2K-3

VLIW 実験環境 CHA-MEN を開発基盤とした CMP シミュレータの一実装

青木 勲 † 米田 淳一 † 岡 大輔 † 古川 文人 ‡ 月川 淳 †
大津 金光 † 横田 隆史 † 馬場 敬信 †
† 宇都宮大学工学部情報工学科 ‡ 帝京大学ラーニングテクノロジー開発室

1 はじめに

現在, マイクロプロセッサの多くが，チップ内に複数の要素プロセッサを搭載するチップマルチプロセッサ (CMP) の構成を採用するようになってきている. その CMP の要素プロセッサとして, 低消費電力で命令レベル並列性を抽出できる VLIW プロセッサを採用した CMP の研究が行われている. チップ上のトランジスタ数が増大を続ける将来において, CMP の性能向上は, その豊富な資源をいかに有効活用するかが重要である. そこで, 将来のデバイス技術及び応用分野に応じてチップ内構成を詳細に検討するための実験環境が必要となる.

しかし, その CMP を開発するにあたり, 実現しようとするアーキテクチャの性能を評価する有効な手段は無い. 現在は単一 VLIW プロセッサシステムの性能評価のためのシミュレーション環境の開発基盤として, 実験環境 CHA-MEN[1] を開発済みである. 本稿では CHA-MEN を利用して, ある特定のチップマルチ VLIW プロセッサのシミュレータを実装することで, 実験環境の有効性, 拡張の容易性などについて実験的に検証した.

2 開発基盤 CHA-MEN

図 1 に実験環境 CHA-MEN を示す. この開発基盤となる CHA-MEN は, 単一 VLIW プロセッサの詳細な性能評価を目的としている. CHA-MEN は, 単一 VLIW プロセッサを対象としたクロックレベルシミュレータと, VLIW 向け命令スケジューラを含む言語処理系とから構成される. 本言語処理系では, まず C 言語で記述された評価プログラムのソースコードをコンパイルし, 逐次的に記述されたアセンブリコードを作成する. その後スケジューリングを行い, 命令レベルでの並列化を行い, VLIW 命令形式のアセンブリコードとして生成する. そして, アセンブラによりオブジェクトコードを作り, ランタイムライブラリとリンクして, 単一 VLIW プロセッサシミュレータの入力であるメモリイメージを出力する. 単一 VLIW プロセッサシミュレータは, メモリイメージ, HW 情報ファイル, 命令セット定義ファイルを入力としてシミュレーションを行い, その実行結果と統計情報を出力する.

この単一 VLIW プロセッサシミュレータは C++で記述され, 可読性に優れた既存のプロセッサシミュレータ SimCore/AlphaRealScalarVersion0.9.30[2] をベースとして作成されている. 本稿では, この単一 VLIW プロセッサシミュレータを拡張し, チップマルチ VLIW プロセッサシミュレータを開発した.

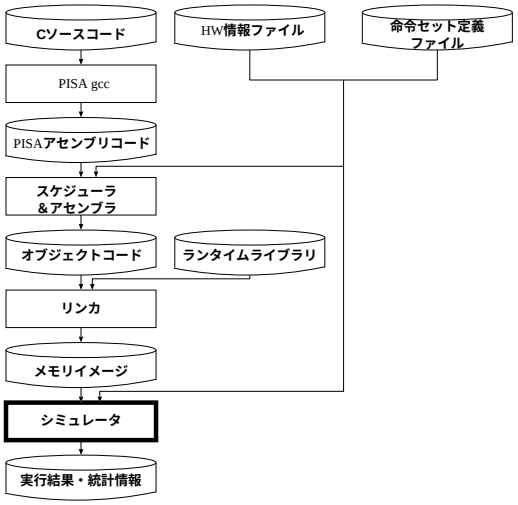

図 1: CHA-MEN シミュレーション環境

3 シミュレーションモデル

チップマルチ VLIW プロセッサシミュレータのシミュレーション対象となるプロセッサ構成を説明する. 構築する実験環境は, 今後のアーキテクチャ検討のために開発される, さまざまなチップマルチ VLIW プロセッサのベースとなるシミュレーション環境を提供することを目的としている. それを利用した一実装例のモデルとしてシンプルなモデルを策定した.

