計算機アーキテクチャ　140－10
（２０００．　１１．　３０）

メモリレイテンシ隠蔽アーキテクチャSCALT

宮坂 和幸*　　清水 尚彦**　　孕石裕昭***
*東海大学大学院工学研究科　　** ***東海大学工学部

概要

我々はソフトウェアコンテキストとしてのバッファを持つ SCALT アーキテクチャプロセッサを提案する。メモリレイテンシ隠蔽のため提案するバッファへ到着するデータをチェックする命令がある。このレポートは通常のパイプラインプロセッサをベースとした SCALT プロセッサの設計について述べる。

SCALT ARCHTECTURE TO CONCEAL MEMORY LATENCY

Kazuyuki MIYASAKA*, Naohiko SHIMIZU**, Hiroaki HARAMIISHI***
*Graduate School of Eng.,Tokai Univ.,** ***Faculty of Eng.,Tokai Univ.

Abstract

We have been proposed a processor architecture SCALT which has a buffer as a software context. For the deviation of the memory latency problem, an instruction which checks existence of the data arrival to the buffer has been proposed. This report describes SCALT processor design whiches based on a conventional pipeline processor design.

1　はじめに

現在のプロセッサ性能をより引き出す方法の 1 つとしてメモリレイテンシ (メモリシステムに対して要求を出してから結果が戻ってくるまでの時間) の隠蔽が考えられる。その方法としてメモリ- プロセッサ間にメモリのコピーを保持する高速なキャッシュメモリを置く事は一般的であるが、キャッシュメモリに入りきらない大きなデータを扱う科学技術計算分野などでは有効ではないことが知られている。そこで我々は、データの格納先をキャッシュメモリではなくキャッシュメモリと同階層に位置するバッファ(SCALT バッファ) を設け、ソフトウェアによって SCALT バッファのデータをプリフェッチすることでメモリレイテンシを隠蔽することを考案する。

SCALT バッファはアーキテクチャに対して独立であり、容量を増やすことによりプロセッサの実装が困難になる事はない。そのためプロセッサ性能が増大し、相対レイテンシが増大しても単にバッファを増やすことで容易に対応が可能である。また、従来の CPU アーキテクチャに数命令を追加するだけで実現できるため、ソフトウェアコンパチビリティを保つことができる。

現在設計している CPU は FPGA をターゲットとしている。そこで NTT が開発した HDL(Hardware Discription Language) である SFL(参考文献 [1]) によって設計を進めている。本報告では SCALT アーキテクチャの動作について説明する。

2　SCALTアーキテクチャ

SCALT アーキテクチャの特徴は SCALT バッファにある。SCALT バッファはバッファエントリ数、バッファエントリサイズを増やすことによってアーキテクチャに依存せずにメモリスループットを容易に上げることができる。

SCALT バッファはキャッシュメモリと同一のメモリ素子から成り、同一階層に位置する。また、キャッ

シュメモリのようなコンシステンシは要求されない。そのため、メインメモリ-SCALTバッファ間でデータ転送するための専用命令を持つ。この専用命令はBF(Buffer Fetch)、BW(Buffer Write)命令(図2参照)であり、128bit(1ライン)のデータを4回に分けてバースト転送する。BF命令はメインメモリからの戻りデータの格納先を区別するためにタグ情報を読みだし要求と共にメインメモリへ発行する。メインメモリ側では指定されたアドレスのデータと共にタグ情報をCPUにバースト転送する。メインメモリ-SCALTバッファ間はBF,BW命令によって転送を行うが、CPU-SCALTバッファ間は通常のLW(Load Word)、SW(Store Word)命令によって32bitのデータ転送を行う。また、SCALTバッファに関してBC(Buffer Check)命令がある(図2参照)。SCALTバッファはエントリ数分のタグテーブルを持つ。これはSCALTバッファが有効であるときは1がセットされ、無効なときは0がセットされるような1bitの情報を保持するレジスタ群である。BC命令はこのエントリ番号で指定されたタグテーブルの1bit情報を指定汎用レジスタに転送することでSCALTバッファに有効なデータが到着しているか確認することができる。この命令により、プログラムレベルでレイテンシの変動に対応することができる。

また、SCALTバッファはクリティカルパスを考慮して、通常のTLBと別に専用のTLBを持ち、仮想アドレスと比較することによってSCALTバッファにデータが存在するかチェックする。SCALTバッファへのアクセスはIN/OUT命令によって行われる。

