向上する。従来はキャッシュサイズを大きくすることでヒット率を向上させてきたが、キャッシュサイズを大きくするとハードウェア量の増加や消費電力の増加などデメリットも多い。

キャッシュサイズを削減するには二つの提案がなされている。一つ目は動的にループだけを検知、ロードしてループからいつ抜け出すかを検知する動的タグレスループキャッシュ法[1]である。単純なコントローラを使用し、キャッシュからデータを読み出すときにタグ比較の必要をなくしている。このような動的にロードしたループキャッシュ法は設計者にはわかりやすいが、分岐を取らないかサブルーチンコールのないループのサポートに制限されている。二つ目は最も頻繁に実行されるループがループキャッシュへプリロードされているかもしれないことを利用するプリロードループキャッシュ法[2]である。プリロードされたループキャッシュ法はより複雑なループをサポートすることができ、より大きな省電力を提供するが、どれだけのループがプリロードできるかには限界がある。

本研究では、ループアドレスレジスタ（LAR）を用いた命令キャッシュ高性能化手法[2]について述べる。ループキャッシュ、LAR、ループマッチメモリを導入することで、ループ部分をキャッシュに格納する方法である。従来では、LARに格納できるターゲットアドレスの数に関係なく、ループキャッシュは一つであった。そこで提案手法では、LARに格納できるターゲットアドレスの数と同数のループキャッシュを導入して、それぞれのターゲットアドレスに対応したループキャッシュへアクセスさせる。これにより個々のループキャッシュのサイズ変更を可能にする。以上のことをアーキテクチャレベルのシミュレータを用いて検証し、キャッシュサイズを削減しつつプロセッサの性能を比較する。

[†] 山形大学大学院理工学研究科 情報科学専攻
Graduate School of Science and Engineering, Yamagata University

ⓒ2009 Information Processing Society of Japan

Vol.2009-ARC-185 No.5
2009/10/26

2. 関連技術

2.1 クリティカル領域

プロセッサの性能を向上させる基本的な規則がある。実行時間の約 90%は最初の 2～4 つの最も頻繁に実行されたループに費やされている[3]。この頻繁に実行されたループはクリティカル領域であり、クリティカル領域の検出はプロセッサの性能を向上する上で非常に重要な要素となる。クリティカル領域はプログラム中のループ、動的に実行されたサブルーチンである。このクリティカル領域をキャッシュに格納することでプロセッサに高速に命令を送ることができ、性能向上につながる。このループに着目したキャッシュとしては、以下で述べる動的タグレスループキャッシュ、プリロードループキャッシュ、および本研究で用いているハイブリッドループキャッシュが挙げられる。

2.2 動的タグレスループキャッシュ[1]

動的にロードしたループはループの開始アドレスにジャンプする命令によって形成される。これは短い後方への分岐命令を意味し、短い後方への分岐命令はプログラムカウンタ相対分岐命令である。短い後方への分岐命令はループキャッシュコントローラを活性化させ、命令をループキャッシュに格納する。ループ反復中の命令はプロセッサとループキャッシュの両方に供給される。また、ループがループキャッシュに完全に入らない場合、最初の命令だけが格納される。

動的にロードしたループキャッシュはミスを受けず、実行オーバーヘッドがない。それはタグ比較を含んでなく､結果として 1 つのアクセスの電力を消費しなくてすむ。

2.3 プリロードループキャッシュ[2]

プリロードループキャッシュは単に別のレベルのキャッシュではなく、頻繁なループをキャッシュから削除されないように使用するテーブルのようなものである。プリロードループキャッシュはキャッシュがマイクロプロセッサの非常に近くに置かれる。そのため、非常に速いアクセス時間を実現することができる。また、小型化したプリロードされたループキャッシュは非常に小さいフェッチ電力をもたらし、命令メモリアクセスの 70%の電力を削減する。

また、各クリティカル領域の開始と終了のアドレスを維持し、キャッシュ中の命令の開始位置を保持することにより、領域のどんな数も格納することが可能である。動的タグレスループキャッシュのようにプリロードキャッシュはタグが必要なく、マイクロプロセッサに統合され、低電力命令フェッチを提供する。システムリセット中に実行することにより、あらかじめループキャッシュをロードする方法も可能である。

