修士論文

題目

MIPS32を対象としたTLB機構の改良による設計コスト削減及び高性能化

指導教員

近藤 利夫

平成 27 年度

三重大学大学院 工学研究科 情報工学専攻
コンピュータ・アーキテクチャ研究室

武藤 郡 (414M522)

内容梗概

近年，組み込みプロセッサの高性能化の要求に伴い，複数の命令を同時にフェッチ，実行するスーパースカラが注目を集めている．スーパースカラを研究する上では，プロセッサの設計データである RTL(Register Transfer Level) 記述を容易に変更できるプロセッサデザインが求められている．このようなスーパースカラプロセッサデザインとして FabScalar が提案されている．FabScalar はフェッチ幅や発行キューのサイズ等のパラメータをパラメータファイルとして渡すことで，様々なスーパースカラコアの RTL を生成する．FabScalar は様々な命令セットに対応しているが，本研究ではその中でも研究分野や組込み機器などで広く用いられている MIPS32 ベースのプロセッサに着目する．一般に，汎用プロセッサは仮想メモリを実現するための物理アドレスと仮想アドレスとの変換を高速に行うために，プロセッサ内に TLB(Translation Lookaside Buffer) と呼ばれる高速なバッファを持っている．MIPS32 ISA では TLB を構成するために CAM(Content Addressable Memory) を要求しているが，メモリコンパイラを利用できる RAM とは異なり CAM は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 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，MIPS32 プロセッサ用 TLB のような連想キーのビット幅やエントリ数の多いものをスタンダードセルで設計するのは非現実的である．

そこで本研究ではオリジナルの MIPS32 で用いられている CAM 型 TLB と同じ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 機構を提案している．この手法は設計の困難な CAM を用いず，かつ CAM を用いた場合と比較して性能を殆ど劣化させることなく TLB を実装可能なため設計コストを大幅に削減できるが，MIPS32 ISA の仕様に準拠させているため，オリジナルの MIPS32 よりも高い性能を出すことはできない．一般的に TLB は保持できるエントリの容量を増加させることでヒット率が上がりページウォークによるメモリアクセスの性能低下を回避することができる．提案手法である疑似 CAM 型 TLB 機構は実装効率が高いため，低コストで TLB のエントリ数を増やすことができるが，MIPS32 ISA においては TLB のエントリ数の最大値が 64 と定められているため既存のソフトウェアとの互換性を考慮すると，TLB のサイズを増加させられないという問題がある．そこで，本研究では更に，プログラム実行時に動的にコードの解析を行い，通常の 64 エントリの TLB と同等の振る舞いを実現する MIPS32 ベースのプロセッサ向け高性能 TLB を提案する．提案する TLB は内部的には 64 以上のエントリ数を持つが，命令実行時に使用するレジスタでは 64 エントリ時と同等の値を使用するため従来の OS などのソフトウェアを改変することな

く TLB の性能を向上させることができる．提案 TLB を 4way のスーパースカラコアに実装し評価を行ったところ，MIPS32 オリジナルの CAM を使用する TLB と比較してヒット率を最大約 20%，平均約 6%向上させ，面積を約 29%削減させることに成功した．

三重大学大学院　工学研究科

Abstract

In recent years, superscalar processors that fetch and run multiple instruction in same time attracts attention with the need for high-performance of built-in processor. In order to research and develop a superscalar processor, it is required that designer can change the processor design implemented by HDL (hardware description language) easily. To meet above demand, FabScalar has been proposed. FabScalar generates HDL design of various superscalar core by passing the parameter file which describes parameters such as fetch width, pieline length and issue queue size. However FabScalar supports variety of instruction sets, this study focuses MIPS32 base processor that is widely used in research fields and embedded devices.

General-purpose processor, including MIPS32, has the high speed buffer called TLB(Translation Lookaside Buffer) in the processor to translate virtual address to physical address promptly to support virtual memory mechanism. In MIPS32 ISA, the TLB is required to be constructed by CAM (Content Addressable Memory). However, that has problem that design cost is very high, because design effort of a CAM is very high. In contrast, a RAM can be designed easily using memory macros or a memory compiler. However CAM can be written in register transfer level (RTL) and be synthesized, the area of CAM designed with standard-cell is very large because search key and bit width of TLB are large.

Hence, this study proposes an implementation methodology of a pseudo CAM approach TLB using only standard-cell and RAM macros. This methodology realizes almost same behavior as CAM approach TLB which is used in original MIPS32. However, this methodology compliant to specification of MIPS32. Therefore, it is impossible to achieve better performance than the original MIPS32 TLB. In general, hit rate of TLB raise and memory access performance is improved by increasing TLB entries. Because the proposed pseudo CAM approach is small footprint, TLB entries can be increased with low cost. However, MIPS32 ISA limits the maximum TLB entries to 64, so it is not possible to increase the size of the TLB. To achieve better performance than the original MIPS32 TLB, this study also proposes dynamic code analyzer for more than 64 TLB entries. The proposed approach supports more than 64 entries without existing software/OS modification by analyzing executing code and compensating the difference of behavior between the original MIPS32 TLB and our large TLB. This study implements proposed two approach and evaluates its effectiveness. According to the evaluation results, the pro-

posed approach improves hit rate in max 20%, in average 6%, reduces area by 29%.