3 SCALT化による性能向上

参考文献[2]では、C言語を用いたSCALTアーキテクチャの性能評価を行っている。このシミュレーション結果から一般的なRISCプロセッサと比較した場合、SCALTアーキテクチャの特徴であるBC命令を使用することで性能を約20%向上できる。また、図3よりSCALTバッファエントリ数の増加に対する性能向上も見て取れる。現在我々は、SCALTバッファエントリ数を128エントリとして設計を進めている。

4 SCALTアーキテクチャの動作の詳細

SCALTアーキテクチャを実現するに当たって、SFLを使用した。以下、SFLにる設計の詳細を説明する。SCALTアーキテクチャはソフトウェアコンパチビリティを保つため従来のCPUアーキテクチャに数命令を加えて実現している。SCALT化を実現するためのCPUは32bitCPUである。以下にCPUの簡単な概要を示す。

- 汎用レジスタ数, 専用レジスタ数 … 32個,17個
- 命令形式 … R,I,J形式(図4参照)
- 命令数 … 43種類
- パイプライン … IF,ID,EXEC,MEM,WBステージを持つの5段パイプライン(+キャッシュパイプライン)

以下の説明は図6,7を参照しながら説明する。BF,BW命令はMEMステージに於いて仮想アドレスからSCALTバッファTLBを見てSCALTバッファにデータが存在するか確認する。それと同時にSCALTバッファタグとデータ読み出しを行う。ここまではBF,BW命令共に共通動作である。

BF命令はSCALTバッファにデータが存在し、タグがセットされているとき2つのBF命令専用バッファの空いている方に有効ビットを立てアドレス情報を格納すると共に次のステージ(cachereq)へ転送される。このときパイプラインレジスタにはデータの格納先であるSCALTバッファエントリ番号を格納する。SCALTバッファにヒットしない場合は命令がリトライされる。

BW命令に関しても同様であるが、この命令の場合にはメインメモリに転送される128bitのデータもBW命令専用バッファに格納する。

cachereqステージは何の処理もせず、このステージと並行に位置するcachectlステージで起動が確認されるとcachectlステージが起動する。cachectlステージは前のステージから送られて来たパイプラインレジスタの情報をキューに格納する。キューに格納するのは次の理由からである。次のmembusステージではメインメモリへの書き込み、読み出しが行われている。メインメモリには幾つかのキュー

が用意されているためキューが一杯にならない限り membus ステージでは要求を出し続けることができるが、メモリからのビジー信号によって書き込み、読み出し要求を出すことができないとき membus ステージで要求を出している命令はメインメモリのキューが空くまで membus ステージに留まる。その時、後続命令はクロックが進むたびに cachectl ステージに入って来くるためパイプラインレジスタの情報を保持する手段が必要になる。そのためメインメモリからビジー信号が出されてから空きのキューができるまでパイプラインレジスタからの情報を格納するためのキューが必要になる。もし cacheclt ステージのキューが一杯になった時は、命令をリトライする。メインメモリとのインターフェースについては後述する。

また、cachectl ステージではメインメモリのアドレスを各命令専用のバッファから読みだし、レジスタに格納している。このレジスタは次の membus ステージで参照される。

membus ステージではメインメモリに対するアクセスを行っているわけであるが、BF 命令はメモリキューが空いているならば SCALT バッファのエントリ番号と共にメモリリード要求を出す。また、BW 命令もメモリキューが空いていた場合、mem ステージでバッファに格納した 128bit のデータを 32bit づつメインメモリへバースト転送する。BW 命令は最後のデータをメインメモリへ転送した段階で BW 命令専用バッファをクリアし終了する。