3.1 マルチスレッド実行モデル

CMP の最大の特徴である, 高速なプロセッサ間通信を活かすプログラムの実行モデルとして, マルチスレッド実行が有効であると考えた. 今回策定したマルチスレッド実行では, あるプロセッサでのスレッド実行中に, 空いているプロセッサ上で次に実行されるスレッドを次々に起動する. また, スレッドの実行を終えたプロセッサが新たなスレッドを実行可能とするために, スレッド実行終了を知らせる機能が必要となる. マルチスレッド実行では, 複数のスレッドが複数のプロセッサ上でオーバーラップして実行される. 計算に必要なデータを複数のスレッド間で共有している場合, そのデータの更新が正しい順序で行われなければ, 正しい計算結果が得られない. このため, スレッド間で同期をとり, 共有データの通信を正確に行う必要がある.

図 2(a) に, マルチスレッド実行の様子を示す. 本実行モデルでは, スレッドの起動を fork 命令によって行う. fork 命令は, この命令を実行したプロセッサのコンテキストを, 次に実行するスレッドが起動されるプロセッサ上にコピーし, スレッドの開始アドレスを知らせる. ここで, 1 つのスレッドから起動できるスレッドは, スレッドの管理とスレッド間の同期処理を簡単

Implementation of VLIW CMP simulator based on CHA-MEN simulation environment

†Isao Aoki, Junichi Yoneda, Daisuke Oka, Atsushi Tsukikawa, Kanemitsu Ootsu, Takashi Yokota and Takanobu Baba, Departmentof Information Science, Faculty of Engineering, Utsunomiya University

‡Fumihito Furukawa, Learning Technology Laboratory, Teikyo University

にするために 1 つとする．また，スレッドの実行終了の通知は，term 命令によって行う．スレッド間での共有データの通信は，共有メモリのみを介して行う．このようなスレッド間同期通信を，release 命令，wait 命令を使用し，図 2(b) のように実現する．共有データの読み出しが，共有データへの更新を不正に先行することを防ぐために，wait 命令によりスレッドの実行を停止する．共有データへの更新が完了したことを，そのデータを使用するプロセッサに release 命令によって通知し，通知されたプロセッサは実行を再開するようにする．

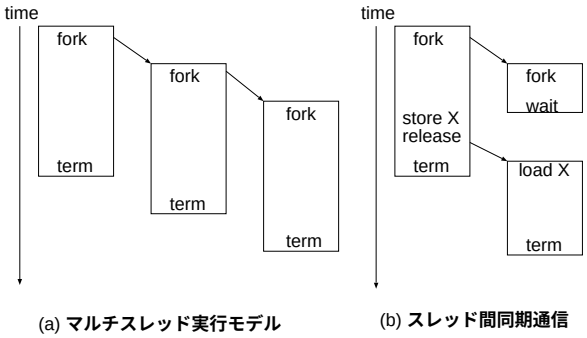

図 2: マルチスレッド実行モデル

3.2 CMP アーキテクチャ

マルチスレッド実行モデルの処理を行う CMP の構成図を図 3 に示す．

本シミュレーションモデルは，各要素プロセッサ (PE) 間と 2 次キャッシュをメモリネットワークによって結合したシンプルなものとする．メモリネットワーク，2 次キャッシュでの競合を抑えるために，各要素プロセッサには 1 次キャッシュを持たせる．1 次キャッシュには共有データは保存しないこととし，アドレス空間を共有データとそれ以外で区別して使用する．本システムでは 2 次キャッシュをチップ上に集積することにより，低オーバーヘッドのマルチスレッド実行を支援する．

