# 組込みプロセッサのための超高速なオンチップメモリ最適化技術

代表研究者　　戸　川　　望　　早稲田大学理工学術院　教授

## 1 まえがき

近年，CPU 等の処理装置の高速化により，メモリ等の記憶装置との性能差が拡大し，問題となっている．これを解決するために用いられる手法の 1 つがキャッシュあるいはキャッシュメモリである．しかし，対象となるアプリケーションに対して，キャッシュ容量が小さすぎるなどキャッシュ構成が適合しない場合はキャッシュミスが頻発し，十分に効果を発揮しないこともあるため，こうした問題を考慮したキャッシュ構成を選択することが求められている．

組み込みシステムの場合，システム上で動かすアプリケーションは少数であり，これらさえ高速に実行できれば，他のアプリケーションの実行が遅くても問題ない．よって，キャッシュ構成をアプリケーションに合わせて最適化することができる．

キャッシュ構成を何らかの指標に基づいて最適化するためには，各構成のキャッシュヒット数およびキャッシュミス数を得る必要がある．これをもとに評価式を導入し，最適なキャッシュ構成を決定する．しかし，単純に全ての構成をシミュレーションした場合，対象アプリケーションのメモリアクセスごとにキャッシュヒットあるいはミスを判定していく必要がある．このキャッシュヒット/ミス判定回数が大きくなると，シミュレーションの実行時間が長くなる．この問題を解決するため，既存手法では主に 2 つのアプローチをとっている．1 つ目は，文献[1]のように解析的手法を用いることで，誤差は大きいものの，高速に処理を行う手法である．2 つ目は，文献[2]-[4]のようにシミュレーションベースの手法を用いることで，実行時間は遅くなるものの正確に各構成の結果を得る手法である．

L1 キャッシュ構成に対するシミュレーションベースの手法として，我々はこれまで CRCB1 手法，CRCB2 手法[6]が提案され，文献[5]の手法と組み合わせることで，単純なキャッシュシミュレーションと比較して平均 205.40 倍の高速化を可能とした．これらの手法は探索範囲内の全ての構成をシミュレーションするため，必ず最適な構成を選択することができる．しかし，L2 キャッシュを含めた 2 階層キャッシュ構成を対象とする場合，シミュレーションを行うキャッシュ構成数が爆発的に大きくなる．2 階層を対象とした場合の探索構成数は，1 階層を対象とした場合と比較して約 100 倍にも達する．このため，既存手法ではシミュレーションベースの手法を採用しているものはシミュレーションする構成を削減したものが多く，探索範囲内のすべてのキャッシュ構成をシミュレーションする手法は少ない．

本報告では，文献[5],[6]の手法をベースに，2 階層キャッシュ構成に適用し，さらに Inclusion Property を利用することで CRCB-T (CRCB for Two-level caches)手法を提案する．これらの手法を用いることで高速かつ正確に全構成をシミュレーションすることを可能にする．さらに,近年はスクラッチパッドメモリ (SPM) をキャッシュの代用として利用する場合もあり，その最適化手法も存在している．しかし，2 階層キャッシュ構成に SPM を含めた最適な構成を探索する場合，探索構成数は 2 階層を対象とした場合の約 15 倍になり，さらに大きくなってしまう．そこで，Inclusion Property を利用することで，CRCB-S (CRCB for Scratch pad memory)手法を提案し，実行時間を抑えつつ，必ず最適な構成を選択することを可能とする．

## 2 シミュレーションの高速化手法

本報告では図 1 のようなメモリアーキテクチャを対象とする．データの置き換え手法は LRU 法 (Least Recently Used 法) を用いる．さらに SPM へのデータマッピング手法にはアクセス回数の多いデータから順にマッピング手法を採用する．キャッシュ・SPM 構成 c は，SPM 容量 t，セット数 s，ブロックサイズ b，連想度 a によって c = ((t), (s, b, a), (s', b', a')) のように与えられる．(t) は SPM 構成，(s, b, a) は L1 キャッシュ構成，(s',b',a')は L2 キャッシュ構成を示している．また，セット数の最大値を sm，ブロックサイズの最大値を bm，連想度の最大値を am，SPM 容量の最大値を tm とする．さらに，セット数の最小値を s0，ブロックサイズの最小値を b0，SPM 容量の最小値を t0 とする．

