Vol.2015-ARC-216 No.40
2015/8/6

スタンダードセルベース設計用のCAM型TLBの実装手法の提案

武藤 郡[1,a)]　佐々木 敬泰[1]　深澤 祐樹[1]　近藤 利夫[1]

概要：近年組み込みプロセッサは高性能化しており，OS を動作させ様々な処理を行うことが増えている．一般的な OS の多くは動作にメモリ管理機構 (Memory Management Unit：MMU) や割り込み処理機構を必要としている．MMU には通常，アドレス変換の高速化のために TLB (Translation Lookaside Buffer) を内蔵している．TLB の実現方法には，仮想アドレスをキーとして検索する CAM (Content-addressable memory) 方式と，仮想アドレスをインデックスとして検索する RAM 方式があり，プロセッサ毎に採用している方式が異なる．ARM プロセッサでは，RAM 方式を採用しているが，MIPS をはじめとする幾つかのアーキテクチャは，MMU 内の TLB を CAM で構成することを前提としている．一般にスタンダードセルベースの LSI 設計や FPGA での実装において CAM を利用することは，面積が非常に大きくなってしまことから非現実的である．しかしながら，MIPS 等の CAM ベースのプロセッサの TLB を RAM で実装すると TLB へのアクセス方法が変わるため，既存の OS 等が動作しなくなるという問題が発生する．そこで本研究ではこれらの問題を解決するために，RAM ベースの TLB でありながら，既存の CAM ベースの TLB を前提として設計されたコードを動作させることのできる TLB の実装手法を提案する．本論文では，提案手法を MIPS ライクな命令セットを持つスーパースカラプロセッサに組込み，既存の MIPS 用コードがそのまま動作することを示す．

1. はじめに

近年，組み込みプロセッサは高性能化が進んでおり，GUI やマルチメディア処理等の機能を実現するために OS を動作させることが多くなってきている．プロセッサ上で OS を動作させるためにはアドレス変換やメモリ保護を行う MMU(Memory Managemenet Unit) や，例外処理を行うモジュールをプロセッサに実装する必要がある．

MMU は仮想メモリを実現するために物理アドレスと仮想アドレスの変換を行うが，アドレス変換を行うにはメモリ上の物理アドレスと仮想アドレスの対応表であるページテーブルを引かなければならないため，これを高速に行うために TLB(Translation Lookaside Buffer) と呼ばれるバッファが存在する．TLB はページテーブルの一部を保持するキャッシュメモリとして働き，その実装には大きく分けて RAM を用いた RAM 型と，CAM (Content-addressable memory)[1] を用いた CAM 型の 2 種類がある．MIPS をはじめとする幾つかのアーキテクチャは TLB が CAM 型動作を前提として設計されているため，そのようなアーキテクチャを使用する際には CAM 型 TLB を実装する必要がある [2]．

しかし，CAM はメモリマクロやメモリコンパイラを利用して容易に設計が行える RAM とは異なり，手設計する必要があるため設計コストが非常に高いという問題がある．また，CAM を RTL(Register Transfer Level) で記述し，論理合成することも可能であるが，スタンダードセルを用いて設計した CAM は面積が非常に大きくなるため，連想キーのビット幅やエントリ数の多い TLB をスタンダードセルで設計するのは非現実的である．そこで本研究では CAM 型としての動作を実現しながら，RAM マクロとスタンダードセルのみで設計可能な疑似 CAM 型 TLB 機構を提案する．提案手法を用いた場合，ISA から見える TLB は CAM を用いた物と同じ様に振る舞うため，MIPS などのアーキテクチャ上で OS のカーネルコードを変更すること無く実行出来る．

本稿は，以降次のように構成される．まず，第 2 節では一般的な CAM ベース TLB，及び本研究で対象としている MIPS プロセッサの TLB の仕様について述べ，第 3 節では CAM の実装方法に関する先行研究について述べる．第 4 節では RAM を用いた擬似 CAM ベース TLB を提案し，第 5 節で性能評価を行う．最後に，第 6 節でまとめと今後

[1] 三重大学大学院工学研究科
Graduate School of Information Engineering, Mie University
[a)] muto@arch.info.mie-u.ac.jp

