Vol.2016-ARC-220 No.7
2016/5/9

Bloomフィルタを用いた 自動メモ化プロセッサのハードウェアコスト削減手法

藤井 政圭[1]　佐藤 裕貴[1]　津邑 公暁[1]　中島 康彦[2]

概要：マイクロプロセッサの高速化手法として，我々は計算再利用をハードウェアにより動的に適用する自動メモ化プロセッサを提案している．自動メモ化プロセッサは，関数およびループを再利用対象区間と見なし，実行時にその入出力を再利用表に記憶し，それを利用することで実行自体を省略する．なお，再利用表の検索に伴うオーバヘッドを抑制するため，自動メモ化プロセッサでは高速な連想検索が可能なCAMの使用を想定している．しかしCAMは消費電力，面積，製造コストが非常に大きく，自動メモ化プロセッサの実用性向上のためには，このCAMサイズを抑える必要がある．そこで本稿では，RAMとBloomフィルタの組み合わせによりCAMの一部を代替することで，自動メモ化プロセッサのハードウェアコストを削減する手法を提案する．シミュレーションによる評価の結果，再利用表の消費エネルギーを，平均50.4%，最大67.5%削減できることを確認した．

1. はじめに

これまで提案されてきたプロセッサ高速化手法は，粒度の違いはあれそのほとんどが，プログラムの持つ並列性に基づくものである．しかし，プログラム中から並列性を抽出する困難さや，プログラム中に存在する並列性にも限りがあることから，これらの手法にも限界がある．

これに対し，我々は計算再利用技術に基づいた高速化手法である**自動メモ化プロセッサ** [1] を提案している．並列性に基づく手法が処理全体の総量を変化させず複数の処理を同時実行することにより高速化を図るのに対し，計算再利用は処理自体を省略し，処理の総量を減らすことで高速化を図る手法である．自動メモ化プロセッサは，関数およびループを再利用対象区間とみなし，実行時にその入力と出力の組を再利用表と呼ばれる表へ登録する．そして，再び同一命令区間を同一入力を用いて実行しようとした際に，再利用表に記憶した過去の出力を利用することで，その命令区間の実行を省略する．ここで，計算再利用を適用する場合，再利用表の連想検索が必要となる．この連想検索によるオーバヘッドを抑制するため，自動メモ化プロセッサでは，高速な連想検索が可能なメモリであるCAMを用いて再利用表を構成することを想定している．

CAMは，多くのエントリを対象とした高速な一致比較が必要な場合に有効なハードウェアである．しかし，そのために必要な回路は大きく，複雑であるため，消費電力，面積，製造コストなどのハードウェアコストが非常に大きい．よって，自動メモ化プロセッサの実用性向上のためには，CAMサイズは極力小さく抑えることが望ましい．そこで本稿では，RAMとBloomフィルタを用いることで，既存の自動メモ化プロセッサと同等の性能を維持しつつ，必要なCAMサイズを削減する手法を提案する．

2. 自動メモ化プロセッサ

計算再利用（Computation Reuse）とは，プログラムの関数やループなどの命令区間に対し，その入力の組（入力セット）と出力の組（出力セット）を実行時に記憶しておき，再び同じ入力によりその命令区間が実行されようとした場合に，過去に記憶された出力を再利用することで命令区間の実行自体を省略し，高速化を図る手法である．また，この手法を命令区間に適用することは，**メモ化（Memoization）**[2] と呼ばれる．

このメモ化を，ハードウェアを用いて動的に適用するプロセッサとして，我々は**自動メモ化プロセッサ（Auto-Memoization Processor）**[1] を提案している．自動メモ化プロセッサは，プログラム実行時に動的に関数およびループを検出し，メモ化を適用することで，既存のバイナリを変更することなくプログラムを高速に実行できる．自動メモ化プロセッサは，一般的なプロセッサ同様，コア内部にALU，レジスタ，1次データキャッシュなどを持ち，コア

[1] 名古屋工業大学
Nagoya Institute of Technology
[2] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

© 2016 Information Processing Society of Japan

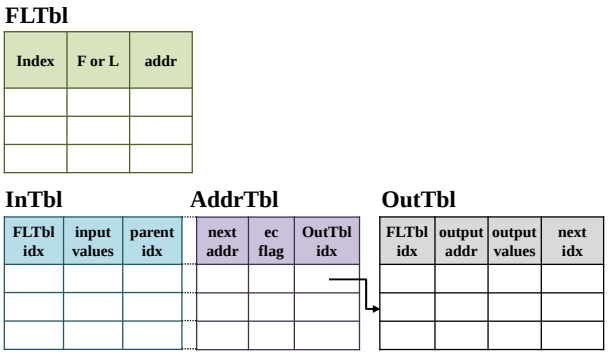

図 1　MemoTbl の構成

の外部に 2 次データキャッシュを持つ．また，独自の機構として，命令区間およびその入力と出力の組（入出力セット）を記憶しておく表である，**再利用表（MemoTbl）**と**メモ化制御機構**，および MemoTbl への書き込みバッファとして働く**再利用バッファ（MemoBuf）**を持つ．