キャッシュでは，どのセットにデータを格納するのかを決定するため，メモリアドレスを offset，index，

tag の 3 つのフィールドに分割処理している(図 2)．キャッシュでは，index の値を用いてデータの格納先セットを決定する．キャッシュに目的のデータがあるかどうか確認する際には，格納先セットの各ブロックの tag を確認し，tag が一致した場合はヒット，一致しなかった場合はミスとなる．図 1 のような構造の場合，SPM にマッピングされていないデータに対しては，L1 キャッシュを確認する．ここでキャッシュミスとなった場合，L2 キャッシュを確認する．ここでもキャッシュミスであった場合，メインメモリからデータを取ってくる．

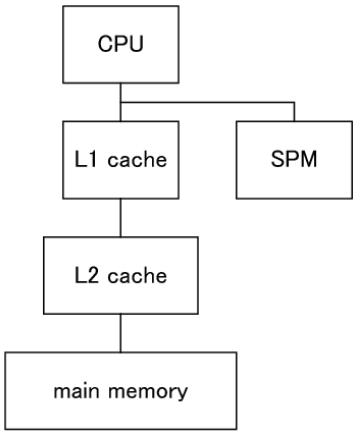



図 1 対象メモリアーキテクチャ

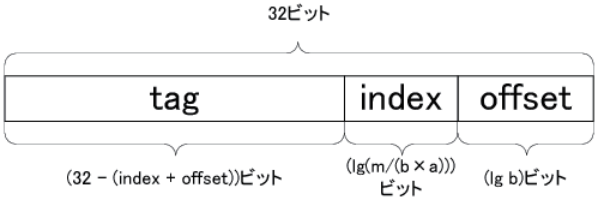



図 2 メモリアドレスと tga, index, offset

### 2-1 1 階層キャッシュ構成シミュレーション[6]

2 つのキャッシュ構成 s1，s2 について，キャッシュ構成 s1 の要素がすべてキャッシュ構成 s2 の要素に含まれていれば s1⊂s2 と書き,キャッシュ構成 s2 でキャッシュミスが起こればキャッシュ構成 s1 でもキャッシュミスが起き，キャッシュ構成 s1 でキャッシュヒットが起こればキャッシュ構成 s2 でもキャッシュヒットする．この性質は一般的に Inclusion Property と呼ばれる．文献[6] では，文献[5]の手法と組み合わせた L1 キャッシュシミュレーション手法が提案されている．この手法では，ある 1 つのメモリアクセスに対して，キャッシュ構成(-, (s0, b0, 1), -)...(-, (sm, bm, am), -)をシミュレーションするため，以下のような操作を行う．

**[複数連想度同時探索手法+CRCB1 手法+CRCB2 手法]**

(0-0) キャッシュ構成を(-, (s0, b0, am), -) と設定する．

(0-1) メモリアドレス ma から，index iL1，tag tL1 を算出する．index iL1 に対応するセットには，am 個のブロックが配置されている．

(0-2) am 個のブロックについて，1 番目から順に，tag tL1 が存在するかどうか調べる．

(0-3) (0-2)において，j 番目のブロックで tag tL1 がヒットした場合，セット数 s が一定である構成(-, (s, b, a), -) (ただし，j≦a≦am)の全キャッシュでヒットしており，セット数 s が一定である構成(-, (s, b, a), -) (ただし，1≦a<j)の全キャッシュでミスすることになる．このヒットしたブロックを 1 番目のブロックとする．

(0-4) (0-2) において，tag tL1 がミスした場合，セット数 s が一定である構成(-, (s, b, a), -) (ただし，1≦a≦am)の全キャッシュでミスしていることになる．am 番目のブロックと tag tL1 を置換し，1 番目のブロックとする．

(0-5) [CRCB1] (0-3) において，1 番目のブロックで tag tL1 がヒットした場合，キャッシュ構成(-, (s′ , b, am), -) (ただし，s≦s’≦sm)で必ずヒットするため，キャッシュ構成(-, (s′ , b, am), -) のヒット/ミス判定を省略する．

(0-6) [CRCB2] (0-3) において，i 回目のアクセスと(i-1)回目のアクセスの offset および tag が等しい場合，i 回目のアクセスはキャッシュ構成(-, (s’, b’, a), -) (ただし，b≦b’≦bm, s≦s’≦sm)で必ずヒットするため，キャッシュ構成(-, (s’, b’, a),-) でのヒット/ミス判定を省略する．