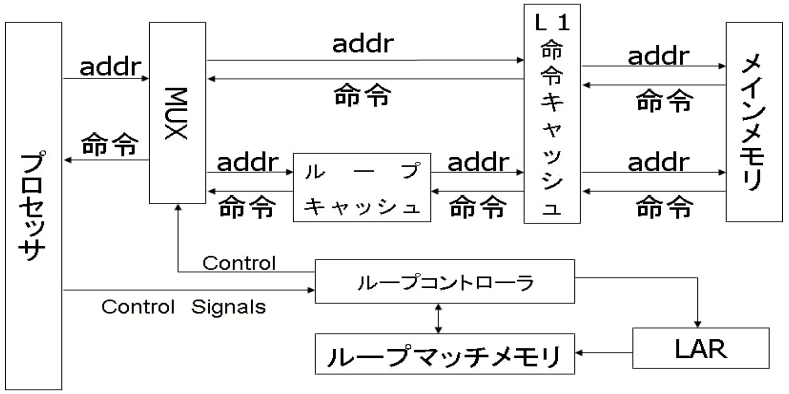

図 1 LAR を用いた命令キャッシュアーキテクチャ
Figure 1 Instruction cache architecture that uses LAR

3. LAR を用いた命令キャッシュ機構

3.1 LAR を用いた命令供給機構[1]

LAR を用いたキャッシュは動的に動作させるか、プリロードされた記憶装置の別レベルから動作させる。主なループキャッシュはマイクロプロセッサに統合されるため、より小さくする必要がある。一方で、プリロードされた記憶装置はアクセスが稀少になり、重要な電力にはなりえないため、サイズ制限は考慮しない。また、LAR を用いたキャッシュの重要な特徴は設計者があらかじめアプリケーションにおいて余分な分析ステップを取りたくない場合、動的に動作するようにのみ設定できることである。

3.2 アーキテクチャ

LAR を用いた命令供給機構のアーキテクチャを図 1 に示す。主な構成は 2 つの命令キャッシュ、ループコントローラ、LAR、ループマッチメモリからなる。ループキャッシュはループ内命令を格納する専用のキャッシュである。ループキャッシュを用いることにより、頻繁に実行されるコードがキャッシュに残される可能性が増加し、キャッシュサイズの削減が可能となる。ループコントローラは分岐命令・ジャンプ命令が実行されるときに動作し、MUX、LAR、ループマッチメモリを制御する。LAR はジャンプ命令・分岐命令のターゲットアドレスを保存する。LAR を用いることで、後方へジャンプする命令のみならず、前方へジャンプし、その先でループとなる場合も

ⓒ2009 Information Processing Society of Japan

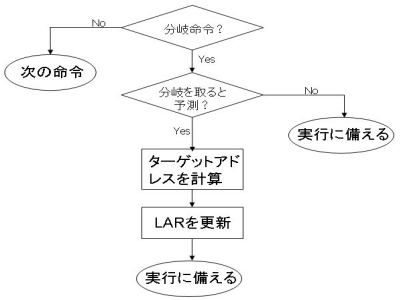

図 2　分岐予測時フローチャート
figure 2　Flow chart at branch prediction

対応することができる。ループマッチメモリは LAR にターゲットアドレスが保存されているかの判断を行う。

図 2 に分岐予測時のフローチャートを示す。分岐を取ると予測した場合、コントローラは LAR に予測したターゲットアドレスを格納し、実行に備える。分岐を取らないと予測した場合は LAR にターゲットアドレスを保存しないで実行に備える。