自動メモ化プロセッサは再利用対象命令区間に進入する際，MemoTbl を参照し現在の入力セットと MemoTbl に記憶されている過去の入力セットを比較する．これを**再利用テスト**と呼ぶ．もし，現在の入力セットが MemoTbl に記憶されているいずれかの入力セットと一致する場合，その入力セットに対応する出力セットをレジスタやキャッシュに書き戻し，命令区間の実行を省略する．一方，現在の入力セットが MemoTbl のいずれの入力セットとも一致しない場合，その命令区間を通常実行しながら，その入出力セットを MemoBuf に登録し，実行終了時に MemoBuf の内容を MemoTbl に登録することで将来の再利用に備える．

次に，この木構造で表現された全入力パターンを登録するための表である MemoTbl の詳細な構成を**図 1** に示す．MemoTbl は，命令区間を記憶する**命令区間表（FLTbl）**，入力を記憶する**入力表（InTbl）**，入力アドレスを記憶する**アドレス表（AddrTbl）**，および出力を記憶する**出力表（OutTbl）**の 4 つの表で構成される．FLTbl，AddrTbl，OutTbl は RAM で実装し，InTbl は **3 値 CAM（Ternary Content Addressable Memory）**で実装する．CAM はエントリの数に依存しない高速な連想検索が可能なメモリであり，メモリ内の全ビットに対して比較回路を持つため，全てのエントリに対して同時に一致比較を行うことができる．これにより，高速な連想検索を実現している．

さて，一般に命令区間内では，複数の入力値が順に参照される．しかし，同じ命令区間でも，その入力アドレスの列は変化する場合がある．例えば，条件分岐命令を実行すると，次に参照されるアドレスはその条件分岐命令の分岐結果によって変化してしまう．そこで，自動メモ化プロセッサは，全入力パターンを木構造で表現し，MemoTbl に登録する．例えば，**図 2** に示すプログラムを実行する場合，関数 calc の全入力セットは**図 3** に示すような木構造で表現されることになる．なお，この木構造のノードは命令

```
1  int a = 3, b = 4, c = 8;
2  int calc(x){
3    int tmp = x + 1;
4    tmp = tmp + a;
5    if(tmp < 7)
6      tmp = tmp + b;
7    else
8      tmp = tmp + c;
9    return(tmp);
10 }
11 int main(void){
12   calc(2);          /* x = 2, a = 3, b = 4 */
13   b = 5; calc(2);   /* x = 2, a = 3, b = 5 */
14   a = 5; calc(2);   /* x = 2, a = 5, c = 8 */
15   a = 3; calc(2);   /* x = 2, a = 3, b = 5 */
16   return(0);
17 }
```

図 2　サンプルコード

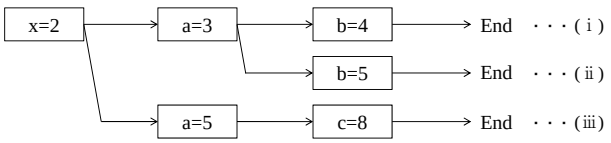

図 3　入力パターンの木構造

区間の入力を，エッジは入力と次に参照される入力との対応関係を表しており，End はそれ以上の入力が存在しないことを示す．なお，図 3 の (i)，(ii)，(iii) はそれぞれ図 2 の 12 行目，13 行目，14 行目における関数呼び出しの際の入力セットに対応する．この例において，入力セット (i) と (ii) では変数 b が 3 番目に参照されるのに対して，(iii) では変数 c が 3 番目に参照される．これは，2 番目に参照される変数 a の値が異なることにより，プログラムの 5 行目における条件分岐の結果が変化したためである．

3. 自動メモ化プロセッサのハードウェアコスト削減手法

本章では，自動メモ化プロセッサの実用化に向けた問題点を述べ，それを解決するために提案するハードウェアコスト削減手法の全体像について述べる．

3.1 自動メモ化プロセッサのハードウェアコスト

2 章で述べたように，自動メモ化プロセッサは InTbl を CAM で実装している．これは，再利用テストの際，現在の入力値と一致する入力値を持つエントリの検索，つまり InTbl の全エントリに対する高速な連想検索が必要となるからである．しかし CAM は高性能である反面，その機能を実現するために要する回路は大きく，複雑なものとなるため，RAM と比較して，面積，電力消費，製造コストが大きいという問題点を持つ．そのため汎用プロセッサでは，TLB やロードストアキューなど，高速な連想検索が必要と

© 2016 Information Processing Society of Japan

なる部分にのみ CAM が使用され，そのサイズはせいぜい 16KBytes 程度である．

一方で既存の自動メモ化プロセッサで想定している CAM のサイズは 128KBytes と，汎用プロセッサで用いられている CAM と比較して非常に大きく，オンチップで実装するのは現実的ではない．このことから，自動メモ化プロセッサの実用性を向上させるためには，CAM のサイズを抑制し，ハードウェアコストを削減する必要がある．

3.2 ハードウェアコスト削減手法の概要

前節で述べたように，自動メモ化プロセッサの実用性向上に向けて，ハードウェアコストの問題を解決する必要がある．そこで，既存の自動メモ化プロセッサと同等の性能を維持しつつ，低コストなハードウェア構成によって CAM のサイズを削減する手法を提案する．