(0-7) ブロックサイズ b を 2 倍にし，(0-1)から(0-6)を b=bm となるまで繰り返す．

(O-8) ブロックサイズ b を b0 に，セット数 s を 2 倍にし，(O-1)から(O-7)を s=sm となるまで繰り返す．

### 2-2 2 階層キャッシュ構成シミュレーション

前節で紹介した手法を 2 階層キャッシュ構成シミュレーションに拡張する．この場合，ある 1 つのメモリアクセスに対して，1 階層キャッシュシミュレーションに加えて，以下の処理を行う．

**[2 階層キャッシュ構成シミュレーション]**

(T-0) (O-3),(O-4)でキャッシュミスが発生する構成がある場合，各処理に加えて以下の処理を実行する．
(T-1) ある L1 キャッシュ構成の連想度 j 以下の構成でキャッシュミスが発生するとする．ここで，キャッシュ構成を(-, (s, b, j), (2s, b, am))と設定する．
(T-2) メモリアドレス ma から，L2 キャッシュ構成の index iL2，tag tL2 を算出する．index iL2 に対応するセットには，am 個のブロックが配置されている．
(T-3) am 個のブロックについて，1 番目から順に，tag tL2 が存在するかどうか調べる．
(T-4) (T-3)において，k 番目のブロックで tag tL2 がヒットした場合，セット数 s が一定である L2 キャッシュ構成(-, (s, b, j), (2s, b, a))(ただし，k≦a≦am)の全キャッシュでヒットしており，セット数 s が一定である L2 キャッシュ構成(-, (s, b, j), (2s, b, a))(ただし，1≦a<k)の全てでミスすることになる．このヒットしたブロックを 1 番目のブロックとする．
(T-5) (T-4) において，tag tL2 がミスした場合，セット数 2s が一定である L2 キャッシュ構成(-, (s, b, j), (2s, b, a))(ただし，1≦a≦am)の全てでミスしていることになる．am 番目のブロックと tag tL2 を置換し，1 番目のブロックとする．
(T-6) [CRCB1] (T-4)において，1 番目のブロックで tag tL2 がヒットした場合，L2 キャッシュ構成(-, (s, b, j), (s'', b, am))(ただし，2s≦s''≦sm)で必ずヒットするため，L2 キャッシュ構成(-, (s, b, j), (s'', b, am))のヒット/ミス判定を省略する．
(T-7) L2 キャッシュ構成のブロックサイズ b を 2 倍にし，(T-1)から(T-6)を b=bm まで実行する．
(T-8) L2 キャッシュ構成のブロックサイズ b を(T-1)で設定した値に戻し，セット数 2s を 2 倍にし，(T-1)から(T-7)を s=sm まで実行する．
(T-9) L2 キャッシュ構成のブロックサイズ b，およびセット数 s を(T-1)で設定した値に戻し，L1 キャッシュ構成の連想度 j を 1 つ減らし，(T-1) から(T-8)を j=1 まで実行する．

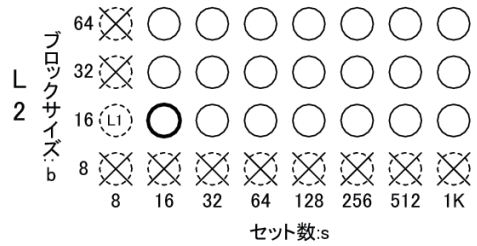
このように Inclusion Property を利用することで,2 階層キャッシュ構成シミュレーションが可能となる．この場合，1 階層キャッシュシミュレーションと同様にシミュレーションが開始される．1 階層キャッシュシミュレーションにおいて，ブロックサイズ b=16，セット数 s=8 のシミュレーションでキャッシュミスが発生した場合，L2 キャッシュのシミュレーションに移行する．この場合，L2 キャッシュは図 3(a) の太い円の構成から，順にシミュレーションしていく．この際，L1/L2 キャッシュの仕様により，L1 キャッシュ構成と同じキャッシュ構成や，図 3(a) 上で点線の円に×としている構成については L2 キャッシュシミュレーションを行わない．その後，図 3(b) のように，L1 キャッシュのシミュレーションに戻る(図 3 上ではシミュレーション済みの構成を点線の正方形で示している)．

