修 士 論 文

ページアドレス予測によるTLBプリローディングの研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

請園 智玲

2003年3月

修 士 論 文

ページアドレス予測によるTLBプリローディングの研究

指導教官　田中清史 助教授

審査委員主査　田中清史 助教授
審査委員　日比野靖 教授
審査委員　堀口進 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

110019 請園 智玲

提出年月: 2003 年 2 月

概 要

現代のプロセッサは，仮想記憶実装によるオーバヘッドをページテーブルのキャッシュといえる TLB をハードウェアで用意することで軽減している．しかし許容量を超えるページを使う処理を実行する場合，TLB がうまく機能しない状況が起こる． 本研究では次にプログラムが発生させるページアクセスを予測し，あらかじめメモリからプリロードすることによって TLB の記憶容量を軽減しつつ TLB ヒット率を向上させる機構を提案する．

目 次

図 目 次

表 目 次

第1章　はじめに

今日のマイクロプロセッサは仮想記憶をサポートすることが一般的である．仮想記憶は計算機上の実メモリ容量を越えるプログラムを実行するための機構である．仮想記憶の特徴を以下に示す．

1. プログラムの大きさが，主記憶装置容量の制約を受けない．
2. プログラムは，OS によりページあるいはセグメントとよばれる単位に分割される．
3. 主記憶装置の空いている部分，ないしは空けられる部分を探して，仮想記憶装置から必要とされるページあるいはセグメントの内容を取り出して割り当てることができる．
4. 主記憶装置と二次記憶装置の間のプログラムの取り出しや格納はシステムがページ単位あるいはセグメント単位に分けて行う．

プログラムはその特性上，一定時間で頻繁にアクセスするメモリ領域に局所性持つ．仮想記憶とはこの局所性を利用し，頻繁にアクセスされるメモリ領域を実メモリに残し，それ以外のメモリ領域を二次記憶装置に置くことによって実メモリ容量を越えるプログラムの実行を可能とする．また，仮想記憶の他の利点として仮想アドレス概念導入によるプログラムの動的再配置の実現とメモリ資源の効率化／メモリ保護の実現が挙げられる．動的再配置が実現されることにより，プログラマレベルでメモリ管理を行う必要が無くなり，プログラム開発効率が飛躍的に向上すると共に，マルチプログラム実行環境とマルチユーザ環境におけるメモリ保護を容易に実現している．また，複数プログラム実行により，常にメモリ使用状況が動的変化する場合において柔軟かつ効率的に実メモリ資源を利用することが可能となる．通常，計算機上における仮想記憶環境は OS が提供するものであるが，OS が仮想記憶環境を構築するためにはハードウェアサポートが不可欠である．本論文では，この仮想記憶をサポートするハードウェア機構に着目し，その特徴と問題点を検証し，より効率的に仮想記憶環境をサポートするハードウェアの提案を行う．

近年，半導体技術の発展によりマイクロプロセッサの動作周波数の向上，計算機システムのメモリの大容量化が成されてきた．それに伴い，実行するプログラムのワーキングセットサイズが増加している．この増加は仮想記憶を実現する計算機システムにおいてプログラム実行効率上重大なボトルネックになる．仮想記憶をサポートするプロセッサは通常，TLB（Translation Lookaside Buffer）を備えている．TLB は仮想記憶におけるページテーブルをプロセッサ内にキャッシュし，アドレス変換および保護を高速化する機構である．TLB の性能を決定する要素として，一度にマップすることができるメモリの範囲

（TLB リーチ）がある．プログラムが TLB リーチを超えるサイズのワーキングセットを扱う場合，TLB 内のエントリの入れ替えが必要となり，これが頻繁になるとプログラム実行に大きな影響を及ぼす．TLB ミスによる実行時間損失が 12.5%，キャッシュミスを考慮すると 20%に近づくアプリケーションが存在すると報告されている [1].

TLB リーチ増大のために TLB エントリ数を増加あるいはページサイズを拡大することは，動作周波数の低下やメモリフラグメンテーションを引き起こす可能性が大きくなる．これまで，TLB の性能を向上させるために選択的にページサイズを大きくすることにより，メモリフラグメンテーションを抑える方式（スーパーページ [2]，subblock TLB[3]）が研究されてきた．しかしこれらの方法は，ページフォルト時にスーパーページの作成（ページプロモーション）に要するオーバーヘッドが存在し，また各エントリにページサイズや有効ビットを付加する必要があり，ハードウェア量を増大させる傾向がある．

本研究では TLB リーチ増加による TLB 性能向上という方法を採らず，次にアクセスされるページを予測し，あらかじめページテーブルからロードしておくプリロード機構を新たに TLB に付け加えることにより，エントリ数およびページサイズに対するヒット率依存が小さい TLB 機構を提案し，評価を行う．

本論文 2 章では最初に提案するページアドレス予測について説明する．その上で本論文において提案，評価を行った線形ページアドレス予測機構の詳細を説明する．3 章では線形ページアドレス予測機構の完全線形ページアクセス予測の制限を緩和するための線形ページアドレス予測機構の非線形対応構成法について意義とその詳細を説明する．4 章では本論文において設計を行った CPU の詳細を説明する．5 章では設計したハードウェアの評価方法と実験の方法論について論じる．6 章では提案機構の評価と示す．7 章では本論文の関連研究を紹介する．8 章で本論文のまとめを行う．

第2章　ページアドレス予測

ページアドレス予測機構を実現するためには，適切な予測方針が必要となる．本章ではページアドレスを予測を行うための予測方針として 2.1 で履歴予測と傾向予測について論じ，具体的予測方針を論ずる．その上で傾向予測に着目した線形ページアドレス予測を提案する．2.1.3 では線形ページアドレス予測機構の具体的実装方法について言及し，2.2 では本機構の特徴といえるバッファへのプリロードについてその利点と従来の TLB との相違点を論じる．

2.1　履歴予測と傾向予測

本論文は予測機構を実現するための方針を**履歴予測**と**傾向予測**の 2 つに分割して考える．

2.1.1　履歴予測

通常，マイクロプロセッサ内部で予測機構を実現する場合には，過去の状態を記録しておくハードウェア的なテーブル (予測テーブル) を用意し，それを参照することによって実現する．動的分岐予測機構はその代表的一例として挙げられる．これをページアドレス予測に適用する場合，同様に予測テーブルを用意しておく必要がある．これを本研究において履歴予測という．履歴予測の特徴はプロセッサが予測テーブルの能力で遡ることが可能な遠さの過去と同じ振る舞いをする場合に，高確率で予測が的中する点にある．予測テーブルは，通常 SRAM で構成されたメモリを使うことからその容量が直接ハードウェア量と遅延量に関係する．そのため予測テーブルの記憶容量には限界が存在する．従って履歴予測は用意できる予測テーブルの容量に予測的中率が依存する．履歴予測を命令ページアクセスとデータページアクセスに分割して考察する．

命令ページアクセス　一般的なプログラムの命令メモリ消費の総容量はデータメモリ消費の総容量に比べ小さい．そのことから予測に使用される予測テーブルも小容量で実現が可能となる．本節では更に予測テーブルを小さくする手法として動的分岐予測機構の予測テーブルを利用した履歴予測機構を提案する．命令アクセスでページアクセスが遷移する状況は 2 つある．1 つは，1 命令実行を終了し次の命令をフェッチするため，PC(Program Counter) が命令サイズ分インクリメントされ，前回フェッチした命令のアドレスと今フェッチしようとする命令のアドレスがページ境界を越え

ている場合である．2つめは分岐／ジャンプ命令でページ境界を越えた命令へフェッチをかける場合である．32 ビット命令長，4KB ページサイズの場合，1 の状況のみが起こる場合において，命令アクセス時のページ遷移の確率は 1/1024 である．それに対し 2 の状況はプログラマ又はコンパイラ依存であるため確率を計算することができないが，実行プログラムの割合の殆どループ構造の実行で占められることから，1 の状況より頻繁に起こることが考えられる．動的分岐予測は分岐命令の飛び先アドレスを格納した予測テーブルを持っていることから，この予測テーブルを利用し，ページ境界を越える分岐先アドレスを抽出しプリロードしておく機能を有することで，命令ページアドレス予測の予測テーブルを縮小することが可能である．

データページアクセス データページアクセスはプログラマ依存であると共に実行時に動的に変化するアクセスである．ページアクセス先がスタティックデータである場合，そのデータはプログラム実行中永久に保持されるため，予測テーブルにそのページアクセスの履歴が残っている場合は，高確率で予測的中する．しかしながら，大規模データを扱うプログラムの場合，プログラム実行フォーマットのヒープ空間にダイナミックデータを確保することは一般的である．そのため，予測テーブルは出来るだけ大きなテーブルを用意しなければならなくなる．加えて，このメモリ領域は生成／消滅が動的なため履歴情報を予測テーブルに格納しても無駄になる場合がある[1]．これらのデータ対しプログラムが頻繁にページアクセスを行う場合，履歴予測は要求されるハードウェア量に見合った予測的中率を保つことが難しい．

以上の考察から履歴予測の予測的中率は予測テーブルの容量で決められることがわかる．履歴予測は通常の TLB やキャッシュ等で一般的に行われている LRU(つまり，時間的局所性) の拡張行為に帰着する．この履歴予測によるページアドレス予測研究はソフトウェアレベルで既に行われた [1]．本論文では今までの予測機構の中心となる履歴予測の概念を廃し，プログラム実行に潜在する傾向に基づいた予測，つまり傾向予測を提案する．本論文は傾向予測の提案の一例として**線形ページアドレス予測**を提案し，その実装を行い検証する．線形ページアドレス予測の詳細は 2.1.2 で詳細を説明する．

2.1.2 傾向予測と線形ページアドレス予測

傾向予測とはプログラムに潜在する傾向を元に予測を立てる予測方針である．これらに類する予測として既に在る機構は，通常予測機構とみなされていないが，キャッシュのブロック転送 (つまり，空間的局所性) 等がこの予測に当てはまる．この傾向予測をページアドレス予測に適用する一例として線形ページアドレス予測を提案する．

実行プログラムの構造上，プログラムテクスト，スタティック／ダイナミックデータおよびスタック領域は，それぞれ連続する仮想アドレス空間に存在する．最も明確なページ

[1] これは命令ページアクセスにおいても起こる．ヒープ空間へのデータ領域の動的確保や消滅は命令メモリにおけるダイナミックリンクに相当するためである．

アクセス傾向は逐次的にアクセスされる場合である．プログラムテクストは分岐，ジャンプ命令実行の場合を除いて命令サイズ分インクリメントされる仮想アドレスでアクセスされる．データアクセスの場合，配列形式で定義されたデータは逐次的に参照される傾向がある[2]．この逐次的アクセスに対して，最後にヒットしたページアドレスの周囲のページアドレスを予測の対象とする．線形ページアドレス予測とはこの傾向を用い次に必要とされるページを予測することである．

本論文で提案する線形ページアドレス予測機構の概要を図 2.1 に示す．

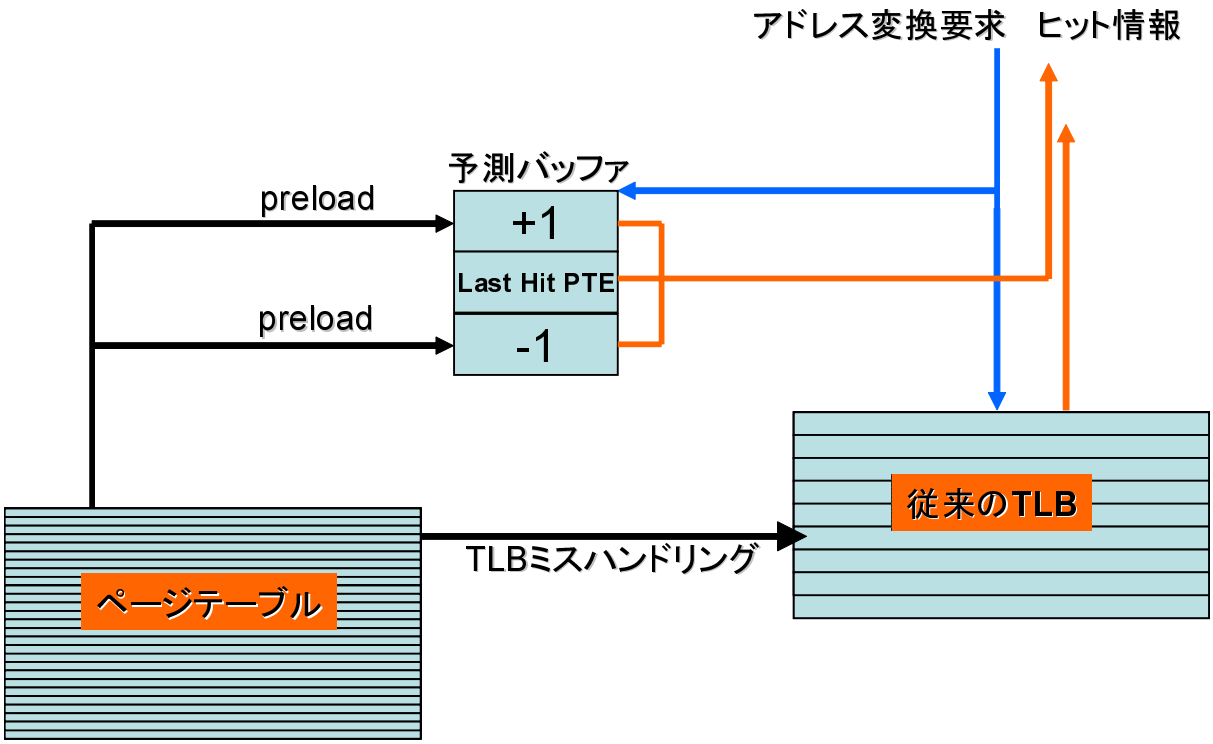

図 2.1: 線形ページアドレス予測．

本機構の実現はパイプラインによるプログラム実行のバックグランドで予測およびプリロードを行うために，全てハードウェアで実現される．(ソフトウェアによる実現はシステムソフトウェア設計上，予測機構を意識しなければならない上通常のハードウェア構成[3]では予測のタイミングが TLB ミス時に限定される．) また，命令アクセス，データアクセスに独立して TLB を儲け，それぞれのアクセス傾向に沿った予測を行う．予測機構は最後に参照した仮想ページの ±1 の仮想ページ番号のページテーブルエントリ（PTE）をページテーブルからプリロードする機能を有する．線形アクセスが発生した場合，この機構により従来の TLB では必ずミスする初回のページアクセスに対して，予測的中時に

[2]ワーキングセットが大きな場合，その大部分は配列データで構成される傾向がある．

[3]予測ミスのタイミングで例外処理を発生させソフトウェアで解決することも可能であるが，予測はミスが頻発することもあるため，現実的ではない．

ミスの大部分を防ぐ効果が得られる．またプリロードした PTE を TLB でなく，特別にバッファを設け，バッファに格納するプリロード方針をとる．この方針については 2.2 で詳しく説明する．

2.1.3 線形ページアドレス予測機構のハードウェア仕様

本研究において設計された線形ページアドレス予測機構のブロック図を図 2.2 に示す．

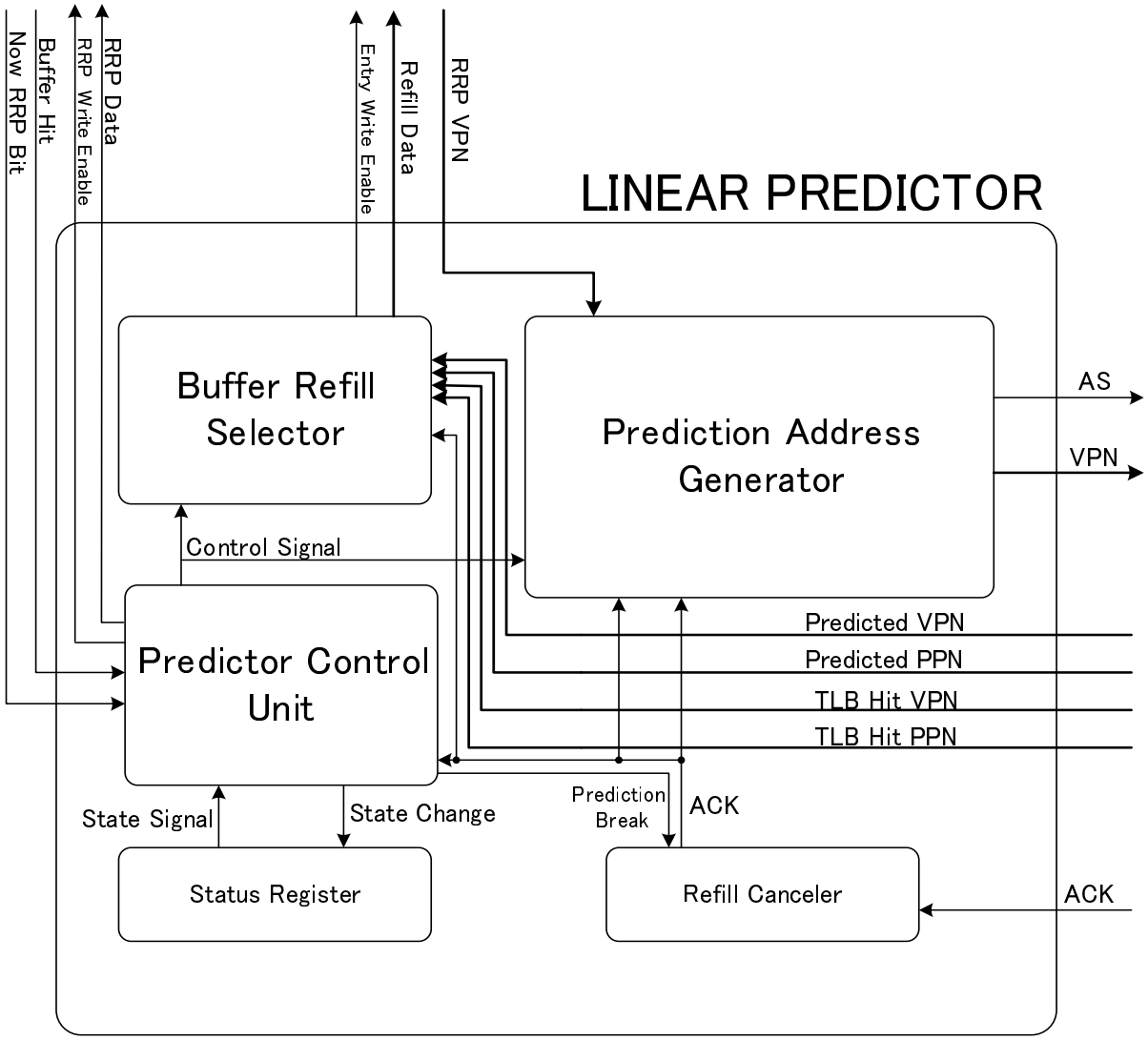

図 2.2: 線形ページアドレス予測機構ハードウェア構成ブロック図．

線形ページアドレス予測機構は大きく `Predictor Control Unit`，`Prediction Address Generator`，`Buffer Refill Selector` の 3 つの機能ユニットから構成される．その他に

予測を中断，破棄するために使用される Refill Canceler，予測機構の状態を記録する Status Register が存在する．線形ページアドレス予測機構は予測バッファと共に動作する．予測バッファについては 2.2 おいてに詳しく論ずる．Predictor Control Unit は予測バッファの状態と Status Register を判断し線形ページアドレス予測機構全体のコントロールを行う．Prediction Address Generator は Predictor Control Unit の制御で予測 VPN(Virtual Page Number) 発行のタイミングの取得後，Last Hit PTE から RRP VPN(Reamer Reference Position Virtual Page Number) を取得し，±1 の予測 VPN を生成する．生成された予測 VPN は AS(Address Strobe) と共に TLB ミスハンドラに供給される．Buffer Refill Selector は Predictor Control Unit の制御で Predicted VPN, Predicted PPN(Physical Page Number) と TLB Hit VPN, TLB Hit PPN から到着するバッファにリフィルするため情報 (PTE) をバッファの適切な位置に格納する機能を有する．この線形ページアドレス予測機構の動作時の入出力波形を図 2.3 に示す.