の課題について述べる．

2. 研究背景

2.1 MMU

MMU はアドレス変換やメモリ保護の機能を持つ，図 1 のような仮想記憶を実現するためのユニットである．仮想記憶を実現するためにはプログラム上で用いられる仮想アドレスを，実記憶上で用いられる物理アドレスに変換する機構が必要となる．アドレス変換を行うためには，実記憶上に存在する物理アドレスと仮想アドレスの対応表であるページテーブルを引かなければならない．メモリアクセスのたびにページテーブルを引くとレイテンシが膨大になってしまう．この問題を解決するためにプロセッサ内に TLB が置かれる．

TLB は高速な小容量のメモリであり，キャッシュシステムのような役割をする．プロセッサはページテーブルの一部を TLB に格納しておき，アドレス変換を行う場合には TLB を参照することで高速にアドレス変換を行う．

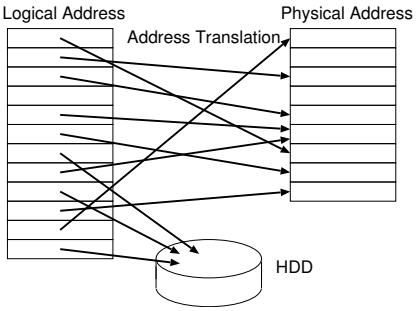

図 1　アドレス変換

2.2 一般的な TLB の実装手法

以下に，一般的なプロセッサにおける TLB の実装手法を 2 つ，すなわち RAM を用いた TLB の実装手法と，CAM を用いた TLB の実装手法を挙げる．

2.2.1 RAM を用いた TLB

図 2 は RAM を用いて TLB を実装する手法である．MMU はロード/ストア命令や命令フェッチによりメモリにアクセスする前に，プログラム中に使用されている仮想アドレスから物理アドレスへ変換するために TLB に仮想アドレスの上位部分，すなわち仮想ページ番号 (Virtual Page Number : VPN) を渡す．TLB は VPN の一部をインデックスとしてタグメモリ（図 2 の左側のメモリ）を読み出し，読み出された VPN と一致したら TLB ヒットとする．同時に物理フレーム番号 (Physical Frame Number : PFN) を格納したメモリ（図 2 の右側のメモリ）を読みだし，TLB ヒットであれば PFN を MMU に返す．MMU は PFN と仮想アドレスの下位部分であるオフセットを結合し，物理アドレスを生成する．一方，ミスヒットの場合は，インデックスで示される行に VPN，及び PFN を登録する．従って，RAM 方式の場合は，仮想アドレスにより，格納されるエントリが一意に決まる．このように，RAM 方式ではダイレクトマッピング方式のキャッシュと同じように振舞う．

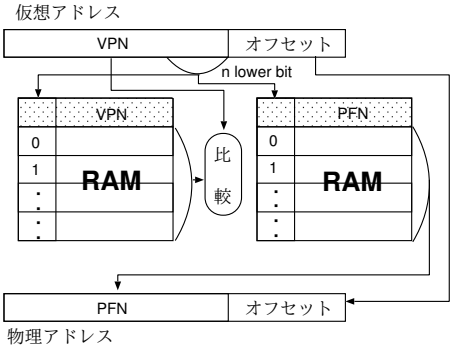

図 2　RAM を用いた TLB

2.2.2 CAM を用いた TLB

図 3 は CAM を用いて TLB を実装する手法である．MMU は RAM ベースの場合と同様に仮想アドレスを VPN とオフセットに分け，VPN を用いて CAM を読み出すが，RAM ベースの場合と異なり，VPN をすべて用いて CAM を検索する．CAM 内にマッチする VPN が存在した場合は，同じ行に格納されている PFN を読みだし，オフセットと結合して物理アドレスを生成する．RAM を用いた TLB の場合，ある仮想アドレスがメモリの何行目に格納されるかは決定的であったのに対し，CAM を用いた TLB の場合は仮想アドレスと格納される番地は無関係である．そのため，ミスヒットの場合は，MMU は何らかの方法でリプレースする行を決定し，TLB に対し VPN 及び PFN を登録する．換言すれば，OS は仮想アドレスと無関係に任意の番地（行）に VPN, PFN のペアを書き込む．