3.2.1 RAM とフィルタによる CAM サイズの削減

既存の自動メモ化プロセッサでは CAM で構成している，連想検索が必要となる InTbl の一部を，RAM で構成することを考える．しかし，RAM 上で連想検索を行う場合，各エントリを逐次読み出し，値の一致比較を行う必要がある．そのため InTbl の CAM を単純に RAM で置き換えた場合，再利用テストにおける InTbl の検索オーバヘッドが増加し，計算再利用に成功した場合であっても，それによるサイクル数削減の効果を十分に得られない，もしくは命令区間を通常実行した場合よりも多くのサイクル数が必要となってしまう可能性がある．

そこで提案手法では，RAM と合わせて，該当エントリが存在するか否かを判定可能なフィルタを用いることで，再利用テスト失敗時のオーバヘッドを削減する．また，全入力セットを木構造で表現している点に着目し，RAM を木構造の検索に適した構成にすることで，再利用テスト成功時のオーバヘッドを抑制する．このようにして，必要となる CAM のサイズを削減しつつ，既存の自動メモ化プロセッサと同等な性能の維持を目指す．

3.2.2 再利用テスト失敗時のオーバヘッドの削減

既存の自動メモ化プロセッサが再利用テストを行う際，InTbl は CAM で構成されているため，1 つの入力と全エントリを同時に一致比較できる．そのため，エントリの数や，入力と一致するエントリの有無に関係なく，1 つの入力に対する検索オーバヘッドは一定である．しかし，InTbl を RAM で構成した場合，再利用テストの際，各エントリを逐次読み出し，一致比較する必要がある．そのため，一致するエントリを発見するまでに読み出したエントリの数に比例して，1 つの入力に対する検索オーバヘッドは増加する．さらに，再利用テスト失敗時，つまり一致するエントリが存在しない場合，InTbl に登録されている全てのエントリに対して一致比較を行うため，非常に大きな検索オーバヘッドが発生する．

そこで提案手法では，**Bloom フィルタ** [3] を用いることでこれを解決する．Bloom フィルタは，ある値が集合に登録されているか否かを，その集合を検索することなく判定可能なフィルタである．これを InTbl に応用し，Bloom フィルタによる登録の有無の判定を，RAM 構成の InTbl に対する検索に先立って行う．もし Bloom フィルタによる判定の結果，現在の入力セットと一致する過去の入力セットが InTbl に登録されていないと判定された場合，その時点で再利用テスト失敗とし，RAM を検索することなく再利用テストを終える．これにより，再利用テスト失敗時の検索オーバヘッドは Bloom フィルタによる判定に必要なサイクル数のみに抑えられ，RAM を用いた場合の検索オーバヘッドを抑制できる．

3.2.3 再利用テスト成功時のオーバヘッドの削減

前項で述べたように，再利用テスト失敗時の検索オーバヘッドは，Bloom フィルタを用いることで抑制できる．一方で，現在の入力と一致するエントリが存在する場合には，逐次読み出しによる一致比較が必要となる．その際に要した RAM の読み出し回数が多いと，検索オーバヘッドが増加してしまい，前述したように，計算再利用によるサイクル数削減の効果を十分に得られなくなる．そのため，再利用テスト成功時の検索オーバヘッドも抑制する必要がある．

そこで，自動メモ化プロセッサでは入力パターンを木構造で管理しているという点に着目する．ある命令区間に対する再利用テストにおいて，その 1 つの入力値を格納したエントリを発見したとする．その際，当該命令区間の次の入力を格納するエントリは，入力の木構造において先程のエントリに対応するノードの，子ノードにあたるエントリである．そのため次は，その子ノードにあたるエントリのみに対し値の一致比較を行うだけで十分であり，それら以外のエントリを参照する必要はない．この考えに基づき，木構造におけるノードの親子関係を考慮して InTbl を検索することで，再利用テストにおける無駄なエントリの読み出しを無くし，検索オーバヘッドを削減することができる．

しかし，ある 1 つの入力から多くの入力パターンに枝分かれするような命令区間では，ノードの親子関係を元に InTbl を検索しても，読み出しが必要となるエントリを十分に絞り込めず，大きな検索オーバヘッドが発生する場合がある．そこで，InTbl の一部を，各行に複数エントリを記憶可能な構成とし，次に検索すべき複数エントリを同一行からまとめて読み出すことができるようにすることで，そのような場合に RAM へのアクセス回数が大きく増大してしまうことを抑制する．

4. フィルタを用いたオーバヘッド削減

本章では，3.2.2 項で述べた，フィルタを用いて再利用テスト失敗時のオーバヘッドを削減する方法について，まず Bloom フィルタの概要を述べたのち，提案手法に用いるパ

© 2016 Information Processing Society of Japan

ラレルカウンティング Bloom フィルタの概要と具体的な利用法を述べる．

4.1 Bloom フィルタ

Bloom フィルタ (Bloom Filter) は，ある値が集合のメンバーであるか否かを判定するために用いられるデータ構造である．例えば，InTbl に適応する場合，値は入力値，集合は InTbl，集合のメンバーは InTbl に記憶した入力値にあたる．Bloom フィルタは 1 ビット幅の要素を複数持つ配列と複数のハッシュ関数からなる．Bloom フィルタに対する操作には，新たな値を Bloom フィルタに追加する「登録」と，ある値が Bloom フィルタに登録されているか否かをチェックする「判定」が存在する．登録の操作では，ハッシュ関数を用いて，Bloom フィルタに追加したい値のハッシュ値を生成し，配列において，生成したハッシュ値に対応する全ての要素のビットをセットする．判定の操作では，ハッシュ関数を用いて，登録されているか否かを判定したい値のハッシュ値を生成し，配列において，生成したハッシュ値に対応する全ての要素のビットがセットされているか否かをチェックする．全てセットされている場合は登録済，1 つでもセットされていないビットがあった場合は未登録と判定される．