目 次

図 目 次

表 目 次

1　はじめに

近年，組み込み分野ではプロセッサの高性能化の要求に伴いスーパースカラが注目を集めている．スーパースカラは複数の命令を同時にフェッチ，実行するプロセッサアーキテクチャで一度に 1 つの命令しか実行できないシングルパイプラインと比べ命令実行の効率を向上させることができる．しかし，スーパースカラは命令間の依存解決やアウトオブオーダー実行の機構などを必要とし構造が複雑なため設計時間が膨大になることが研究を行う上で障害となっている，そのため，スーパースカラに関する研究ではオープンソースなスーパースカラコアのプロセッサデザインが求められている．このようなプロセッサデザインとしてノースカロライナ州立大学で FabScalar[1] が提案されている．FabScalar はフェッチ幅，パイプライン段数，発行キューや各種バッファのサイズ等のパラメータをパラメータファイルとして与えることで任意のスーパースカラコアを自動生成するツールセットである．FabScalar を使用することで様々な構成のスーパースカラコアを自動生成することができるため，設計時間を短縮することができる．FabScalar は様々なアーキテクチャへの対応を目指しているが，本研究では特に研究分野や組み込み機器で広く用いられている MIPS32 を対象とする．MIPS32 は仮想メモリを実現するための物理アドレスと仮想アドレスとの変換を高速に行うためにプロセッサ内に TLB(Translation Lookaside Buffer) と呼ばれる高速なバッファを持っている．MIPS32 ISA では TLB は CAM(Content Addressable Memory)

で設計される仕様となっているが，メモリコンパイラを使用できる RAM とは異なり CAM はそれぞれ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 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，MIPS32 プロセッサ用 TLB のような連想キーのビット幅やエントリ数の多いものをスタンダードセルで設計するのは非現実的である．

そこで本研究ではオリジナルの MIPS32 で用いられている CAM 型 TLB と同じ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 [2] を提案する．この手法は設計の困難な CAM を用いず，かつ CAM を用いた場合と比較して性能を殆ど劣化させることなく TLB を実装可能なため設計コストを大幅に削減できるが，MIPS32 ISA の仕様に準拠させているため，オリジナルの MIPS32 よりも高い性能を出すことはできない．一般的に TLB は保持できるエントリの容量を増加させることでヒット率が上がりページウォークによるメモリアクセスの性能を向上させることができる。提案手法は，実装効率が高いため，オリジナルの CAM 型 TLB と比べてのエントリ数を低コストで増やすことができるが，MIPS32 ISA においては TLB のエントリ数の最大値が 64 と定められているため既存のソフトウェアとの互換性を考慮すると，TLB のサイズを増加させられないという問題がある．そこで，本研究では更に，プログラム実行時に動的にコードの解析を行い，通常の

64 エントリの TLB と同等の振る舞いを実現する MIPS32 ベースのプロセッサ向け大容量 TLB[3] を提案する．提案する TLB は内部的には 64 以上のエントリ数を持つが，命令実行時に使用するレジスタでは 64 エントリ時と同等の値を使用するため従来の OS などのソフトウェアを改変することなく TLB の性能を向上させることができる．提案 TLB を 4way のスーパースカラコアに実装し評価を行ったところ，従来の CAM を使用する TLB と比較してヒット率を最大約 20%，平均約 6%向上させ，面積を約 29%削減させることに成功した．

以降，本稿は次のように構成される．まず，第 2 節では本研究のので改良を行う TLB について述べ，第 3 節で本研究で対象としている MIPS32 ベースのプロセッサの TLB の仕様について述べる．第 4 節では既存の CAM を RAM に置き換えるための研究について述べ，第 5 節では RAM を用いた疑似 CAM 型 TLB の実装手法を提案し，第 6 節で擬似 CAM 型 TLB の性能評価を行う．第 7 節では動的コード解析を用いた MIPS32 ベースプロセッサ向け高性能 TLB の提案し，第 8 節で高性能 TLB の性能評価を行う．最後に，第 9 節でまとめについて述べる．

2　背景

2.1　MMU(Memory Management Unit)

MMU はアドレス変換やメモリ保護の機能を持つ，仮想記憶を実現するためのユニットである．仮想記憶とは図 2.1 のように物理アドレスやハードディスクに対して仮想的なアドレスを割り振るメモリ管理手法の一種である．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．この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．しかしメモリアクセスのたびにページテーブルを引くとオーバーヘッドが膨大なものとなってしまう．MIPS32 ISA ではこの問題を解決するために TLB が使用される．

2.2　TLB(Translation Lookaside Buffer)

TLB はプロセッサ内部に搭載される高速な小容量のバッファであり，一種のキャッシュメモリとして動作する．プロセッサは主記憶上の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．TLB の実装手法はエントリの保持に CAM を用いる CAM 型と RAM を用いる RAM 型が存在する．MIPS32 ISA はこの内 CAM 型に当てはまるため，TLB の実装時に CAM が必要となる．しかし，CAM はメモリコンパイラで設計可能な

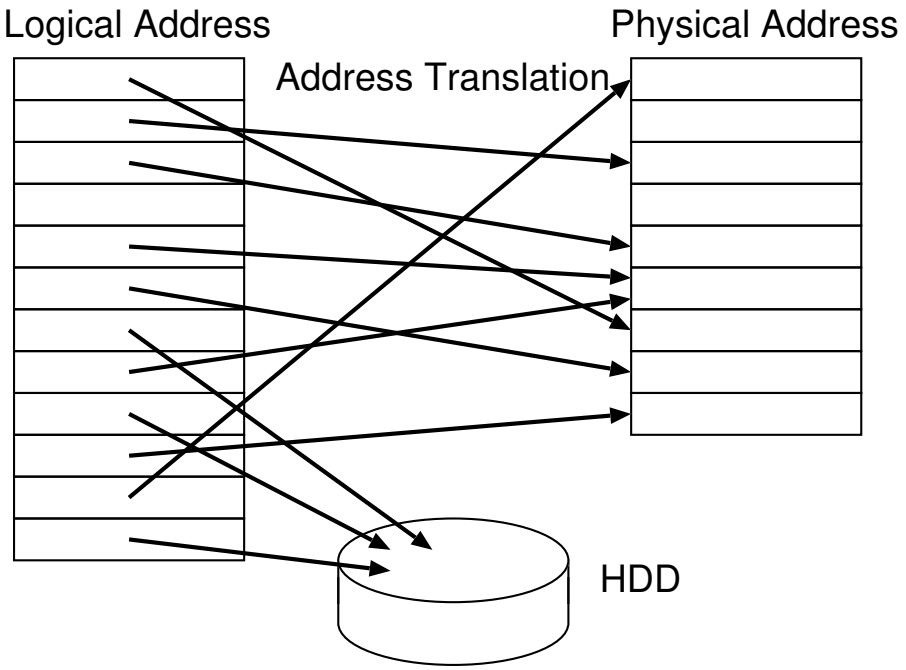

図 2.1: アドレス変換

RAM とは異なりそれぞれ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 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，MIPS32 プロセッサ用 TLB のような連想キーのビット幅やエントリ数の多いものをスタンダードセルで設計するのは非現実的である．そこで本研究ではオリジナルの MIPS32 で用いられている CAM 型 TLB と同じ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．

