計算機アーキテクチャ 150－14
（2002．11．28）

線形ページアドレス予測によるTLBプリローディング

請園智玲† 田中清史†,††

今日のプロセッサはTLBを内蔵することにより仮想記憶のオーバヘッドを軽減している．しかしTLBサイズを超えるページ参照が頻繁に発生する場合にはその効果が得られない．本稿では次に参照されるページアドレスを予測し，あらかじめメモリからプリロードすることによって，TLBのサイズを削減しつつTLBヒット率を向上させる機構を提案し，予備評価を行う．

TLB Preloading by Linear Page Address Prediction

TOMOAKI UKEZONO† and KIYOFUMI TANAKA†,††

Modern processors with a built-in TLB reduce overheads of virtual memory. However, the TLB does not work when a running program frequently accesses more pages than it can cover. In this paper, we propose a mechanism that predicts page addresses to be accessed and preloads the corresponding entries from memory in advance. This can reduce the TLB size and improve the hit ratio. Moreover, we present the preliminary evaluation.

1. はじめに

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つつある．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性能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機構を提案する．

2. 線形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．動的分岐予測[4]に代表される動的予測機構は，プログラム実行中にアドレスの履歴情報を記録しその情報を元に予測を行うが，本研究における予測機構はプログラムに潜在するページアクセス傾向にしたがって予測を行う．このことから履歴情報を格納する必要が無く，少量のハードウェアで実現可能である．

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．最も明確なページアクセス傾向は逐次的にアクセスさ

† 北陸先端科学技術大学院大学 情報科学研究科
Japan Advanced Institute of Science and Technology
†† 科学技術振興事業団，さきがけ研究21，「機能と構成」領域
"Information and Systems", PRESTO, Japan Science and Technology Corporation(JST)

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☆．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．これを線形ページアドレス予測とする．

2.1 線形ページアドレス予測器

本研究で提案する線形ページアドレス予測機構の概要を図1に示す．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，本機構は全てハードウェアで実現される．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，ミスの大部分を防ぐ効果が得られる．

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かつプリロードにより回避するが，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．

3. 非線形ページアクセス対応

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

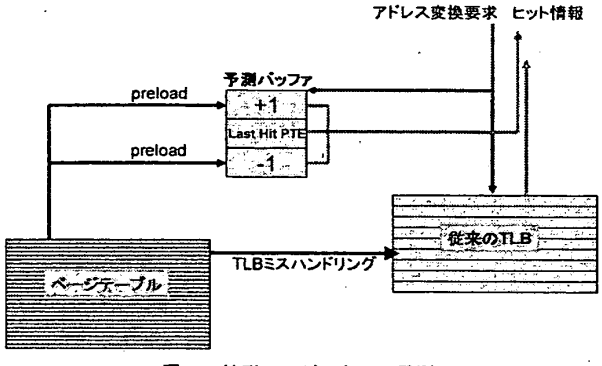

図1 線形ページアドレス予測．

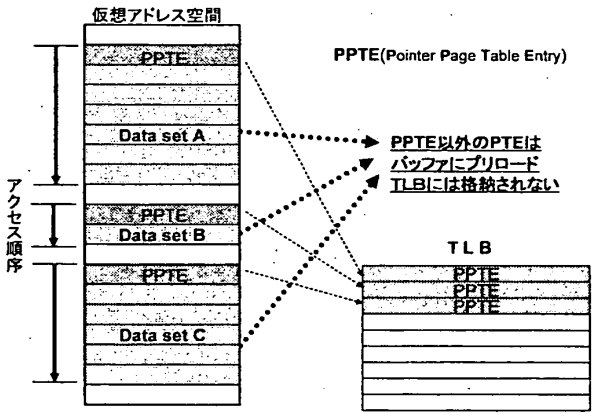

図2 PPTEと予測PTE．

☆ 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．

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．本節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．

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に示す．これにより仮想アドレス空間上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．

予測範囲の過度の拡大はプリロードのためのメモリアクセスを多発させるため，メモリバスが占有される可能性がある．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することが重要である☆．

3.2 Multiple Operands support

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．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．

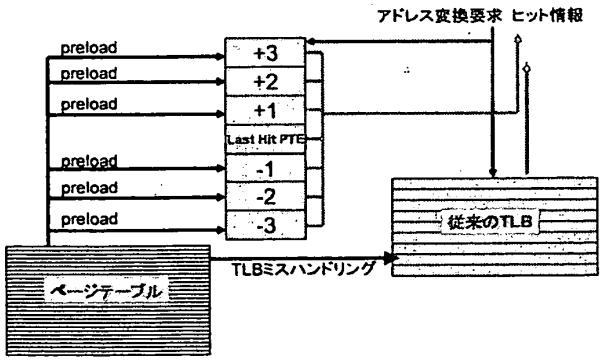

図3 Wide Range Support（WRS）．

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4に示す．MOS構成は予測器の並列度にしたがってメモリアクセスが増加するため，プリロードによるメモリアクセスの増加とバッファアクセス遅延の増大が顕著にならない程度の並列度に抑えることが重要である．