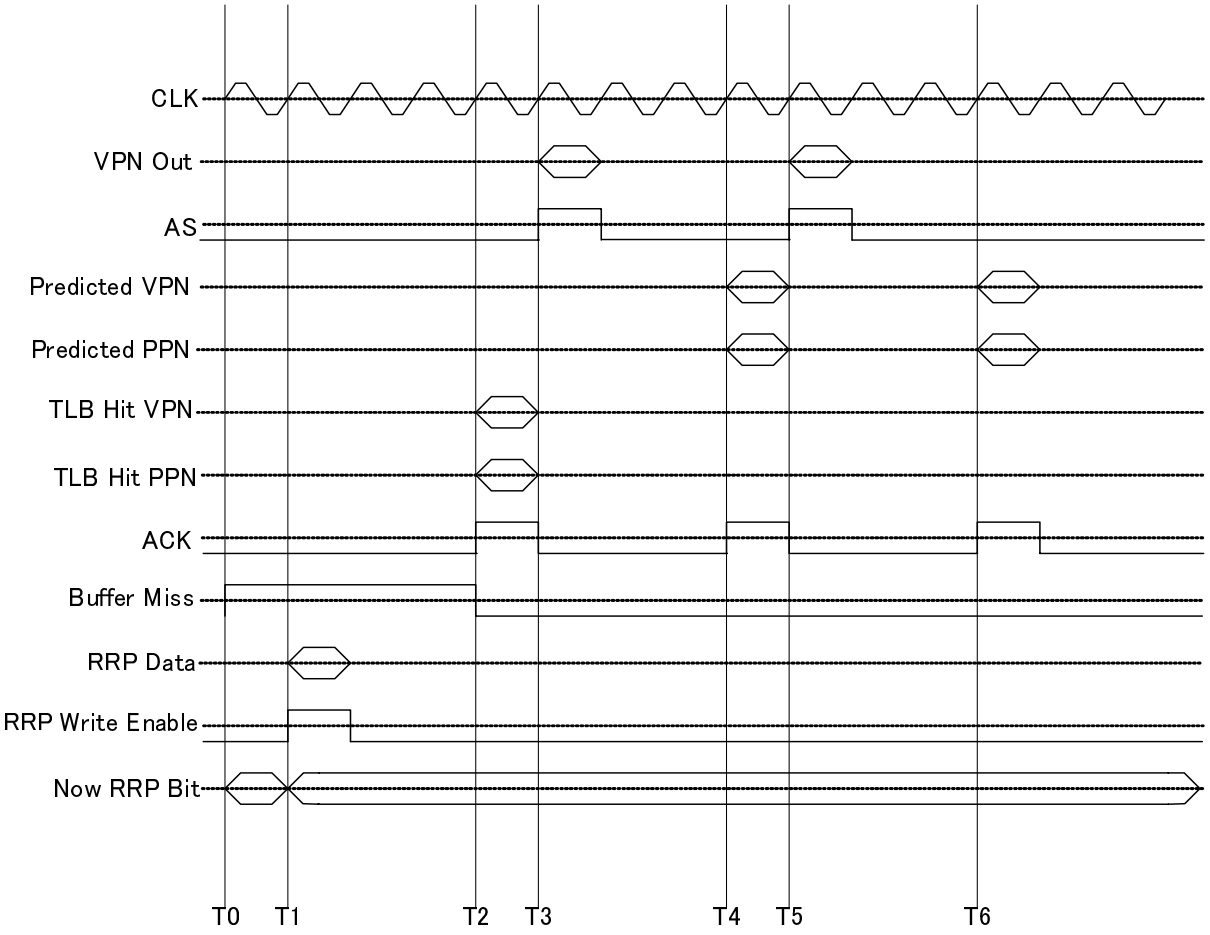

図 2.3: 線形ページアドレス予測機構の動作時入出力波形.

図の波形は図 2.2 の外部入出力をプローブしたものであり，システムブートアップ直後の正常な予測開始時の波形を示したものである．この波形図における $T0$ から $T6$ 迄の波形変化を見ることで正常予測完了プロセスを詳解する．

〔時刻 $T0$〕 ブートアップ直後のため，バッファが初期状態で何も変換情報 (PTE : Page Table Entry) が存在せず，バッファはミスし線形ページアドレス予測機構にバッファからミスシグナルが到着する．

〔時刻 $T1$〕 線形ページアドレス予測機構はバッファが初期状態であるため，リフィル時に PTE を格納する位置を決定する RRP ビットフィールドを設定しなくてはならない．そのためにバッファに対し，初期値となる `RRP Data` と `RRP Write Enable` を供給する．その後 TLB からのヒット情報の待ち状態へ移行する．

〔時刻 $T2$〕 線形ページアドレス予測機構がミスした場合，TLB のミス／ヒットに関わらず，今までの予測を破棄，再予測を行う．予測を行うために予測機構は TLB から予測の元となる Last Hit PTE が供給されなくてはならない．ここではブートアップ直後のため TLB もミスとなり TLB に PTE を格納するための TLB ミスハンドリングが行われる．($T1$〜$T2$)　通常の TLB ミスハンドリング終了後，TLB は TLB でヒットした PTE を線形ページアドレス予測機構に転送する．時刻 $T2$ では TLB から TLB がヒットした情報が `TLB Hit VPN` と `TLB Hit PPN` から `ACK` シグナルと共に供給されている．同時に到着した PTE はバッファに格納される．

〔時刻 $T3$〕 `TLB Hit VPN` と `TLB Hit PPN` が到着した次のクロックサイクルで最初の予測 VPN を発行する．発行の順序は VPN+1，VPN−1 の順である．この時刻 $T3$ では VPN+1 を `VPN Out` から発行している．発行した VPN は外部で TLB ミスハンドラに渡され TLB ミスハンドリングと同様に処理された後，予測機構は予測した PTE を得ることが可能となる．

〔時刻 $T4$〕 最初に予測をかけた VPN+1 の PTE が `Predicted VPN` と `Predicted PPN` から到着する．同時にバッファ内部にこの PTE が格納される．

〔時刻 $T5$〕 時刻 $T4$ の次のクロックサイクルで時刻 $T3$ において既に生成された VPN−1 の予測が発行される．

〔時刻 $T6$〕 2 回目の予測 VPN による PTE が到着する．これがバッファに格納され予測機構の準備は終了する．

図において TLB 又はページテーブルからの待ち時間 ($T1$〜$T2$，$T3$〜$T4$，$T5$〜$T6$) はメモリアクセスレイテンシである．図では縮小のため 3 クロックサイクルとなっているが，実際にはバスクロックサイクルで 12 クロックサイクル程度，近年の CPU はこのバスクロックの 10〜20 逓倍で動作していることを考慮に入れると 120〜240 クロックサイクルの待ち時間となる．この予測 PTE 待ち時間に参照ページが変わり本来予測が的中するはず

の VPN±1 の参照が発生すると，予測バッファはミスと判断し再度 TLB ミスハンドリング終了後の Last Hit PTE 待ち状態へと遷移する．以上が一連の線形ページアドレス予測機構の動作である．

2.2 予測バッファ

予測機構は予測した PTE を格納するための専用の予測バッファを有する．予測した PTE を従来の TLB でなく予測バッファに入れる理由は 3 つある．1 つは予測によってページテーブルから読み込まれた PTE を格納する際，既に格納されている重要な PTE がリプレースされないようにするためである．（擬似 LRU でリプレース制御されている TLB では頻繁にアクセスされる PTE がリプレースの対象となる可能性がある．）2 つ目の理由は TLB に格納される PTE を限定することで，ページテーブルの PTE を線形アクセスされる PTE とそうでない PTE に区別することができることである．2.1.2 節で示した線形ページアドレス予測器は，線形アクセスが発生した場合に初回のページミスを予測しプリロードすることにより回避するが，線形アクセスが成立しない状況では予測が外れてミスとなる．このようなミスが発生すのは線形予測されるデータセットの先頭の PTE に対してである．本研究ではこの PTE を PPTE（Pointer Page Table Entry）と呼ぶ．ページテーブルに存在する PTE を PPTE と非 PPTE に区別したとき，TLB に挿入される PTE は PPTE のみとなる．PPTE 以外の PTE は全て予測バッファに格納され，予測が遷移したときに取り除かれる．これにより，複数のページに跨る線形アクセスされるデータセットに対して，TLB エントリの使用は一つの PPTE のみとなる．すなわち，データセットのページ数を無視できるという意味を持つ．

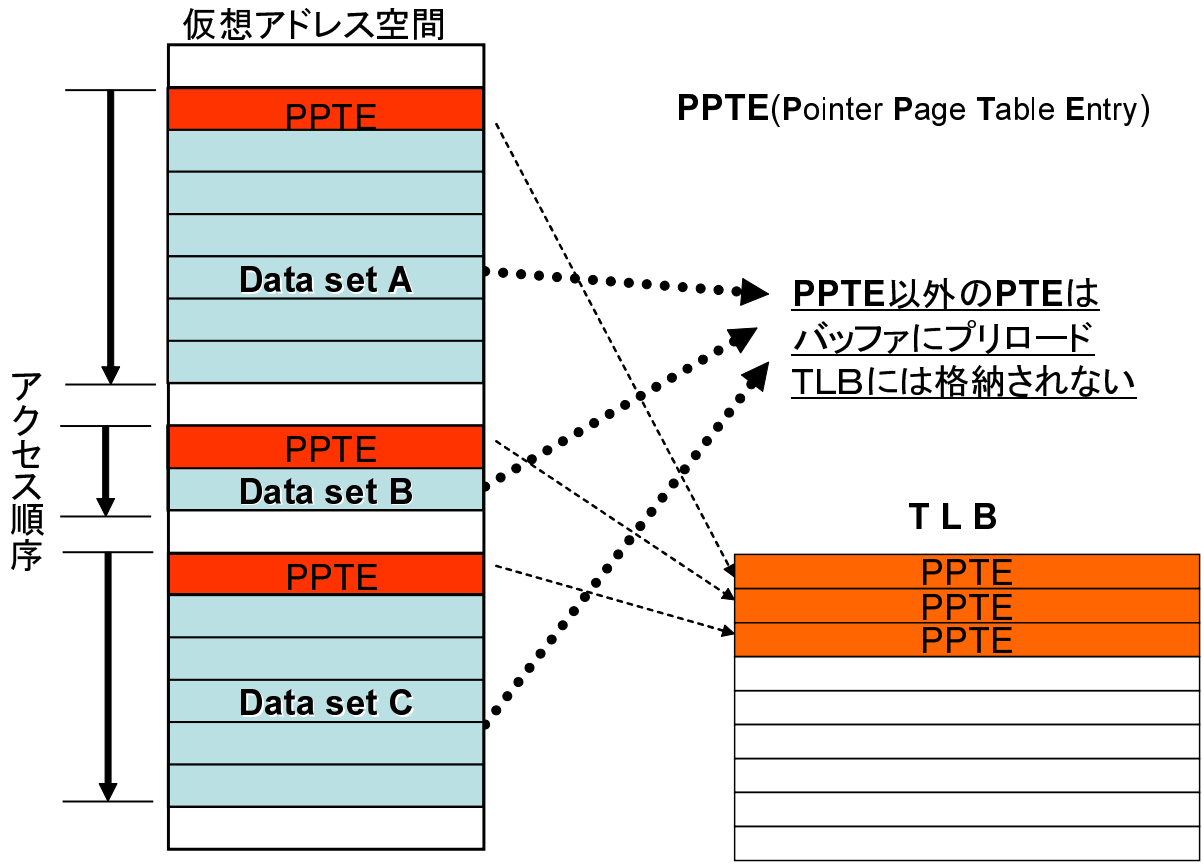

図 2.4: PPTE と予測 PTE.

図 2.4 において，データセット A，B，C に関して使用する TLB エントリはそれぞれの PPTE の 3 つのみとなる．最後の理由は，プログラムの実行中に非線形のページアクセスが発生した場合に，従来の TLB より性能が低下することが無い点である．すなわち，線形アクセスされないページの PTE は全て PPTE として TLB に格納されるため，従来の TLB と同じ振る舞いをする．この 3 つの理由により本機構は予測バッファ格納方式を採用する．

本機構を実装するに当たり，予測 PTE の格納先として TLB を使わずに新たにバッファを設ける方針はハードウェア量及び遅延の増大につながるということをハードウェア設計上念頭におかなければならない．バッファエントリの数は 3 エントリの固定で有り，その 1 エントリは TLB の 1 エントリと同じ構成のフィールドを持っている．また，アドレス変換参照時にバッファと TLB は同時に参照しなくてはならない．このことからシステムに予測機構を備えた 32 エントリの TLB が存在する場合，ハードウェア量，遅延量的にその TLB が 35 エントリの TLB になったことに等しい．プログラムが扱う 1 データセットに着目した場合，本機構がその実装に伴うハードウェア量と遅延量増加に見合う性能向上を実現するためには，TLB リーチの観点から，非 PPTE が 3 エントリ分あればよい．

また，非 PPTE が 3 エントリ以上ある場合は TLB 費やしたハードウェア以上の性能向上が得られる．この特性により，命令 TLB に関して線形ページアドレス予測を適用した場合，プログラムテクストへのアクセスはいかなる場合も線形傾向にありることから，Split TLB[4]実装の場合は命令 TLB のサイズ縮小，Shard TLB[5]の場合は共有 TLB 圧迫の減少につながるものとなる．

2.2.1 予測バッファのハードウェア仕様

本機構により，プログラムの実行で発生するページアクセスの中から非線形アクセスの情報のみが TLB に格納されるため，TLB エントリ数を削減可能である．すなわち，本機構は TLB のエントリ数およびページサイズを大きくせずに TLB 性能を向上させる方法である．実装した予測バッファの回路図を図 2.5 に示す．

[4]TLB 構成の一例．命令アクセスとデータアクセスにそれぞれ独立した個々の TLB を配置する構成．個々のアクセスの傾向によって個々のアクセスの参照に影響を与えない特性を持つ．

[5]TLB 構成の一例．命令アクセスとデータアクセスで一つの TLB を共有する構成．MIPS 等の実装では命令アクセスには 共有 TLB の上の階層に マイクロ TLB と呼ばれるごく小さい独立した TLB を用意する場合もあるが，2 つのアクセス系統が同じ TLB を利用する場合はこう呼ばれる．

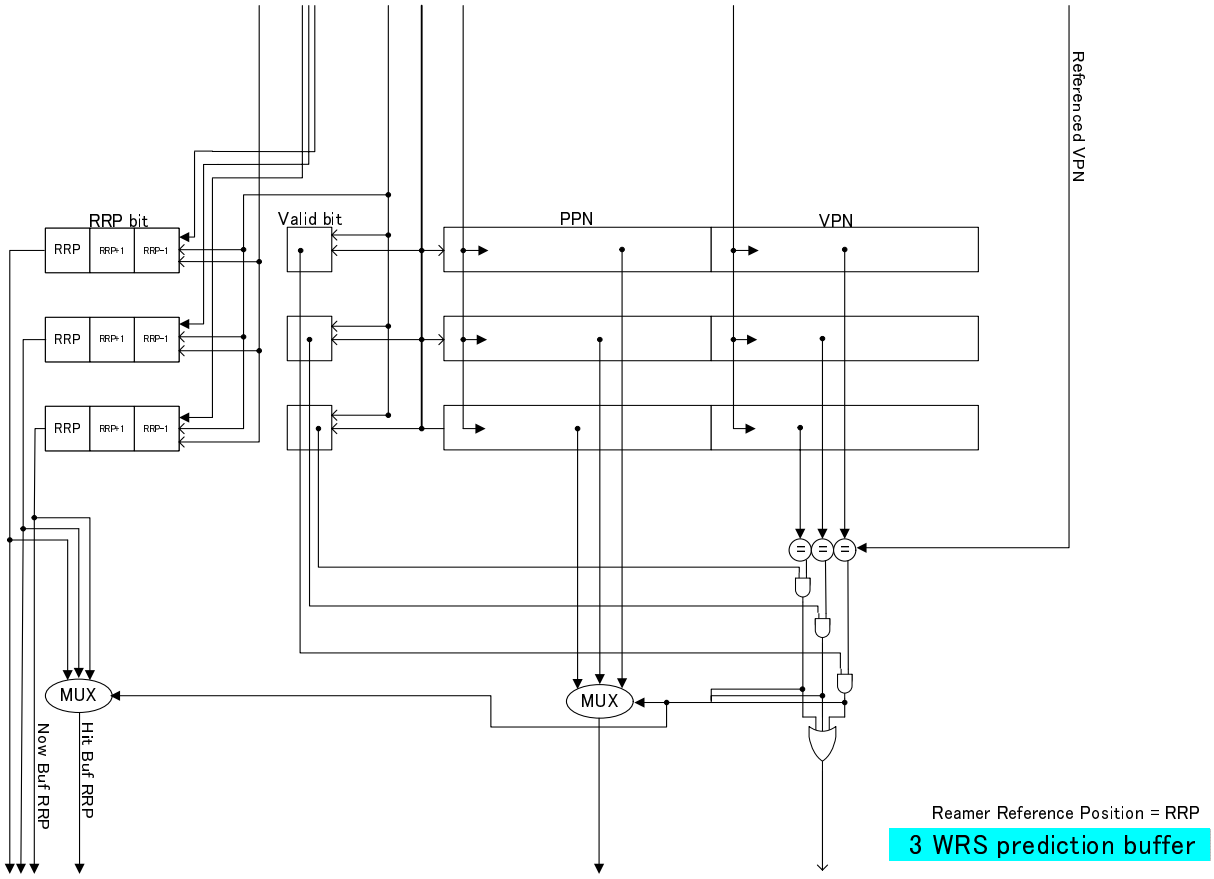

図 2.5: 予測バッファ実装回路図.

