大規模データパス・アーキテクチャに適したロードストアユニット構成

入江英嗣[†] 安島雄一郎[†] 辻秀典[†]
坂井修一[†] 田中英彦[†]

我々は大規模データパス (VLDP)・アーキテクチャを提案している。VLDP アーキテクチャでは複数パス実行を用いた大規模投機実行により実効 IPC8 を目指している。VLDP メモリアクセス機構はプログラム中のロード/ストア命令によるメモリアクセスを仮想化し、実行部の負荷を軽減する。また、曖昧な依存関係を解消したメモリアクセスを大規模に並列実行し、必要とされるスループットを得る。本稿ではこのようなメモリアクセス機構実現のためのハードウェアとして、分散したアクセスコントローラや、高速に依存関係をチェックする機構等の構想を示し、VLDP に適したロードストアユニットを提案する。

Load Store Unit Structure for Very Large Data Path Architecture

Hidetsugu Irie,[†] Yuichiro Ajima,[†] Hidenori Tsuji,[†]
Shuichi Sakai[†] and Hidehiko Tanaka[†]

We propose the Very Large Data Path Architecture which achieves actual IPC 8 by using Multi-Path execution and Massive Speculative Execution. VLDP Memory Access section deals with aliasing between store and load instruction and applies Multi-Path execution. It also controlls enormous memory access in parallel and sustains data throughputs. This paper describes VLDP-suited load store unit which consists of distributed access controll units.

1. はじめに

現在、半導体デバイス技術の進歩はとどまることを知らず、マイクロプロセッサの設計には大量のトランジスタを投入することが可能となっている。

我々が提案している大規模データパス (Very Large Data Path, 以降 VLDP) アーキテクチャでは豊富なトランジスタ資源を仮定した新しいアーキテクチャによって、実効 IPC8 を目指している。

このように大量のハードウェアを活かして大規模に命令を実行するアーキテクチャにおいて、ロード/ストア命令をどう処理するかは重要な課題となっている。メモリアクセス命令は全実行命令中の 1/3 を占め、従来から指摘されているアクセスレイテンシの問題のほか、メモリアクセス命令の大規模実行を目指す場合、曖昧な依存関係の問題がスループットの向上を妨げている。アドレス計算はパイプライン後段で行われるため、動的かつ高速な解析機構が必要とされるためである。

フェッチ、実行機構の大幅な性能向上に対し、メモリアクセスの大規模化が求められるが、従来の解析方式では、高速化が困難となることが予想される。

† 東京大学 大学院工学系研究科
Graduate school of Engineering, The University of Tokyo

本稿では VLDP アーキテクチャについて、仮想化したメモリアクセスを実現するメモリ部の概要について述べ、ロードストアユニットの構成を提案する。

2. 関連研究

プロセッサの動作速度と主記憶として用いられる DRAM の動作速度との差は年々広がる一方であり、メモリウォール問題として従来から指摘されている。近年では、DRAM のインターフェイスを改良することにより、大幅にスループットを高めた SDRAM、RDRAM 等が主流になっているが、アクセスレイテンシに関してはそれほど大きな改善はない。

このギャップを緩衝するために用いられている方式がキャッシュを用いた階層記憶構造である。高速に動作する SRAM 上にメモリ空間のサブセットを保持し、キャッシュがヒットを続けている間は低いレイテンシでデータを供給できる。キャッシュの性能はプロセッサの性能に大きな影響を与えるため、ミス率を軽減したり、ミスペナルティを隠蔽したりするための研究が数多く行われている。

一方、進歩を続けるプロセッサアーキテクチャは、キャッシュシステムにスループットの面からも改善を迫っている。全実行命令の 3 割がメモリアクセス命令であるため、IPC3 を越えるプロセッサを実現するためには、キャッシュは毎サイクル複数のデータを供給

し続けなければならない。

キャッシュ設計には、ミス率、動作速度、ポート数の間でトレードオフがあり、単純な大容量化で性能を上げることは難しい。ハードウェア技術の進歩を想定したキャッシュについて、大容量化やウェイ数の強化によって見込まれるミス率の変化の研究や、複数パス実行に対応したキャッシュの研究が行われている。また、様様な技術によるキャッシュのマルチポート化がIPC に与える影響が研究されている[4)]。