3.3 実装時の予測器ハードウェア構成

命令TLBに線形ページアドレス予測を適用し，分岐，ジャンプ命令で非線形アクセスが発生した場合，ページ内とその±1のページへのアクセスは線形予測に吸収されるため，非線形拡張をしない予測機構のみで十分な性能を確保できる．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有効に実行効率を改善するため，本メカニズムを採用する場合，命令TLBのエントリ数を大幅に減らすことが可能となる．データアクセスは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5に示す．

4. 実装と方法論

本研究はハードウェア実装／計測による研究である．提案したメカニズムは全てハードウェア記述言語

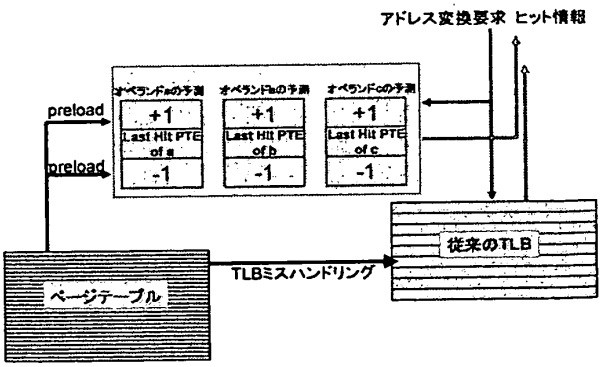

図4 Multiple Operands Support（MOS）．

☆ 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．

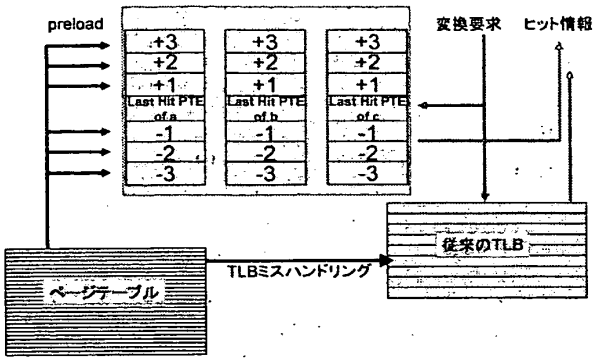

図 5 WRS と MOS を組み合わせた予測バッファ構成.

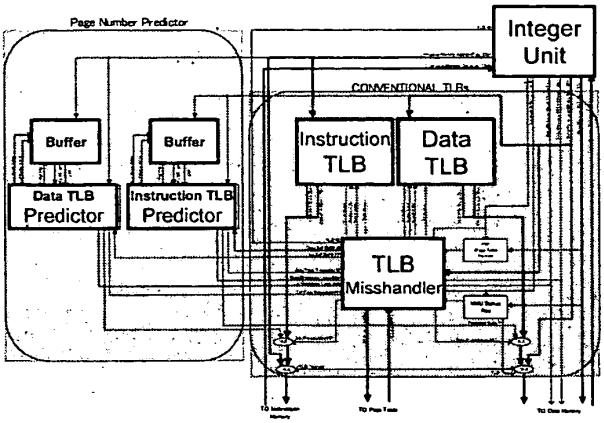

図 6 実験環境ハードウェアブロック図.

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，提案したメカニズムの有用性を示すためである．今回測定したハードウェアの概要を示す図を図 **6** に示す．

設計したハードウェアは大きく Integer Unit, MMU, Predictor の 3 つに分けられる．従来からのハードウェア構成は Integer Unit と MMU の組み合わせである．予測プリロード機構を備えた TLB を実現するため，線形ページアドレス予測機構を基本的な構成に組み合わせることで実現した．従来からの修正点は予測器のページテーブルへのアクセスのためのインターフェースとして TLB ミスハンドラに少量の修正を加えたのみである．