図は予測バッファのハードウェア構成を示しており，矩形部分は記憶素子の配列を意味する．線形ページアドレス予測を実現するため，バッファは Last Hit PTE の ±1 の PTE を格納しなけれはならない．よって，バッファには 3 エントリ分の記録領域が用意されている．1 つのエントリのフィールドはページテーブルの PTE 情報が VPN と PPN のみで構成されていることを想定して設計を施したため, Attribute フィールドは存在しない. TLB のエントリフィールドも同じである．これらエントリ毎にコントロールビットが存在する．エントリの有効を示す Valid ビットと，仮想アドレス空間上の線形位置を記録する RRP ビットフィールドがそれにあたる[6]．この RRP ビットフィールドは 2.2.1 で説明したリフィ ル時のエントリの選択に使われる．RRP ビットフィールドは予測ヒット時に変更され，予測がヒットしてバッファが遷移する場合に既にバッファに格納されている PTE を有効に使うことを可能とする．バッファヒット時の内部状態遷移を示す図を図 2.6 に示す.

[6]MIPS 実装の場合 Dirty ビットと Valid ビットは共通化され Attribute フィールドに Dirty ビットとして存在するが，本論文の実装では Attribute フィールド実装しないため独立しコントロールビットとして配置した.

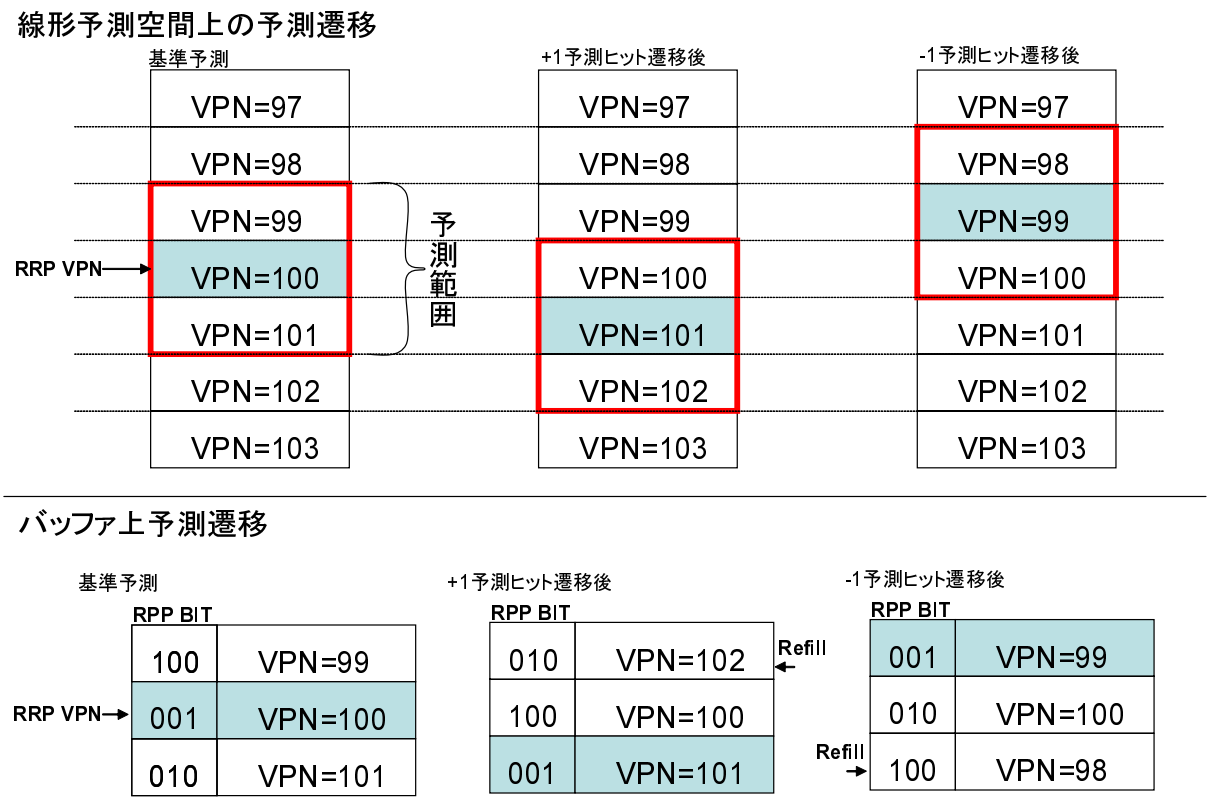

図 2.6: 予測バッファヒット時遷移.

図の中央，線で区切られた上方は線形予測が対象とする連続した VPN に対応する予測空間（連続する VPN によりインデックスされるページテーブル空間）上での予測遷移を示している．図左は今プログラム実行中の参照で VPN=100 の　参照があり，予測準備が完了している状態である．連続した VPN 空間の VPN=97 〜VPN=103 迄の中で VPN=100 を RRP VPN として ±1 の VPN=101 と VPN=99 を範囲とした太線枠が予測範囲である．この状態で+1 ヒット，つまり VPN=101 の参照が発生しバッファヒットした場合，線形予測機構は VPN=101 を RRP VPN とし予測を開始しなくてはならない．この場合図中央の太枠に予測範囲が遷移する．また，-1 ヒット，VPN=99 参照時は図右の太枠に予測範囲が遷移する．これを実現するためにはバッファの物理的位置を予測範囲の並び方にバインドし，遷移をするたびに全ての情報を格納する方法も考えられるが，本研究の実装は基準位置参照ビットフィールド (RRP ビットフィールド) を設けることでバッファヒット時の遷移は最小のラッチで行えるように設計されている．RRP ビットは基準位置 (RRP) を示すビットと VPN+1，VPN-1 を示すビット，で構成されている．RRP VPN を示す場合このフィールドは ”001 ”，+1 は ”010 ”，-1 は ”100 ”である．これをバッファヒットした次のクロックサイクルでヒットの種類に合わせ書き換えることによって 2.1.3 で示した `Buffer Refill Selector` によって必要のあるバッファエントリのみに必要な情報を格納することができる．図の中央，線で区切られた下方の図は上方で示された例で，実際

のバッファにおける遷移を示している．左の基準予測のバッファの状態から，中央の +1 予測ヒット遷移後と右の -1 予測ヒット遷移後のバッファの状態を見ると， +1 予測ヒット遷移後のバッファ状態では VPN=100 と VPN=101 の物理位置が変わっておらず，不要となった VPN=99 の位置に遷移後必要とされる VPN=102 がリフィルされている ことがわかる．また，-1 予測ヒット遷移後は VPN=100 と VPN=99 の物理位置が変わっておらず，不要となった VPN=101 の位置に遷移後必要とされる VPN=98 がリフィルされていることがわかる．RRP ビットフィールドの導入はバッファへの PTE 格納先の柔軟化だけでなく，バッファ制御回路を単純化し，また VPN+1 ヒットの後，予測リフィルが完了する前に VPN-1 参照があった場合などにメモリアクセスを発生させること無く，RRP ビットフィールドの変更のみで以前の予測状態に戻すことを可能にしている．

第3章　線形ページアドレス予測機構の非線形対応構成

命令アクセスは通常，プログラムカウンタの値をインクリメントしてメモリアクセスを行うため，線形アクセス傾向がある．ページアクセス単位でもこの傾向は変わらない．一方データアクセスに関してはデータ構造とそのアクセス方法はプログラム依存である．このため提案した線形予測機構が十分に機能しない場合がある．本章ではこの問題に対処するために，完全線形ページアクセスの制限を緩和する 2 つの手法を提案する．この手法を用い実装をすることにより，よりプログラム実行において現実的な線形ページアクセス抽出と線形予測機構のバッファの効率使用が可能となる．また 2 つの手法の具体的実装方法について論ずる．

3.1　Wide Range Support

Wide Range Support（WRS）は予測の範囲を拡大する非線形アクセス対応手法である．線形予測は最後に参照したページの ±1 のページの PTE をプリロードするものであった．WRS では ±2，±3 のようにプリロードする PTE を増やすことにより，予測範囲を拡大して非線形アクセスを吸収する．WRS 構成図を図 3.1 に示す．ここでは WRS の大きさの指標を予測範囲が ±2 の時は `2WRS`，±3 の時は `3WRS` と表す．図は 3WRS 実装のものである．

図は図 2.1 で示した通常の線形ページアドレス予測機構よりバッファ部分が上下に伸びている．これは図 2.6 で示した予測空間の枠を拡大したことに他ならない．厳密には VPN±2 ヒットや VPN±3 ヒットは線形ページアクセスとはいえない．しかし，2 次元配列の列空間への逐次アクセスや構造体配列への逐次アクセスなど 1 データセット内部でのアクセスが全て線形アクセスとなるわけではなくプログラマレベルで意識されない逐次アクセスの非線形ページアクセスが起こる場合もある．加えて，1 データセットが WRS 範囲内の数ページに収まるほど小さく，それが仮想アドレス空間上連続しており，そのデータセットが順に逐次アクセスがある場合も WRS 構成をとることで多量の TLB ミスを解消することができる．つまり，WRS 構成を導入することで仮想アドレス空間上，データセット内部の非線形アクセスや比較的近い位置のデータセット同士が結合され，TLB に挿入される PPTE が一つにまとめられることが可能となる．

また，この拡張を採用することはバッファエントリを増加させることにつながり 2.2 で

論じたハードウェア量と遅延の増大問題が同じように問題定義される．この WRS 構成は線形ページアドレス予測の付加的な機能であり，WRS 構成を大きくし予測プリロード PTE を増やすことは性能向上に比例するものではない．このことから WRS 構成を実際にハードウェアに実装する場合は，メモリアクセス遅延のクリティカルパスに大きな影響を与える構成をとらず，効率に対する遅延増加が妥当な構成を実装することが重要である．

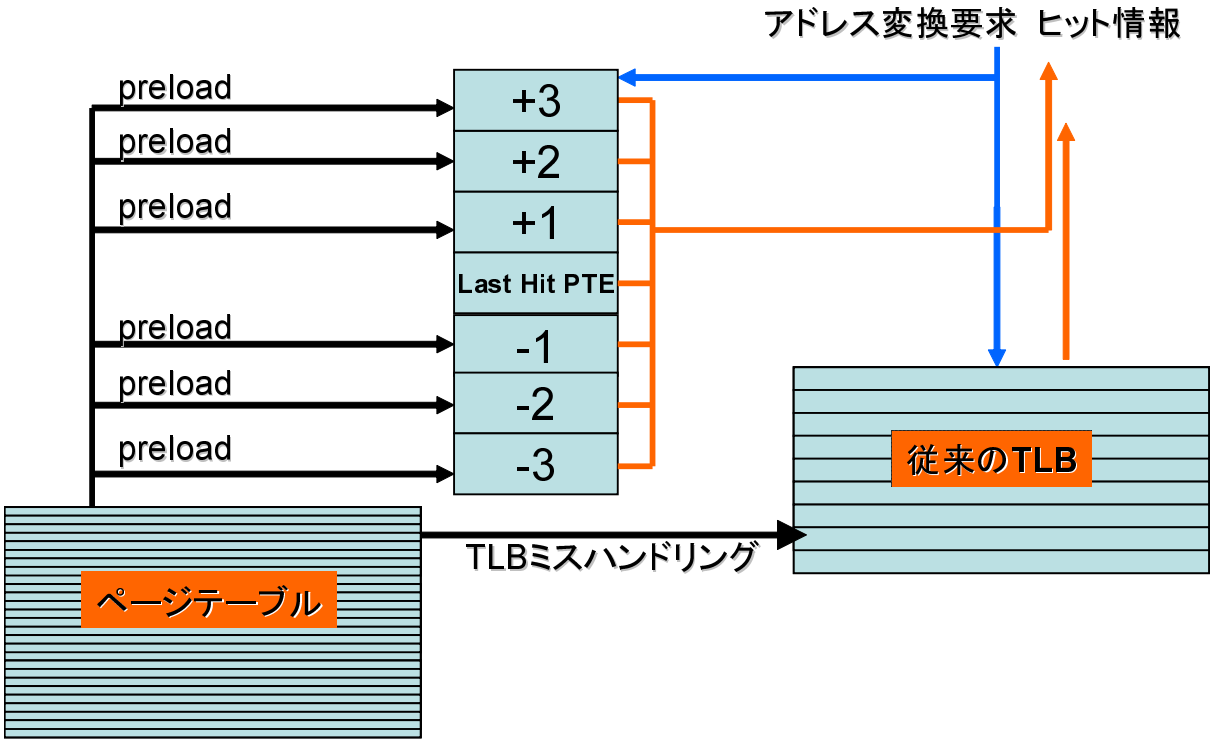

図 3.1: Wide Range Support（WRS）.

3.1.1 WRS 構成へのバースト転送適用

2.1.3 で詳解したように WRS を実装した場合，1PTE 要求毎にメモリリクエストを要求するとなるとメモリリクエスト数は WRS 構成の大きさに比例する．このリクエスト増加は遅延量増加と同じく実行プログラムのメモリアクセスリクエストを圧迫しプログラム実行効率を減少させる原因となる可能性がある．仮想アドレス上連続するページの PTE はページテーブル内に連続して配置されるため，予測する複数の PTE をメモリからバースト転送することが可能である．このことから本論文では WRS 構成の大きさはプリロードする全エントリのサイズがメモリへの一回のバーストアクセスに収まる程度に予測範囲を設定し，この問題を解決する．WRS 構成時 PTE プリロードをバースト転送する概念図を図 3.2 に示す．

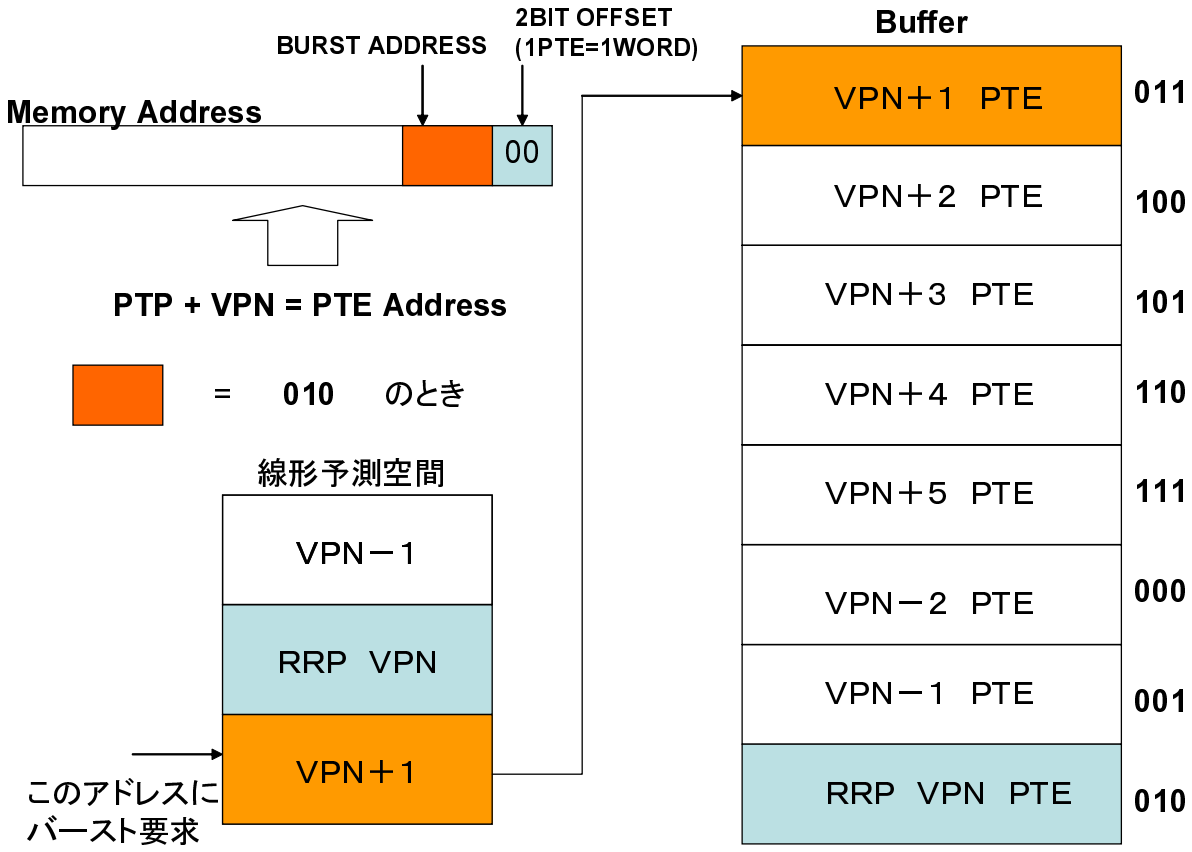

図 3.2: WRS バースト転送対応実装概念図.

図はバースト転送での WRS 予測 PTE のプリロードを示している．32Bit メモリアドレス空間を使用するプロセッサの場合，通常は PTEA(Page Table Entry Address) はプロセッサ内部にある PTP(Page Table Pointer) と参照アドレスの VPN をオフセット加算したものである[1]．このアドレスで示せるページテーブルの構成はオフセットの VPN により静的に PTE の位置が決められている．4KB ページの場合，VPN は $32bit - 12bit = 20bit$ となり，ページテーブルは 1PTE32 ビットで 4byte アラインされているならば，2 ビットのオフセットが PTEA に存在するため 4MB の大きさである[2]．このアドレスの 1WORD オフセットより上のビットがバーストアドレスである．(bit 長はバースト長によって決定される．8 バースト時は 3bit) 図は WRS が 8 エントリ存在する 4WRS 構成のバッファである．本来は $8 + 1 = 9$ エントリであるが，RRP VPN に対応するエントリは実装的に別に用意されている為，図では 8 エントリとした．(この理由については後で論ずる) このバッファをバースト転送で埋めるために RRP VPN の +1 つまり VPN+1 を 1 回だけバースト転送要求する．その VPN+1 のバーストアドレス部分の下位 3 ビットが 8 エントリを埋めるバーストスターティングアドレスである．図はこの 3 ビットのアドレスが 010 であるときの例である．通常のバースト転送はこのアドレスを与えられるとアドレス的に 011，100，101 の順に 3 ビットのアドレスが一回転するまで情報を転送する．一般的にこのバーストの順番をバーストシーケンスといい，採用したこのシークケンスをシーケンシャル方式という．現在 SDRAM で使われている代表的なバーストシーケンス 2 つを表 3.1 に示す.

シーケンシャル方式はスターティングアドレスから逐次に転送されるのに対し，インターリーブ方式は 2 番目の転送がスターティングアドレスより低いアドレスのデータが転送されるスターティングが存在する．順方向 (VPN の + 方向) へはバースト転送中早い段階でバッファにフィルしたい理由からシーケンシャル方式を採用した．しかしながら逆方向 (VPN の − 方向) への予測を優先してフィルしたい場合はインターリーブ方式が有効となる．その理由はシーケンシャル方式と比べてスターティングアドレスより逆方向に 1 つ低いアドレスのデータの到着が早いためである．しかしながら，バースト方式は予測方針によって決められるものではなく CPU キャッシュのフィル方法によって決められるため，選択の自由度は無いといえる．

[1]通常ページテーブルはそのページサイズにアラインされていなければならない．よって加算とは PTP の下位ビットに VPN を挿入するのみである (つまり，PTP の下位ビットはページサイズ分 0 セットされていなくてはならない)．また，ページテーブルはページアラインされていなければならない制約を取り除くため，PTEA の生成に算術的加算を行った場合，複数プロセスで動作するプロセッサにおいては複数のページテーブルが存在するため同一の PTE アドレスが存在してしまう理由から，算術加算によるアドレス生成は通常行われない.