さらに SPM を含めたシミュレーションを行う場合，次のような処理を行う．

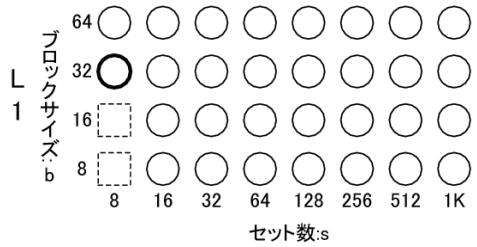
**[SPM を含めたシミュレーション]**

(S-0) 各アドレスのアクセス回数を計測する．
(S-1) (O-1) 以下の処理を全メモリアクセスに対して実行する．
(S-2) SPM 容量 t を t0 に設定する．
(S-3) (S-0)で得られたデータを利用し，アクセス回数の多いデータから順に SPM にマッピングしていく．
(S-4) (S-1) を再度実行する．
(S-5) SPM 容量 t を 2 倍にし，(S-3)，(S-4)を繰り返す．これを t=tm となるまで実行する．

SPM については図 4(a) の太い円の構成で示している SPM 容量 t=0 の場合について，図 3 と同様にシミュレーションしていく．その後，SPM 容量を t=t0（この場合は t=256）と設定し，同様の操作を繰り返していくことになる(図 4(b)，シミュレーション済みの構成を点線の正方形で示している)．

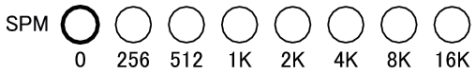



(a) L2 キャッシュ構成の探索




(b) L1 キャッシュ構成の探索の続き

図 3 2 階層キャッシュ構成の探索




(a) t=0 の場合


(b) t=256 の場合

図 4 SPM 構成の探索

### 2-3 CRCB-T 手法

L1 キャッシュと L2 キャッシュの構成に対して以下の性質がある．

**性質 1.** L1 キャッシュ構成の連想度が a=1 となる 2 階層キャッシュ構成((t), (s, b, 1), (s', b, a'))（ただし，s<s'≦sm, 1≦a'≦am)を考える．この構成((t), (s, b, 1), (s', b, a')) のキャッシュミス数は，1 階層キャッシュ構成((t), (s', b, a'), -)のキャッシュミス数と等しくなる．

**証明.** 1 階層キャッシュ構成((t), (s, b, 1), -)と((t), (2s, b, a), -)を考える．この 2 つのシミュレーションを CRCB1 手法を導入して行う場合，キャッシュ構成((t), (s, b, 1), -)でヒットしたデータはキャッシュ構成((t), (2s, b, a), -)のシミュレーションを実行しない．よって，キャッシュ構成((t), (s, b, 1), -)でミスしたデータのみがキャッシュ構成((t), (2s, b, a), -) でシミュレーションされる．2 階層キャッシュ構成((t), (s, b, 1), (2s, b, a))をシミュレーションする場合，1 階層キャッシュ構成((t), (s, b, 1), -)でヒットしたデータは 2 階層キャッシュ構成((t), (s, b, 1), (2s, b, a)) のシミュレーションを実行しない．この場合も，1 階層キャッシュ構成((t), (s, b, 1), -) でミスしたデータのみが 2 階層キャッシュ構成((t), (s, b, 1), (2s, b, a)) でシミュレーションされる．つまり，1 階層キャッシュ構成((t), (2s, b, a), -)のシミュレーションを行うメモリアクセスと，2 階層キャッシュ構成((t), (s, b, 1), (2s, b, a))のシミュレーションを行うメモリアクセスは等しくなる．よって，この 2 つのキャッシュ構成((t), (2s, b, a), -)と((t), (s, b, 1), (2s, b, a))のシミュレーションを行うメモリアクセスが等しくなるため，キャッシュミス数も等しくなる．2 階層キャッシュ構成の L2 キャッシュのセット数を L1 キャッシュの 4 倍の 4s とした場合，CRCB1 手法によって L2 キャッシュ構成((t), (s, b, 1), (2s, b, 1))でミスしたメモリアクセスのみが L2 キャッシュ構成((t), (s, b, 1), (4s, b, a)) でシミュレーションされる．1 階層キャッシュシミュレーションでも CRCB1 手法によってキャッシュ構成((t), (2s, b, 1), -) でミスしたメモリアクセスのみがキャッシュ構成((t), (4s, b, a),.) でシミュレーションされる．このため，セット数が 8 倍，