4.1 線形ページアドレス予測器の実装

線形ページアドレス予測機構は予測器とバッファから構成される．ページテーブルへの予測 VPN(Virtual Page Number) の発行とバッファへの格納は全て予測器が行う．予測バッファと TLB は命令参照に対して同時に参照され，TLB リフィルに関しては基本的にバッファミスが発生しかつ TLB ミスが発生した時のみ行われる．バッファの内容が変更される要因はバッファがミスが発生した時とバッファの基準エントリ（最後にバッファか TLB でヒットした PTE が格納されるエントリ）以外がヒットした時である．バッファミスが発生した時の振る舞いは，TLB でヒットした PPN を基準エントリにリフィルした後（TLB ミスをしても TLB リフィルが完了し TLB ヒットになるまで予測機構はロックする），予測 VPN を生成する．その場合，+1 予測の後 −1 予測を行う．バッファ内でヒットし，内容が遷移する場合，ヒットしたエントリを基準エントリに変更し，同時に +1 ヒットの場合，+1VPN 予測を −1 ヒットの場合 −1VPN 予測を行う．また，予測はパイプラインの完全にバックグラウンドで行われるため（TLB ミスはパイプラインをロックさせる）予測が完了してない状態でバッファが参照される場合がある．その場合，非線形ページアクセスでのバッファヒット，線形ページアクセスでのバッファミスが発生する場合がある．これらの参照は全てバッファミスと評価され，全て TLB ヒット情報待ちとなる．この振る舞いをするハードウェアを設計し，このメカニズムを線形ページアドレス予測機構とした．

4.2 ハードウェア量の評価

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既にセクション 2 で論じた．しかしながら，実装面において線形ページアドレス予測メカニズム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を測定するために Synopsys 社の FPGA Compiler II を利用し論理合成をった．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(Look Up Table) の使用量，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，提案したメカニズムのシステム内の相対的大きさを示す指標ために十分であるため，この情報を利用した．

4.3 計測環境

本メカニズムを計測するにあたり設計し，実際のプログラム実行上での評価のため簡単な Integer Unit と MMU を設計した．Integer Unit は MIPS1 命令セット 32bit 5 段パイプランを採用した．パイプラインは Branch Prediction や Out of Order 等の投機実行機構や，スーパースカラ等の並列機構，浮動小数演算機構を持たないシンプルな構成とした．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，本研究では予測／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を前提としたためハードウェア

TLB ミスハンドリングを採用した．TLB の構成はセクション 2.1 で論じたように命令アクセス，データアクセスそれぞれに独立したスプリット TLB を採用し，それぞれの TLB に傾向にあった予測機構を追加した．

4.4 予備評価環境

本稿における評価は予備評価である．評価の目的は実際のプログラム上でのデータアクセスにおける提案したメカニズムの効果を示すことを目的とした．計測は設計した回路を回路シミュレータ Model Technology 社 Model Sim でシミュレートすることで行った．評価した回路の構成は WRS 構成と MOS 構成を採用しない単純な線形ページアドレス予測機構をそれぞれの TLB に追加するハードウェア構成である．TLB の性能は 4 KB ページサイズで 16 エントリとした．また，TLB に格納される情報は VPN に対応する PPN(Physical Page Number) のみとし，ページ保護ビット，ダーティビット，属性ビットのフィールドは存在しない．ページテーブルは TLB ミスハンドラの簡易化のため，PTP(Page Table Pointer) に VPN をオフセットとして加算することで，PTE のアドレスを算出できるページテーブルとした．キャッシュは，回路シミュレーションレベルでは物理アドレスでインデックスされており，全てヒットする理想的な L1 キャッシュが存在すると想定した．そのため，ページテーブルアクセス以外のメモリアクセスはメモリアクセスレイテンシが発生しない．ページテーブルへのメモリアクセスレイテンシは 12 クロックサイクルに設定した．このレイテンシ設定は対象とする主記憶を一般的なシングルエッヂ SDRAM に想定した場合，メモリアクセスのアドレスストーブを供給してから最初のデータが到着するまでに 12 サイクル要するという理由からである．しかしながら，このレイテンシはバスクロックサイクルにおける 12 クロックサイクルであり，最新の CPU はこのバスクロックサイクルの 10 倍以上の逓倍数で動作していることを注意にいれなくてはならない．本計測では CPU はバスクロックサイクルに対して 1 逓倍で動作していことが前提となる．計測用プログラムは図 7 に示す．プログラムを MIPS クロスコンパイラでコンパイル後，バイナリを VHDL で記述した仮想的な ROM にロードさせ，実行中，命令フェッチさせることでパイプラインの命令実行を実現した．

計測対象としたプログラムは test_data という配列のデータセットに最初の for ループでインクリメントされる誘導変数 i を格納していき，次のループで全ての配列の内容を加算するプログラムである．ループ回数とデータセットの大きさは PAGE_SIZE と NUM_OF_TLB_MISS で決定される．計測は NUM_OF_TLB_MISS の定数を変化させ行った．NUM_OF_TLB_MISS はデータ TLB ミスをおこす期待値で設定したが，実際の実行は実行時のスタック構造，データセットのアラインで厳密には異なってくることに注意しなくてはならない．