DEC Alpha AXP21264 では L1 キャッシュは毎サイクル 4 データを供給することが可能となっている反面、ヒットレイテンシは 4 サイクルとなっている。

以上の研究などから、VLDP ではヒットレイテンシに数サイクルを見込めば、メモリアクセススループット要求を満たすキャッシュを利用することが可能であると予測している。

キャッシュのバンド幅を活かす、メモリアクセスの並列実行について、メモリアクセスの曖昧な依存関係が、性能に大きな影響を与えている。メモリアクセス命令の実行に際しては、アドレス計算、依存関係の解消、キャッシュアクセスの 3 段階の処理が必要となり、これらの処理全ての効率を高めなくてはならない。動的に依存関係を解析し、メモリアクセスを out of order に実行するための機構として、ロードストアキュー (Load Store Queue, 以降 LSQ) 等が利用されている。

実際にはアクセスにかかる時間とほぼ同じ時間が、依存関係解消待ちのストールに費やされている。このため、様々な動的な依存関係予測手法や、値予測機構が研究されている。

3. VLDP アーキテクチャとメモリシステム

3.1 VLDP アーキテクチャ

VLDP アーキテクチャはスーパスカラ、VLIW など従来のアーキテクチャの延長ではない新しいアーキテクチャである[1)]。大規模にハードウェア資源を利用することにより大幅な性能向上を目指し、実効 IPC8 達成を目標としている。

VLDP ではサイクルあたりの高い処理スループットを得るために、32 命令幅の命令ブロック (Instruction Block, 以降 IB) を処理単位として、各種処理を高速化する。IB 単位の処理を行う実行ユニット (Exectution Unit, 以降 EU) は複数実装され、複数の IB を並列に実行することで、大規模な命令実行を行う。また、複数パス実行を行い、分岐予測ミスによるペナルティを隠蔽する。

VDLP のブロック図を図 1 に示す。VLDP は大きく 3 つの部分に分けることができる。制御部では分岐予測によるフェッチ制御[3)] を行い、実行部では複数の EU によって、IB を実行する[2)]。メモリ部では実行中のロード/ストア命令によるメモリアクセスを制御する。

3.2 VLDP メモリ部の仕様

VLDP メモリ部では実行部に対し、仮想化したメモリアクセスを提供する。メモリアクセスに関する制御を EU から切り離すことで、EU の負担を軽減し、メモリに関する操作を独立させる。

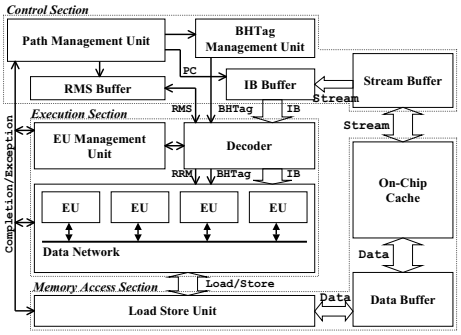

図 1 VLDP ブロック図

EU はアドレス計算を行い、メモリ部に送信するだけで、ロード/ストア命令に必要な処理を済ませることができる。この送信はプログラムオーダーの制約なく各 EU が独立して行うことができる。

ロード命令の場合は、アドレスを送信すると、不定サイクル後に、プログラムオーダを反映した、正しいロード値が返る。ストア命令の場合には、アドレスと値を送信した時点で実行部にとっての処理は完了したことになる。また、投機実行の破棄時には、破棄信号を送ると、キャッシュアクセスの状況に関わらず、短い規定サイクル後には、新しい IB のメモリリクエストを送信することが可能となる。

このような実行部-メモリ部インターフェイスを実現し、かつ VLDP プロセッサの実行性能に見合うデータスループットを得ることがメモリ部の課題である。

VLDP メモリ部は大きくロードストアユニットと階層メモリから構成される。

複製、インターリーブ等の技術で、キャッシュのスループットを大きく設計すると同時に、ロードストアユニットによる大規模な動的スケジューリングにより、このバンド幅を有効に利用する事によって、メモリアクセス処理のスループットを確保する。