図 3 に実行時のフローチャートを示す。分岐命令・ジャンプ命令以外が実行されるときは直前の命令がアクセスしたキャッシュにアクセスする。ジャンプ命令・分岐命令が実行される場合、プロセッサはループコントローラにプログラムカウンタ（PC）とネクストプログラムカウンタ（NPC）、オペコードを送る。ループコントローラはターゲットアドレスを計算し、ループマッチメモリにターゲットアドレスと LAR に目的とするアドレスが格納されているかチェックを行わせる。LAR に目的とするアドレスがあった場合、ループコントローラはループ内命令と判断し、MUX を制御、LAR を更新し、次の命令からループキャッシュにアクセスさせる。ループキャッシュでキャッシュミスした場合は L1 命令キャッシュにアクセスする。LAR に目的とするターゲットアドレスが保存されていない場合、ループコントローラは L1 命令キャッシュにアクセスさせ、LAR を更新する。

3.3 LAR 置換法

LAR は保存するターゲットアドレスの個数を設定することができ、ターゲットアドレスを格納する際に、どれを置換するかを設定する必要がある。そこで提案手法は LRU 法（Least Recently Used）、FIFO 法（First-In First-Out）、頻度カウンタを使用する方法の三種類の置換法が設定可能である。頻度カウンタは何回実行されたらループと判断させるか、2 番目に少ないものを置換するなど様々な設定が可能であるが、今回

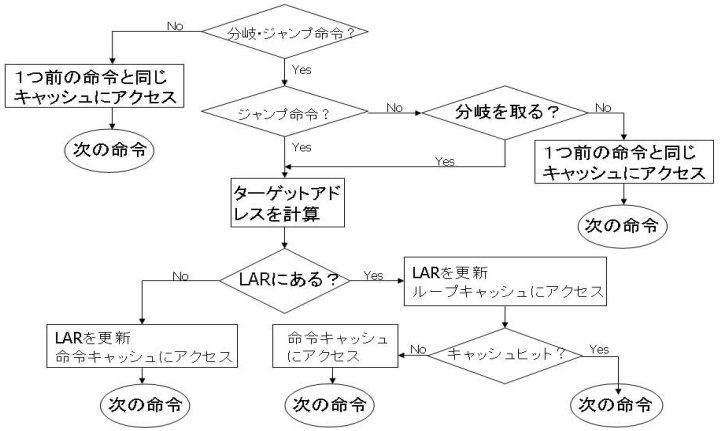

図 3　実行時のフローチャート
figure 3　Flow chart when executing

の実験では実行された回数のもっとも少ないものを置換する。頻度カウンタを使用する手法は参照された回数を保存する必要があり、LAR の数が増えた場合でも頻度カウンタの個数を増やすだけでいいので比較的容易に実現が可能である。しかし、頻度カウンタを使用する手法は時間的局所性が利用できない欠点がある。

3.4 ループキャッシュの分割

本研究では、ループキャッシュの分割手法を提案する。前述したように、LAR はターゲットの保存数を設定することができる。そこで、保存できるターゲットアドレスと同数のループキャッシュを導入する。図 4 は LAR に保存できるターゲットアドレスの数を 4 とした場合の提案手法アーキテクチャの一部である。ループマッチメモリがチェックを行い、LAR に目的とするアドレスがあった場合、ループコントローラはループ内命令と判断する。ループキャッシュにアクセスする際は、4 つのターゲットアドレスはそれぞれのループキャッシュへと対応している。これにより個々のループキャッシュのサイズ変更を可能にする。また、ループキャッシュの個数を増やしたので、それぞれのループキャッシュサイズも必然的に小さくする必要がある。従来手法で使用するループキャッシュが 1KB である場合、提案手法は、ループキャッシュが 4 つあるため、例えば 128byte、128byte、256byte、512byte と合計を 1KB 以下にしなくてはならない。ただし、今回の実験では、ループキャッシュサイズをすべて同じ値にして行うため、従来手法のループキャッシュが 1KB、512byte のときは、提案手法のループキャッシュサイズはそれぞれ、各 256byte、各 128byte としている。