[2]ページテーブルの大きさは PTEA を仮想アドレスとすることで可変とすることが可能であり，MIPSR4000 シリーズ等で実装もされている．この実装を行う場合，PTE をバースト転送する際に全ての PTE が物理アドレス的に連続していない為，PTE でないデータをバーストすると思われるが，バーストアドレスがページサイズのオフセットを超えない限りそれは起こらない．また現実的に，バースト長はキャッシュブロックサイズで設定されており，通常バースト長がシステム起動中に可変となることは無いためこの問題は起こりえない.

表 3.1: バースト長 8 の場合の 2 種類のバーストシーケンス

Starting Address			Addressing(Decimal)	
A2	A1	A0	シーケンシャル方式	インターリーブ方式
0	0	0	0, 1, 2, 3, 4, 5, 6, 7,	0, 1, 2, 3, 4, 5, 6, 7,
0	0	1	1, 2, 3, 4, 5, 6, 7, 0,	1, 0, 3, 2, 5, 4, 7, 6,
0	1	0	2, 3, 4, 5, 6, 7, 0, 1,	2, 3, 0, 1, 6, 7, 4, 5,
0	1	1	3, 4, 5, 6, 7, 0, 1, 2,	3, 2, 1, 0, 7, 6, 5, 4,
1	0	0	4, 5, 6, 7, 0, 1, 2, 3,	4, 5, 6, 7, 0, 1, 2, 3,
1	0	1	5, 6, 7, 0, 1, 2, 3, 4,	5, 4, 7, 6, 1, 0, 3, 2,
1	1	0	6, 7, 0, 1, 2, 3, 4, 5,	6, 7, 4, 5, 2, 3, 0, 1,
1	1	1	7, 0, 1, 2, 3, 4, 5, 6,	7, 6, 5, 4, 3, 2, 1, 0,

3.1.2 バースト転送適用による弊害と解決法

このように元々システムに存在するバースト転送を採用することによって 1 メモリアクセス要求でバッファを全てリフィルすることが可能となる．しかしリフィル後のバッファを見ると図 **??**で示すように予測された VPN の ± が揃っていないことがわかる．例では + 方向に VPN+5，− 方向に VPN−2 となってる．本来予測完了後のバッファ内容は ±4 になっていなくてはならない．これは WRS バッファのリフィルをバースト転送に依存したことの弊害である．更に，もう一つの弊害として線形予測上最も重要である VPN±1PTE のどちらかのリフィルが行われない可能性がある．バーストアドレスアラインと線形予測空間におけるバースト転送の関係を図 3.3 に示す．

図において新しく予測する RRP VPN を `Case A` と `Case B` に分けて考える．そのときの ±1VPN は図に示す位置に在る．RRP VPN の PTEA にバースト転送をかけて 4WRS のバッファをリフィルしようとする．そのとき，太枠で囲われたバースト境界の内部の PTE がバッファにフィルされる．つまり，バーストスターティングアドレスがどのような値をとる場合もこの境界の間の PTE だけがバッファにフィルされることとなる． `Case A` において，バーストスターティングアドレスが 000 のとき完全線形予測では VPN-1 が必要とされるのにもかかわらずバッファに直上の 111 の PTE はフィルされない．また `Case B` においても同じようにバーストスターティングアドレスが 111 のとき完全線形予測では VPN+1 が必要とされるのにもかかわらず，直下の 000 の PTE はフィルされない．つまり 8 バーストの WRS バッファリフィルのスターティングアドレスを RRP VPN から供給した場合，1/4 の確率で線形ページアドレス予測ができないこととなる．

この問題は RRP VPN の PTEA で得られる PTE をバッファとは別領域に格納し VPN+1 の PTEA でバースト転送をかけたことで，順方向の逐次ページアクセスは予測が的中することを確保した．この方法では逆方向のアクセスは線形ページアドレス予測が確保さ

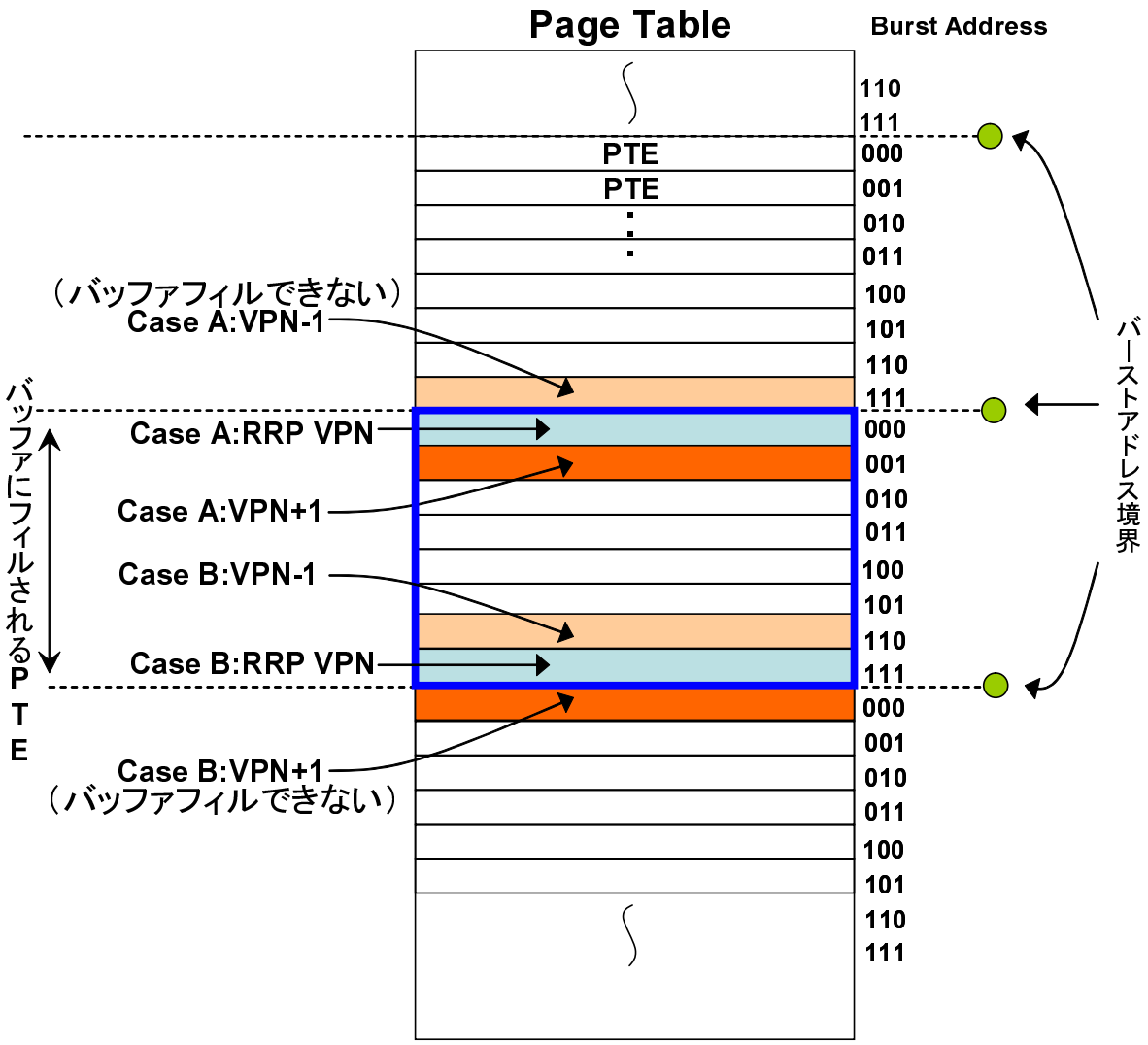

図 3.3: WRS バーストアドレス境界とバッファリフィル関係図.

れない場合がある．しかしながら，8 バースト時に 1/4 で予測が成立しない条件を 1/8 にまで緩和している．但し，配列を逆順にアクセスしていくような参照が発生する場合には，必ず一定の割合でバッファミスを起こすことになる．この解決方法としてバッファをもう一つ設け VPN−1 の PTEA にもバースト転送をかける方法が考えられるが，その場合，各バッファ間には必ず重複する PTE がが存在することとなり，バッファは平均して半分の利用効率となる，本論文ではこの実装は非効率であると判断し，より簡素な実装を選択した．

3.1.3 バースト転送適用 WRS 構成のハードウェア仕様

2WRS バースト適用線形ページアドレス予測を実装した場合の実装ブロック図を図 3.4 に示す．

一般的にバースト長はシステム起動時に静的に設定されるため CPU に搭載されるキャッシュのブロックサイズに合わせ WRS を設定することが重要となる．一般的なプロセッサのキャッシュブロックサイズは 16byte である．本論文ではそれに合わせ実装する構成は 2WRS の 32 ビット 4 バーストであることを前提として設計を施した．

バースト転送を適用した WRS 構成の線形ページアドレス予測機構が完全線形ページアドレス予測機構[3]と大きく異なる点は従来のバッファを `RRP VPN Register` と `Burst Buffer` に分離し RRP ビットフィールドを廃した点にある．この実装を選択した理由は 3.1.2 で既に論じた．従来の完全線形予測はバッファヒット時に予測に要する RRP VPN の遷移は RRP ビットを書き換えることで行ったが，WRS 構成は非線形ヒット時の場合にも RRP VPN の遷移が発生するため，エントリ数が増加するにしたがい RRP ビットフィールドが増大，増加する．そのため，完全線形予測時と同じ手法では回路規模が増大すると共に複雑になる．実装したバースト適応 WRS 構成線形ページアドレス予測機構は従来の予測器制御を廃し，より簡略かつ小規模な回路構成を主眼として設計を施した．バースト適応 WRS 構成を採用した線形ページアドレス予測機構と採用しないない完全線形ページアドレス予測機構のバッファヒット／ミス時の予測機構の状態遷移を図 3.5，図 3.6 を示す．

図 3.5 と図 3.6 ではバッファヒット／ミスと RRP VPN Register のヒット／ミスを分離し，予測発行を 1 回にしていることがわかる．発行済み予測のキャンセル機構を採用しないためこの処理も取り除かれている[4]．これらの単純化した実装で，状態数と最長パスが減少している．この変更は制御回路 (`Predictor Control Unit`) とアドレス生成回路 (`Address Generator`) の単純化と高速化を促す結果となった．しかしながらこれらの変

[3] 3.1.2 で論じたようにバースト適用 WRS は完全な線形予測ができないことからこれ以外の実装の予測機構をこう呼ぶ事とする．

[4] 完全線形ページアドレス予測の実装の場合はバッファの RRP ビットフィールドに論理的位置を記録していた．予測機構はこの情報を元にリフィルしていたが，予測 PTE 待ち中に参照が変わりバッファがミスする場合，現在の予測が破棄され (先に出してしまった要求に対する応答を受け付けない状態に移行)，新しい参照で予測が再遂行される．この実装で予測 VPN に対応していない PTE がリフィルされる場合，バッファ内で線形が成立しないため予測キャンセル処理は必要である．しかしながらバースト適用 WRS の場合，転送される PTE が既に線形であることから，転送される全ての PTE をバッファに格納する保証があればバッファ内部の線形は保証される．故に予測キャンセル処理は必要ない．

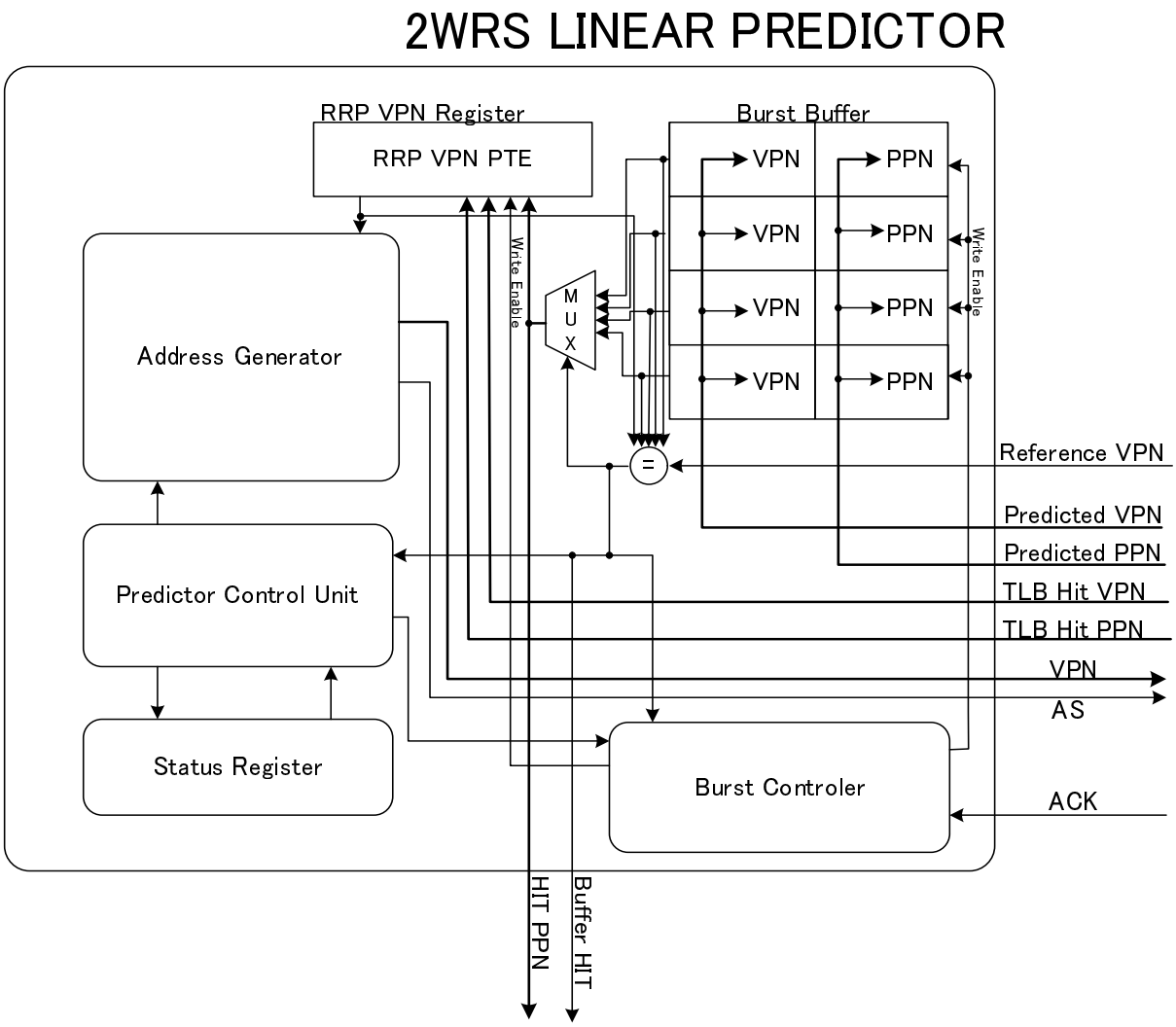

図 3.4: 2WRS 実装ブロック図.

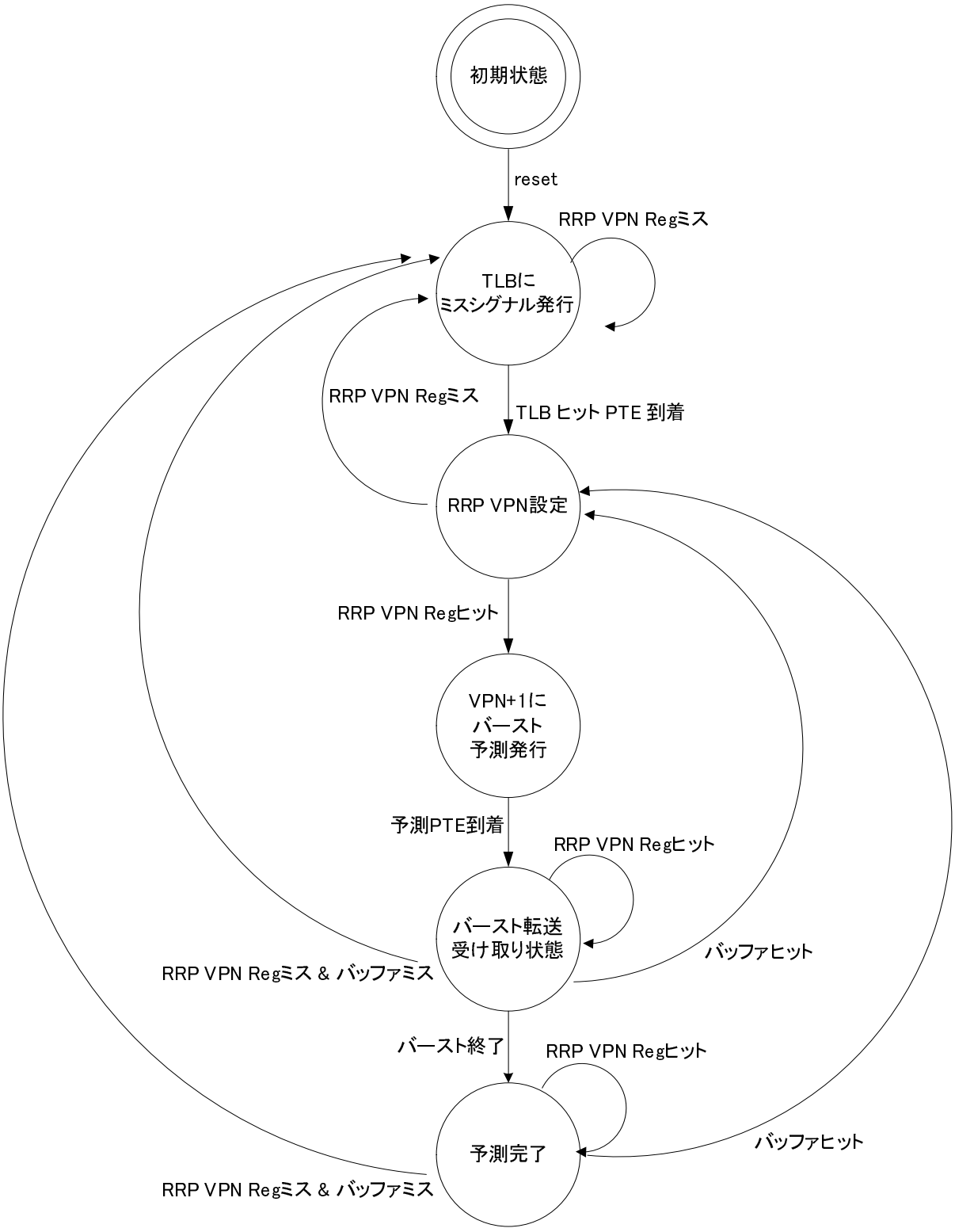

図 3.5: バースト転送を適用した線形ページアドレス予測回路の状態遷移図.