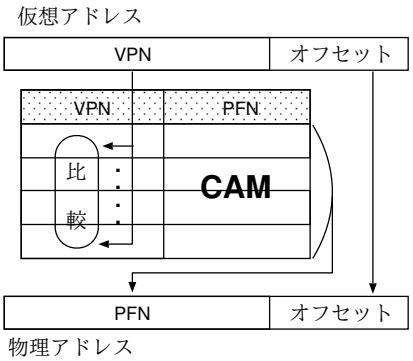

図 3　CAM を用いた TLB

商用プロセッサでも TLB の実装方法としては，RAM ベースのものも CAM ベースのものも存在するが，TLB エントリのリプレース方法が全く異なるため，実装方法を変えた場合，既存の OS やシステムソフトウェアの互換性は

無くなるという問題がある．

2.3 スタンダードセルベース/FPGA 設計における CAM ベース TLB の実装

プロセッサをスタンダードセルベースの LSI や FPGA で実装する場合，CAM はハードマクロとして用意されていないケースが多い．CAM を RTL で記述し，論理合成することもできるが，その場合は大量の Flip Flop と比較器で構成することになる．このような構成は回路規模が非常に大きくなってしまうため，TLB のような規模の大きいモジュールでは非現実的である．そこで本研究ではスタンダードセルベース/FPGA 設計でも設計可能な RAM を用いた擬似 CAM 型 TLB を提案する．

2.4 MIPS プロセッサの TLB 仕様

本研究では MIPS を対象として TLB の実装を行う．そこで，提案手法の説明をする前に，まず MIPS における TLB の扱いについて述べる．MIPS では TLB は 1 ～ 64 までの任意のエントリ数の CAM によって構成される．また，TLB を操作するために幾つかのレジスタと命令を持っている．

TLB 関連の特殊レジスタ

- Random Register:
  TLBWR 命令で書き込みを行うエントリを指定するレジスタ．毎サイクル，1 づつデクリメントされる．
- Index Register:
  TLB 関連命令がアクセスするエントリを指定するレジスタ．
- EntryHi Register:
  仮想ページ番号を格納するレジスタ．
- EntryLo Register:
  物理フレーム番号を格納するレジスタ．

TLB 関連の特権命令

- TLBWI:
  TLB の Index Register で指定するエントリに EntryHi と EntryLo の書き込みを行う．
- TLBWR:
  TLB の Random Register で指定するエントリに EntryHi と EntryLo の書き込みを行う．
- TLBR:
  TLB の Index Register で指定するエントリを EntryHi と EntryLo に読み出す．
- TLBP:
  EntryHi で指定する仮想ページ番号が入っているエントリ番号を Index Register に書き込む．

ロード/ストア命令の実行や命令フェッチによってメモリへのアクセスが発生すると，仮想アドレスが TLB に渡され物理アドレスが生成される．与えられた仮想アドレスのエントリを TLB が持っていない場合は TLB ミス例外が発生し，EntryHi Register に仮想アドレスが書き込まれた後，例外処理ルーチンの TLB 処理部に PC が飛ばされる．例外処理ルーチンではまずロード命令によってページテーブル上の EntryHi Resiter のエントリを読み込む．読み込んだエントリを EntryLo Register に書き込み，TLBWR 命令を実行することによって Random Register で指定するエントリに TLB ミスが発生した VPN と，それに対応する PFN が書き込まれる．その後，PC を TLB ミスが発生した命令に戻し再度実行する事で正常にアドレス変換を行う事が出来る．

また，ソフトウェアから TLB 内のエントリにアクセスする場合は，TLBP 命令を実行することで EntryHi に対応するエントリ番号を Index Register に得る事が出来，その後 TLBR 命令を実行する事で EntryHi，EntryLo にエントリを読み出す事が出来る．

このように，MIPS の ISA は TLB を CAM で実装することを想定しており，既存の OS やシステムソフトウェアの TLB ハンドラはこの仕様に合わせて設計されている．

3. 関連研究

CAM はプロセッサに限らず，ネットワークルータや暗号化/符合化等を行うハードウェアにも広く利用されているが，前述の通り実装が困難であるため，RAM を用いた様々な実装手法が提案されている．