しかし，Bloom フィルタによる判定には偽陽性があり，ハッシュ値の衝突によって，未登録の値に対して登録済と判定する場合がある．そのため，登録済と判定された値が本当に登録済か否か，実際に検索して確かめる必要がある．一方で登録済の値に対して未登録と判定する偽陰性はない．

Bloom フィルタはこのような性質を持つが，自動メモ化プロセッサの再利用テストに用いることは，2 つの点において好ましくない．1 つ目は，Bloom フィルタは基本的に値の削除が出来ない点である．自動メモ化プロセッサでは，限られたサイズの MemoTbl を有効に利用するため，不要と判定されたエントリを追い出す機構を持つため，適切に値を削除できるフィルタを用いる必要がある．

2 つ目は，Bloom フィルタの操作に伴うアクセスオーバヘッドが大きい点である．Bloom フィルタは登録，判定の度に，ハッシュ値の数に応じた回数分，配列へのアクセスが必要となる．配列を構成する RAM が 1 ポートであった場合，アクセス回数に比例して必要なサイクル数が増加し，Bloom フィルタの動作に要するオーバヘッドが性能に影響を与えてしまう恐れがある．また，RAM のポート数を増やした場合，RAM の面積が増加し，ハードウェアコストの問題が発生してしまう．このように，Bloom フィルタは提案手法に用いるには適さないため，これらの問題を解決可能なフィルタを用いる必要がある．

4.2 PCBF による再利用テストの成否判定

提案手法では，Bloom フィルタの一種である，**パラレルカウンティング Bloom フィルタ（Parallel Counting Bloom Filter; PCBF**）[4] を用いる．これは通常の Bloom フィルタと異なり，ハッシュ関数と同数に分割された配列を持ち，この配列の各要素は n ビットカウンタで構成される．配列の各要素を n ビットカウンタに変更したことで，登録の際の操作が，ハッシュ値に対応する要素ビットのセットから，要素カウンタのインクリメントに変わる．また，判定の際にはハッシュ値に対応する要素が全て非ゼロであるか否かをチェックする．この変更によって，各要素に対応するハッシュ値を持つ値がいくつ登録されているかを記憶できるようになり，カウンタのオーバフローが発生しない限り，削除したい値のハッシュ値に対応する要素をデクリメントすることで値の削除が可能となる．

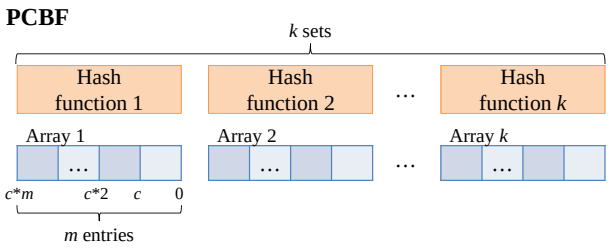

図 4　PCBF の構成

また，ハッシュ関数と分割された配列は一対一で対応するため，競合することなく，各配列へ並行してアクセスすることができる．これにより，通常の Bloom フィルタに比べ，配列の操作に伴うオーバヘッドを抑制できる．このような性質をもつ PCBF を自動メモ化プロセッサに追加し，InTbl の検索に先立って再利用テストの成否を判定することで，InTbl の一部を RAM で構成した場合に増加する，再利用テスト失敗時の検索オーバヘッドを抑制することが可能となる．

PCBF の詳細な構成を図 **4** に示す．PCBF は 6 個のハッシュ関数，およびそれと同数の配列からなり，各配列の要素数は 1366 とする．また，配列の各要素は 8 ビットカウンタとする．ハッシュ関数と配列は 1 対 1 に対応し，各ハッシュ関数が生成するハッシュ値の値域は [0, 1365] とする．

5. 木構造内の親子関係を考慮した検索によるオーバヘッド削減

本章では，InTbl に RAM を用いた場合の再利用テスト成功時のオーバヘッドを削減するため，3.2.3 項で述べた，入力パターンの木構造内の親子関係を考慮して検索する方法について，その詳細を述べる．

5.1 入力の木構造の傾向

計算再利用の対象命令区間の中には，特徴的な入力パターンを持つ命令区間が存在する．その特徴の 1 つに，ある 1 つの入力から多くの入力パターンに分岐するというも

© 2016 Information Processing Society of Japan

```
1  int a = 3, b = 0;
2  int calc(x){
3    int tmp = x + 1;
4    tmp = tmp + a;
5    tmp = tmp + b;
6    return(tmp);
7  }
8  int main(void){
9    for(i=0;i<5;i++) {
10     b = b + i;
11     calc(2);
12   }
13   return(0);
14 }
```