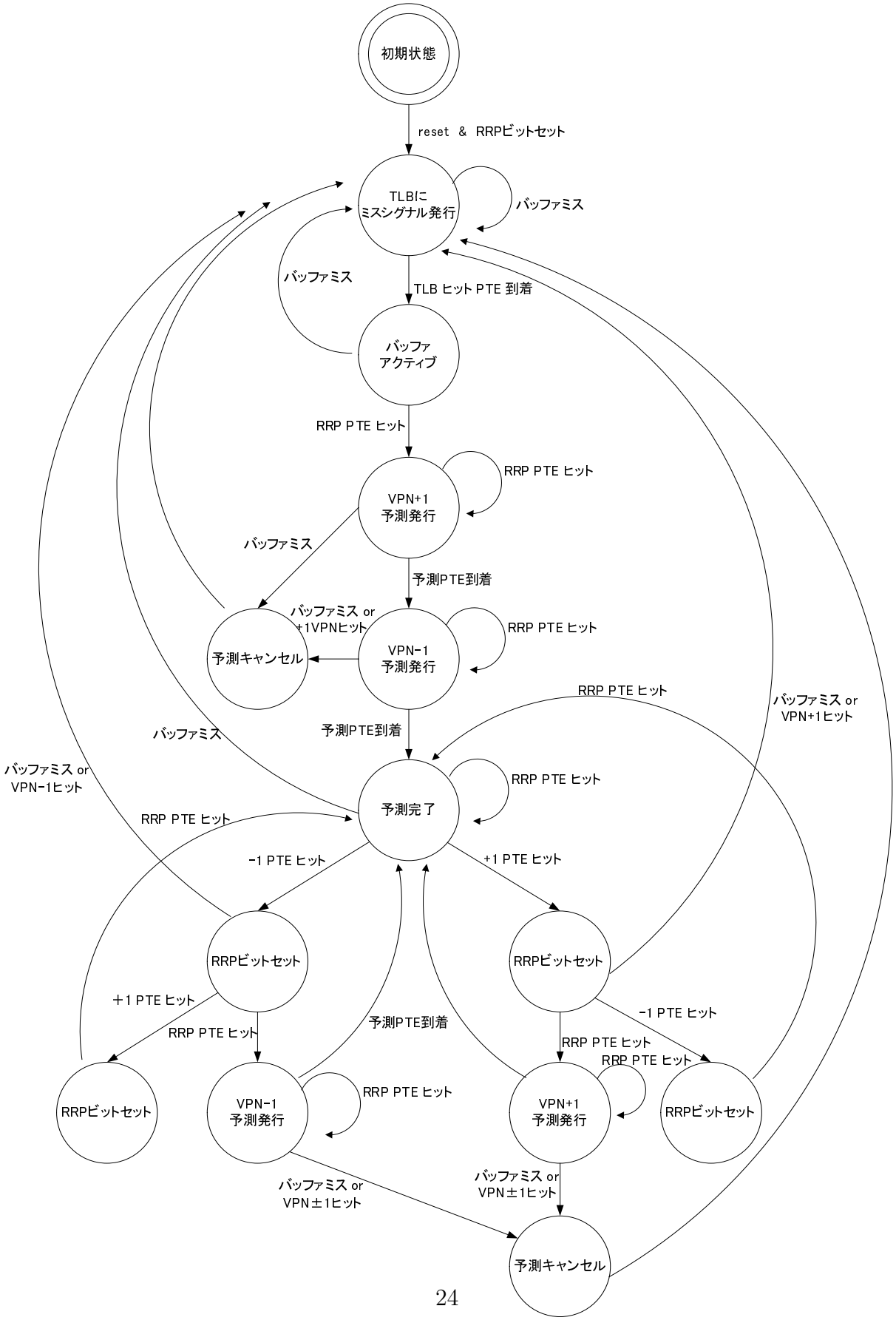

図 3.6: バースト転送を適用しない完全線形ページアドレス予測回路の状態遷移図.

更は線形ページアドレス予測発生時に全く同じ PTE の転送を複数回行うことを許容した設計であるため，メモリバスとバッファの利用効率の問題では完全線形ページアドレス予測機構の実装に劣る．更に，完全線形ページアドレス予測機構の `Refill Selector` の代わりに `Burst Controler` が存在する．`Refill Selector` は RRP ビットフィールドがなくなったことにより必要が無くなり，代わりに到着するデータバースト長をカウントし適切なエントリに分配する回路が必要となったためこれに置き換えた．

3.2 Multiple Operand Support

Multiple Operands Support（MOS）はオペランド数分の予測器を並列に用意し，同時に複数の予測を実現する構成である．これにより，仮想アドレス空間上の遠く離れたページ同士が交互にアクセスされるような非線形アクセスへ対応する．MOS 構成図を図 3.7 に示す．

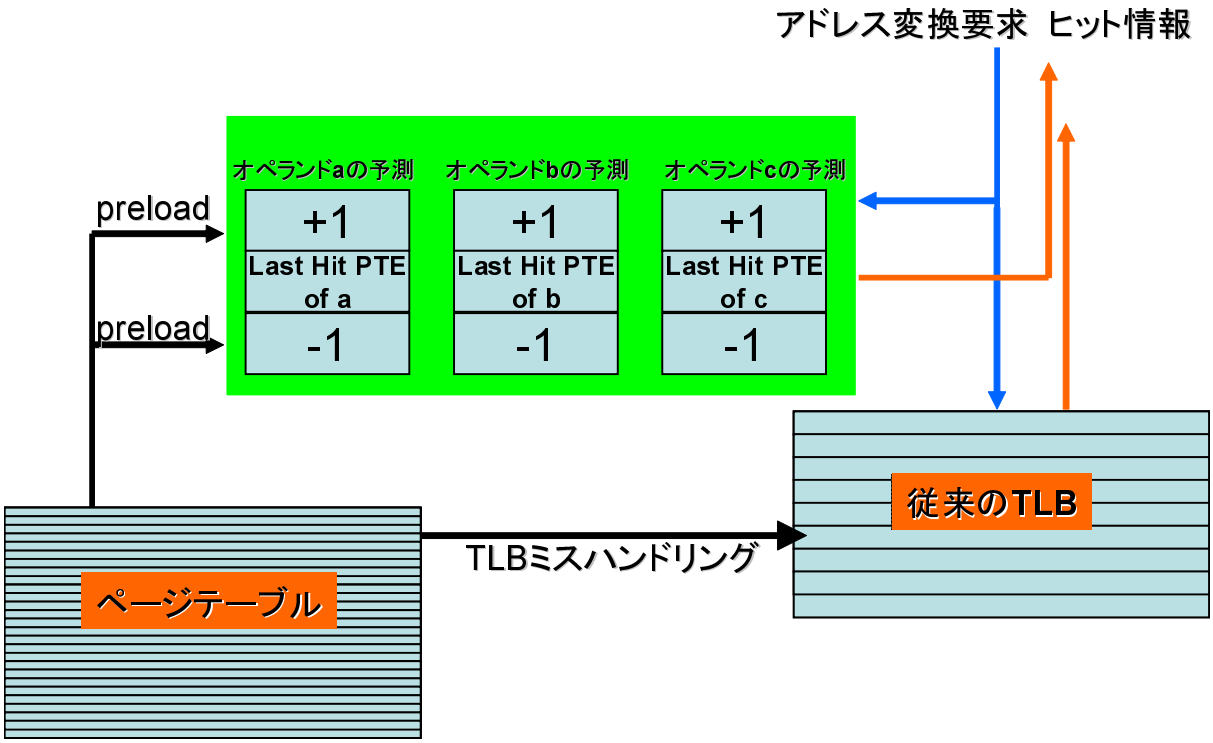

図 3.7: Multiple Operands Support（MOS）.

実際のプログラム実行では演算によりオペランドのアクセス順序が決定される．例えば，

```
for(i=0;i<n;i++)
    a[i]=b[i]+c[i];
```

という配列演算の実行を想定する．それぞれの配列データはページをいくつも跨ぐ大きさのデータセットであり，お互いが仮想アドレス空間上 WRS の予測範囲を超えた位置に存在する場合，b の要素と c の要素がロードされるときにそれぞれ予測が外れ，更に a にストアするときにも予測が外れる．それぞれのデータセットの中では線形にアクセスされているにもかかわらず，実際にアクセスされるアドレス列はデータセット間を移動する，すなわち非線形になる．また，例のプログラムはデータセット間のみのアクセスが発生し

ているが，実際のプログラム上ではプログラマには意識されないローカルスタックへのアクセスが起こる場合がある．特に関数呼び出しの際に生じるスタックポインタ・フレームポインタや引数受け渡しなどスタックフレームを操作する動作がその原因となる[5]．このような状況では，現在アクセスされるオペランドに関する予測が直前にアクセスされたオペランドに関する予測結果を置き換えるため，予測機構がうまく機能しない．MOS はこのようなページアクセスに対応する構成である．ここでは MOS の並列度の指標を並列予測数が 2 の時は 2MOS，3 の時は 3MOS と表した．図 3.7 は 3MOS 実装のものである．例のプログラムを図に示す 3MOS 構成で動作させる場合 3 つの予測機構はデータセット a，b，c 個々の予測状態を保持しつつデータセット内部に関して独立して線形ページアドレス予測を行う．その結果，1 つの線形ペーアドレス予測機構での実行時に発生したオペランド同士の予測器競合が避けられたといえる．

3.2.1 予測置換方式

プログラムで定義されるデータセットの数はプログラマが定義する．使われるデータセットの数が MOS 構成で用意した予測機構の数を上回る場合，新しい予測を始めるために予測の置換を行わなければならない．置換方式にはいくつかの方法が考えられる．

1. キュー型 先入れ先出しで予測の置換を行う．MOS 機構上，最も古い予測が置換対象となる．

2. スタック型 MOS 機構上で一番最後に発生した予測が置換対象となる．例えば，この置換方式を 3MOS 構成に適用する場合，最初に発生した二つの予測は永久に MOS 機構に残りつづけることになる．

3. プログラム同期型 特殊な命令を命令セットに追加，もしくは既存の命令を代用して MOS 機構に予測置換のタイミングを知らせる．

4.LRU(Least Reacently Used) 型 最も最近に使われなかった (つまり，最も最近ヒットしていない) 予測が置換対象となる．

1 のキュー型を採用した場合，ハードウェア的に最もシンプルな構成となる．しかしながらオペランド数が MOS 数を上回るとき，この置換方式は必ずスラッシングを引き起こす置換方式となる．この構成は十分な大きさの MOS 構成が導入できる場合に有効である．2 のスタック型を採用した場合，オペランド数が MOS 数を上回る状況で全ての予測がスラッシングする状況を避けることができる．しかしながら最後の予測のみが置換対象となることから，MOS 構成で扱うことができるデータセット数が静的に決められる．このことから予め決められた数のデータセットのみ局所的に扱う場合にのみ有効である．3 のプログラム同期型を採用した場合，プログラム処理による予測置換は大規模データセットを

[5] レジスタウインドウの機能を備えた SPARC CPU を除いては必ずこのメモリアクセスは発生する．

使う際にプログラマが意識して高速化を図ることが可能となる．しかしながらプログラマは予測器中の現在予測中の RRP VPN を知ることが難しい．そのため個々の予測器を指定して置換できない．この場合の予測置換は全ての予測機構の予測破棄を行うことによって実現される．また，実際のシステムにおいてプログラマがユーザレベルで TLB 周辺の制御を行うことは難しく，実装のためには命令セットの変更と共にオペレーティングシステム等の変更も要求される．4 の LRU 型を採用した場合，ハードウェア構成的に 4 つの構成中最も複雑な構成となるが，プログラマの定義するデータセットに対して柔軟性のある置換方式となる．つまり最もよく使われる予測は確実に残される．更に，3 よりもプログラムレベルで留意する必要も無くなりプログラム上では完全に予測機構の存在を隠蔽することができる．しかしながら 1 つの式のオペランド全てを予測しようとする場合，LRU は最初に開始された予測が最もよく使われない予測だと判断し置換してしまうことから，1 のキュー型と同じ振る舞いする．したがって，LRU 型を採用する場合は 1 のキュー型と同様に十分な大きさの MOS 構成が必要となる．本論文では 4 の LRU 型置換方式を採用した．

3.2.2 MOS 構成のハードウェア仕様

4MOS 構成を採用する線形ページアドレス予測機構の実装ブロック図を図 3.8 に示す．

MOS 構成は線形ページアドレス予測機構を並列に配置した機構である．図における線形ページアドレス予測機構は `LP` で示されるブロックである．4MOS 構成であるので，`LP0`〜`LP3` の 4 つの予測機構が存在する．しかしながら，1 参照においてアクティブな線形ページアドレス予測機構は 1 つであることから現在参照されている VPN に対応する予測機構を MOS 制御機構が選択し，選択された予測機構が参照された VPN に対し個々に対処しなければならない．そのため，個々の予測機構のバッファを個々の予測動作と別に参照する必要がある．それを伝えるシグナルが `Preliminary Buffer Check` である．このシグナルを予測器選択制御回路 `Active MOS Switcher` に入力し現在その VPN を扱っている予測機構を検出し，その予測機構のみをアクティブにする必要がある．更に `Preliminary Buffer Check` にヒットが存在しない場合（つまり存在する全てのバッファがミスした場合）予測器選択制御回路は予測置換を行わなければならない．MOS 機構内部の制御フロー図を図 3.9 に示す．

採用した置換方式に関しては 3.2.1 で既に論じた．本論文で正確な LRU を必要としたため，LRU 置換に一般的に TLB 等で使われる擬似 LRU ではなく，正確な LRU を実現する通常の LRU 方式を採用した．4MOS 構成で設計を施したので $4bit \times 4 = 16bit$ で 16bit 分フィールドを用意した[6]．(`LRU Bit Field`) これは 4MOS 程度なので現実的な記憶容量である．しかしより大きい nMOS 構成を実装する場合，必要となるビット数は n^2 で増加す

[6]LRU は置換候補となる対象数分のビット数をある候補の置換候補順位として候補数分用意した．(10 進記録) この実装で LRU ビット比較を行う回路を単純化することができる．LRU ビットフィールドを削減するために 2 進記録を用いることも可能であるがこの比較には加算器が必要となるためこの実装では MOS 増加時，ランダムロジック部の増大が問題となる．

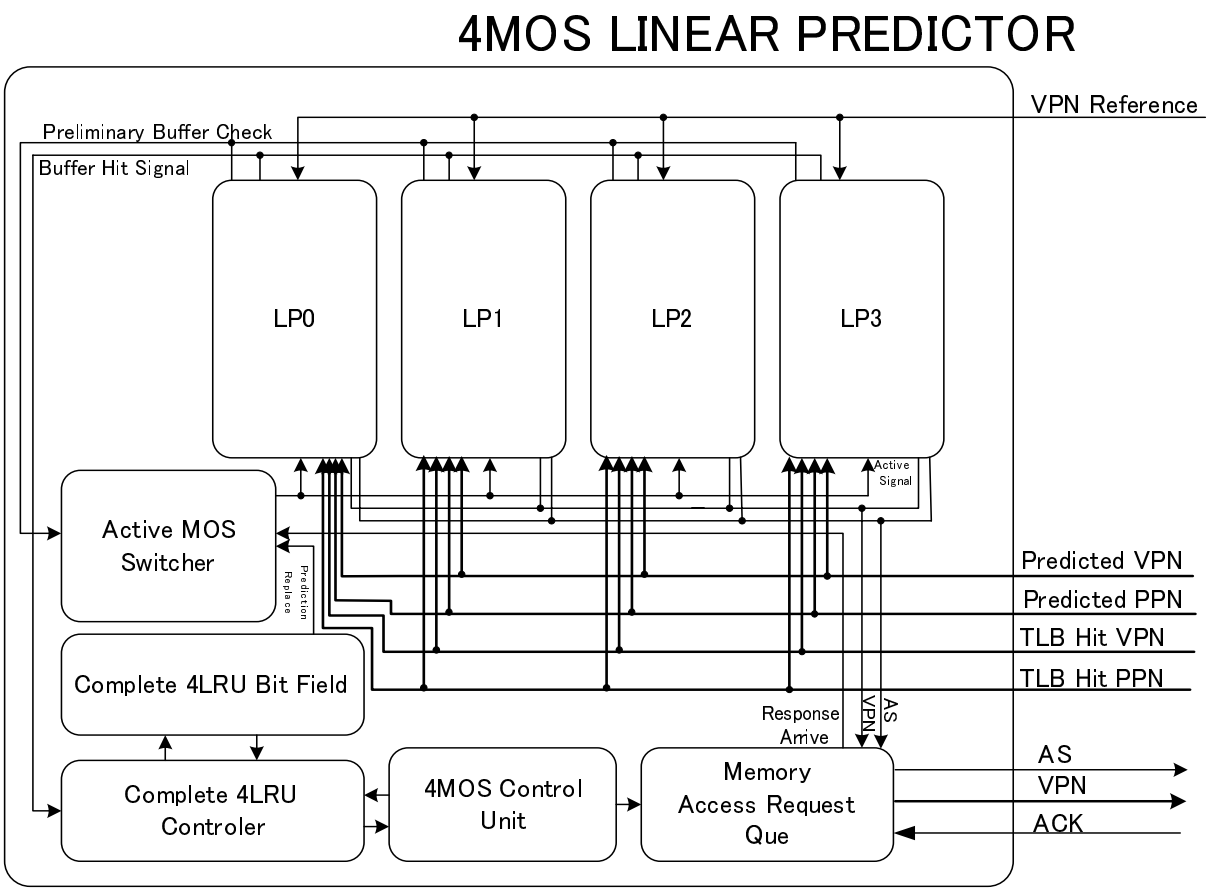

図 3.8: 4MOS 時の実装ブロック図.