3.3 ロードストアユニットの役割

VLDP ロードストアユニットはプログラム中のロード、ストア命令について階層メモリアクセスの制御を行う。マルチパス、out of order 実行の EU と、シングルパス、in order ステートの階層メモリの間で、このギャップを吸収する働きをする。ロードストアユニットは制御回路と、投機的なストア値等を保持するバッファとで構成される。ロードストアユニットは各ロード/ストア命令に対し、以下のような処理をする。

- 曖昧な依存関係の解決、発火の動的スケジューリング
- ロードした値の EU への出力
- 依存関係のあるストア-ロード間でのフォワーディング
- 分岐予測ミス、複数パス実行に伴う、進行中のロー

ド/ストア命令の破棄処理

特に依存関係の解決、スケジューリングは、複数パス実行を行い、各 EU が独立してアドレス計算を行う VLDP においては複雑な解析を伴うため、高速化が課題である。

このようなロードストアユニットをロードストアキューやストアバッファなどの単純な拡張、大型化などによって論理的に実現することは可能である。しかし、大きなバッファはそれだけ動作速度が遅くなり、アクセス負荷の集中にも耐えられなくなる。VLDP ロードストアユニットのハードウェア構成の設計には、この点を考慮しなくてはならない。

4. VLDP ロードストアユニットの構想

4.1 ロードストアユニットの構成

VLDP ロードストアユニットの構成を図 2 に示す。

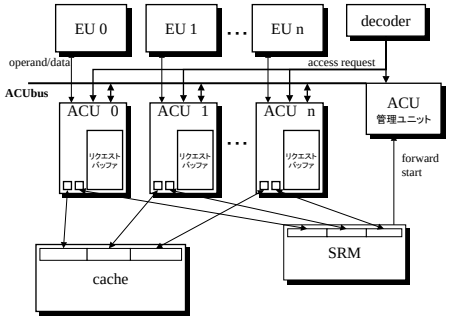

図 2 VLDP ロードストアユニットの構成

各 EU に対応してロード/ストア命令を処理するアクセス制御ユニット (Access Control Unit, 以降 ACU) が同時並列に動作することにより、大規模にメモリアクセス命令を制御、実行する。各 ACU は割り当てられた IB 内のロード/ストア命令についてアクセスの発火制御を行う。ACU はリクエストバッファと呼ばれるバッファを持ち、IB 内メモリリクエスト情報を保持する (図 3)。IB リクエスト情報としては、アドレスやストア値が含まれる。

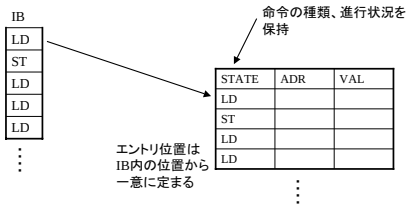

図 3 リクエストバッファ

ACU 管理ユニットは各 ACU の状態を管理し、IB 間にまたがった処理を可能とする。各 ACU および ACU 管理ユニットは、ACU バスと呼ばれるバスで接続されており、制御情報や、フォワーディングデータのやりとりを行う。

キャッシュおよび SRM と呼ばれるユニットは各 ACU で共有される。

以降の節で、VLDP ロードストアユニットの構想と各ユニットの役割について述べる。

4.2 単一キュー方式の問題

ロード/ストア命令を積極的に out of order 実行する場合、一般に次の制約を満たしながら動的スケジューリングすることとなる。

- ロード命令のアクセス開始は、先行ストア命令のアドレスが全て判明するまで待機する
- ストア命令のキャッシュへの書き戻しは、先行ロード命令が全てアクセスを終えるまで待機する
- 同じアドレスへのストア命令は in order に書き戻しを行う

さらに積極的な out of order 実行を目指す場合、新しいメモリモデルや投機実行を導入する必要がある。

この制約を満たすための解析には現行ではロードストアキュー (以降 LSQ) が用いられている。LSQ はフルアソシアティブのバッファに in order 順にエントリを設け、ロード/ストア命令のオペランド保持を行う機構で、以下のような利点を持つ。