図 5　ループによる関数呼出し

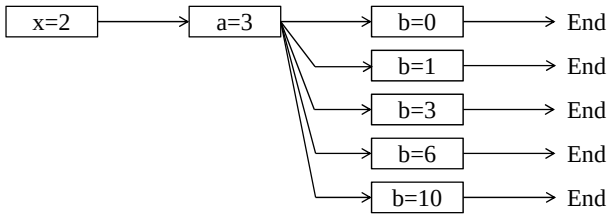

図 6　関数 calc の入力パターンの木構造

のがある．例えば**図 5** に示すプログラムを実行する際，関数 calc の全入力セットは**図 6** に示す木構造で表現される．このプログラムでは，ループのイタレーション毎に変数 b の値のみが変化する．そのため，ループ内で呼び出される関数 calc において，変数 b に対応するエントリがループのイタレーション回数と同数の 5 つ生成される．InTbl を CAM で構成する既存の自動メモ化プロセッサでは，全てのノードに対して同時に一致比較できるため，このような場合でも検索オーバヘッドは一定である．しかし，InTbl を RAM で構成する場合，逐次的に連想検索を行う必要があるため，3.2.3 項で述べたように読み出し対象エントリの絞り込みを行った場合でも，入力 b の一致比較に最大 5 回の読み出しが必要となる．

5.2　木構造の変化に応じた入力情報登録先の決定

提案手法では，InTbl を小容量な CAM と，1 ウェイ RAM，n ウェイ RAM の 3 つのユニットで構成し，入力パターンの木構造に応じて，各入力情報を登録するユニットを選択する．また，木構造に合わせた検索を行うため，各エントリが次に読み出すべきエントリの行番号を新たに MemoTbl に記憶する．この情報を元に InTbl を検索することで，無駄なエントリに対する検索を抑制することができる．それでは図 6 に示した入力セットを登録する様子を，**図 7** を用いて説明する．図 7 は，提案手法における InTbl の概略を示している．なお図中では，テーブルの各要素には，それぞれ一つの入力を格納可能とする．

idx	CAM
000	x=2
001	

	1 way RAM
100	a=3
101	

	n way RAM					
200	b=0	b=1	b=3	b=6	b=10	…
201						

図 7　木構造に合わせた InTbl への登録の様子

まず，図 6 に示す入力パターンの木構造において，ルートノードとなる変数 x の入力情報は CAM へ格納する．ルートノードは親ノードを持たないため，親子関係を考慮した検索ができない．そのため，RAM に格納した場合，一致するエントリを発見するまで，最悪の場合全エントリの読み出しが必要となってしまう．またルートノードは検索頻度が高いため，ルートノードの検索オーバヘッドは性能に影響しやすい．したがって，ルートノードを高速な連想検索が可能な CAM に格納することで，小容量の CAM を有効に活用でき，性能悪化を抑えることができる．

ルードノード以外の入力情報の格納先は，兄弟ノードが存在するか否かによって決定する．図 6 の木構造において変数 a に着目すると，親ノードである変数 x のノードは他に子ノードを持たず，a の兄弟ノードは存在しない．このように兄弟ノードが存在しないノードは 1 ウェイ RAM に格納する．これにより，次に検索すべきエントリが 1 つしか無い場合，そのエントリは 1 ウェイ RAM に格納されており，入力値が不一致の場合，他のエントリを検索することなく再利用テストを終えることができる．

続いて，図 6 の変数 a のノードに注目すると，次に検索すべきノードである変数 b のノードが 5 つ存在する．この変数 b のノードのように，木構造に兄弟ノードが存在する場合，これらの入力情報は n ウェイ RAM の同一行に格納する．これにより，これらのエントリを検索する際，一度の検索で該当行に格納された最大 n 個のエントリを同時に読み出し，一致するエントリを発見することができる．そのため，検索すべきエントリが複数存在する場合であっても，RAM の必要参照回数を抑制することができる．

5.3　MemoTbl の構成

入力パターンの木構造を考慮した InTbl への格納および検索を行うため，InTbl および AddrTbl の構成を**図 8** に示すように変更した．InTbl は（a）CAM，（b）1 ウェイ RAM，（c）n ウェイ RAM で構成され，各エントリは既存の自動メモ化プロセッサの InTbl と同様のフィールドを持つ．また（c）n ウェイ RAM の各行は，n 個のエントリに加えて，n 個の中に一致するエントリが無かった際に次に検索する n ウェイ RAM の行番号を格納するためのインデクス（next set）を持つ．