るためほぼ不可能となる．完全 LRU を実現するコントローラは `LRU Controler` である．この回路は現在の `LRU Bit Field` とバッファヒット情報から置換対象予測機構を常に選択している回路である．

MOS 構成を採用しない線形ページアドレス予測機構はバッファ内部で予測値の遷移が発生する場合にメモリアクセスが発生した．バッファヒットが発生しバッファの RRP ビット変更後，予測が開始され (予測 VPN が発行され)，予測 PTE が到着するまで，次のバッファ遷移は起こらない設計となっていたため，発行したメモリアクセス要求に対する応答があるまでは新しいメモリアクセス要求は発生しなかった．しかしながら，MOS 構成を採用した線形ページアドレス予測機構は個々の LP が独立して予測を行うため，VPN 参照が変わる時に予測機構全体としてのメモリアクセス要求が多重に発生する問題がある．この問題を解決するために `Memory Access Request Que` をメモリアクセス要求の出口に配置した．この回路はメモリアクセスをキューイングする機能を有する．キュー動作は LP からメモリアクセス要求が到着したときエンキューされ，キューイングされているメモリアクセス要求はメモリから送られる ACK のタイミングでデキューされる．LP

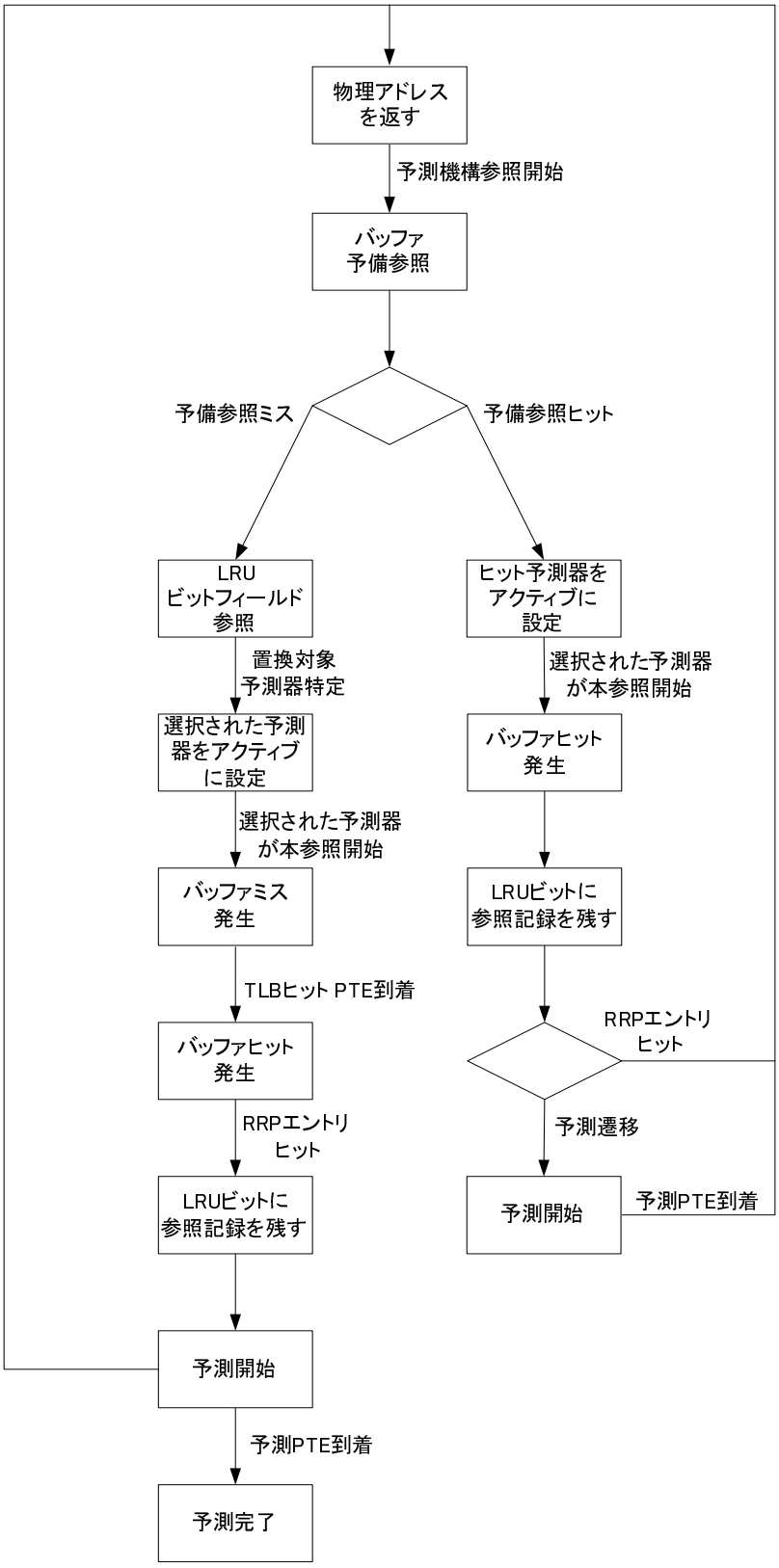

図 3.9: MOS 機構内部制御フロー図.

から到着するメモリアクセス要求は参照 VPN の変化によって発生するため同時に発生することが無いことからこの回路にはメモリアクセスをシリアライズする機能は有していない．また，メモリ側からの ACK を個々の線形ページアドレス予測機構に分配するため `MOS Control Unit` が存在する．ACK は `Memory Access Request Que` に記録されているメモリアクセス要求の履歴情報を元に分配される．

MOS 構成は増やすほどに線形ページアドレス予測機構を有効に利用できる構成である．しかしながら，MOS 構成は 1 つの線形ページアドレス予測機構のハードウェア量の MOS 数倍が必要となり，かつ MOS 制御用のハードウェアも必要となる．加えて，3.2.2 で既に論じたように，MOS 構成時は各予測機構のバッファを全て参照しなければならず，遅延は増大する．これらの理由から MOS 構成はハードウェア量と遅延量の側面から適切な MOS 数を選択することが重要である．

3.3 WRS と MOS の適用と複合構成

プログラム実行の際，発生するメモリアクセスは命令メモリアクセスとデータメモリアクセスである．本論文ではそれぞれのメモリアクセスに予測を適用するために命令 TLB とデータ TLB が独立して存在するスプリット TLB を想定している．WRS と MOS 適用はそれぞれのメモリアクセス傾向に沿って適用しなくてはならない．

3.3.1 命令アクセスへの線形ページアドレス予測機構の適用

命令 TLB は通常，TLB リーチ不足でスラッシングが起きる状況に陥るとプログラム実行にとって深刻なオーバーヘッドとなる．そのため，ハードウェア構成を決定する際，命令 TLB エントリ数を大幅に減らしハードウェア量及び遅延の調整をとることが難しい．しかしながら，明確な線形アクセス傾向が存在する命令アクセスは線形ページアドレス予測機構が最大限有効に実行効率を改善するため，本機構を命令アクセスに適用する場合，PPTE が減少する理由から命令 TLB のエントリ数を大幅に減らすことが可能となる．これは本機構が TLB ミス回数の減少と共にハードウェア量減少を実現することを意味する．WRS と MOS の命令アクセスへの適用はほぼ効果が得られない．1 ページ 4KB の場合，1 ページに存在できる命令数は 1024 命令．WRS 構成を採用しない線形ページアドレス予測機構の場合基準となるページの ± 1 の範囲へのブランチ／ジャンプを予測するので 2048 命令より大きいブランチ／ジャンプが存在しなければ予測は外れない．一般的に実行するプログラムの殆どがループボディである特性から最も大きいアウターループボディが 2048 命令を超えなければ殆どの場合プログラムの実行中予測は的中する．また遠い位置の関数呼び出しで大きくページを跨ぎアクセスされ，予測が外れた場合でもその関数のエントリポイントの存在する PTE が PPTE と認識され，TLB エントリを消費するが，そこからの基本的なループ構造の予測は的中するため，全体的なミスの割合は遠い位置への関数呼び出しの割合のみとなる．このことから WRS と MOS 構成はそのハードウェ

ア，遅延量増大に見合った性能を得ることができない．しかしながら，もっと注意深く命令 TLB ミスを減らしたい場合は，WRS を適用することで関数呼び出し時の TLB ミス数を減らすことは可能である．

3.3.2 データアクセスへの線形ページアドレス予測機構の適用

データアクセスは命令アクセスと比べプログラマ依存であるため予測が成立しにくい．そのためデータアクセスは出来るだけ広範囲の予測をするべきである．このため WRS と MOS を組み合わせた構成が有効である．WRS 型予測器を MOS で並列に配置することで，TLB に挿入される PPTE を更に減少させることが可能となる．WRS と MOS を組み合わせた予測バッファ構成を図 3.10 に示す．

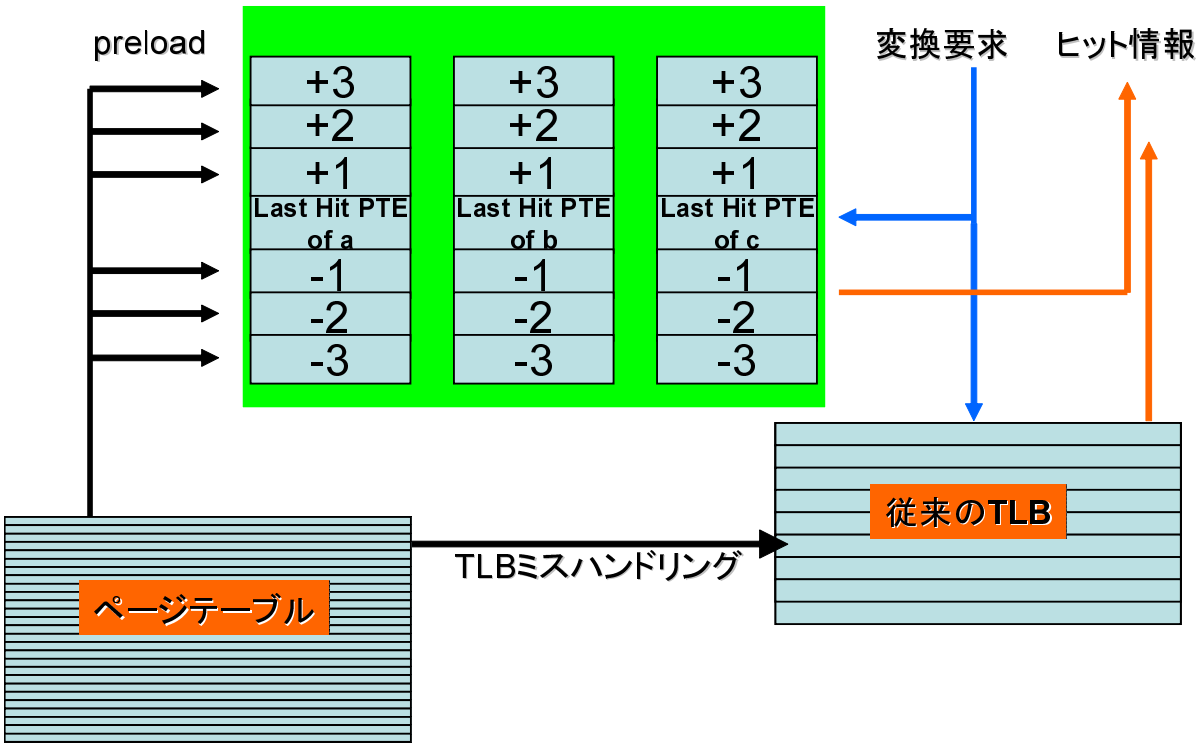

図 3.10: WRS と MOS を組み合わせた予測バッファ構成．

第4章　実装ハードウェア

本研究はハードウェア実装／計測による研究である．提案したメカニズムは全てハードウェア記述言語 VHDL を使用し，実際の回路設計を行った．これはハードウエア量，遅延，実行時間に与える効果を正確に算出することにより，提案した機構の有用性を示す．本章では設計したハードウェアの詳細な説明を行うと共にその特徴，特性を説明する．今回計測した実験環境ハードウェアの概要を示す図を図 4.1 に示す．

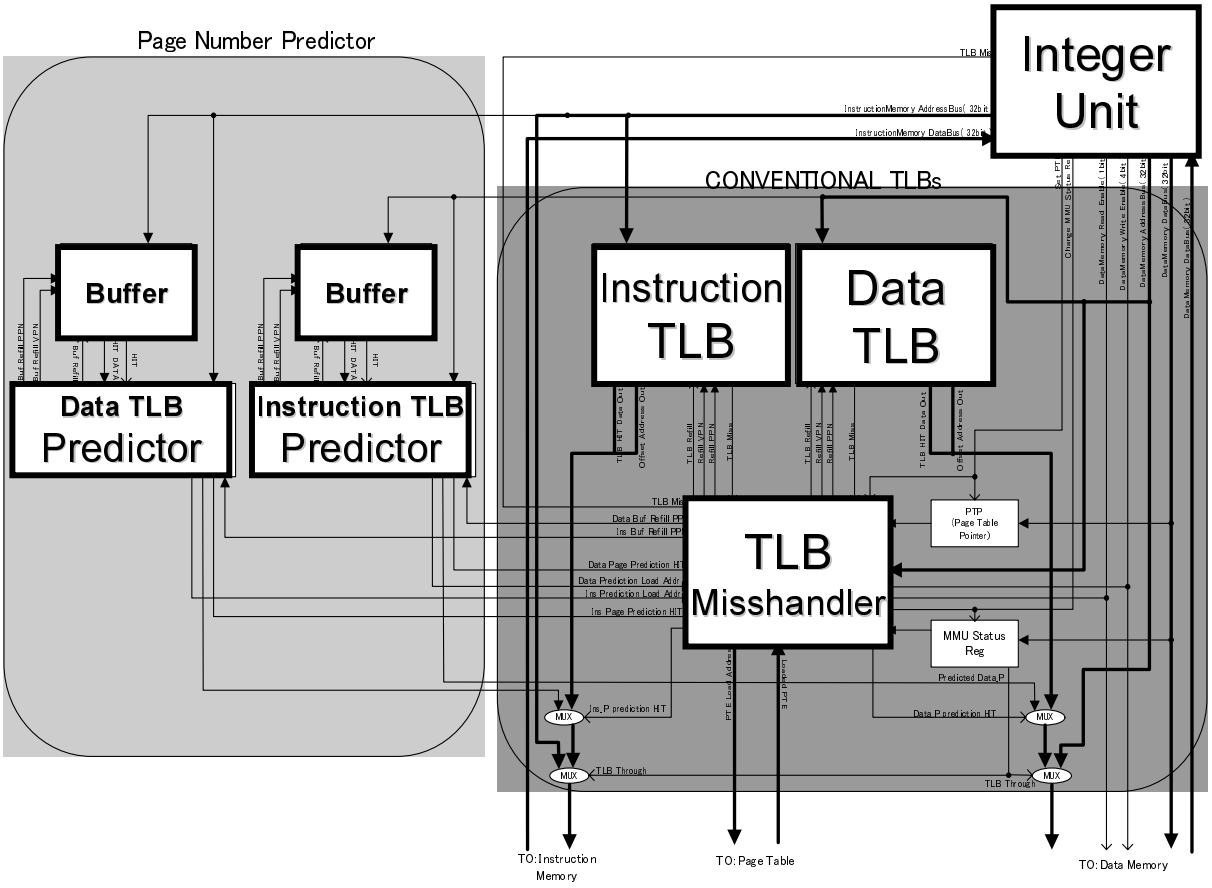

図 4.1: 実験環境ハードウェアブロック図.

図から実装ハードウェアは大きく 3 つのブロックに分けられることがわかる．命令実行パイプライン，TLB と TLB ミスハンドラ，予測機構である．それぞれのブロックは一般

的に整数演算器か MMU(Memory Management Unit) に分類され，TLB，TLB ミスハンドラ，予測機構は MMU に属する機構となる．以降の節は個々のハードウェアの詳細な説明を行う．

4.1 命令実行パイプライン [Integer Unit]

本研究では提案した機構が実際の CPU 上で有効に動作することを検証するために Integer Unit を実装した．設計した Integer Unit の簡略ブロック図を図 8.1 に示す．

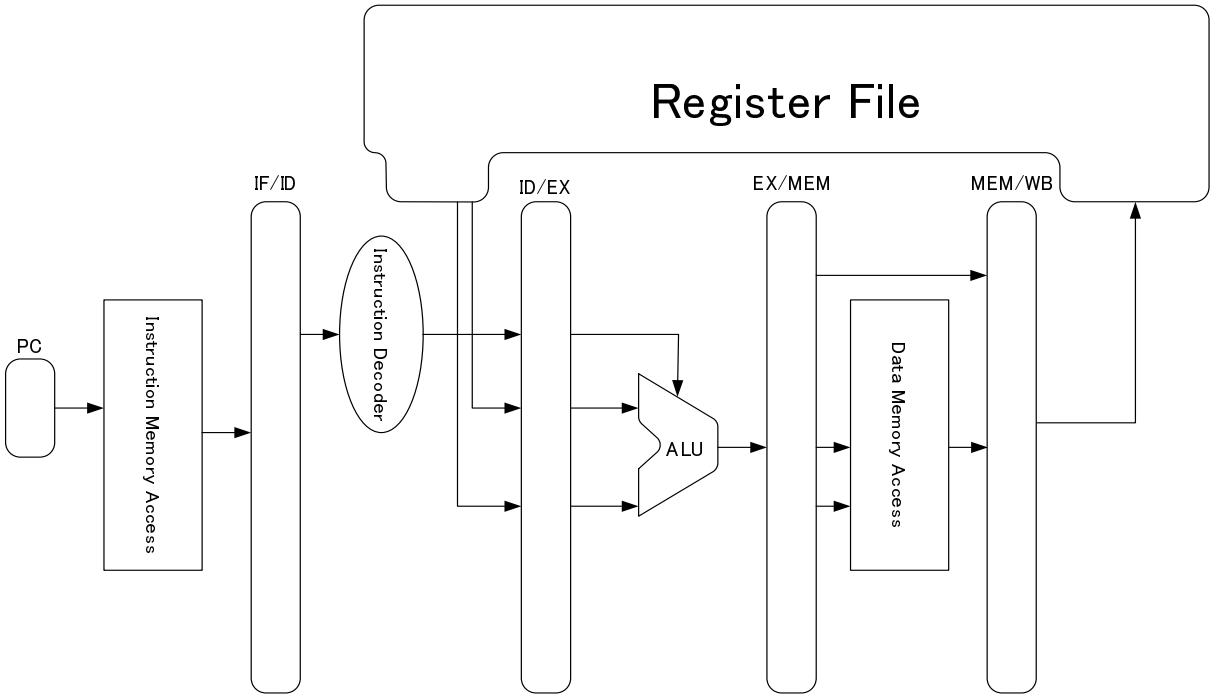

図 4.2: パイプラインの簡略ブロック図．

パイプラインは Out-of-Order，分岐予測，割り込み／例外処理等の機能を持たない 5 段パイプライン構成である．レジスタ，演算，メモリアクセスは全て 32 ビットで実現されている．ステージ分割は命令フェッチを行う**IF**(**I**nstruction**F**etch) ステージ，命令デコード，レジスタファイルからの読み出しを行う**ID**(**I**nstruction **D**ecode) ステージ，命令実行を行う**EX**(**EX**ecution) ステージ，メモリのロード／ストアを行う**MEM**(**MEM**ory access) ステージ，レジスタ書き込みを行う**WB**(**W**rite **B**ack) ステージに分割されている．命令セットは MIPS*I* 命令セットを採用し，乗除算，浮動少数演算以外の命令が実行できる．少数段のパイプライン設計であるため分岐命令実行は動的分岐予測機構を採用しておらず，分岐命令を高速化するため，算術演算を要する分岐命令以外は ID ステージで実行する．

それに伴い，EX ステージ以外にも ID ステージへのフォワディングが実現されている．本論文ではコンテキストスイッチやソフトウェア TLB ミスハンドリングに関して計測／評価を行なわないため割り込み／例外処理は実装していない．その他 MMU 関係を除いたアーキテクチャ依存部分は全て MIPS R3000 シリーズに準じている．図は設計したパイプラインのの概略図である．この他，細部にわたるブロック間結線図は付録 A に示す．

4.2 MMU

本節では MMU 内の予測機構以外の機構を説明する．Integer Unit は MIPS 準拠の設計を施したが，MMU は MIPS の実装には準じていない．MIPS は本来ソフトウェア TLB ハンドリングであるが，本 MMU は予測機構と相性が良いことからハードウェア TLB ミスハンドリングを採用した．設計した MMU を図 4.3 に示す．