- キャッシュへの書き込みのレイテンシを隠蔽できる
- 分岐の投機実行によるキャッシュ書き込みを、分岐確定まで保留できる
- 各ストア命令が別々のバッファエントリを利用することで、メモリアクセス命令同士の WAW、WAR の問題を回避できる

LSQ 内で RAW 依存が生じた場合は、値をキャッシュを介さずフォワーディングする機構が実装されている。また、一般に書き戻しはストア命令のリタイア時に行われる。

一方で、各ロード命令のアクセス実行時に、全てのエントリを参照して、RAW 関係の有無を調べる負荷が発生する。また、RAW 依存が存在した場合、バッファ内容のキャッシュへの書き戻し又はフォワーディングのオーバーヘッドが発生する。

一般に、現行の LSQ はエントリ数が少ないため、全検索はオーバーヘッドとは考えられていない。RAW フォワーディングは、むしろキャッシュアクセスよりレイテンシの少ない、”RAW ヒット”として捉えられている場合もある。

しかし、VLDP においては、LSQ 方式は実装の面から難点を抱えている。VLDP では大量フェッチによる利点を活かして、広い範囲でロード/ストア命令の並列性を抽出し、大規模にアクセスを発火することで、実行機構へのデータ供給を充足させなければならない。

LSQ はフルアソシアティブであるため、動作速度、ポート数、エントリ数全てを向上させることは困難である。また、キュー状であるため、複数パス実行に

よるパス管理を別に行わなければならない。そこで、VLDP ロードストアユニットでは、LSQ とは違った機構で、理想的 LSQ と同等の性能を得るよう、構成を設計している。

4.3 ACU の分散

バッファを分散させることは、動作速度やアクセス負荷の分散の面から効率的である。一方で、分散されたバッファ間の通信が逆にオーバーヘッドとならないよう注意する必要がある。

VLDP は IB 単位でデコード、EU への割付、実行を行う。同様に、メモリアクセスの管理も IB 単位で行う構成をとる。IB 内のリクエスト情報を保持するリクエストバッファとアクセス発火コントローラによって構成される ACU(アクセスコントロールユニット) を、EU と同じ数実装し、対応する EU に割り当てられた IB 内のロード/ストア命令のアクセス処理を行う (図 4)。

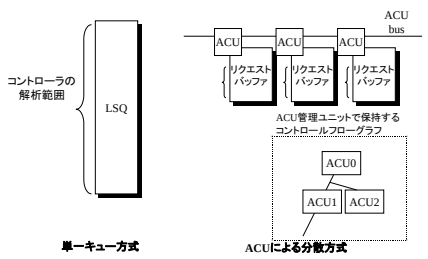

図 4 LSQ 方式と分散方式の比較

一つの IB についての処理が終了すると、リクエストバッファの保持内容は破棄され、次の IB の処理を開始する。

ACU 内コントローラは、外部からの制御情報と、リクエストバッファ内を参照し、アクセス発火/待機、フォワーディング処理などのコントロールを行う。各 ACU 内のリクエストバッファはフルアソシアティブであるが、1 つの IB 内のリクエストのみを保持するので、集中した LSQ 方式に比べて、エントリ数を数分の 1 程度で実現できる。

4.4 ACU の管理

各リクエストバッファのエントリはプログラム内の順序に従って、in order に確保されるが、IB の割り当ては、アイドル状態の EU へ任意に行われる。そのため、ACU に分散したロード/ストア命令は互いの順序関係の情報が失われた状態にある。

各 ACU の順序関係を把握し、ACU 間にまたがった依存関係を解決するために、ACU を集中管理するテーブルを設ける。このテーブルは ACU の数だけエントリを持ち、ACU に割り当てられた IB の BHTag(IB を識別するタグ) や、ストア値保持が終了したか等の処理進行状況を保持する。ACU とコントロールフローグラフの関連づけを ACU 管理ユニットで行うことにより、複数パスに対応したツリー状の LSQ を論理的に実現する事ができる。