ⓒ2009 Information Processing Society of Japan

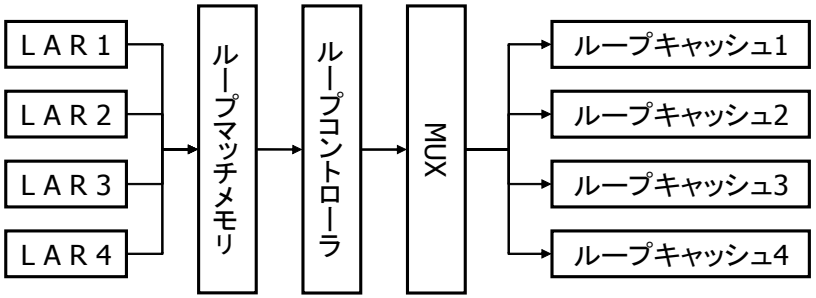

図 4　ループキャッシュ分割時のキャッシュアクセス
figure 4　Cache access at the time of the loop cache division

表 1　頻度カウンタ置換法における IPC
table 1　IPC in frequency counter replacement method

頻度カウンタ	LARSIZE2	LARSIZE4	LARSIZE8
ammp	0.211	0.211	0.211
equake	0.264	0.264	0.264
gcc00	0.289	0.289	0.289
mesa	1.437	1.437	1.437

表 2　FIFO 法における IPC
table 2　IPC in First-In First-Out method

FIFO	LARSIZE2	LARSIZE4	LARSIZE8
ammp	0.195	0.194	0.194
equake	0.239	0.243	0.243
gcc00	0.288	0.288	0.289
mesa	0.959	0.960	0.960

表 3　LRU 法における IPC
table 3　IPC in Least Recently Used method

LRU	LARSIZE2	LARSIZE4	LARSIZE8
ammp	0.194	0.194	0.193
equake	0.243	0.243	0.243
gcc00	0.283	0.282	0.281
mesa	0.959	0.960	0.960

4.　実験

4.1　実験環境

LARのサイズと置換法の検討を行い、その後それぞれの手法のIPCを測定し、性能の評価を行う。以上のことをプロセッサアーキテクチャの研究分野において広く使用されているSimpleScalar Tool Set version3.0[4][5][6]を用いて検証する。

シミュレータは SimpleScalar Tool Set に含まれる sim-outorder を用いる。sim-outorder はパイプライン機構や分岐予測、アウトオブオーダー方式を採用しており、4 命令同時発行/実行可能である。命令セットは DLX 命令セットを拡張した SimpleScalar 独自の PISA 命令セットを使用する。PISA 命令セットの命令長は 64bit である。Bimodal PredictorはBTB（Branch Target Buffer）と呼ばれる 2 ビットのテーブルを使用し、予測を行う。分岐予測ミスのペナルティは 3 サイクルとなっている。連想度は 1 としている。テストベンチはCPUでの標準的な性能指標となったSPEC95、SPEC2000[7]を用いている。

4.2　LAR およびキャッシュサイズの検討

表 1、表 2、表 3 は従来手法において、L1 命令キャッシュ、ループキャッシュをそれぞれ 1KB としたときの LAR サイズと置換手法の IPC への影響を示す。IPC の変化はほとんどみられなかったが、LAR サイズが 4 の構成が最も高い。よって LAR のサイズは 4 とする。

図 5 に分岐予測を用いない従来手法の命令キャッシュおよびループキャッシュの各ヒット率を示す。ループキャッシュは 1KB でもほぼ 100%となる。一方、L1 命令キャッシュは 1KB でのヒット率は低く、IPC を向上させるには 2KB 以上必要である。従って、以降の実験では、sim-outorder（デフォルト）の命令キャッシュ構成は 1KB、2KB、4KB とする。従来手法の各キャッシュ構成は L1 命令キャッシュを 1KB、2KB、ループキャッシュを 1KB とする。提案手法の各キャッシュ構成は L1 命令キャッシュを 1KB、2KB、ループキャッシュを各 256byte、各 128byte とする。