Hoang らは，二分木探索を用いて複数の RAM から必要なデータを探索する手法を提案している [3]．この手法では，探索をパイプライン化することで高スループットを実現しているが，スーパスカラプロセッサでは TLB の検索は 1 サイクル程度で行う必要があるため，スループットよりもレイテンシが重要となる．そのため，Hoang らの手法を TLB に適用した場合，パイプライン段数が大幅に伸びるため，性能が低下する危険性がある．

文献 [4], [5], [6] では，FPGA 向けの CAM の実装手法を提案している．この手法は，探索パターンをアドレスと見做して RAM にアクセスすることで CAM の動作を模擬する手法である．この手法はパターン長の 2 乗に比例する容量のメモリが必要となるため，TLB のような用途には不向きであるという問題がある．文献 [7] では，FPGA 上に論理合成可能なスーパスカラプロセッサである FabScalar[13] を実装することを試みている．スーパスカラプロセッサには，TLB 以外にも命令スケジューラのウェイクアップロジック等，様々な場所で CAM が必要になる．そこで，文献 [7] では，文献 [4], [5] の手法を用いて CAM を実装している．しかしながら，文献 [7] では比較的小規模な CAM のみを用いており，また，実装したプロセッサは仮想メモリをサポートしていないため，TLB の効率的な実装手法は明らかになっていない．

文献 [8], [9] も同様に RAM を用いた CAM の実装手法であるが，TCAM(Ternary CAM) を対象としており，また探索にも複数サイクル掛かるため，TLB の実装には不向きである．

また，一部の FPGA では，CAM を内蔵している製品もある [10], [11]．例えば，Altera 社の APEX20 シリーズでは，ESB (Embedded System Block) 内にハードウェアとして実装されており，組込みメモリと組み合わせることで，MIPS プロセッサの TLB を容易に実装することができる [12]．しかしながら，比較的新しい FPGA ではハードマクロとして CAM を内蔵していないケースが多く，また，実装対象が特定の FPGA に限定されるため他の FPGA シリーズや LSI 設計には適用できない．そのため，本論文では FPGA 内の組込み CAM の利用は考慮しないものとする．

4. RAM を用いた疑似 CAM 型 TLB の実装手法の提案

RAM を用いた CAM 型 TLB 機構は図 4 の様にそれぞれ VPN と PFN を保持する 2 つの RAM から構成される．仮想アドレスが渡されると前述の CAM ベース TLB と同様に VPN とオフセットに分割する．それぞれの RAM へのインデックスは VPN をハッシュ関数に与える事によって求められる．求められたインデックスで RAM を引き，RAM 内の VPN と変換対象の仮想アドレスの VPN とを比較する事で TLB が対象の VPN を保持しているかを判定する事が出来る．CAM で構成したものと比べ VPN の比較が 1 度だけで済むため，回路規模の大幅な削減と消費電力の低下が見込まれる．VPN が一致していた場合 TLB がページテーブルのそのエントリを保持している事がわかるので，RAM 内の PFN を返しオフセットと合わせる事で物理アドレスへの変換が完了する．しかし，MIPS 等のアーキテクチャは CAM による TLB の動作を要求するため，RAM で実装したこの機構では一部の命令で TLB の動作が異なり正常に動作しないことがある．

例えば，前述の TLBWI 命令は Index Register で指定したエントリに書き込む必要があるが，この機構ではアドレスからインデックスが一意に決まってしまうため指定位置への書き込みは行えない．そこで図 5 のような CAM 相当でのエントリ番号と RAM を引くアドレスとのリストである RAM インデックステーブルを作成する．RAM への格納を行う際には RAM インデックステーブルに RAM でのインデックス，すなわちハッシュ値を入れる．TLBR 命令で TLB の Index Register で指定するエントリにアクセスする際は Index Register で RAM インデックステーブルを引き，得られたハッシュ値を使用して RAM を引く事で TLBW 命令で書き込んだエントリを読み出す事が出来るため，RAM と CAM でのインデックスのずれを解消する事が出来る．また，TLBP 命令は本来 CAM でのインデックスを返す．しかし前述の RAM インデックステーブルではそのままだと CAM でのインデックスから RAM のインデックスを得る事しか出来ず，テーブル内のすべての RAM のインデックスとの比較が必要となってしまう．これを解決するために前述の RAM インデックステーブルとは逆の，RAM を引くアドレスと CAM 相当でのエントリ番号のリストである CAM インデックステーブルを作成する．これにより指定した RAM でのアドレスが CAM ではどのエントリに当たるのかが求められ，TLBP 命令を正常に実行する事が出来る．