このテーブルの内容を参照して、毎サイクル各 ACU へ制御情報が送信される。例えば”ロード開始可”信号は、親にあたる全ての ACU で全てのストア値が保持されている ACU に対してのみ送信される。この信号を受けた ACU のみが、アドレスが判明したロード命令のアクセスを開始することができる。このようにすることによって、解析範囲全体を検索することなく、一度に多数のロード命令について、先行ストア終了チェックを行うことができる。

複数パス実行によるメモリ命令同士の干渉は、この ACU 管理ユニットのレベルで解決する。IB の構成上、異なるパスの命令が、同じ ACU に同時に割り当てられる事はない。

4.5 ロード許可と書き戻し許可

ロード命令とストア命令は、各 ACU において基本的に out of order に処理されるが、処理順序について、以下の制限を用い、正しいロード/ストア命令実行を保証する。

- ロード命令のキャッシュアクセスは、パス上流のストア命令が全て値をリクエストバッファへ保持するまで待機する
- リクエストバッファ内容のキャッシュへの書き戻しは、パス上流のロード命令が全てキャッシュアクセスを終了するまで待機する
- 同じアドレスへのストア命令の書き戻しは in order に行う

この制御は ACU 管理ユニットから各 ACU へ送信される制御信号によって行われる。ACU 管理ユニットには、各 ACU におけるロード/ストア命令の進行状況が毎サイクル送信される。この進行情報と各 ACU の BHTag 情報から、制御信号が決定される。

4.6 ストア予約マップ

ロード可制御信号を ACU 管理ユニットから受信した ACU はオペランドの揃っているロード命令からアクセスを発火する。

このとき、キャッシュへのアクセス前に、パス上流かつ同じアドレスへの未書き戻しストアが存在するかどうかをチェックしなければならない。RAW 依存のストア命令が存在した場合は、フォワーディング処理を行うことになる。

このチェックには未書き戻しストア命令の全検索を必要とし、ロード命令のレイテンシを増加させる。また、全てのロード命令がこのチェックを行うため、リクエストバッファの参照負荷集中を生じさせる。

プログラムを正確に実行するためには、このチェックを全てのロード命令について行わなければならない反面、実際に実行中のストア-ロード間で RAW 依存を生じるケースはそれほど頻度が高くない。

そこで、アドレスからの連想検索でキャッシュのようにアクセスできるテーブル、SRM(ストア予約マップ) を設け、処理中のストアの数をアドレス毎に保持する。ストア命令は、値保持時に、対象アドレスの値

をインクリメントし、キャッシュへ書き戻し後、デクリメントする。

各ロード命令は、リクエストバッファを全検索する代わりに、ストア予約マップの対象アドレスの値を参照する。殆どのケースにおいて、この値は 0 であり、RAW 依存がないことを高速にチェックできる。SRM の値が 1 以上であった場合は後述するフォワーディング処理を行う。

4.7 フォワーディング処理

ロード命令の SRM 参照時、値が 1 以上であった場合、キャッシュへの未書き戻しストアに RAW 依存がある可能性がある。この場合、ACU 間でリクエストバッファ内の値を直接転送する。フォワーディング処理を実現するために、各 ACU はフォワーディング用のバスで接続されている。フォワーディングは、次に示す 3 段階の処理で実行される。処理には数サイクル必要とするが、パイプライン化が可能である。

(1) 当該アドレスへの未書き戻しストア命令を持つ ACU の検索
(2) 複数の候補の中から、適切なフォワード元 ACU の特定
(3) ACU 間でデータの送受信

SRM 値が 1 以上であった場合、まず SRM から、当該アドレスが、ACU バスを通じて、全 ACU へ送信される。また、アドレスと、そのロード命令を含む ACU の ID が ACU 管理ユニットへ送信される。各 ACU は、そのアドレスに対するストア命令の有無を ACU 管理ユニットへ通知する。

ACU 管理ユニットでは、各 ACU からのストア有無信号を受け、管理ユニット内の BHTag 情報を元に、フォワーディング元のストア命令を含む ACU を特定する。この特定結果はバスを通じて、全 ACU へ送信される。SRM 値が 1 以上であっても、SRM 値をインクリメントしたストアはロード命令より下流のものであったり、別のパスのものであったりして、RAW 依存関係が存在しない場合が存在する。そのような場合は、フォワーディング処理はキャンセルされる。