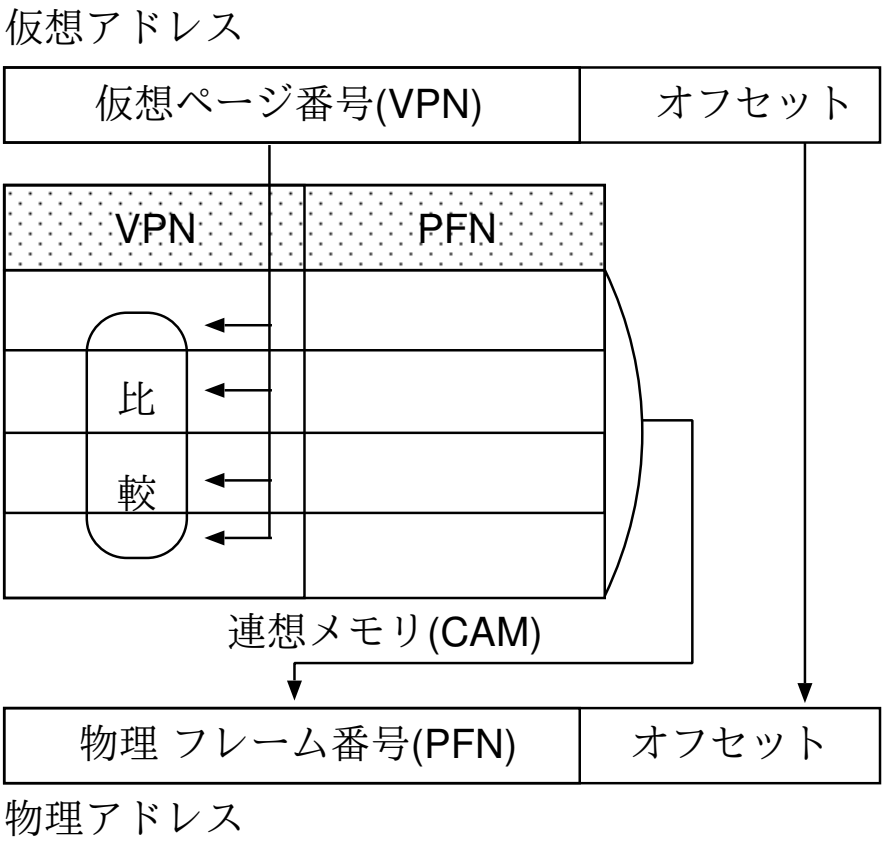
仮想アドレス
仮想ページ番号(VPN)
オフセット
VPN
PFN
比
較
連想メモリ(CAM)
物理 フレーム番号(PFN)
オフセット
物理アドレス

図 2.2: TLB

3 MIPS32 ISA

3.1 MIPS32 ISA の TLB の仕様

まず本研究で対象とする MIPS32 ISA における TLB 関連の仕様について述べる．MIPS32 ISA では TLB は最大 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特殊なレジスタと OS のみが使用できる特権命令を持っている．

TLB 関連の特殊レジスタ

- Random レジスタ

  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 レジスタ

  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 レジスタ

  仮想ページ番号を格納するレジスタ.

- EntryLo レジスタ

  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 レジスタで指定するエントリに EntryHi レジスタと

EntryLo レジスタの書き込みを行う.

- TLBWR

  TLB の Random レジスタで指定するエントリに EntryHi レジスタと EntryLo レジスタの書き込みを行う.

- TLBR

  TLB の Index レジスタで指定するエントリを EntryHi レジスタと EntryLo レジスタに読み出す.

- TLBP

  EntryHi レジスタ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 レジスタに書き込む.

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 レジスタ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 レジスタのエントリを読み込む．読み込んだエントリを EntryLo レジスタに書き込み，TLBWR 命令を実行することによって Random レジスタで指定するエントリに TLB ミスが発生した VPN(Virtual Page Number) と，それに対応

する PFN(Page Frame Number) 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 レジスタと同じ値が入っているエントリ番号を Index レジスタに書き込み，その後 TLBP が見つけた位置に TLBR 命令を実行する事で EntryHi レジスタ，EntryLo レジスタにエントリを読み出す事が出来る．

3.2 MIPS32 ISA の TLB の動作

通常の MIPS32 ISA における TLB のブロック図は図 3.3 のようになる．TLB が使用される状況は大きく分けてロード/ストア命令やフェッチによるメモリアクセスの際のアドレス変換時と TLB 関連の特権命令実行時である．1 つ目のメモリアクセスが発生する際には，仮想アドレスから物理アドレスへの変換が必要となる．TLB は仮想アドレスを渡されるとまずそのアドレスを VPN と offset に分割する．分割された VPN は CAM 内の全てのエントリの VPN と比較され，一致したエントリが有れば対応する PFN を返す．返ってきた PFN と offset を合わせることで物理アドレスが生成される．また，一致するエントリが見つからなかった場合は TLBHit が 0 となり TLB ミス例外が発生し例外処理ルーチンによって TLB のリプレースが行われる．2 つ目の特権命令実行時は命令によって異なる動作

を行う．TLBP 命令は EntryHi レジスタと CAM 内の全てのエントリの VPN を比較し一致した箇所が有ればインデックス番号を Index レジスタとして返す．TLBR/TLBWI/TLBWR 命令はそれぞれ Index/Random レジスタを使用して CAM の指定した箇所に対する読み出し又は書き込みを行う．

MIPS32 ISA ではこのように TLB が CAM で実装されていることを前提とした設計がなされているため CAM の使用が避けられず，CAM を手設計することによる設計コストや，D-flip flop で実現することによる膨大な面積が問題となっていた．そこで我々はこのような 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．