4.2.1 MMU 仕様

ハードウェア TLB ミスハンドラ導入により MIPS の TLB ミス解決命令 TLBI(TLB Index)，TLBR(TLB Read)，TLBW(TLB Write)，TLBP(TLB Probe) は使用できない設計とした．コプロセッサ 0 アクセス命令 MTC0(Move To Coprocessor 0)，MFC0(Move From Coprocessor 0) が実装されれている[1]．これを用い MMU の環境設定及び予測機構の環境設定ができるよう従来のコプロセッサレジスタのバインドを変更している．新しいコプロセッサ 0 レジスタバインドを表 4.1 に示す．

通常の MIPS 実装は TLB のアクセスを仮想空間的なセグメントに分割し制御している．MIPS R3000 における TLB アクセス制御を表 4.2 に示す．

表ではキャッシュのアクセス制御も含むが，本研究ではキャッシュ機構をシステム内に想定しない．この MIPS 特有の TLB アクセス制御を本 MMU 機構は廃し，新たに Reserved となっていた COP0 の 31 番レジスタに**MMU and Predictor Status(MPSR)** レジスタを設けアクセス制御と新たに追加する線形ページアドレス予測機構の制御をするためのフィールドとする．新たに追加した**MMU and Predictor Status** レジスタのビットフィールドを図 4.4 に示す．

MPSR には 1 ビットの制御フィールドが 4 つ存在する．TA(TLB Active) は TLB を有効にする場合に 1 をセットするビットである．このビットが 0 の場合，TLB 参照は行われずパイプラインから出されたアドレスがそのままメモリアクセス要求時のアドレスとなる．つまり，パイプラインが物理アドレスを扱える状態となる．TF(TLB Flush) は TLB を初期状態に戻す場合に 1 をセットする．このビットが 1 の場合，TLB の全エントリの Valid

[1]MIPS CPU のメモリマネジメントは Integer Unit とは別に用意された機能ユニットの**コプロセッサ**で機能する．通常の MIPS 実装では，プロセッサステータス管理及びメモリ管理をコプロセッサ 0(**COP0**)，浮動少数処理をコプロセッサ 1(**COP1**) が行う．

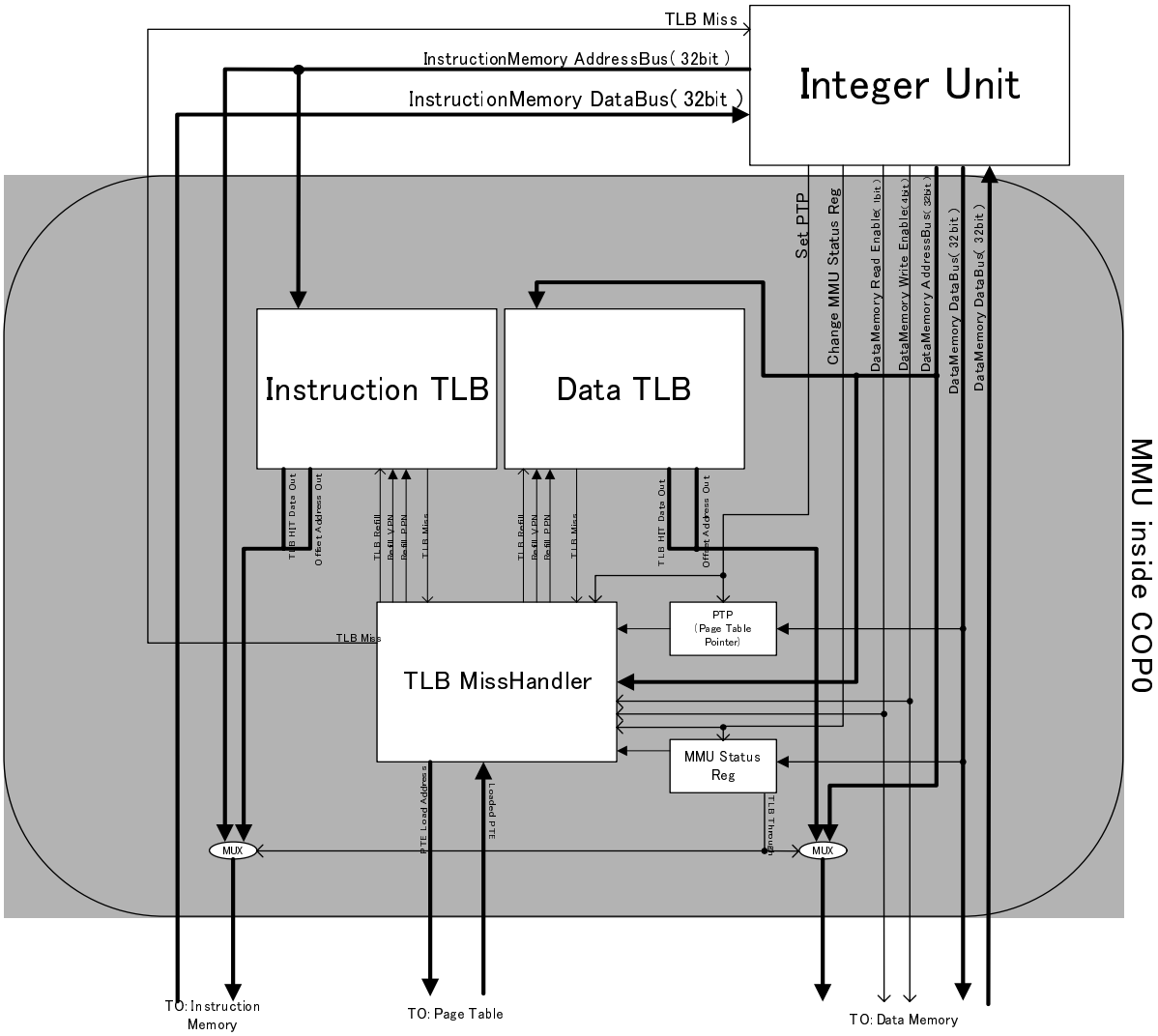

図 4.3: MMU の実装ブロック.

表 4.1: コプロセッサ 0 レジスタバインド.

COP0 レジスタ番号	従来の COP0 レジスタニーモニック	設計した COP0 レジスタニーモニック
0	*Index*	＜ *Reserved* ＞
1	*Random*	＜ *Reserved* ＞
2	*EntryLo0*	＜ *Reserved* ＞
3	*EntryLo1*	＜ *Reserved* ＞
4	*Context*	*PTP Register*
5	*PageMask*	＜ *Reserved* ＞
6	*Wired*	＜ *Reserved* ＞
8	*BadVaddr*	＜ *Reserved* ＞
10	*EntryHi*	＜ *Reserved* ＞
12	*SR*	*SR*
13	*Cause*	＜ *Reserved* ＞
15	*PRID*	*PRID*
14	*EPC*	＜ *Reserved* ＞
31	＜ *Reserved* ＞	*MMU and Predictor Status*

表 4.2: MIPS R3000 におけるセグメント別アクセス制御.

アクセス制御	セグメント名	アドレスレンジ	容量
TLB OFF & Cache ON	*kseg0*	0xa000 0000〜0xbfff ffff	0.5GB
TLB OFF & Cache OFF	*kseg1*	0x8000 0000〜0x9fff ffff	0.5GB
TLB ON & Cache ON	*kseg2*	0xc000 0000〜0xffff ffff	1GB
	kuseg	0x0000 0000〜0x7fff ffff	2GB

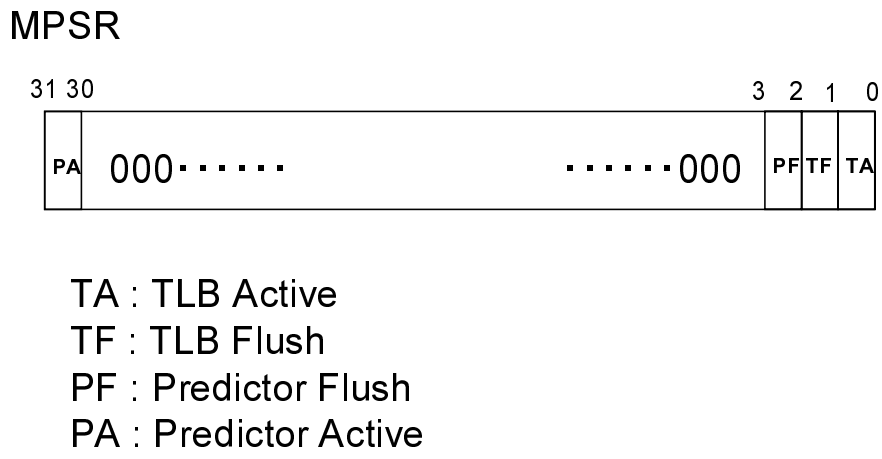

図 4.4: MPSR フィールド.

ビットが強制的に 0 になる．ブートアップ時はハードウェアリセットで Valid が 0 になるためブートストラップでの操作は不要である．コンテキストスイッチ時など TLB 使用中に TLB の全てのエントリを無効にしたい場合に使用する[2]．PF(Predictor Flush) は予測機構を初期化したい場合に使用する．予測バッファの全エントリの Valid フィールドを強制的に 0 にする．予測バッファはハードウェアリセット時に Valid を 0 にするためブートストラップでの操作は不要である．コンテキストスイッチ時や明示的に全予測を破棄したい場合に有効である．但し，複数の予測機構が存在する場合には予測機構を指定して予測破棄を行えず，全ての予測機構の予測を 破棄しなければならない．PA(Predictor Active) 予測機構を有効にする場合に 1 をセットする．このビットが 0 の場合，予測機構は予測を中断し，待機モードに移る．カーネルモード動作時や TLB 非動作時，また明示的に予測を中断したい場合に有効である．複数の予測機構が存在する場合予測機構を指定して操作できないため，全ての予測機構の予測が中断する．予測機構は TLB が動作していない状態では動作しないので注意が必要．このビットに 1 を立てる場合は TA ビットが必ず 1 でなければならない．

本論文で設計したハードウェア TLB ミスハンドラはアドレス変換機能のみに注目した回路であり，VPN から PTEA を算出する回路と そのロード回路のみで構成されており，Dirty ビットのある TLB エントリのライトバック，メモリ保護等を行う回路は存在しない．また，これらの機能を実装する場合は TLB エントリだけでなく，予測機構のバッファのエントリにも必要となることに注意が必要である．

[2]実際の MIPS 実装では TLB エントリのフィールドに ASID フィールドが存在しコンテキストスイッチ時に TLB を Flush する必要はない．本論文での実装はコンテキストスイッチを意識していないため ASID フィールドを付 け加えなかった．詳細は 4.2.2 で示す.

4.2.2 TLB

本 TLB 機構は通常の MIPS の TLB 機構と根本的に異なる．対比のため通常の MIPS R3000 の TLB のエントリのフィールドと本 TLB 機構の TLB エントリフィールドの図を図 4.5 に示す

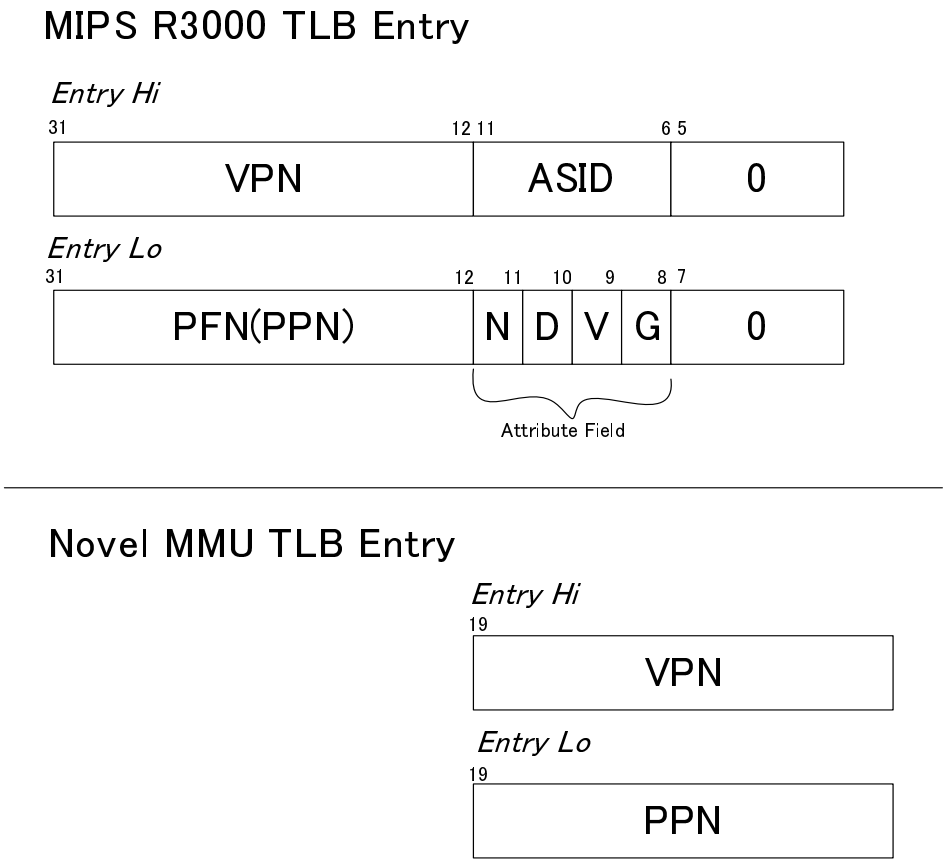

図 4.5: TLB エントリフィールド比較図.

図は 32 ビットアーキテクチャ，4KB ページサイズの場合の TLB エントリの構成である． TLB エントリは MIPS の形式上 Entry Hi と Entry Lo に分けられる．Entry Hi はハードウェアによって参照されるフィールドでエントリのヒットを検索するためのインデックスとなるフィールドである．MIPS R3000 の Entry Hi のフィー ルドは仮想ページ番号の VPN とコンテキスト番号の ASID と 0 オフセット[3]で構成される．設計した TLB の Entry Hi は VPN フィールドのみで構成されている．本論文ではコンテキストスイッチ時の振る舞いは想定していないためエントリ設計では ASID フィールドは存在しない．Entry Lo はヒット時に必要となる情報が格納されている． MIPS R3000 の Entry Lo は

[3]0 オフセットは TLB のソフトウェアハンドリング時にプログラムがこのレジスタの内容を 32 ビットデータとして扱うことができるために存在する．

PFN(Physical Flame Number)[4]とページ属性を示す Attribute フィールドと 0 オフセットで構成される．PFN は参照ヒット時仮想〜物理アドレス変換としてハードウェア的に使用されるフィールドである．Attribute フィールドは N，D，V，G ビットが存在し，ページのこのビットの組み合わせで属性が表現される．設計した TLB の Entry Lo は PPN ールドのみで構成されている．本論文ではページ保護等ページ属性を使用する処理は想定していないためエントリ設計では Attribute フィールドは存在しない．MIPS R3000 における Attribute フィールドの各ビット要旨を表 4.3 に示す．

表 4.3: MIPS R3000 における Attribute フィールドの各ビットの要旨

ビット名	要旨
N	ノンキャッシャブルビット．1 がセットされると，そのページへのアクセスはキャッシュスルーとなる．
D	ダーティービット．1 がセットされていない状態で Write オペレーションが発生した場合に例外が発生する．書込み保護の意味があるのと同時にページアウト時にページ内容の一貫性を保つためにも使われる．
V	エントリが有効であることを示すビット,1 がセットされていない TLB エントリがヒットするとミス例外発生がする．
G	グローバルビット，1 にセットされると Entry Hi 参照時に ASID 比較が無視される．

これらのビットは本論文ではアドレス変換のみに注目したため未実装としたが，実際のシステム実装においては必要不可欠なビットである．MIPS 実装においてこれらのビットフィールドの構成はソフトウェアハンドリング用に最適化されたフィールドであることから，ハードウェアハンドリング実装時はこのフィールドとは全く異なるフィールドを用意しなければならない．線形ページアドレス予測機構はソフトウェアハンドリングの TLB に対しても実装は可能であるが，線形ページアドレス予測機構から発行される要求に対するミスハンドリングはハードウェアミスハンドリングで行わなければならない．異なるミスハンドリング機構が共存するシステムの場合，線形ページアドレス予測機構のバッファに実装する Attribute ビットへの TLB からのヒット PTE 転送が極端に難しくなる可能性がある．特に，MIPS 実装の場合，例外を引き起こす Attribute ビットがあるため注意が必要となる．望ましい実装は予測機構に合わせて TLB もハードウェアミスハンドリングする MMU である．その場合ハードウェアミスハンドラが予測機構と TLB で共有化でき，無駄なハードウェア量増大とならない．

[4]PPN(Physical Page Number) と同意．MIPS R6000 シリーズから広義の意味で PPN と呼ばれるようになった．

本論文で設計された TLB の回路図を図 4.6 に示す.

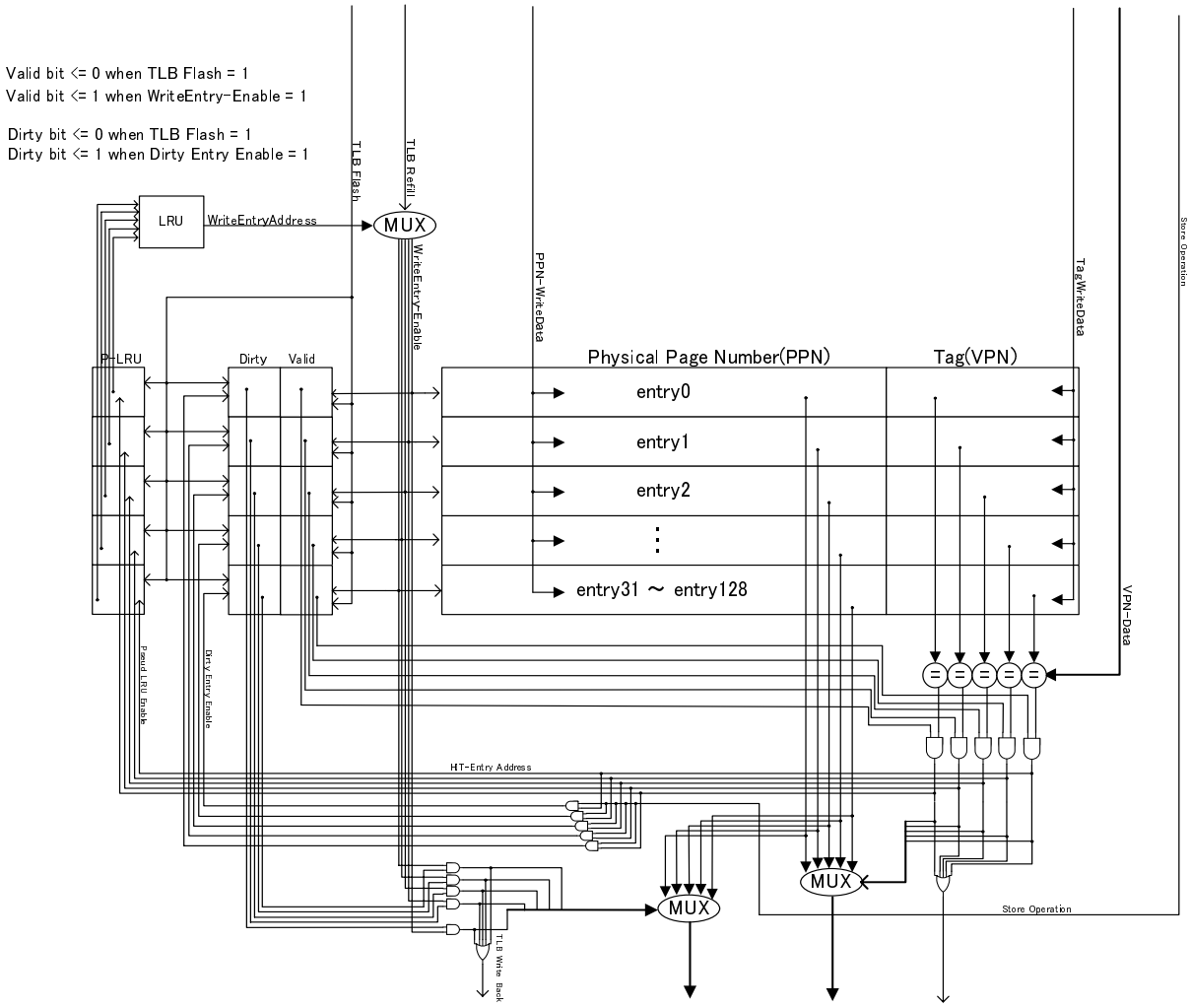

図 4.6: TLB 回路図.