この RAM を用いた方式では本来の CAM 型では消えているはずのエントリが残ってしまい，それをアドレス変換時や TLBP 命令実行時に参照してしまい誤ったヒットを起こしてしまう可能性が生じるので，RAM インデックステーブルに valid bit を追加し valid bit が落ちている時はヒットさせない事により誤ったヒットを起こさないようにしている．

しかし，この機構では同じハッシュ値を持つエントリは 1 つしか保持できないため，TLB ミス例外を起こしたロード/ストア命令の命令アドレスのハッシュ値が，ロード/ストア命令のアクセス先アドレスのハッシュ値と一致してしまうと問題が起きる．TLB ミス例外を解消し PC を戻すと，その命令をフェッチするために再び TLB ミス例外を起こし 1 度目の TLB ミス例外処理時に確保したエントリを破棄してしまい命令実行がストールしてしまう．そこで 2-way RAM 方式を用いてこの問題を解決する．

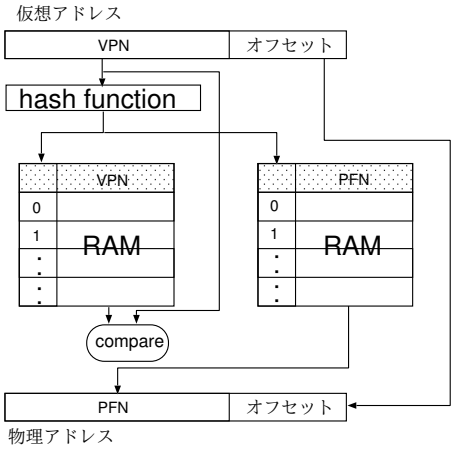

図 4　提案手法

4.1 2-way RAM 方式

2-way RAM 方式では図 6 に示すようにタグメモリとデータメモリについて，それぞれのインデックスに 2 つのデータを保持できるようにする．TLBWR,TLBWI による TLB へのエントリの書き込みを行う際にはそれぞれのイン

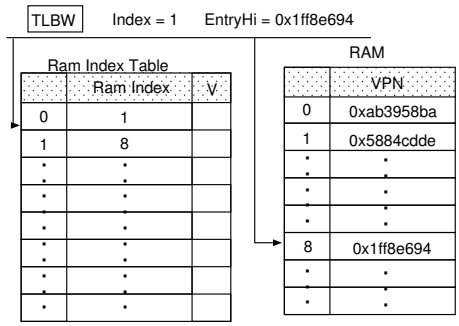

図 5　インデックステーブル

デックスごとに LRU(Least Recently Used) を用いて書き込む．また，RAM インデックステーブルにどちらのウェイか，を示すビットを追加する事でアドレス変換時に使用するウェイを判定する．

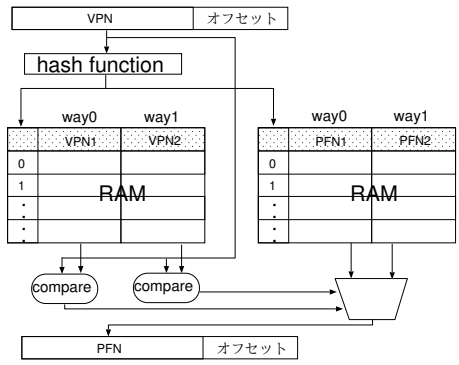

図 6　2-way RAM

5. 性能評価

提案手法の TLB と既存の CAM を用いて実装された TLB についてそれぞれ評価を行った．以下 5.1 にて評価環境の詳細について述べ，5.2 にて評価結果について述べる．