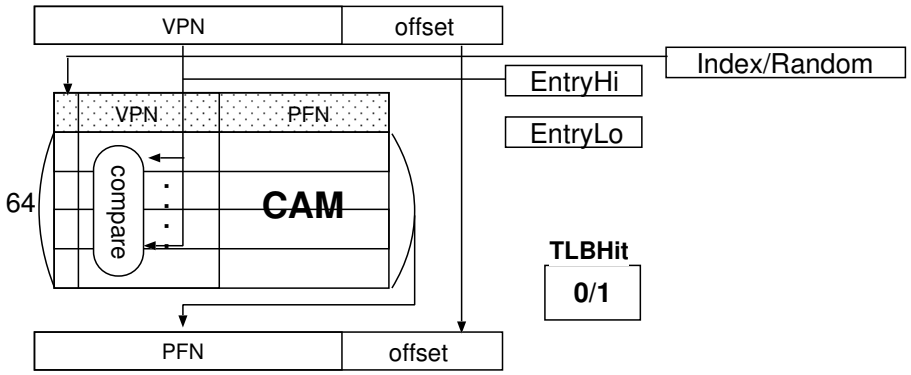

図 3.3: MIPS32 ISA の TLB

4　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 [4]．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 [5, 6, 7] 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 [8] 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[1] 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 [8] では，文献 [5, 6] の手法を用いて CAM を実装している．しかしながら，文献 [8] 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 [9, 10] 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 [11, 12]．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 [13]．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．

5　RAMを用いた疑似CAM型TLBの実装手法の提案

5.1　擬似CAM型TLBの構成

RAMを用いた擬似CAM型TLB機構は図5.4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．まずメモリアクセス発生時のアドレス変換時の動作について述べる．

擬似CAM型TLBは仮想アドレスを渡されるとVPNとオフセットに分割する．つぎに分割されたVPNをハッシュ関数に与えることでハッシュ値を得る．このハッシュ値がRAMのインデックスとして用いられる．求められたインデックスでRAMを引き，得られたVPN0，VPN1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を保持しているかの判定を行う．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，MIPS32 ISAでは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TLB関連の特権命令が正常に動作しないことがある．

例えば，前述のTLBWI命令はIndexレジスタ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5.5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 TLBWI 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．データアクセスによる 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するという動作を繰り返してしまうため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．

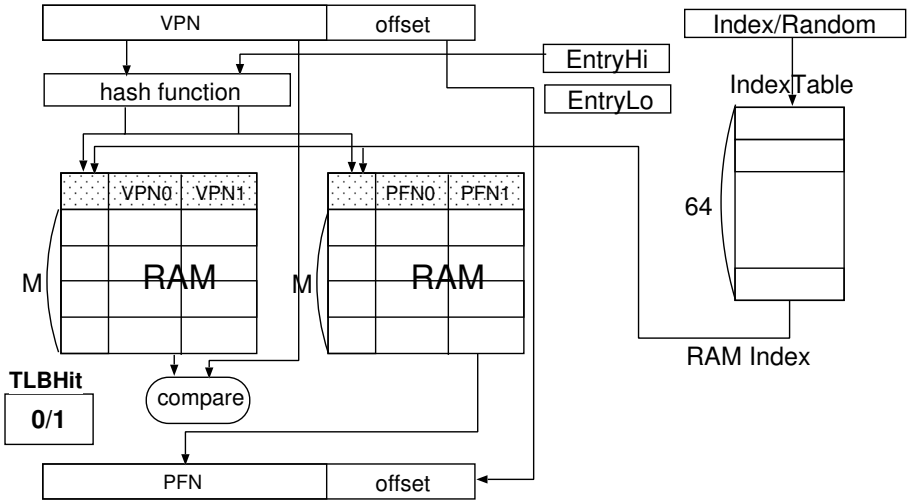

図 5.4: RAM を用いた擬似 CAM 型 TLB

5.2 2-way RAM 方式

2-way RAM 方式では図 5.6 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

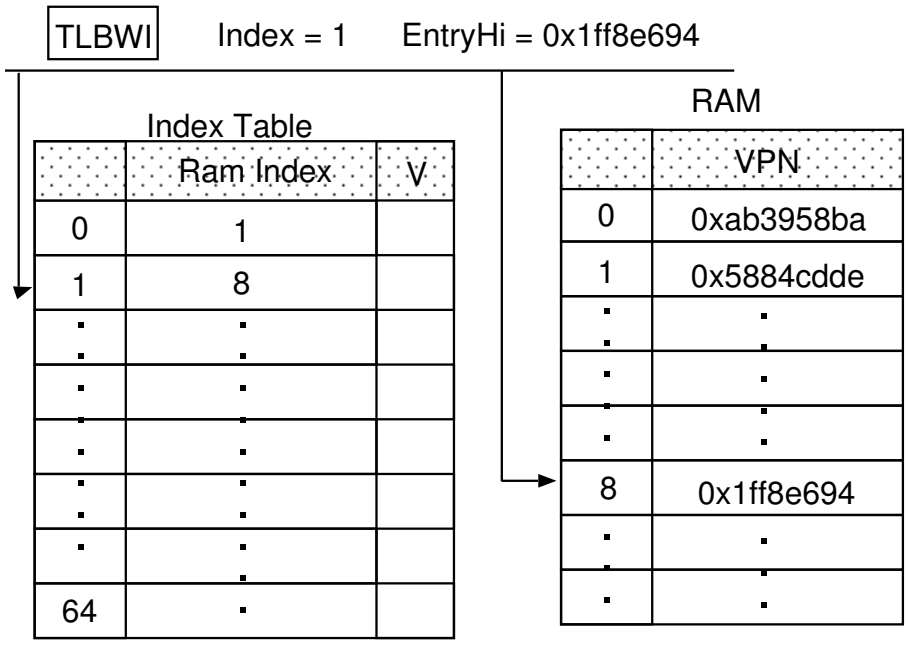

図 5.5: インデックステーブル

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．これにより命令とデータアクセスのアドレスのハッシュ値が一致しても両方のエントリを TLB に保持できるようになり正常な動作が実現する．