ⓒ2009 Information Processing Society of Japan

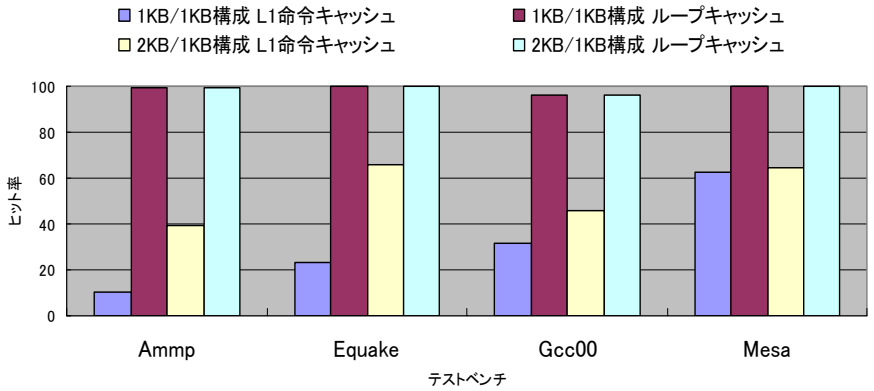

図 5 命令キャッシュおよびループキャッシュのヒット率
figure 5 A hit rate of a instruction cache and the loop cache

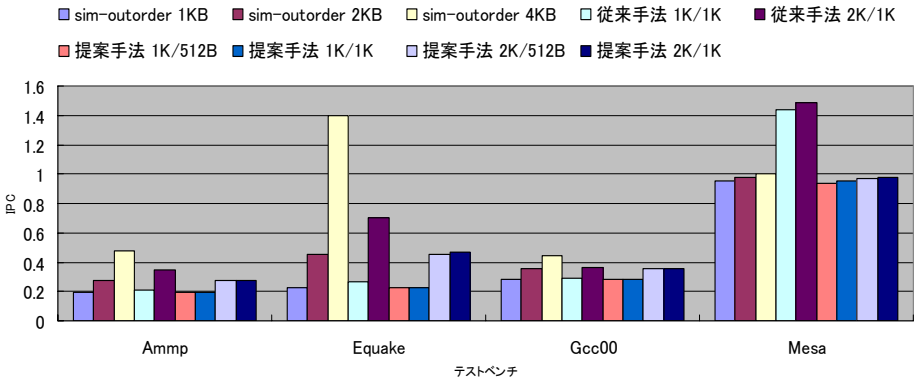

図 6 頻度カウンタ置換法を用いた各テストベンチの IPC
figure 6 IPC of each test bench where used frequency counter replacement method

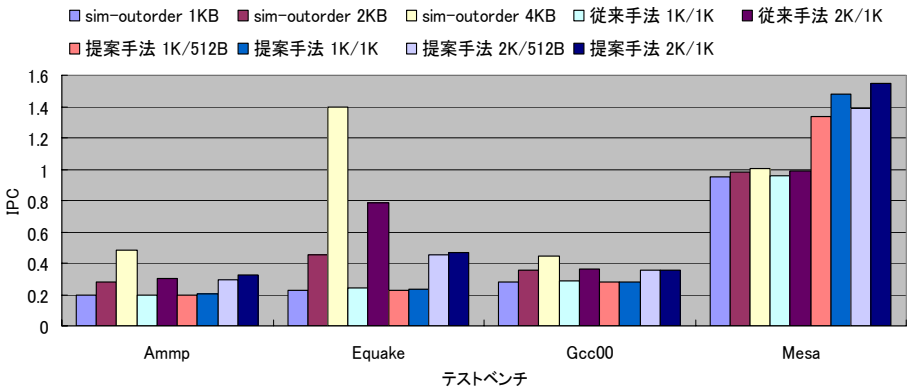

図 7 FIFO 置換法を用いた各テストベンチの IPC
figure 7 IPC of each test bench where used First-In First-Out replacement method