図は TLB のハードウェア構成を示しており，矩形部分は記憶素子の配列を意味する．エントリは VPN フィールドと PPN フィールドが存在し Attribute フィールドが存在しない．この理由に関しては既に 2.2.1 で論じた．コントロールビットは LRU 情報を記録する LRU ビット，ページ書き込みを記録する Dirty ビット，エントリの有効を示す Valid ビットで構成される．本論文の TLB は LRU ビットを 1 ビットで実現する疑似 LRU を実装した．ページサイズ，エントリ数は論理設計時可変設計とした．実装時は 4KB ページサイズを想定したため VPN，PPN フィールドは各 20 ビット，エントリ数は 32 エントリとした．尚，本論文の計測において Dirty ビットは関係しないため実際のミスハンドラへの配線は行われていない.

第5章　実験方法論

本研究はハードウェア実装／計測による研究である．提案した機構は全てハードウェア記述言語 VHDL を使用し，実際の論理回路設計を行った．これはハードウエア量，遅延，実行時間に与える効果を正確に算出することにより，機構の有用性を示すためである．本章では提案機構のハードウェア量評価と実行時間評価の方法を論じ，実行時間計測の際使用した C 言語プログラムを示す．

5.1　ハードウェア量、遅延の評価

本方式が従来の TLB 性能を下げることなく線形ページアクセスが発生した場合に有効に機能することは 2.2 で既に論じた．しかしながら，実装面において線形ページアドレス予測機構の有用性を示すためには，性能向上だけでなく性能向上に見合う十分に少ないハードウェア量と遅延で予測機構が実現されていることが重要である．設計したハードウェアのハードウェア量と遅延を測定するために Synopsys 社の FPGA Compiler II を利用し Xilinx FPGA VIRTEX2 2V6000FF1152 をターゲットデバイスとして目標周波数値 (制限) を 20MHz にして論理合成を行った．論理合成後のレポートに記述される Primitive reference count を参照しハードウェア量とした．この情報としては，FPGA 用に論理合成されたネットリストの LUT(LookUp Table)，フリップフロップ等の使用量がレポートされる．実際の LSI としての厳密なゲート使用量の指標にはならないが，提案した機構のシステム内の相対的大きさを示す指標としてこの情報を利用した．また、Timing Path Groups と Clocks を参照し遅延とした。

5.2　実行時間計測

本機構の実際のプログラム実行上での評価のため，Model Technology 社の論理回路シミュレータ ModelSim 上で C 言語で記述したプログラムを実行し実行クロックサイクル数と TLB ミスペナルティの計測を行った．また今回は純粋な TLB の性能を評価するために，命令/データキャッシュはシステム構成の中に入れていない．MMU と TLB に関して，本来の MIPS の方式ではソフトウェア TLB ミスハンドリングを前提としているが，計測システムハードウェア TLB ミスハンドリングを採用している．このことにより予測／ミスハンドリングを全てパイプラインのバックグラウンドで実行することが可能となった．

予測／ミスハンドリングは 1 クロックサイクルで実行できる設計となっている.

5.2.1 計測プログラム

本計測で使用したプログラムを図 5.1 に示す.

```
#define PAGE_SIZE 1024 /* 4KB/4 */
#define NUM_OF_TLB_MISS 5
int test_data[ PAGE_SIZE*NUM_OF_TLB_MISS ];
int dataset_access();
int main( void ){
    register int a;
    a = dataset_access();
}
int dataset_access(){
    int i,sum;
    for ( i=0;i<PAGE_SIZE*NUM_OF_TLB_MISS;i++)
        test_data[ i ] = i;

    for ( i=0;i<PAGE_SIZE*NUM_OF_TLB_MISS;i++)
        sum += test_data[ i ];

    return sum;
}
```

図 5.1: 計測用プログラム 1.

第6章　評価

本章では提案、設計した機構の評価を行う。評価は 6.1.1 ハードウェア量、6.1.2 遅延の評価とプログラム実行効率に分けて評価する。

6.1　設計ハードウェア評価

本節では設計したハードウェアの結果と評価を示す。本論文でハードウェア評価の対象としたのは次に示すハードウェアである。実行パイプライン単体、予測機構無し MMU 単体、1WRS-1MOS の完全線形ページアドレス予測機構単体、1WRS-4MOS 完全線形ページアドレス予測機構単体、2WRS-1MOS バースト適用線形ページアドレス予測機構単体、予測機構を備えない CPU、1WRS-1MOS の完全線形ページアドレス予測機構を備えた CPU、1WRS-4MOS 完全線形ページアドレス予測機構を備えた CPU、2WRS-1MOS バースト適用線形ページアドレス予測機構を備えた CPU である。予測機構を備えない CPU の内部構成は命令実行パイプラインと MMU で構成される。1WRS-1MOS の完全線形ページアドレス予測機構を備えた CPU の内部構成は命令実行パイプラインと MMU と命令、データメモリアクセスに対してそれぞれ一つずつの 1WRS-1MOS 完全線形ページアドレス予測機構で構成される。1WRS-4MOS 完全線形ページアドレス予測機構を備えた CPU は命令実行パイプラインと MMU と命令メモリアクセスに 1WRS-1MOS の完全線形ページアドレス予測機構とデータメモリアクセスに 1WRS-4MOS 完全線形ページアドレス予測機構で構成される。2WRS-1MOS バースト適用線形ページアドレス予測機構を備えた CPU は命令実行パイプラインと MMU と命令メモリアクセスに 1WRS-1MOS の完全線形ページアドレス予測機構とデータメモリアクセスに 2WRS-1MOS バースト適用線形ページアドレス予測機構で構成される。

6.1.1　ハードウェア量評価

設計したハードウェアのハードウェア量の測定結果を表 6.1 に示す。

表の機構名は略称となっている。それぞれの意味を示す。実行パイプライン単体 (PIPELINE)、予測機構無し MMU 単体 (MMU)、1WRS-1MOS の完全線形ページアドレス予測機構単体 (1WRS-1MOS)、1WRS-4MOS 完全線形ページアドレス予測機構単体 (1WRS-4MOS)、2WRS-1MOS バースト適用線形ページアドレス予測機構単体 (2WRS-1MOS)、予測機構

表 6.1: ハードウェア量測定結果

機構名	使用セル名						
	FDE	FD	LUT	XORCY	TLB	MUX	CMP
PIPELINE	1502	0	3765	16	0	0	0
MMU	2723	0	2411	8	0	1	0
1WRS-1MOS	160	7	330	4	0	0	0
1WRS-4MOS	740	40	1736	0	0	0	0
2WRS-1MOS	213	0	246	2	0	0	1
Normal CPU	4224	0	6087	23	0	0	0
1WRS-1MOS CPU	4718	16	7661	27	0	0	0
1WRS-4MOS CPU	2610	45	5998	23	2	0	0
1WRS-4MOS CPU'	5298	45	9100	23	0	0	0
2WRS-1MOS CPU	4795	10	7658	25	0	0	0

を備えない CPU(Normal CPU)、1WRS-1MOS の完全線形ページアドレス予測機構を備えた CPU(1WRS-1MOS CPU)、1WRS-4MOS 完全線形ページアドレス予測機構を備えた CPU(1WRS-4MOS CPU)、2WRS-1MOS バースト適用線形ページアドレス予測機構を備えた CPU(2WRS-1MOS CPU) である。表 6.1 で示される使用セル名を説明する。FDE はイネーブル付き D フリップフロップの数，FD は D フリップフロップの数、LUT はランダムロジックに使用されたルックアップテーブルの数、XORCY はキャリーロジックに使われる特殊な XOR 素子の数である。TLB，MUX、CMP に関しては FPGA コンパイラが論理設計上一まとまりになるブロックをまとめて表示したものである。MUX はマルチプレクサ、CMP は比較器である。これらのハードウェア量は LUT に加算されるものであるが一つのハードウェア量は不明である。しかしながらどちらも回路設計上極小さな部品であることから CPU 全体から見た場合には無視できる大きさである。TLB は本論文において設計した TLB である。これは LUT，FDE に関係するものであり、ハードウェア量も大きいため回路設計データから個別に抜き出し単体で論理合成を行った。その結果、1TLB は FDE が 1344、LUT が 1551 という結果になった。TLB 数のレポートが出されているのは、1WRS-4MOS CPU である。1WRS-4MOS CPU の FDE と LUT にこの値を 2 倍し足すことで FDE，FD，LUT，XORCY のみでハードウェア量を比較することができる。この値を足し合わせた結果が表の 1WRS-4MOS CPU' である。

ハードウェア量評価を相対的大きさの見地から見る場合、それぞれの予測機構は十分に小さいハードウェア量で実現されていることがわかる。Normal CPU に対して記憶容量では 1WRS-1MOS が約 3.5% 程度、1WRS-4MOS が約 17.5% 程度、2WRS-1MOS が約 5.0% 程度、ランダムロジックでは 1WRS-1MOS が約 5.5% 程度、1WRS-4MOS が約 28.5% 程度、2WRS-1MOS が約 4.0% 程度のハードウェア量である。MOS 機構のランダムロジッ

クが若干大きいが、これらのハードウェアは今日、ダイ面積の 50%〜80% を占めるキャッシュと FPU が Normal CPU の構成に含まれていないことから、極小さいハードウェアであることがわかる。実際の CPU 構成上での実装はこのほかにも TLB-予測器間の制御回路が付属するため若干量大きくなるが、ほぼ無視できる大きさである。実際の CPU 構成におけるハードウェア量は Normal CPU に対して記憶容量では 1WRS-1MOS CPU が約 11.5% 増加、1WRS-4MOS CPU' が約 24.5% 増加、2WRS-1MOS CPU が約 13.5% 増加した。ランダムロジックでは 1WRS-1MOS CPU が約 25.85% 増加、1WRS-4MOS CPU' が約 49.4% 増加、2WRS-1MOS CPU が約 25.8% 増加した。

6.1.2 遅延評価

設計したハードウェアの遅延の測定結果を表 6.2 に示す。

表 6.2: 遅延測定結果

機構名	In-¿RC(ns)	RC-¿Out(ns)	RC-¿RC(ns)	周波数 (MHz)
PIPELINE	2.57	2.77	11.67	85.69
MMU	5.21	6.02	5.78	166.11
1WRS-1MOS	4.77	4.70	5.34	187.27
1WRS-4MOS	4.77	3.65	5.34	187.27
2WRS-1MOS	3.10	3.82	3.58	261.78
Normal CPU	3.89	11.30	11.67	85.69
1WRS-1MOS CPU	4.77	10.86	11.67	85.69
1WRS-4MOS CPU	4.20	10.86	11.67	85.69
2WRS-1MOS CPU	5.21	10.86	11.67	85.69

機構名は 6.1.2 で説明したものと同一である。遅延は In-¿RC(ns)、RC-¿Out(ns)、RC-¿RC(ns) の 3 つのがレポートされた。それぞれインプットからフリップフロップまでの遅延、フリップフロップからアウトプットまでの遅延、フリップフロップからフリップフロップの間の遅延である。この他入力から出力の直接パスの遅延もレポートされたが、値が小さすぎるため表には記載しなかった。動作速度的に遅延のクリティカルパスとなっているのは PIPELINE の 11.67ns である。これがどの CPU 構成に対しても最大遅延となり動作周波数を決定し、Normal CPU、1WRS-1MOS CPU、1WRS-4MOS CPU、2WRS-1MOS CPU は 85.69MHz で動作することとなった。PIPELINE の入出力は直接メモリアクセスなので、PIPELINE における In-¿RC は RC-¿Out メモリアクセスの遅延と見ることができる。RC-¿Out は TLB 参照を行うため Normal CPU，1WRS-1MOS CPU，1WRS-4MOS CPU，2WRS-1MOS CPU に共通して大幅に増大している。しかし、それぞれの遅延の差は殆ど無い。参照 VPN 数が増加したことにより遅延が増大するはずであったが、若干量

減少する結果となった。この結果については不明であるが、これはFPGAのゲート/配線遅延特性とFPGA Compilerの最適化が影響していると推測している。

6.2 性能評価

本節では提案機構の性能を評価する。

第7章　関連研究

第8章　まとめ

本研究において提案した機構はソフトウェア実行において，従来の MMU よりもプログラム実行効率を向上させることが可能である．メモリ中の線形アクセスされるページ数が多いほど本機構は効力がある．また，単純に TLB ミス回数を抑制するだけでなく TLB スラッシングを抑制する効果があり，将来さらに増大を続けるプログラムのワーキングセットサイズに対する TLB リーチ問題の改善策となる．本機構は従来の MMU を少ない修正で追加可能であり，実装が容易であると共に，十分に小さいハードウェア量で実現可能である．加えて，予測機構を追加することによる従来の TLB 性能を落とすことが無く，線形ページアクセス部分のみの改善を行えることから，他のページアクセス予測機構と共存ができる．

付録

ここでは本文中に参照された付録の説明を記載する.

付録 A：パイプライン結線図

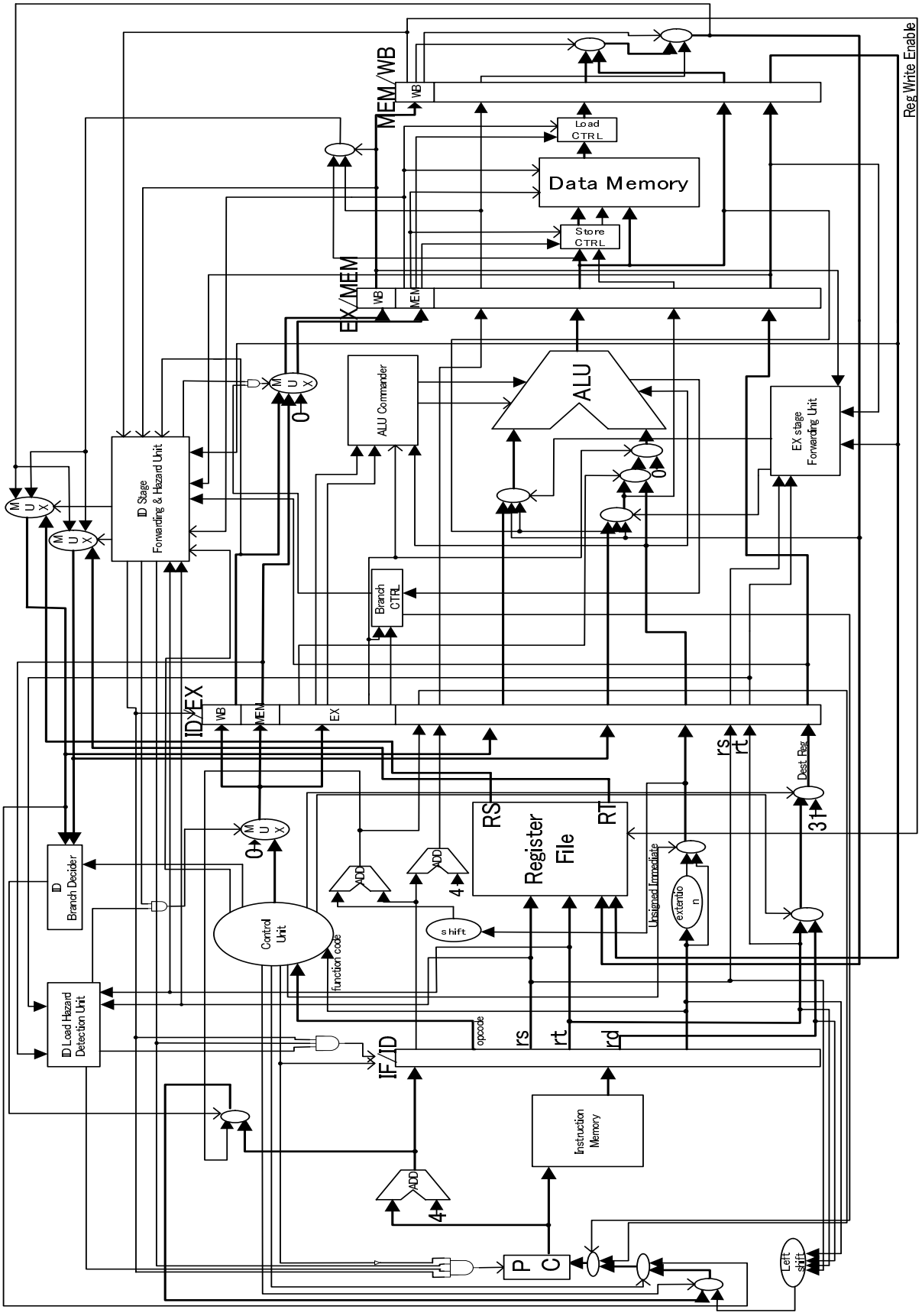

図 8.1: パイプライン結線図.

謝辞

本報告を終えるにあたり，御指導，御教授を頂いた 田中 清史 助教授に心より感謝します．
本研究に御意見，御助言を頂いた 日比野 靖 教授，宮崎　純 助手に深く感謝します．
本研究をサポートしてくださり，議論をしてくださった 北陸先端科学技術大学院大学 計算機アーキテクチャ講座 田中研究室 一同に深く感謝します．
本研究において，Synopsys 社と Model Technology 社の University Program を用いた．深く感謝します．
最後に日頃よりあたたかく見守ってくださった父，母，姉に深く感謝します．

参考文献

[1] Ashley Saulsbury , Fredrik Dahigren and Per Stenstrom: "Recency-Based TLB Preloading" Proceedings of the 27th annual international symposium on Computer architecture, Pages 117–127,2000

[2] M.Talluri, S.Kong, M.D.Hill and D.A.Patterson: "Tradeoffs in Supporing Two Page Sizes" Proc of ISCA, pages 415–424, 1992

[3] Madhusudhan Talluri and Mark D.Hill："Surpassing the TLB Performance of Superpages with Less Operating System Support" In Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems, pages 171–182, 1994.

[4] J. E. Smith, "A Atudy of Branch Prediction Strategies", Proc of ISCA, pp.135–148, May, 1981