16 倍とさらに大きくしていっても同様の理由から性質 1 が成立する．

性質 1 を用いた場合，前節で提案したアルゴリズムをさらに以下のように変更する．

**[2 階層キャッシュ構成シミュレーション + CRCB-T 手法(T-1 に適用する)]**

(T-1') ある L1 キャッシュ構成の連想度 j 以下の構成でキャッシュミスが発生するとする．ここで，キャッシュ構成を(-, (s, b, j), (2s, b, am))と設定する．L1 キャッシュ構成の連想度 j が 1 となる場合のみ，キャッシュ構成を(-, (s, b, 1), (2s, 2b, am))と設定し，キャッシュ構成(., (s, b, 1), (s′, b, a′))(ただし，2s≦s'≦sm, 1≦a'≦am)のシミュレーションを省略する．キャッシュ構成(-, (s, b, 1), (s', b, a'))(ただし，2s≦s'≦sm, 1≦a'≦am) のキャッシュミス数には，キャッシュ構成(-, (s′, b, a′), -) のキャッシュミス数を利用する．

CRCB-T 手法を適用した場合，図 5 のように L1 キャッシュ構成とブロックサイズが等しくなる構成はシミュレーションを省略することができる(シミュレーションを省略するキャッシュ構成を点線の三角形で示した)．

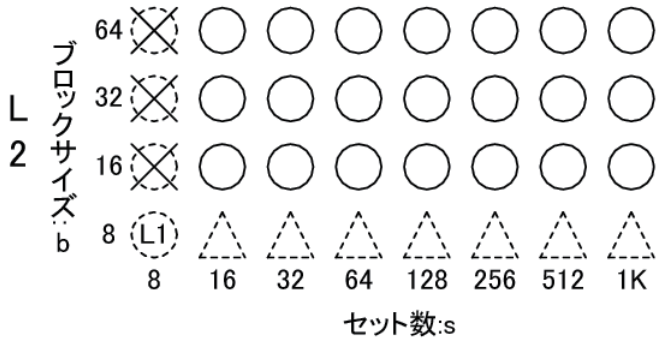



図 5 CRCB-T 手法の適用

### 2-4 CRCB-S 手法

性質 1 と同様に，SPM の構成について以下の性質がある．

**性質 2.** 構成が((t), -, -) となる SPM でヒットするデータは，構成が((t'), -, -)(ただし，t≦t'≦tm)となる全ての構成でヒットする．

**証明.** SPM 構成((t), -, -)と((2t), -, -)を考える．SPM には最もメモリアクセス回数の多いデータから格納されることとしている．このため，((t), -, -)となる構成では，アクセス回数が 1, 2, …, t 番目に多いデータが格納されることになる．((2t), -, -)となる構成の場合は，アクセス回数が 1, 2, …, 2t 番目に多いデータが格納される．よって，((t), -, -)となる構成でマッピングされているデータは((2t), -, -)となるデータに必ずマッピングされることになる．容量を 4 倍，8 倍と大きくしていった場合でも同様の理由から，性質 2 が成立することになる．

性質 2 を用いた場合，2.2 節で提案したアルゴリズムをさらに以下のように変更する．

**[SPM を含めたシミュレーション+ CRCB-S 手法(S-1, S-4 に適用)]**

(S-1') 現在の SPM 容量を t1 とする．(O-1)以下の処理を全メモリアクセスに対して実行する．この際，L1 キャッシュ構成のセット数 s=s0，ブロックサイズ b=b0 の構成において，CRCB2 手法が成立するメモリアクセスは SPM 容量 t>t1 以降のシミュレーションを省略する．

(S-4') 現在の SPM 容量を t1 とする．(S-1)を再度実行し，L1 キャッシュ構成のセット数 s = s0，ブロックサイズ b=b0 の構成において，CRCB2 手法が成立するメモリアクセスおよび SPM にヒットするメモリアクセスは SPM 容量 t>t1 以降のシミュレーションを省略する．

(S-5') SPM 容量 t を 2 倍にし，(S-3),(S-4) を繰り返す．これを全てのデータが SPM に格納されるか，t=tm となるまで実行する．

(S-1') および(S-4')において，L1 キャッシュ構成のセット数 s=s0，ブロックサイズ b=b0 の構成で CRCB2 手法が成立するメモリアクセスは以降のシミュレーションを省略している．これは，この条件を満たしたメモリアクセスは以降の全ての SPM・キャッシュ構成でヒットするため，このデータのシミュレーションを行わなくとも，結果に影響を与えないためである．(S-5') において，全データが SPM に格納された場合，全てのアクセスが SPM でヒットすることになる．このため，性質 2 より，それ以降全ての SPM・キャッシュ構成をシミュレーションしなくとも，全てのデータが SPM にヒットし，キャッシュにはアクセスされないことがわかる．よって以降のシミュレーションを省略することが可能になる．