6　擬似 CAM 型 TLB の性能評価

提案手法の擬似 CAM 型 TLB とオリジナルの CAM を用いて実装された TLB についてそれぞれ性能，面積評価を行う．以下第 6.1 節にて評価

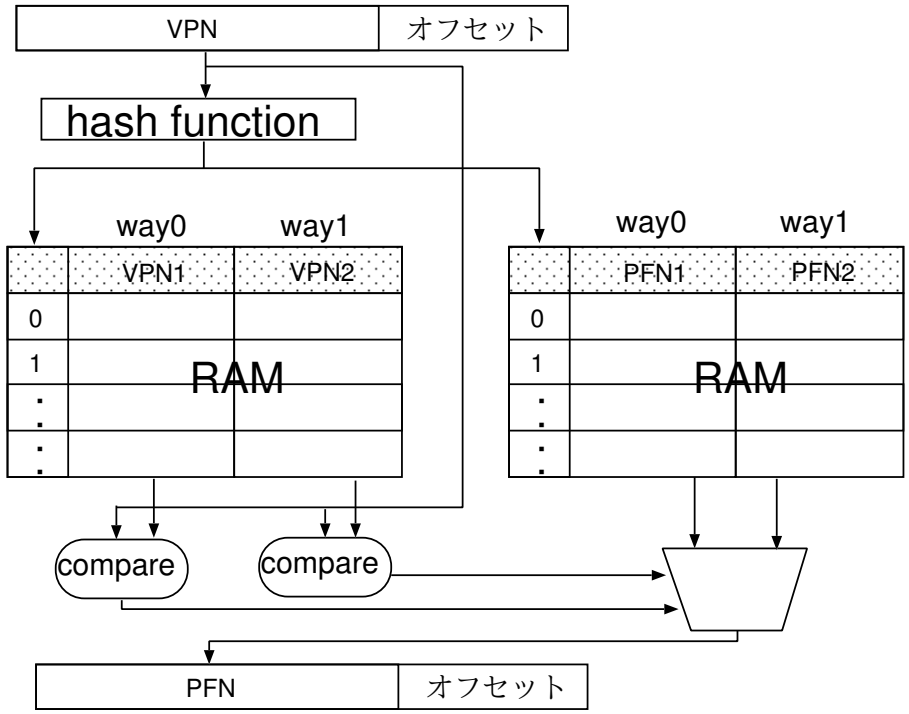

図 5.6: 2-way RAM

環境の詳細について述べ，第 6.2 節にて評価結果について述べる．

6.1 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 6.7 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．

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 ヒット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

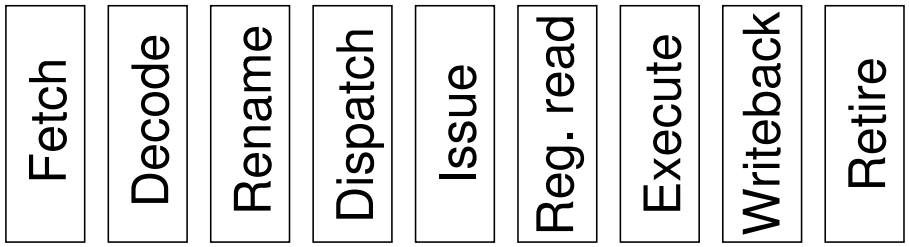

図 6.7: Canonical superscalar processor.

表 6.1: 使用した EDA/CAD ツール

工程	ベンダ名	ツール名	バージョン
Functional verifiation	Cadence	NC-Verilog	12.20
Synthesis	Synopsys	Design Compiler	2013.03-SP2
Power estimation	Synopsys	PrimeTime PX	D-2010.06

メモリマクロを使用している．表 6.1 に使用した EDA/CAD の詳細を掲載する．

6.2 評価結果

6.2.1 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 6.8 に示す．この結果から，オリジナルの CAM を用いた TLB と比較して提案の RAM を用いた擬似 CAM 型 TLB では TLB ヒット率の低下が起きている事が分かる．オリジナルの TLB と，それと同様に総エントリ数が 64 となるエントリ数 32×2 の擬似 CAM 型 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 128 の時では，CAM を用い

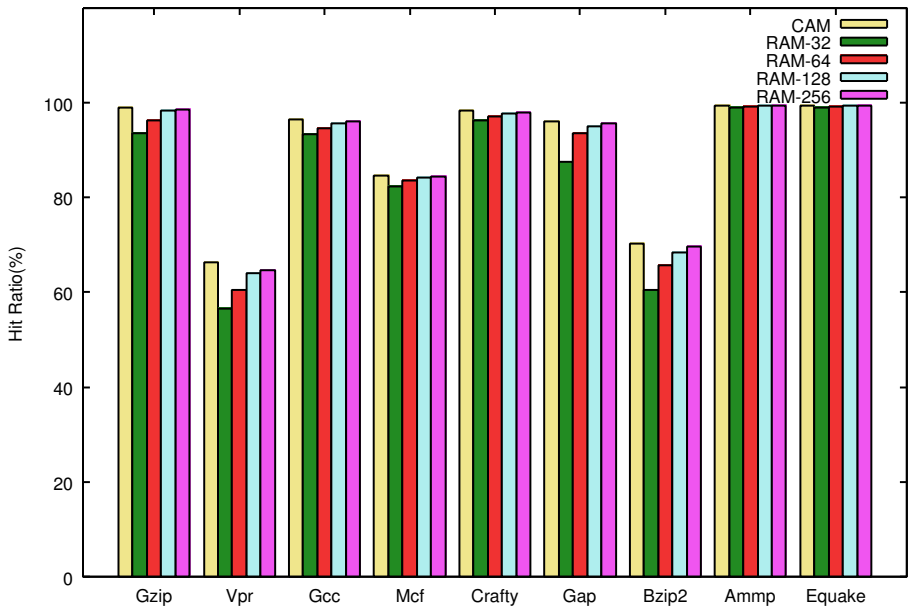

図 6.8: 擬似 CAM 型 TLB のヒット率

たものと比べ平均約 1%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となっている．しかし，Vpr ではエントリ数を 256 に増やしても約 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．

6.2.2 面積の評価

CAM の 64 エントリと RAM の 128 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 44,039 ゲートとなり，