BF 命令はメインメモリからのデータ転送信号が送られて来ると同時に redata ステージを起動する。そして rd1～rd4 ステージに於いてバースト転送されて来るデータを 32bit づつレジスタに格納する。この時メインメモリからは SCALT バッファのエントリ番号も送られて来るため rd1 から rd4 ステージまでのパイプラインレジスタにこの情報も格納していく。rd4 ステージでは SCALT バッファのエントリ番号と共に次のステージ (cachew) へ移る。

cachew ステージではレジスタに格納された 32bit データを連結した 128bit のデータと SCALT バッファエントリ番号から SCALT バッファへの書き込みを行っている。そしてそれと同時に使用した BF 命令専用バッファの有効ビットをクリアしている。ソフトウェアによる SCALT バッファのデータプリフェッチを行う上で欠かせない命令が BC(Buffer Check) 命令である。BC 命令は mem ステージに於いて指定された SCALT バッファタグのエントリ番号より 1 ビットの情報を読み出し、その情報と格納先の汎用レジスタ番号と共に WB ステージを起動する。WB ステージでは送られて来た情報をもとに 32bit の汎用レジスタへ 1bit の情報を符号拡張して格納して終了する。

5 メインメモリとのインターフェース

以下にメモリコントローラのブロック図を示す。現段階ではメインメモリ (SDRAM) の容量は 64MB としている。

CPU からのメモリに対するリクエストを受け付けるユニットが Device core side に相当し、PCI からのリクエストを受け付けるユニットが PCI Bus side に相当する。各デバイスからリクエストがあった場合は転送されて来た情報を SDRAM Access Queue (SAQ) に格納する。同時に Locked Address Register(LAR) に受けたアドレスを転送する。LAR は各デバイス毎のアドレスロック情報を格納している。CPU,PCI ユニットからリクエストがあった場合、このレジスタ情報と比較を行いその結果を SAQ へ送る。既にロックされているアドレスの場合は CPU,PCI ユニットと SAQ にロックできなかったことを知らせる。CPU,PCI の各ユニットはこの情報を受け取ると各デバイスに通知する。

BF 命令がリクエストされた時は、SAQ と LAR にアドレス転送して内部インクリメンタアドレスを加算する。BW 命令の場合は、リクエストがあった次のクロックで SAQ にアドレスとデータを転送してアドレスをインクリメントする。BF,BW 命令は共にバースト転送される。

SAQ では受け付けられたリクエストを順次 SDRAM へ発行する。それと同時に Return Data Queue(RDQ) にタグ情報 (SCALT バッファエントリ番号) を転送する。また同一の row アドレスのリクエストがある場合は、リクエストの順序を変更して最適化を行っている。SAQ が一杯になった時は PCI、CPU に対して制御信号を送りリクエスト

が受け付けられないことを通知する。

RDQ ではSDRAMからのデータとCPUに送る際に用いられるタグ情報の同期を取っている。両方のデータが揃った時点でデータ転送を知らせる制御信号とタグ情報、データを各デバイスに送る。タグ情報が送られるのはBF命令のときだけである。

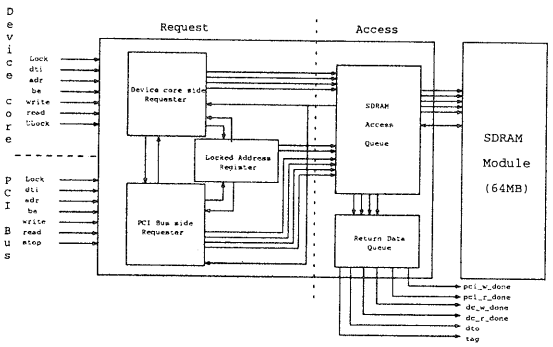

図 1: Strage Controller

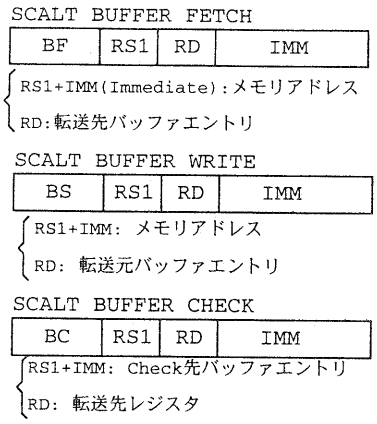

図 2: SCALT 専用命令

6 今後の予定

今後は細かいシミュレーションを行った後、SCALT 化した CPU を PCI ボードに実装し動作確認をする予定である。そのためには CPU の動作周波数や容量に制限があるため、HDL 記述の最適化をしなければならない。