## 3 実験結果

提案したキャッシュ構成探索アルゴリズムを C++言語によって計算機上に実装した．対象アプリケーションは MediaBench を利用した．SPM を含めない 2 階層キャッシュシミュレーションの計算機実験結果を表 1 に，SPM を含めた 2 階層キャッシュシミュレーションの実験結果を表 2 に示す．表中のアプリケーションの項目は(E)がエンコーダ，(D)がデコーダのデータキャッシュ最適化結果，adpcm の(I)は，adpcm エンコーダの命令キャッシュの最適化結果を示している．Access の項目はアプリケーションのメモリ参照回数，実行時間の括弧内に表記されている数値はヒット/ミス判定回数を示している．なお，全ての手法がシミュレーションベースかつ全ての組み合わせの結果を正確に得られる．また，表 3 の全探索の部分で数値が入っていない欄は，24 時間実行してもシステムが終了しなかったアプリケーションである．

表 1 から，2 階層キャッシュシミュレーションにおいて，提案した手法は全探索手法より平均 1465.12 倍高速化されている．さらに表 2 から，SPM を含めた 2 階層キャッシュシミュレーションにおいて，本手法は徹底的な探索手法より最大 3172.94 倍高速化されている．既存手法と比較し，探索構成数の面で本手法は非常に有利である．また，解析的手法と比較した場合，システムの実行速度では劣るものの，必ず最適解を得られる上，最大でも 40 分程度でシミュレーションが終了することから本手法が有利であると考えられる．

表 1 2 階層キャッシュ構成最適化システムの実行時間

| Application | Access [百万回] | Cache $c_{opt}$ $(s,b,a)$ | $(T,E)$ ([ms],[mW]) | [5]+[6]+CRCBT,S [sec] (百万回) | [5]+[6]+CRCBT [sec] (百万回) | [5] [sec] (百万回) | 全探索 [sec] (百万回) |
|---|---|---|---|---|---|---|---|
| ADPCM(E) | 0.52 | ((1024,8,1),(2048,32,1)) | (0.95,40.63) | 1.63 (19.75) | 1.84 (22.08) | 7.47 (128.52) | 599.81 (17360.50) |
| ADPCM(E) | 0.52 | ((1024,16,1),-) | (0.99,30.56) | 1.62 (19.75) | 1.85 (22.08) | 7.48 (128.52) | 599.74 (17360.50) |
| ADPCM(I) | 6.60 | ((64,16,1),(8192,16,1)) | (5.72,291.03) | 2.81 (68.37) | 3.07 (73.61) | 34.58 (1013.71) | 6685.40 (208586.67) |
| ADPCM(I) | 6.60 | ((64,16,1),(128,16,4)) | (5.73,137.84) | 2.82 (68.37) | 3.07 (73.61) | 34.58 (1013.71) | 6661.24 (208586.67) |
| JPEG(E) | 5.19 | ((2048,32,1),(8192,32,1)) | (12.34,1170.39) | 32.91 (709.49) | 37.71 (807.10) | 193.93 (2550.07) | 5602.61 (191019.50) |
| JPEG(E) | 5.19 | ((4096,32,1),-) | (14.75,792.77) | 32.92 (709.49) | 37.63 (807.10) | 193.53 (2550.07) | 5617.61 (191019.50) |
| JPEG(D) | 1.22 | ((2048,32,1),(8192,32,1)) | (3.15,301.83) | 10.79 (180.69) | 12.37 (207.83) | 61.29 (756.34) | 1437.07 (48342.29) |
| JPEG(D) | 1.22 | ((256,16,8),-) | (4.24,148.96) | 10.79 (180.69) | 12.37 (207.83) | 61.27 (756.34) | 1437.02 (48342.29) |
| EPIC(E) | 7.23 | ((16384,16,1),(32768,64,1)) | (39.07,16845.30) | 204.00 (4484.96) | 234.49 (5060.07) | 763.68 (9663.75) | 8357.84 (347144.83) |
| EPIC(E) | 7.23 | ((256,16,4),-) | (41.40,4638.87) | 203.64 (4484.96) | 234.35 (5060.07) | 761.22 (9663.75) | 8337.61 (347144.83) |
| EPIC(D) | 1.72 | ((8192,64,1),(16384,128,1)) | (19.14,9955.29) | 99.71 (2053.05) | 114.39 (2327.35) | 257.53 (3411.27) | 2080.40 (87315.85) |
| EPIC(D) | 1.72 | ((4096,64,2),-) | (21.53,2549.97) | 100.33 (2053.05) | 114.07 (2327.35) | 257.22 (3411.27) | 2079.53 (87315.85) |
| G721(E) | 47.95 | ((256,8,1),(4096,32,1)) | (54.39,1242.89) | 13.55 (373.64) | 15.44 (419.31) | 209.82 (656.51) | 46833.79 (1485361.15) |
| G721(E) | 47.95 | ((1024,8,1),(2048,16,1)) | (56.01,1118.72) | 13.55 (373.64) | 15.43 (419.31) | 209.20 (656.51) | 46851.73 (1485361.15) |
| G721(D) | 49.00 | ((1024,8,1),(4096,32,1)) | (57.17,1141.62) | 13.43 (359.99) | 15.51 (409.38) | 200.76 (5852.16) | 47813.14 (1512574.30) |
| G721(D) | 49.00 | ((1024,8,1),(2048,16,2)) | (57.19,1141.52) | 13.44 (359.99) | 15.50 (409.38) | 201.14 (5852.16) | 47753.54 (1512574.30) |
| GSM(E) | 51.39 | ((64,32,1),(2048,32,1)) | (48.83,1077.32) | 15.92 (622.68) | 17.56 (669.92) | 225.60 (6908.79) | 50361.77 (1602874.42) |
| GSM(E) | 51.39 | ((64,32,1),(2048,32,1)) | (48.83,1077.32) | 15.94 (622.68) | 17.59 (669.92) | 224.48 (6908.79) | 50351.82 (1602874.42) |
| GSM(D) | 8.31 | ((256,8,1),(4096,16,1)) | (10.15,243.44) | 6.08 (129.59) | 6.90 (143.44) | 48.92 (1191.23) | 8342.77 (260573.84) |
| GSM(D) | 8.31 | ((1024,8,1),(4096,16,1)) | (10.88,238.58) | 6.08 (129.59) | 6.90 (143.44) | 48.51 (1191.23) | 8332.62 (260573.84) |