提案手法は CAM と比べて約 46%減となった．

6.2.3　擬似 CAM 型 TLB の有効性及び問題点

提案の擬似 CAM 型 TLB はヒット率の低下を 1%程度に抑え，かつ面積を 46%削減することに成功している．この結果から擬似 CAM 型 TLB は小面積に TLB を実装できることが分かる．つまり，本研究の評価結果より，提案手法を用いることで ASIC や FPGA 等の設計フローにおいて，オリジナルの MIPS32 と同等の性能の TLB を少ないハードウェア資源で実現できることを明らかにした．

しかし，この手法は MIPS32 ISA の仕様に準拠しているため，オリジナルの MIPS32 で使用される TLB の性能を上回ることができないという制限があった．そこで本研究ではプログラム実行時に動的にコードの解析を行い，通常の 64 エントリの TLB と同等の振る舞いしつつも高いヒット率を実現する MIPS32 ベースのプロセッサ向け高性能 TLB を提案する．

7 MIPS32ベースプロセッサ向け高性能TLBの提案

前述の通り，疑似 CAM 型 TLB はオリジナルの MIPS32 と同等の性能を少ないハードウェア量で実現できるが，MIPS32 ISA の仕様上，TLB のヒット率向上には限界がある．提案手法は TLB を小面積で実装できるため，エントリ数を増加させることも容易にできるが，MIPS32 ISA の仕様から外れるため，既存のソフトウェアや OS が動作しなくなるという問題がある．そこで，本章では MIPS32 の ISA を変更することなく，TLB のヒット率を向上させるため，TLB の大容量化及び動的コード解析を用いた高性能 TLB を提案する．

7.1 高性能TLBの構成

提案手法の高性能 TLB は図 7.9 のように構成される．TLB の本体部分であるページエントリを格納する箇所は図 5.4 の TLB と擬似 CAM 型 TLB と同様に RAM が使用される．RAM は VPN を格納するものと PFN を格納するものの 2 つあり，それぞれ任意の M エントリを持つ 2way の RAM になっている．そのため，RAM 内のエントリにアクセスする時にはエントリ番号とともにウェイの番号も指定する必要がある．提案する TLB では高性能化のために，CAM と RAM でのインデックスの齟齬を解消するためのインデックステーブルの容量が 64 × セット数 N(N は任意の数) をとしている．また，この容量の増加に伴い発生する幾つかの問題

に対応するためにオーダーアナライザーというモジュールを実装している．以下でTLBが使用される際の高性能TLBの動作を述べる．ロード/ストア命令やフェッチによるメモリアクセスが発生する際には通常のTLBと同様に仮想アドレスをVPNとoffsetに分割する．分割されたVPNはハッシュ関数に渡され，得られたハッシュ値を用いてVPN，PFNのRAMにアクセスする．次に，RAMから得られたVPN0，VPN1と変換対象のVPNとを比較する．図3.3のものとは異なり比較するエントリは2つで済んでいる．VPNがVPN0，VPN1のどちらかと一致していれば，一致していた方のウェイ番号を取得する．RAMから得られたPFN0，PFN1の内VPNが一致していたウェイ番号の方をPFNとして返し，offsetと結合することで対応する物理アドレスが得られる．また，一致するエントリが見つからなかった場合は通常のTLBと同様に，TLBHitを0としTLBミス例外が発生し例外処理ルーチンによってTLBのリプレースが行われる．

7.2 TLB関連の特権命令実行時の動作

次に高性能TLBでMIPS32 ISAの特権命令であるTLBP, TLBWI, TLBWR, TLBRの4命令を処理する手法について説明する．

7.2.1 TLBP

TLBP 命令は図 7.10 に示した箇所によって実行される．TLBP 命令は EntryHi レジスタで指定した VPN が TLB 内に有るかどうか検索し，有った場合はそのエントリのインデックス番号を Index レジスタに返す．提案 TLB ではエントリの保持に RAM を使用しているため前述のインデックステーブルが実装されてされている．通常 MIPS32 ISA における TLB の CAM のサイズは 64 となるため，インデックステーブルのサイズも 64 となるが，提案 TLB ではこのサイズを 64 × セット数 N(N は任意の数) としている．TLBP 命令を実行する際には，まず (1) で EntryHi レジスタがハッシュ関数に渡され，ハッシュ値を得る．次に得られたハッシュ値を用いて VPN0,VPN1 を取得する．そして，(2) で EntryHi レジスタと VPN0,VPN1 が比較され一致しているものがあれば RAM のアドレス (EntryHi レジスタによるハッシュ値) とウェイ番号を得る．このアドレスは RAM のインデックスであるため，第 5 項で述べた CAM インデックステーブルを使用して，(3) で CAM でのインデックスに変換を行う．また高性能 TLB では CAM に相当するサイズは 64 × N となっているため，CAM インデックステーブルは RAM のインデックスに対応する CAM のインデックスだけではなくウェイ番号も書き込むようにする．これにより TLBP 命令はヒットした位置の CAM のインデックスとウェイ番号を得ることができる．しかし Index レジスタに書き込まれる値は 64 を超えることはできないためヒットしたウェイ番号を Index レジスタに書き込

むことはできない．そこで高性能 TLB にはオーダーアナライザーに Way Buffer が搭載されている．Way Buffer には TLBP 命令による検索がヒットした場合，(4) で RAM インデックステーブルでの変換後のウェイ番号が書き込まれる．TLBR 命令による Index レジスタを用いた TLB へのアクセスの際にはこの Way Buffer を用いることで TLBP 命令がヒットした箇所へのアクセスが可能となる．これにより TLBP 命令の実行が正常に行われる．

7.2.2 TLBWI/TLBWR

また，TLBWI 命令は図 7.11 に示した箇所によって実行される．TLBWI 命令が実行されるとまず，(1) で EntryHi レジスタがハッシュ関数に渡され，ハッシュ値を得る．得られたハッシュ値を用いて，(2) で VPN0 または 1 に EntryHi レジスタを，PFN0 または 1 に EntryLo レジスタを書き込む．この際の 0 か 1 かの決定は LRU に従って古いものに書き込みを行う．次に，(3) でインデックステーブルの更新を行う．ここでは Index レジスタをアドレスとし，書き込むウェイの選択には LRU を用い古いものを優先している．RAM のインデックスとして EntryHi レジスタをハッシュ関数に通して得られたハッシュ値を書き込むデータとする．また，同時に CAM インデックステーブルの更新も行う．これにより TLBWI 命令の実行が正常に行われる．