フォワードされるデータを含む ACU は、ACU 管理ユニットにおける特定結果を受けて、バスにストア値を流す。ロード命令を含む ACU がこの値を受信して、フォワーディング処理は完了する。

プロセッサアーキテクチャに大きく依存するが、一般にロード命令の 30%程度が実行中のストア命令に RAW 依存を持つと言われている。フォワーディング処理の発生する頻度について概算すると、VLDP では毎サイクル平均 1 命令となる。

4.8 実行の破棄

複数パス実行を行う VLDP では、実行中の命令が頻繁に破棄される。破棄処理にオーバーヘッドが存在すると、複数パス実行によって得る性能向上を相殺してしまうおそれがある。そのため、実行の破棄は高速に行われなければならない。

破棄処理は基本的に、ACU の保持する内容を無効化するだけで完了するが、SRM をインクリメントしているストア命令が存在する場合、SRM の更新を行わなければならない。また、フォワーディング、キャッシュアクセスには、競合、ミスなどから不定サイクルの実行時間がかかり、これらの実行中に破棄処理が生じた場合でも、高速に破棄が行える機構が必要である。

各 ACU は SRM デクリメントの為のキューを持ち、実行破棄時には、SRM デクリメントが必要なアドレスの値をリクエストバッファからコピーする。ACU では新しい IB の処理と、破棄された IB の SRM デクリメントが並行して行われる。

アクセス実行中の IB 破棄処理を可能にするための機構として、各 ACU はポートセマフォを持つ。ポートセマフォは BHTag を保持するキューで、キャッシュ等へアクセスを行う時に書き込まれ、キャッシュ等からデータが返ってきた時に読み出される。ポートセマフォから読み出された BHTag が処理中の IB の BHTag と異なる場合、データは破棄される。

ポートセマフォは各 ACU に、SRM 読み込み用、キャッシュ読み込み用、フォワーディング処理用、下位メモリ読み込み用の 4 つが実装され、それぞれ独立して動作する。

5. メモリアクセス処理の流れ

この節では、ロードストアユニットでどのような処理が行われるかをロード命令、ストア命令について述べる。

5.1 ストア命令の処理

ストア命令の処理の流れを図 5 に示す。

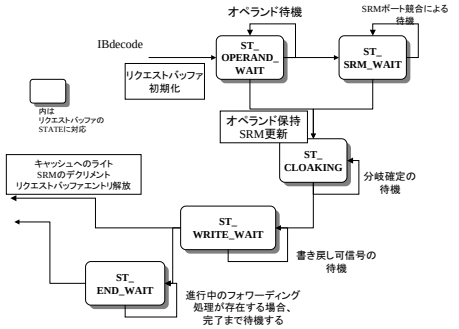

図 5 ストア命令処理の流れ

IB デコード時に、その IB 内の全てのメモリリクエストについて、命令の種類 (ロード/ストア) 及び、リクエストの IB 内位置情報、そして IB の割り当てられる EU の ID がロードストアユニットに送信される。ロードストアユニットでは、この情報を受けて、IB の割り当てられた EU に対応する ACU をアクティブにし、リクエストバッファを初期化する。各ロード/ストア命令は、IB 内位置に従って、リクエストバッファ内にエントリを割り当てられ、オペランド待機状態となる。

VLDP では毎サイクル 1IB がデコードされるため、この初期化処理も 1IB/cycle のスループットを持つよ

うハードウェアを設計する。

リクエストバッファの初期化を終えたストア命令はEU からのストア値とアドレスの送信を待機する。オペランドを受信した時点で、実行部にとってのストア処理は終了する。

オペランドの揃ったストア命令は、SRM の当該アドレスの値をインクリメントし、キャッシュへ書き戻す条件が整うまで値をリクエストバッファへ保持する。この間、ACU バスからフォワーディング要求があった場合は値の供給を行う。

キャッシュへの書き戻しを開始する条件は、まず、そのストア命令の実行が確定していることと、上流のロード命令がキャッシュアクセスを終了し、ストア命令が属している ACU へ”書き戻し可”制御信号が送信されていることである。この条件が揃ったストア命令は、SRM へアクセスを行う。