表 2 SPM を含めた 2 階層キャッシュ構成最適化システムの実行時間

| Application | Access [百万回] | Cache $c_{opt}$ $(s,b,a)$ | $(T,E)$ ([ms],[mW]) | [5]+[6]+CRCBT,S [sec] (十億回) | [5]+[6]+CRCBT [sec] (十億回) | [5] [sec] (十億回) | 全探索 [sec] (十億回) |
|---|---|---|---|---|---|---|---|
| ADPCM(E) | 0.52 | ((4096),-,(32,8,1)) | (0.54,14.00) | 6.61 (0.06) | 7.43 (0.06) | 23.67 (0.36) | 20973.14 (39.77) |
| ADPCM(E) | 0.52 | ((4096),-,(32,32,1)) | (0.54,13.95) | 6.61 (0.06) | 7.45 (0.06) | 23.76 (0.36) | 20826.61 (39.77) |
| ADPCM(I) | 6.60 | ((512),-,(32,8,1)) | (4.57,94.71) | 12.00 (0.10) | 16.66 (0.10) | 58.61 (1.31) | - (-) |
| ADPCM(I) | 6.60 | ((512),-,(32,32,1)) | (4.57,94.28) | 12.02 (0.10) | 16.66 (0.10) | 58.83 (1.31) | - (-) |
| JPEG(E) | 5.19 | ((131072),-,(32,8,1)) | (9.24,1184.09) | 240.73 (4.93) | 242.27 (4.93) | 1086.64 (13704.67) | - (-) |
| JPEG(E) | 5.19 | ((8192),-,(32,8,1)) | (10.81,416.22) | 238.80 (4.93) | 242.31 (4.93) | 1092.50 (13704.67) | - (-) |
| JPEG(D) | 1.22 | ((32768),-,(32,8,1)) | (1.78,94.16) | 61.96 (0.93) | 62.07 (0.93) | 294.21 (3.58) | 55137.87 (201.20) |
| JPEG(D) | 1.22 | ((16384),-,(32,32,1)) | (1.90,72.24) | 61.78 (0.93) | 62.08 (0.93) | 292.95 (3.58) | 53914.16 (201.20) |
| EPIC(E) | 7.23 | ((524288),-,(32,8,1)) | (21.86,10111.2) | 2164.59 (47.30) | 2182.94 (47.30) | 7880.08 (100.56) | - (-) |
| EPIC(E) | 7.23 | ((512),-,(32,8,1)) | (33.51,1230.43) | 2170.80 (47.30) | 2193.50 (47.30) | 8034.96 (100.56) | - (-) |
| EPIC(D) | 1.72 | ((524288),-,(32,8,1)) | (4.96,2291.73) | 1051.70 (21.72) | 1055.79 (21.73) | 2690.49 (35.40) | 69210.15 (793.20) |
| EPIC(D) | 1.72 | ((1024),-,(32,8,1)) | (11.94,460.41) | 1052.70 (21.72) | 1049.52 (21.73) | 2680.28 (35.40) | 68219.69 (793.20) |
| G721(E) | 47.95 | ((512),(1024,16,1),(4096,32,1)) | (33.19,1692.45) | 44.76 (0.62) | 95.03 (0.62) | 358.35 (7.17) | - (-) |
| G721(E) | 47.95 | ((512),-,(32,32,1)) | (35.37,769.41) | 43.50 (0.62) | 94.80 (0.62) | 358.77 (7.17) | - (-) |
| G721(D) | 49.00 | ((512),(1024,16,1),(4096,32,1)) | (33.89,1727.56) | 45.79 (0.59) | 95.98 (0.59) | 339.45 (6.85) | - (-) |
| G721(D) | 49.00 | ((512),-,(32,32,1)) | (36.07,783.29) | 42.53 (0.59) | 95.71 (0.59) | 340.76 (6.85) | - (-) |
| GSM(E) | 51.39 | ((512),(256,16,1),(4096,16,1)) | (40.60,1508.96) | 74.90 (1.09) | 110.66 (1.10) | 507.24 (12.16) | - (-) |
| GSM(E) | 51.39 | ((2048),-,(32,32,1)) | (47.10,1099.56) | 74.28 (1.09) | 111.05 (1.10) | 508.30 (12.16) | - (-) |
| GSM(D) | 8.31 | ((512),(1024,16,1),(8192,16,1)) | (7.43,417.48) | 29.62 (0.39) | 35.07 (0.39) | 149.52 (2.58) | - (-) |
| GSM(D) | 8.31 | ((2048),-,(32,32,1)) | (8.13,199.54) | 29.14 (0.39) | 35.12 (0.39) | 149.26 (2.58) | - (-) |