5.1 評価環境

本項では，本研究でベースとして用いるスーパースカラプロセッサである FabScalar について説明する．FabScalar は System Verilog で記述された論理合成可能なスーパスカラプロセッサコアである．図 7 に FabScalar の最も基本的な構成を示す．FabScalar の基本構成は 9 つのパイプラインステージにより構成される，すなわち，フェッチ，デコード，リネーム，ディスパッチ，発行，レジスタ読出し，実行，書戻しとリタイアである．FabScalar ではこの構成を標準とし，様々なパラメータ，例えばフェッチ幅や演算リソース数などを変更したスーパースカラコアの RTL コードを自動生成できる．FabScalar では，フェッチ幅，発行幅，コミット幅，発行キューのエントリ数，リオーダバッファのエントリ数，などスーパースカラコアを構成する様々な要素がパラメータ化されており，異なる構成のスーパースカラコアを容易に生成することが可能である．

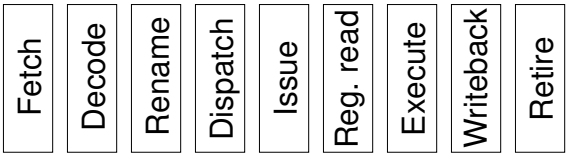

図 7　Canonical superscalar processor.

しかしながら，FabScalar には Co-processor 0 が実装されておらず，割り込みや仮想メモリをサポートしていない．そこで，まず FabScalar に Co-processor 0 を追加し，仮想メモリ，及び TLB ミス発生時の割り込み処理のためのハードウェアを追加した．FabScalar に提案手法を組込み，MIPS32 用の TLB ミスハンドラが正常に動くことを確認した．しかしながら，RTL シミュレーションは時間が掛かるため，規模の大きなベンチマークプログラムを動作させることは難しい．そこで，提案手法の性能を評価するために，TLB の動作を模擬するシミュレータを作成し，SPEC 2000 ベンチマークのトレースデータを用いて TLB ヒット率を算出した．評価には SPEC 2000 ベンチマークの中から無作為に選んだ 9 種類，すなわち，SpecINT から Gzip, Vpr, Gcc, Mcf2, Crafty, Gap, Bzip2 の 7 種類，SpecFP から Ammp,Equake の 2 種類を用いた．また，評価はプログラムを最後まで走らせて行った．

評価においては以下の 5 つの構成を用いる．すなわち，1) 64 エントリの CAM（MIPS32 の標準仕様），2)2-way RAM 方式で 32 エントリ/way，3)2-way RAM 方式で 64 エントリ/way，4)2-way RAM 方式で 128 エントリ/way，5)2-way RAM 方式で 256 エントリ/way のものである．

各 TLB について TLB ミス率と面積について評価を行う．CPU アーキテクチャは 4-way のスーパースカラとしている．面積については実装した RTL を論理合成し評価する．論理合成には Sysnopsys 社の Design Compiler を使用した．また，RAM は ROHM 0.18μm CMOS プロセスのメモリマクロを使用している．

5.2 評価結果

5.2.1 TLB ヒット率の評価

9 つの SPEC ベンチマークについての各 TLB ごとの TLB ヒット率を図 8 に示す．この結果から，CAM を用いて実装を行った TLB と比較して RAM を用いて実装を行った TLB では TLB ヒット率の低下が起きている事が分かる．CAM を用いた TLB と，それと同様に総エントリ数が 64 となる 2way でエントリ数 32 の RAM を用いた TLB を比較すると，平均 5%程度 TLB ヒット率が低下してしまっている．エントリ数を増やすと TLB ヒット率は増加していきエントリ数 256 の時では，CAM を用いたものと比べ平均約 0.4%のヒット率低下に押さえられており，十分なエントリ数が有れば性能低下はほとんど無視できる物

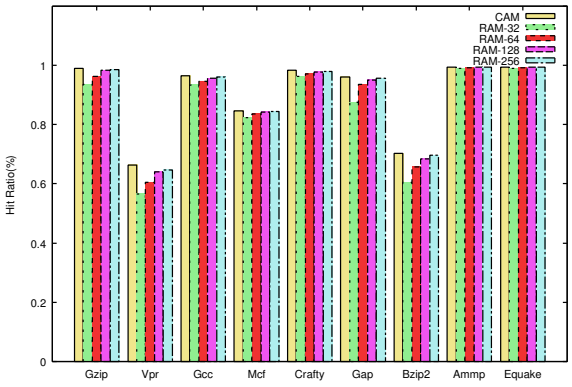

図 8 hit ratio

となっている．しかし，エントリ数 256 においても Vpr では約 3%のヒット率低下が見られる．これは Vpr で RAM の競合が頻繁に発生しているために提案の RAM を使用した TLB が適していないと考えられるので，競合の原因の解析が必要である．