低オーバーヘッドのマルチスレッド実行を実現するためには，他のプロセッサへのスレッド起動時のコンテキストのコピーを高速に行う必要がある．そのための，専用のリソースマネージャーネットワークを設ける．また，fork 命令，term 命令によるプロセッサでのスレッドの実行状況を管理するための専用ハードウェアとしてリソースマネージャを設ける．wait 命令によりスレッドの実行を停止したプロセッサが，release 命令による他のプロセッサからの実行再開の通知を適切に受け取るために，wait 命令と，release 命令を対応づけるための同期テーブルを各要素プロセッサに持たせる．

4 動作検証

実装したシミュレータ上で，簡単なサンプルプログラムを動作させ，正しい結果が得られることを確認した．サンプルプログラムには，配列要素に格納された値を一つずつ調べていき，偶数か奇数かを判別するものを用いた．

また，図 4 のようなトレースログで，正しくスレッド間の同期がとれていることを確認した．図 4 は見やすいように，出力の一部を削除するなどの整形を行っている．2 行目では，PE00 が WB ステージで release 命令 (命令アドレス 060) を実行している．3, 5, 7 行目では，PE01 が wait 命令 (命令アドレス 030) によりスレッドの実行を停止している．PE00 の release 命令が実行されてから 3 サイクル経過後，9 行目で PE01 に release 命令が通知され，スレッドの実行を再開していることが分かる．

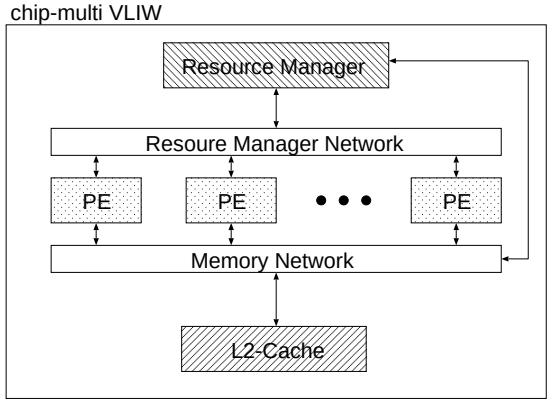

図 3: CMP モデル

```
1:  CYC PE IF  ID  RR  EX  MA  WB   RE  WT
2:  034 00 088 080 078 070 068 060 |-- |--|
3:      01 048 040 038 030 --- --- |-- |-1|
4:  035 00 090 088 080 078 070 068 |-3 |--|
5:      01 048 040 038 030 --- --- |-- |-1|
6:  036 00 098 090 088 080 078 070 |-2 |--|
7:      01 048 040 038 030 --- --- |-- |-1|
8:  037 00 0a0 098 090 088 080 078 |-1 |--|
9:      01 050 048 040 038 030 --- |-- |--|
```

図 4:トレースログの一部

5 まとめ

本稿では拡張性を持った CMP 開発の第一段階として作成した，CMP シミュレータの実装について説明した．今後はより大規模なプログラムによる本シミュレータのテストを行う予定である．そして，拡張性を持ったシミュレータに向けた検討を行い，変更が容易な構造を持った CMP シミュレータの開発を行う予定である．

謝辞 本研究は，一部日本学術振興会科学研究費補助金（基盤研究 (B)18300014，同 (C)16500023，若手研究 (B)17700047）および宇都宮大学重点推進研究プロジェクトおよび帝京大学理工学部教育・研究推進特別補助金の援助による．

参考文献

[1] 古川 文人ほか “チップマルチ VLIW のための拡張性を重視したシミュレーション環境”, 電子情報通信学会コンピュータシステム研究会, 信学技報, vol.107, No.487, pp.37-42, 2005

[2] SimCore Project Homepage, http://www.arch.cs.titech.ac.jp/~kise/SimCore/realscalar.htm