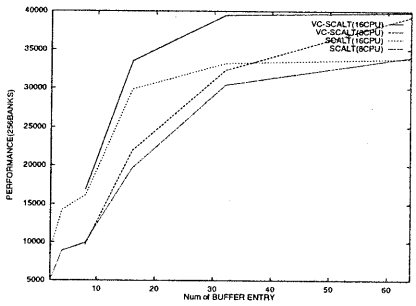

図 3: Performance to change of the number of Buffer entry

参考文献

[1] http://www.kecl.ntt.co.jp/parthenon/

[2] 三竹大輔, 清水尚彦 " 変動するメモリレイテンシに対応するプロセッサ" 東海大学紀要 (工学部),Vol.39,No.1,1999

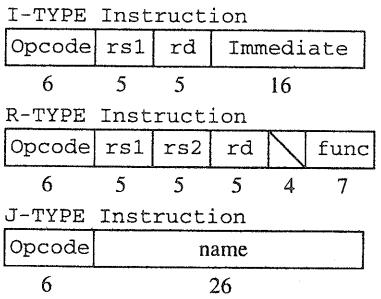

図 4: CPU の命令形式

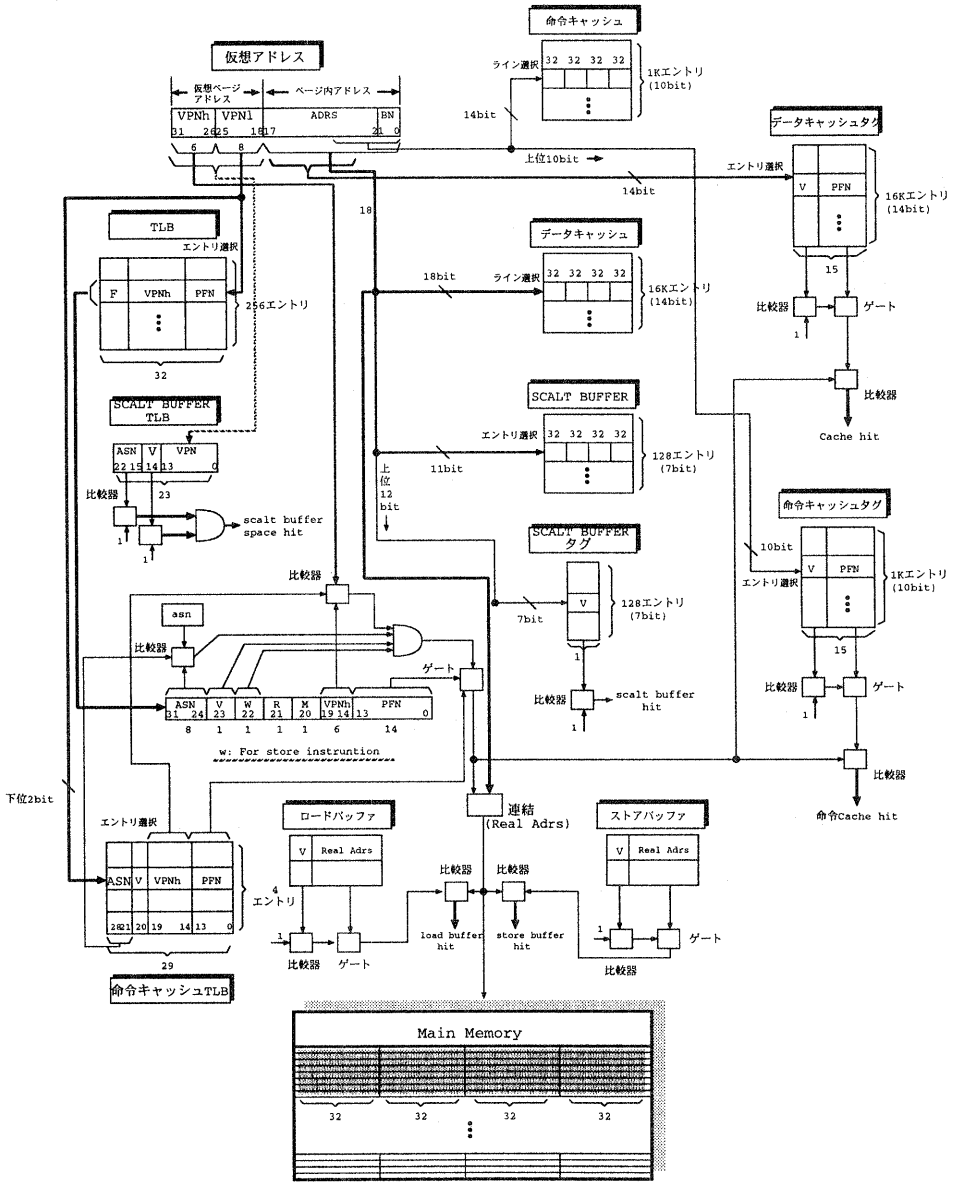
仮想アドレス
仮想ページアドレス
ページ内アドレス
VPNh
VPN1
ADRS
BN
命令キャッシュ
ライン選択
1Kエントリ (10bit)
14bit
上位10bit
データキャッシュタグ
エントリ選択
V
PFN
16Kエントリ (14bit)
TLB
F
VPNh
PFN
256エントリ
データキャッシュ
18bit
比較器
ゲート
Cache hit
SCALT BUFFER TLB
ASN
SCALT BUFFER
128エントリ (7bit)
11bit
scalt buffer space hit
SCALT BUFFER タグ
命令キャッシュタグ
10bit
asn
7bit
scalt buffer hit
w: For store instruntion
下位2bit
命令Cache hit
ロードバッファ
連結 (Real Adrs)
ストアバッファ
Real Adrs
4 エントリ
load buffer hit
store buffer hit
命令キャッシュTLB
Main Memory