SRM 値が 1 であった場合、キャッシュへの書き戻しを行い、終了後、SRM の値をデクリメントする。一方 SRM 値が 2 以上であった場合、当該アドレスへのストア命令が複数あることを示している。この場合、同じアドレスについて、ストア命令は in order に書き戻さなくてはならない。ACU 管理ユニットは、毎サイクル、その時点でコントロールフローグラフの根本にあたる ACU に対して、root 制御信号を送信する。SRM 値が 2 以上のアドレスへのストア書き戻しは、この root 信号を受けた ACU に属するストア命令のみが行うことができる。この場合も、キャッシュへの書き戻し終了後、SRM 値のデクリメントを行う。

書き戻し処理を終了したストア命令は、リクエストバッファのエントリを削除し、処理を終える。

5.2 ロード命令

ロード命令の処理の流れを図 6 に示す。

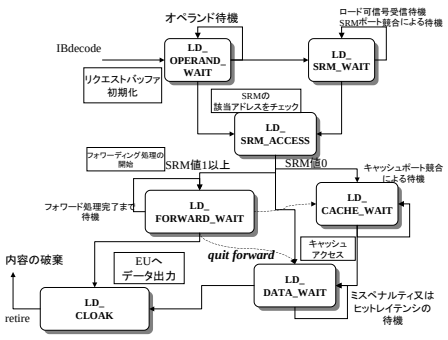

図 6 ロード命令処理の流れ

ロード命令についてもストア命令と同様、初期化処理が IB デコード時に行われる。

EU で計算されたアドレスを受信時に、ACU へ”ロード可”制御信号が送信されていれば、SRM へアクセスを行う。そうでない場合は、”ロード”可信号を受信するまで待機状態となる。

SRM 値の値により、処理が異なる。SRM 値が 0 であった場合、RAW 依存の可能性はなく、キャッシュアクセスを行い、値を得る。

一方、SRM 値が 1 以上の値であった場合、RAW 依存が存在する可能性があり、ACU バスを用いたフォワーディング処理を開始する。フォワーディング処理は、ACU と ACU 管理ユニットによって行われ、結果が返るまで、ロード命令は待機状態となる。また、ACU 管理ユニットでの解析の結果、RAW 依存が存在しないと判断された場合は、改めてキャッシュアクセスを開始する。

キャッシュからのロードまたはフォワーディング処理により値を得たロード命令は、EU へ値を出力し、処理を終了する。

6. まとめと今後の課題

本稿では VLDP アーキテクチャのメモリ部について概要を示し、中核となるロードストアユニットの構成を提案した。

現在 VLDP プロジェクトではクロックレベルシミュレータを作成中であり、これを用いて、各部パラメータの調整を行う。

また、本稿で提案したモデルはメモリアクセス命令の大規模実行によって、高いスループットの達成を目指しているが、プログラム中の RAW 依存が性能を制限してしまう事が考えられる。

今後、依存関係予測や値予測の導入についても検討を行っていく。

謝辞 本研究の一部は、文部省科学研究費補助金 (基盤研究 (B) 課題番号 11480066) および、(株) 半導体理工学研究センターとの共同研究によるものである。

参 考 文 献

1) 辻 秀典, 安島 雄一郎, 坂井 修一, 田中 秀彦. ”大規模データパス・アーキテクチャの提案” 情報処理学会研究報告 2000-ARC-139, pp.49-54(2000)
2) 安島 雄一郎, 辻 秀典, 坂井 修一, 田中 秀彦. ”大規模データパス・アーキテクチャの提案”. 情報処理学会研究報告 2000-ARC-139, pp.55-60(2000)
3) 塚本 泰道, 安島 雄一郎, 辻 秀典, 坂井 修一, 田中 秀彦. ”大規模データパス・アーキテクチャの提案”. 情報処理学会研究報告 2000-ARC-139, pp.61-66(2000)
4) J.A.Rivers, G.S.Tyson, T.M.Austin, and E.S.Davidson. ”On High-Bandwidth Data Cache Design for Multi-Issue Processors”. IEEE MICRO-30, pp.46-56(1997)