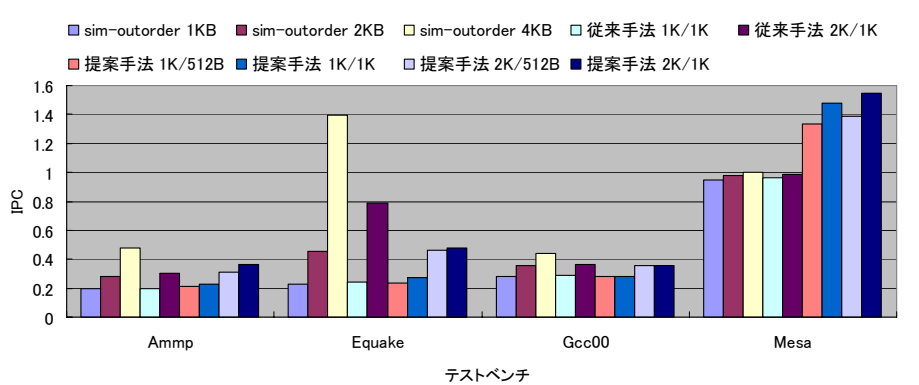

図 8 LRU 置換法を用いた各テストベンチの IPC
figure 8 IPC of each test bench where used Least recently Used replacement method

4.3 結果

図 6、図 7、図 8 に LAR の三種類の置換法を用いたときの各テストベンチにおける IPC を示す。Ammp、Equake および gcc00 テストベンチにおいて、どの置換法でもキャッシュ構成が同じであれば IPC はほぼ変化していない。Equake テストベンチにおいて、提案手法は L1 命令キャッシュを増加すると、sim-outorder の 4KB の構成、および従来手法の 2K/1K 構成に大きく差をつけられている。その差は、従来手法とでは約二倍、sim-outorder とでは約三倍にもなる。
提案手法の場合に、ループ内命令と判断されループキャッシュに格納されるものの、キャッシュサイズが 256byte では命令を格納しきれずキャッシュミスとなることが原因で、従来手法の場合よりも IPC が低下してしまったと思われる。Mesa は提案手法に

ⓒ2009 Information Processing Society of Japan

おいて IPC がもっとも向上したテストベンチであり、キャッシュサイズを削減しつつ IPC 向上が実現できている。FIFO 置換法と LRU 置換法を用いた場合は、従来手法の IPC は sim-outorder と比べほとんど差がないが、提案手法においてはどの構成であっても sim-outorder および従来手法の IPC を上回っている。しかし、置換法に頻度カウンタを用いた場合には、従来手法の手法では IPC が向上しているのに対し、提案手法では IPC の向上は見受けられない。

5. おわりに

本研究では LAR を用いた命令供給機構について述べた。提案手法では、LAR に格納できるターゲットアドレスと同数のループキャッシュを導入したうえで、アーキテクチャレベルのシミュレータとして広く使用されている SimpleScalar Tool Set を使用して、キャッシュサイズを削減しつつプロセッサの IPC 向上を図った。

結果から、LAR を用いた命令供給機構はキャッシュサイズを削減しつつ、IPC が向上する。Mesa テストベンチにおいて、提案手法では、LRU 置換法および FIFO 置換法を用いた際に、sim-outorder および従来手法よりも少ないキャッシュ構成で IPC が最大 50%向上している。また、ループキャッシュを導入することは IPC を向上させるために有用な手法であることもわかった。

今後の課題として、より最適な構成にするための各ループキャッシュサイズの変更、実際の消費電力やハードウェア量なども考慮する必要がある。

参考文献

1) Ann Gordon-Ross and Susan Cotterell and Frank Vahid “Exploiting Fixed Programs in Embedded Systems:AloopCache Example.” IEEE Computer Architecture Letters January 2002
2) Ann Gordon-Ross and Frank Vahid “Dynamic Loop Caching Meets Preloaded Loop Caching –A Hybrid Approach” Proc IEEE Internal Conference on Computer Design (ICCD’02) pp 446 September 2002
3) Ann Gordon-Rossand Frank Vahid “Frequent Loop Detection Using Efficient Nonintrusive On-Chip Hardware” IEEE Transactions On Computers, VOL.54 NO.10,October 2005.
4) SimpleScalar LLC , http://www.simplescalar.com/
5) VDEC監修/浅田邦博・藤田昌宏共編,”システムLSI設計自動化技術の基礎 – パブリックドメインツールの利用法”,pp9-18,培風館,2005,December
6) Doug Burger and Todd M. Austin, The SimpleScalar Tool Set Version2.0users_guide_v2 (SimpleScalar version 2.0 user’s guide)
7) Starndard Performance Evaluation Corporation ,http://www.spec.org/

ⓒ2009 Information Processing Society of Japan