図 5: アドレス変換図

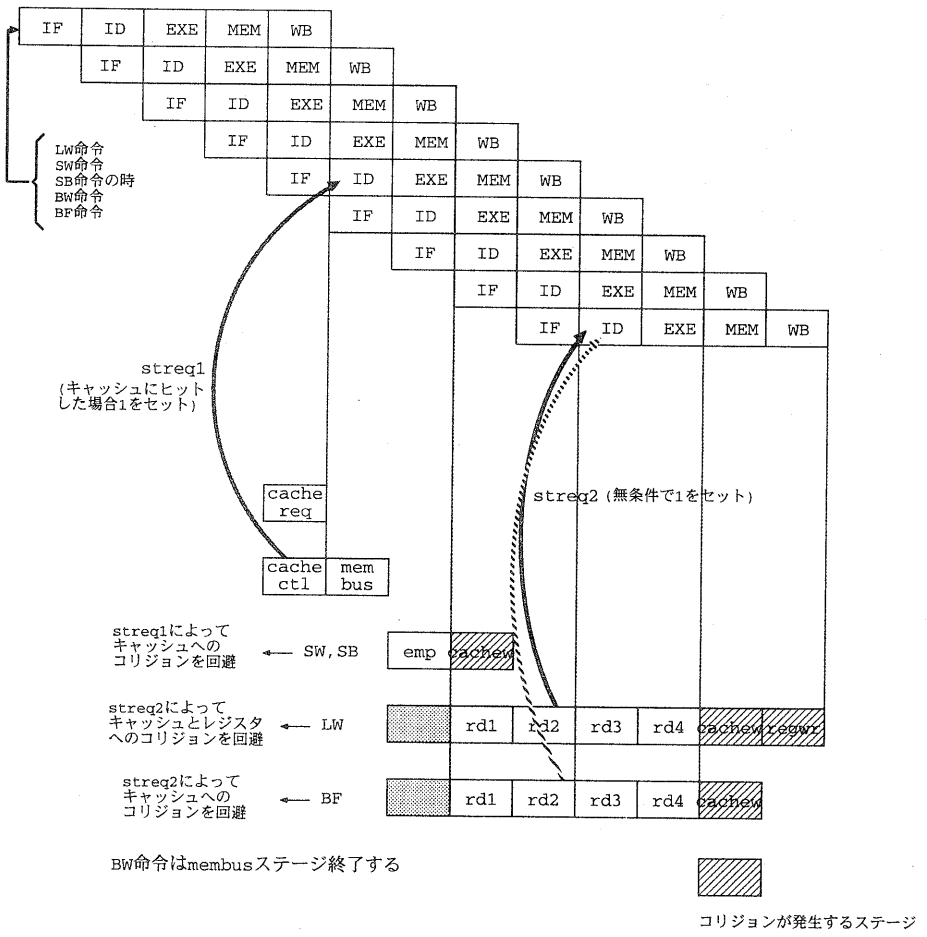

図 6: LW,SW,SB,BW,BF 命令のパイプライン動作