## 4 結論

本報告では，SPM を含めた 2 階層キャッシュ構成最適化アルゴリズムを提案し，その評価実験を行った．従来最速とされたアルゴリズムと比較し，1000 倍をはるかに越える，最大 3172.94 倍高速に SPM・キャッシュ構成を最適化することを示した．

## 【参考文献】


[1] P. Hallschmid and R. Saleh, "Automatic cache tuning for energy-efficiency using local regression modeling," in Proc. of DAC 2007, pp. 732-737, 2007.

[2] J. Edler and M. D. Hill, "Dinero IV trace-driven uniprocessor cache simulator," http://www.cs.wisc.edu/ markhill/DineroIV/.

[3] A. G. S. Filho, P. Viana, E. Barros, and M. E. Lima, "Tuning mechanism for two-lebel cache hierarchy intended for instruction caches and low energy consumption," in Proc. of SBAC-PAD 2006, pp. 125-132, 2006.

[4] M. D. Hill and A. J. Smith, "Evaluating associativity in CPU caches," IEEE Trans. on Computers, vol. 38, Issue. 12, pp. 1612-1630, 1989.

[5] A. Janapsatya, A. lgnjatovic, and S. Parameswaran, "Finding optimal L1 cache configuration for embedded systems," in Proc. of ASP-DAC 2006, pp.796-801, 2006.

[6] N. Tojo, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Exact and fast L1 cache simulation for embedded systems," in Proc. of ASP-DAC 2009, 2009.


## 〈発　表　資　料〉

| 題　　名 | 掲載誌・学会名等 | 発表年月 |
|---|---|---|
| FIFO と PLRU をキャッシュ置換ポリシとする高速なキャッシュ構成シミュレーション手法 | 情報処理学会 DA シンポジウム 2010 | 2010 年 9 月 |
| FIFO をキャッシュ置換ポリシとする正確なキャッシュ構成シミュレーションの高速化 | 電子情報通信学会 VLSI 設計技術研究会 | 2010 年 11 月 |
| 柔軟な置換ポリシをもつ 2 階層キャッシュの正確で高速なシミュレーション手法 | 電子情報通信学会 VLSI 設計技術研究会 | 2011 年 3 月 |
| Exact and Fast L1 Cache Configuration Simulation for Embedded Systems with FIFO/PLRU Cache Replacement Policies | IEEE 2011 International Symposium on VLSI Design, Automation and Test (2011 VLSI-DAT) | 2011 年 4 月 |