© 2016 Information Processing Society of Japan

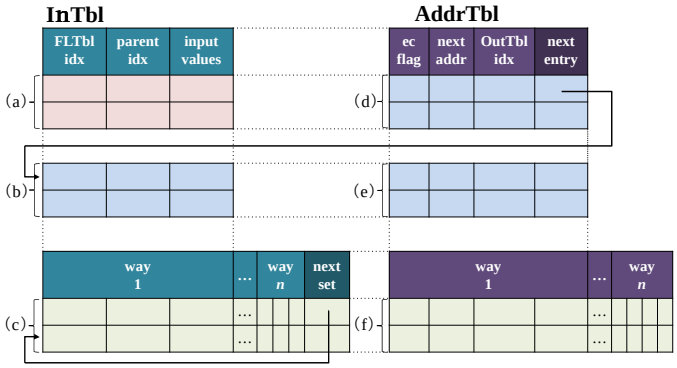

図 8　InTbl および AddrTbl の構成

AddrTbl の各エントリと InTbl の各エントリが 1 対 1 対応するよう，InTbl にあわせて AddrTbl の構成も変更する必要がある．そのため，InTbl の（a）CAM と対応する部分は（d）1 ウェイ RAM，（b）1 ウェイ RAM と対応する部分も同様に（e）1 ウェイ RAM，（c）n ウェイ RAM と対応する部分は（f）n ウェイ RAM で AddrTbl を構成する．また AddrTbl の各エントリは，既存の自動メモ化プロセッサの AddrTbl が持つフィールドに加え，次に読み出すべき InTbl エントリを指すインデクス（next entry）を持つ．

6. 評価

以上で述べた提案手法の有効性を確かめるため，サイクルベースシミュレーションにより評価を行った．

6.1 評価環境

評価では，自動メモ化プロセッサに提案手法を簡易実装し，ベンチマークプログラムの実行に要するサイクル数および消費エネルギーを算出する．簡易実装では，本来 n ウェイ RAM で構成する表を，同サイズの 1 ウェイ RAM で構成する．ただし，サイクル数は n ウェイ RAM を用いた場合を想定して算出する．また，PCBF の判定に要するサイクル数および誤検出の発生率は一定の値になると仮定して PCBF のオーバヘッドを算出し，PCBF を追加したことによる性能への影響を調査する．簡易実装により算出されるサイクル数は，エントリの登録が 1 ウェイ RAM および n ウェイ RAM のどちらかに偏る場合を考慮していないため，本来の実装を行った場合のサイクル数より，少なくなることが予想される．しかし，本来の実装を行った場合と大きくサイクル数が異なるのは，エントリの登録先が大きく偏る特殊なプログラムのみであり，それ以外のプログラムにおいては，本来の実装を行った場合とのサイクル数との差は少ないと考えられる．

評価には，計算再利用のための機構を実装した単命令発行の SPARC V8 シミュレータを用いた．また，消費エネルギーの算出については，アーキテクチャレベルの消費電力シミュレータである Wattch [5] の測定方法を参考にした．評価に用いたパラメータを**表 1** に示す．キャッシュや命令レイテンシは SPARC64-III [6] を参考とした．また，既存手法の InTbl に用いる CAM の構成は MOSAID 社の DC182888 [7] を参考にし，サイズは 32Bytes 幅 ×4K 行の 128KBytes とした．一方，提案手法の InTbl に用いる小容量 CAM の構成は eSilicon 社の eFlexCAM [8] を参考にし，サイズは 32Bytes 幅 ×256 行の 8KBytes および 32Bytes 幅 ×128 行の 4KBytes とした．なお，既存手法の評価では，プロセッサのクロック周波数が CAM のクロック周波数の 10 倍であると仮定して検索オーバヘッドを見積もっている．また，提案手法の評価では，既存手法の 1/16 以下のサイズの CAM を用いるため，プロセッサのクロック周波数が CAM のクロック周波数の 4 倍であると仮定した．PCBF による判定に要するサイクル数は 2 サイクルとし，誤検出の発生率は，参考文献 [4] における計算式を参考に，ハッシュ関数の数 k=6，配列の合計要素数 m=8K の場合を想定し，1.00%と仮定した．

表 1　評価環境

MemoBuf	
size	64 KBytes
MemoTbl	
FLTbl size	8 KBytes
InTbl/AddrTbl/OutTbl size	128 KBytes each
Comparison latency	
reg. ⇔ CAM (128/8/4 KBytes)	9/3/3 cycles/32Bytes
Cache ⇔ CAM (128/8/4 KBytes)	10/4/4 cycles/32Bytes
reg. ⇔ RAM	2 cycles/32Bytes
Cache ⇔ RAM	3 cycles/32Bytes
Write back latency	
MemoTbl ⇒ register	1 cycles/32Bytes
MemoTbl ⇒ Cache	2 cycles/32Bytes
PCBF	
hash function	6 pieces
array size	1366 Bytes
counter	8 bit
false positive rate	1.00 %
latency	2 cycles
D1 cache	
size	32 KBytes (4 ways)
access latency	2 cycles
miss penalty	10 cycles
D2 cache	
size	2 MBytes (4 ways)
access latency	10 cycles
miss penalty	100 cycles
Register windows	4 sets
miss penalty	20 cycles/set