5. 結果と評価

5.1 性能評価

計測結果を表 1 に示す．表中の NUM_OF_TLB_MISS はプログラム中の NUM_OF_TLB_MISS 定数で設定した数，終了 CC はプログラム実行に要したクロックサイクル数，I-TLB ミスは命令 TLB ミス数，D-TLB ミスはデータ TLB ミス数，TLB 消費 CC は TLB ミスハンドリングに要したクロックサイクル数である．表は 2 つあり，上の表が予測機構無しの計測結果，下の表が予測機構がある場合の計測結果である． 今回の計測では NUM_OF_TLB_MISS は 5 と 20 に設定した 2 回の計測を行った．I-TLB ミスに関して，全ての計測に渡ってミス回数は 1 回となっている．この数字は実行したプログラムが小さく，全てのプログラムテクストが 1 ページ以内に収まっているためである．計測に使った C プログラムのマシン語命令数は 81 命令であった．この容量は 324 バイトであり，単一ページ内に収まる容量である．次にデータ TLB ミスは予測器無しの場合 NUM_OF_TLB_MISS が 5 のとき 8 回ミス，20 のとき 41 回ミスしている．NUM_OF_TLB_MISS が 5 のときの 8 回のミスは 16 エントリの TLB にに入りきるミスであるため，初回ページアクセス時のミスのみである．NUM_OF_TLB_MISS が 20 の時の 41 回のミスは，TLB16 エントリに入りきらない PTE を扱ったため，TLB がスラッシングを起こした結果である．これは TLB リーチが引き起こした問題といえる．このミス回数に対して予測機構を付きのハードウエアの

```
#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 ){
    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 += data[ i ];

    return sum;
}
```

図 7 計測用プログラム．

表 1 実行シミュレーション測定結果

予測機構無しのハードウェアでの測定結果				
NUM_OF_TLB_MISS	終了 CC	I-TLB ミス	D-TLB ミス	TLB 消費 CC
5	225446	1	8	108
20	901662	1	41	504
予測機構付きのハードウェアでの測定結果				
NUM_OF_TLB_MISS	終了 CC	I-TLB ミス	D-TLB ミス	TLB 消費 CC
5	225386	1	3	48
20	901206	1	3	48

表 2 ハードウェア量測定結果

Integer Unit のハードウエア量			
FD	FDE	LUT	XORCY
0	1502	3765	16
MMU のハードウエア量			
FD	FDE	LUT	XORCY
0	2723	2354	8
予測機構のハードウエア量			
FD	FDE	LUT	XORCY
3	160	323	3

D-TLB ミスは NUM_OF_TLB_MISS に関係なく 3 である．この結果は TLB に格納されたエントリが 3 エントリのみであるという意味を持ち，予測機構がうまく機能し，TLB に格納される情報を抑制できたといえる．実行クロックサイクル損失は NUM_OF_TLB_MISS が 5 の時約 44 %に NUM_OF_TLB_MISS が 20 の時に 9.5 %に抑えられている．この損失は全体の実行時間から換算すると 0.06 %以下の損失であるが，この値は CPU が 1 逓倍動作時の値なので実際には逓倍数倍の実行損失なる．

5.2 ハードウェア量評価

計測対象のハードウェア量を表 2 に示す．本研究で設計した回路を Integer Unit，MMU，線形予測機構の 3 つに分類し論理合成を行い，レポートされたチップ使用状況の表である．

表中の FD は D フリップフロップの数，FDE はイネーブル付き D フリップフロップの数，LUT は LUT(Look Up Table) の数（つまりランダムロジックに使った素子数），XORCY はキャリーロジックに使用される特殊な XOR 素子数を示す．予測機構を持たないハードウェア構成は Integer Unit + MMU のハードウェア量が最終的なハードウェア量である．予測機構付きハードウェア構成の場合は Integer Unit + MMU + 予測機構のハードウェア量が最終的なハードウェア量である．FD と XORCY は小さいので無視するとしても，予測機構のハードウェア量は FDE が全体の約 3.6 %，LUT が全体の約 5.3 %の大きさであることが分かる．

6. おわりに

本研究で提案したメカニズム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上げることができる．メモリ中に存在する線形アクセスされるページ数が多いほど本メカニズムは効力を発揮する．また，単純に TLB ミス回数を抑えるだけではなく，TLB スラッシングを抑える効果があり，大きなワーキングセットを扱うプログラムに対する TLB リーチ問題の改善策となる．本メカニズムは従来の MMU を少量変更することで追加することができ，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．本稿における計測結果は全て予備評価であり，線形ページアクセスされることが確定しているプログラム上での測定結果である．今後の課題として，一般的なプログラム上での線形ページアドレス予測機構の振る舞いと，非線形ページアクセス対応の WRS と MOS 構成を含めたハードウェア構成での計測を行いたい．

謝辞

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．

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