TLBWR 命令は TLBWI 命令の Index レジスタを Random レジスタに置き換えることで同様に実行ができる．

7.2.3 TLBR

TLBR 命令の実行ではまず Index レジスタと前述した Way Buffer を用いてインデックステーブルを引き，RAM のインデックスとウェイ番号を得る．次に得られたインデックスとウェイ番号で RAM を引き，VPN と PFN を取得する．最後に VPN と PFN を EntryHi レジスタ,EntryLo レジスタに書き込み TLBR 命令の実行が完了する．

7.3 フルフラッシュへの対応

オーダーアナライザーの持つ機能の一つとしてフルフラッシュへの対応がある．フルフラッシュとは TLB が持つエントリを全て無効化する動作であり，0 から 63 まで Index レジスタの値をインクリメントしながら TLBWI 命令を繰り返し実行することによって全エントリを上書きすることによって実現される．既存の OS などにはこのフルフラッシュを行うためのコードが入っている．しかし，高性能 TLB では TLBWI によってアクセスされるエントリ，すなわちインデックステーブルの容量が $64 \times N$ であるため 0 から 63 までのループだけでは残りのが $64 \times (N-1)$ の無効化を行うことができない．そこで，オーダーアナライザーはこの問題を解決するための機構を持っている．TLBWI による TLB の更新があると，

オーダーアナライザーはIndexレジスタの値をLastIndexに書き込む．また，書き込む際にLastIndexとIndexレジスタを確認し，Indexレジスタが$LastIndex+1$であればUpdateNumをインクリメント，そうでなければUpdateNumを0にする．UpdateNumが64になれば，連続した64のエントリに対する更新，つまりフルフラッシュが起きたことが分かるため，TLBの全エントリのvalidを落とすことにより従来のソフトウェアを改変することなくフルフラッシュに対応している．

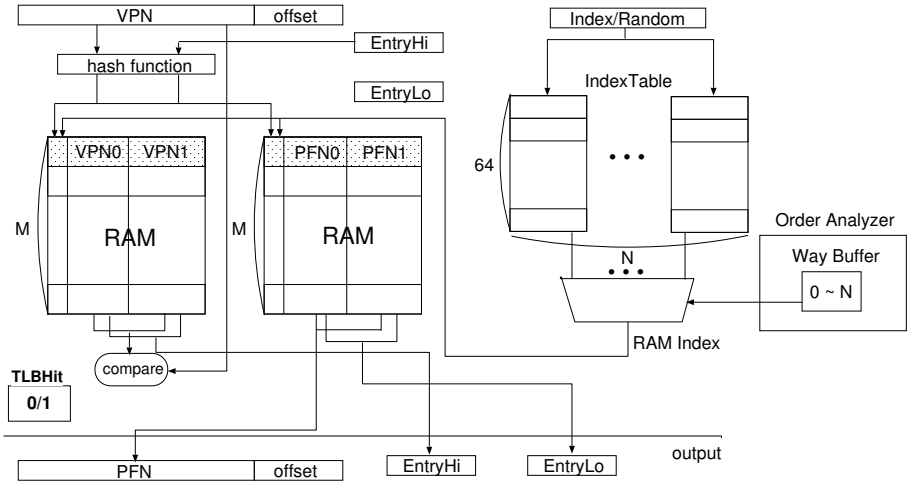

図 7.9: 動的コード解析による高性能TLB

8　性能評価

高性能TLBと，MIPS32 ISAのオリジナルの64エントリのTLBについてそれぞれ性能，面積評価を行う．以下第8.1節にて評価環境の詳細について述べ，第8.2節にて評価結果について述べる．

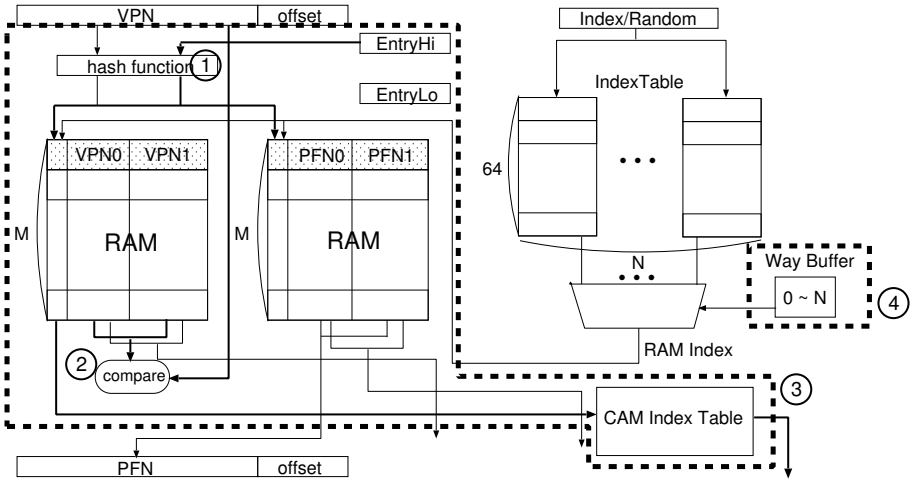

図 7.10: TLBP の動作

8.1 評価環境

評価環境は 6.1 節と同様である．

評価には，64 エントリ（MIPS32 の標準仕様），RAM サイズ 128 かつインデックステーブル 64 エントリ 2way(提案手法) の 2 種の構成を用いる．

8.2 評価結果

8.2.1 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 8.12 に示す．エントリ数を 2 倍にした提案手法の TLB は，全てのベンチマークでヒット率が向上していることが分かる．通常の 64 エントリのものと比べて最大では約 20%，平均で約 6%のヒット率の向上が得られた．特に元々のヒット率が低かった Vpr，Bzip2 では 20%以上大きなのヒット