6.2 評価結果

SPEC CPU95 を用いた評価結果を**図 9** および**図 10** に示す．図 9 のグラフは各ベンチマークプログラムが要した

© 2016 Information Processing Society of Japan

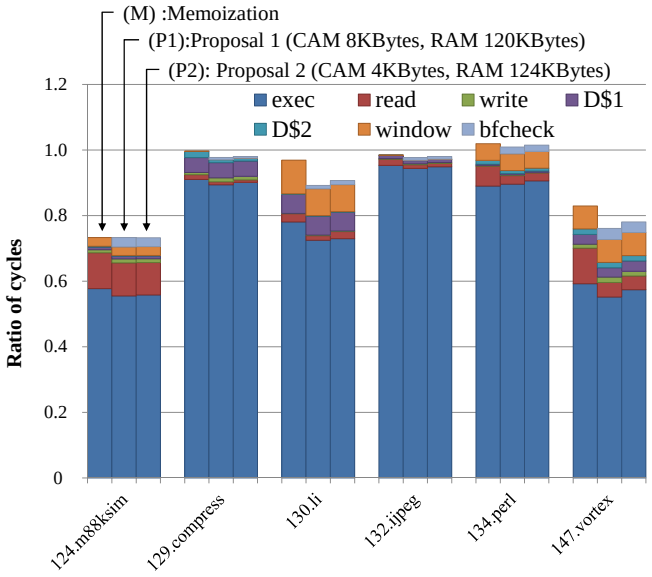

図 9　実行サイクル数

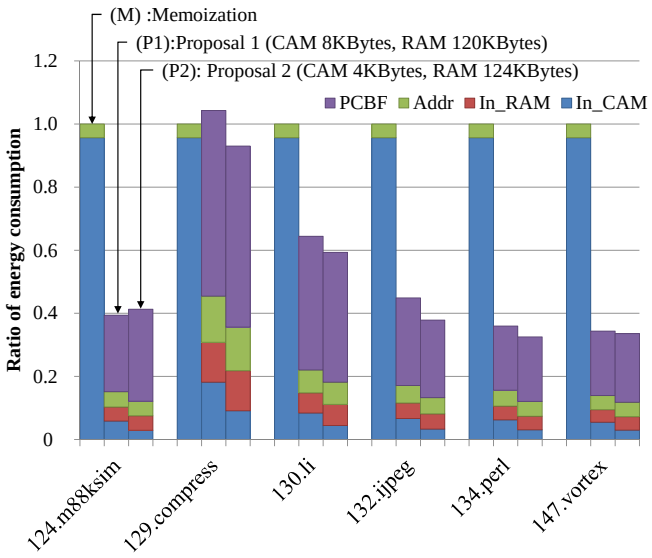

図 10　InTbl，AddrTbl および PCBF の消費エネルギー

総実行サイクル数を，図 10 のグラフは各ベンチマークプログラムに要した InTbl，AddrTbl および PCBF の消費エネルギーを示している．図 9 中で各ベンチマークプログラムの結果を示す 3 本のグラフはそれぞれ

(M)　既存モデル（CAM 128KBytes）
(P1)　提案モデル 1（CAM 8KBytes，RAM 120KBytes）
(P2)　提案モデル 2（CAM 4KBytes，RAM 124KBytes）

が各ベンチマークプログラムの実行に要したサイクル数を表しており，メモ化を行わないモデルの実行サイクル数を 1 として正規化している．凡例はサイクル数の内訳を示しており，exec は命令実行サイクル数，read は入力と MemoTbl の比較に要したサイクル数（検索オーバヘッド），write は MemoTbl の出力をレジスタやメモリに書き込む際に要したサイクル数（書き戻しオーバヘッド），D$1 および D$2 は 1 次および 2 次データキャッシュミスによるペナルティサイクル数，window はレジスタウインドウミスによるペナルティサイクル数，bfcheck は PCBF の判定に要したサイクル数である．

また，図 10 中の 3 本のグラフは，図 9 と同様の 3 つのモデルが各ベンチマークプログラムの実行に要した InTbl，AddrTbl および PCBF の消費エネルギーの合計を表しており，既存モデル (M) の消費エネルギーを 1 として正規化している．また凡例は消費エネルギーの内訳を示しており，In_CAM は InTbl を構成する CAM の消費エネルギー，In_RAM は InTbl を構成する RAM の消費エネルギー，Addr は AddrTbl の消費エネルギー，PCBF は PCBF の消費エネルギーである．

評価の結果，RAM を用いたことによる性能の悪化が予想されたが，総実行サイクル数は既存モデルと比較して，(P1) では平均 2.7%，最大 8.3%，(P2) では平均 2.5%，最大 6.4%削減されており，むしろ性能は僅かながら向上することが確認できた．また，消費エネルギーの削減率は，既存モデルと比べた場合，(P1) では平均 46.1%，最大 65.6%，(P2) では平均 50.4%，最大 67.5%となり，提案手法の有効性を確認した．

6.3　考察

実行サイクル数

まず，図 9 で 130.li など 4 つのベンチマークプログラムにおいて，既存モデルと比べて，提案モデルでは命令実行サイクル数が削減され，性能が向上している．これは，オーバヘッドフィルタと呼ぶ，メモ化の効果が得られない命令区間の再利用を中止する機構が原因である．このオーバヘッドフィルタによって，既存モデルでは再利用を中止されていた命令区間が，提案モデルでは検索オーバヘッドが削減され，そのような命令区間が再利用されるようになったためである．検索オーバヘッドが削減された要因は 2 つ挙げられる．1 つ目は，提案モデルでは InTbl に小容量の CAM を用いた点である．CAM の一回の一致比較に要する時間は，CAM のサイズにより変化し，小容量な CAM ほど少ない時間で一致比較を行うことができる．提案モデルでは，既存モデルの InTbl に用いた CAM よりも小容量なものを用いたことで，CAM の一致比較による検索オーバヘッドが削減された．2 つ目は，アクセス 1 回あたりの遅延が CAM よりも RAM の方が小さい点である．既存モデルでは，CAM を用いているため全エントリを一度のアクセスで一致比較することができるが，一回のアクセスあたりの遅延が大きくなる．一方，提案モデルでは 1 つ目の入力以外は RAM に格納し，AddrTbl に記憶した行番号を用いてエントリを逐次読み出し，一致比較する．RAM は 1 エントリ毎にしかアクセスできないが，アクセス 1 回あたりの遅延は CAM に比べて小さく，もし，少ない回数の一致比較で一致するエントリを発見した場合，CAM より