5.2.2 面積の評価

CAM の 64 エントリと RAM の 256 エントリの構成について SystemVerilog を用いて実装し論理合成して TLB 部分の面積を算出した．NAND ゲート換算で CAM 方式が 81,655 ゲート，RAM 方式が 78,458 ゲートとなり，提案手法は CAM と比べて約 4%減となった．

面積は余り減少していないが，現在提案手法の TLB の RTL 実装は最適化がなされておらず，エントリの検索方法やハッシュ値の算出方法などを改良する事でより少ないエントリ数で CAM での実装と同等の性能を得られるため，実際には面積は更に減少するものと考えられる．

6. 結論と今後の展望

本論文では RAM を用いた擬似 CAM 型 TLB の設計手法を提案した．CAM を用いなくなった事で RAM マクロとスタンダードセルによる設計が可能となり、CAM マクロを使用できない環境において CAM 型 TLB の自動設計が容易となった．

しかし，一部のベンチマークプログラムにおいて比較的大きな性能低下が見られたため，性能低下の原因の解析と改良が必要である．

謝辞 本研究は JSPS 科研費 24700047，15K00074 の助成を受けたものであり，東京大学大規模集積システム設計教育研究センターを通し、シノプシス株式会社日本ケイデンス株式会社の協力で行われたものである。

参考文献

[1] K. Pagiamtzis, A. Sheikholeslami, “Content-addressable memory (CAM) circuits and architectures: a tutorial and survey”, IEEE Journal of Solid-State Circuits, Vol.41, No.3, pp.712–727, 2006.
[2] Srilatha Manne, Artur Klauser, Dirk C. Grunwald, Fabio Somenzi: “Low Power TLB Design for High Performance Microprocessors”, Computer Science Technical Reports. Paper 784, 1997.
[3] Hoang Le, Weirong Jiang, V.K. Prasanna, “Scalable high-throughput SRAM-based architecture for IP-lookup using FPGA”, Proc. of the International Conference on Field Programmable Logic and Applications, pp.137–142, 2008.
[4] Jean-Louis Brelet: Using Block RAM for High Performance Read/Write CAMs, Xilinx application note, XAPP204, 2000.
[5] Kyle Locke: Parameterizable Content-Addressable Memory, Xilinx application note, XAPP1151, 2011.
[6] A.M.S. Abdelhadi, G.G.F. Lemieux: “Deep and narrow binary content-addressable memories using FPGA-based BRAMs”, Proc. of the International Conference on Field-Programmable Technology, pp.318–321, 2014.
[7] Brandon H. Dwiel, Niket K. Choudhary and Eric Rotenberg: FPGA Modeling of Diverse Superscalar Processors, Proceedings of the 2012 IEEE International Symposium on Performance Analysis of Systems & Software, pp.188–199, 2012.
[8] S. Lekshmipriya, Suby Varghese: “FPGA Based Architecture for High Performance SRAM Based TCAM for Search Operations”, International Journal of Science and Research, Vol.4, No.2, pp.1862–1867, 2013.
[9] Prajitha P. B.: “SRAM-Based TCAM Architecture for ATM”, International Journal for Science and Advance Research In Technology, Vol.1, No.4, pp.64–67, 2015.
[10] Altera: “Implementing High-Speed Search Applications with Altera CAM”, Altera Application Note 119, 2001.
[11] Xilinx: “Content-Addressable Memory”, Product Specification DS253, 2008.
[12] Altera: “APEX 20KC Programable Logic Device Datasheet”, Altera, 2001.
[13] N. K. Choudhary，et. al: “FabScalar: Composing Synthesizable RTL Designs of Arbitrary Cores within a Canonical Superscalar Template”, Proc. of the ISCA-38, pp. 11–22, June 2011.