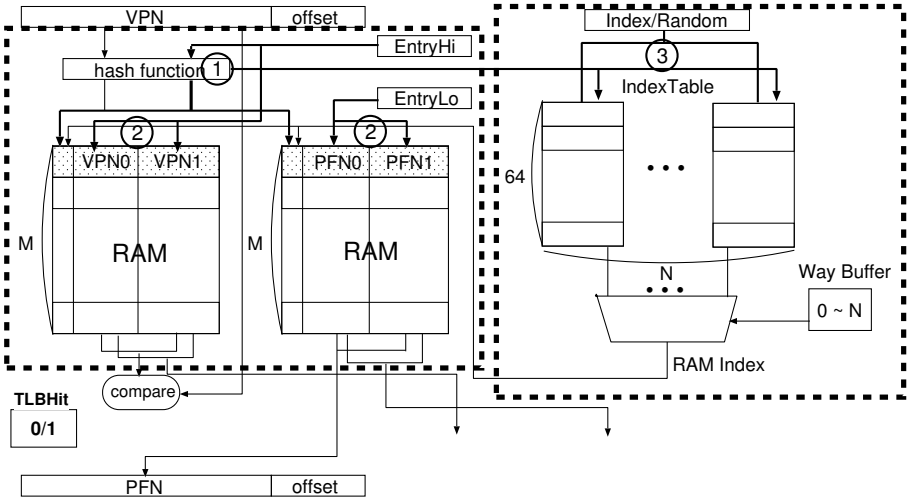

図 7.11: TLBWI の動作

率の向上が見られ，提案手法がヒット率の低いプログラムに効果が高いことが確認された．しかし元々のヒット率が高いプログラムに対しては増加したロジックと比較してそれほどの効果が出ていないため，そういったプログラムの実行時には 1 セットのみを用いることで電力消費の削減が見込まれる．

8.2.2 面積の評価

それぞれの構成について 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 58,284 ゲートとなり，提案手法は CAM と比べて約 29%減となった．

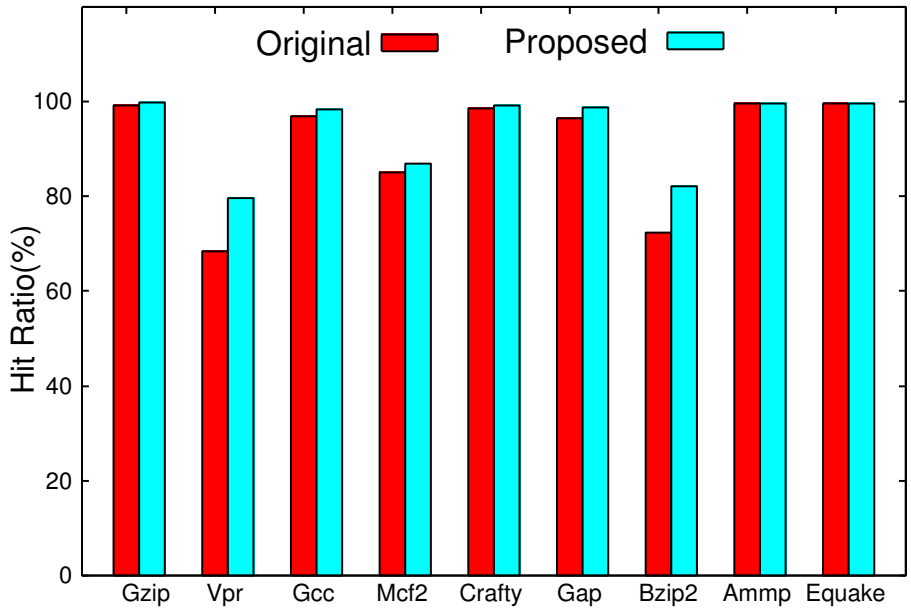

図 8.12: 動的コード解析による大容量 TLB のヒット率

9　まとめ

本研究ではMIPS32を対象としたTLB機構の設計コスト削減及び高性能化を行った．MIPS32にはTLBの高性能化のための容量増加に当たって，CAMを使用しているために設計コストが膨大になることと，ISAの仕様上ソフトウェア等の書き換え無しに容量を増加させられないという二つの課題があった．そこで設計コスト削減のために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機構を提案した．またソフトウェアの書き換え無しに容量の増加を実現するために，動的にコードの解析を行い通常の64エントリのTLBと同等の振る舞いを実現するMIPS32ベースのプロセッサ向け高性能TLBを提案した．提案する2つの手法を適応したTLBを4wayのスーパースカラコアに実装し評価を行ったところ，従来のCAMを使用するTLBと比較してヒット率を最大約20%，平均約6%向上させ，面積を約29%削減させることに成功した．このように，本論文ではMIPS32プロセッサを従来手法では実装が困難であったASICやFPGAの設計フローにおいても，同等以上の性能，かつ小面積で実装できる手法を明らかにした．

謝辞

本研究を遂行するにあたり，日頃から御指導，御助言を頂きました近藤利夫教授，佐々木敬泰助教，深澤祐樹研究員に感謝いたします．また，研究に協力していただいた計算機アーキテクチャ研究室の方々に感謝の意を表します．

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] 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.

[2] 武藤郡, 佐々木敬泰, 深澤祐樹, 近藤利夫. (2015). 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. 研究報告計算機アーキテクチャ (ARC), 2015(40), 1-6.

[3] 武藤郡, 佐々木敬泰, 深澤祐樹, 近藤利夫. MIPS ベースプロセッサの TLB 機構の改良による高性能化. 研究報告 コンピュータシステム研究会 (CPSY), 115, 13-18, 2015

著者, タイトル, 掲載誌, 巻, ページ, 発行年月

[4] 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.

[5] Jean-Louis Brelet: Using Block RAM for High Performance Read/Write CAMs, Xilinx application note, XAPP204, 2000.

[6] Kyle Locke: Parameterizable Content-Addressable Memory, Xilinx application note, XAPP1151, 2011.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] Altera: "Implementing High-Speed Search Applications with Altera CAM", Altera Application Note 119, 2001.

[12] Xilinx: "Content-Addressable Memory", Product Specification DS253, 2008.

[13] Altera: “APEX 20KC Programable Logic Device Datasheet”, Altera, 2001.