© 2016 Information Processing Society of Japan

も少ない時間で一致比較を完了することができる．このように，命令区間の入力パターンによっては，提案モデルで検索オーバヘッドを削減できる場合があり，再利用率，および性能が向上した．

一方，124.m88ksim では，再利用テストの際に PCBF の判定に要したサイクル数により，既存モデルとほぼ同等な性能となっている．このような，計算再利用の効果が大きいプログラムにおいては，PCBF による再利用テスト失敗時の検索オーバヘッド削減の効果が十分に得られない，または PCBF の判定に要するサイクル数による影響が相対的に大きくなる．しかし，予想された RAM を用いることによる性能低下は見られず，提案手法により，既存モデルと同等の性能を維持しつつ，実用性の向上を達成した．

消費エネルギー

図 10 から，129.compress を除く全てのベンチマークプログラムにおいて，提案モデルの消費エネルギーが，既存モデルと比べて大きく削減されていることが分かる．まず CAM については，小容量の CAM を用いたことで一回の一致比較に要する消費電力が削減され，さらにルートノードのみを CAM に格納するようにしたことで，CAM での検索が必要となるのはルートノードのみとなり，CAM のアクセス回数も削減された．また，各行 32Byte からなる 128KByte RAM に対するアクセス一回に要する消費エネルギーは，128KByte CAM の全体に対する一回の検索に要する消費エネルギーと比べ約 1/20 であり，既存モデルの CAM の消費エネルギーと比べて，提案モデルの RAM の消費エネルギーは僅かである．これらのことから，提案モデルで新たに加わる PCBF と RAM による消費エネルギーの増加量を，CAM による消費エネルギーの削減量が大きく上回り，総消費エネルギーが削減された．

ただし 129.compress では，(P1) の消費エネルギーが既存モデル (M) よりも僅かに大きくなっている．129.compress は，総実行サイクル数が少なく，かつ計算再利用の効果があまり得られないプログラムである．このようなプログラムでは，再利用テストにおける再利用表へのアクセス総数，ひいては再利用表のアクセスに要する消費エネルギーがそもそも少ない．その結果として，PCBF による消費エネルギー増加が相対的に大きくなった．

7. おわりに

本稿では，自動メモ化プロセッサの実用化に向け，RAM と Bloom フィルタを用いたハードウェアコスト削減手法を提案した．Bloom フィルタを用いることにより，高コストな CAM の一部を低コストな RAM に置き換えた場合に発生する性能低下を抑制した．SPEC CPU95 ベンチマークを用いてシミュレーションにより評価した結果，性能悪化が予想されたが，逆に平均 2.5%，最大 6.4%サイクル数が削減されることが確認できた．また，再利用表の消費エネルギーを，平均 50.4%，最大 67.5%削減できることを確認した．このことから，提案手法によって，既存手法とほぼ同等の性能を維持しつつ，ハードウェアコストおよび消費エネルギーの削減が可能であることを確認した．今後の課題としては，提案手法の詳細な実装および評価，自動メモ化プロセッサに適した Bloom フィルタの構成の考案，そしてさらなる CAM 削減のための手法の検討などが挙げられる．

参考文献

[1] Tsumura, T., Suzuki, I., Ikeuchi, Y., Matsuo, H., Nakashima, H. and Nakashima, Y.: Design and Evaluation of an Auto-Memoization Processor, *Proc. Parallel and Distributed Computing and Networks*, pp. 245–250 (2007).
[2] Norvig, P.: *Paradigms of Artificial Intelligence Programming*, Morgan Kaufmann (1992).
[3] Bloom, B. H.: Space/Time Trade-offs in Hash Coding with Allowable Errors, *Commun. ACM*, Vol. 13, No. 7, pp. 422–426 (online), DOI: 10.1145/362686.362692 (1970).
[4] 倉田成己，塩谷亮太，五島正裕，坂井修一：ブルーム・フィルタを用いたメモリ・アクセス順序違反検出，情処研報，Vol. 2014-ARC-212, No. 17, 情報処理学会，pp. 1–15 (2014).
[5] Brooks, D., Tiwari, V. and Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *Proc. 27th Annual Intl. Symp. on Computer Architecture*, pp. 83–94 (2000).
[6] HAL Computer Systems/Fujitsu: *SPARC64-III User's Guide* (1998).
[7] MOSAID Technologies Inc.: *Feature Sheet: MOSAID Class-IC DC18288*, 1.3 edition (2003).
[8] eSilicon Corporation: *HiSilicon Licenses eSilicon's 40nm Silicon-Proven TCAMs for High-Performance Network Chips* (2011).

© 2016 Information Processing Society of Japan