1H-2

スーパースカラ版MIPSCOREの実装と評価

坂口　嘉一　†　松村　貴之　†　吉瀬　謙二 †
東京工業大学 大学院情報理工学研究科 †

1　はじめに

In-order実行のスーパースカラプロセッサである，スーパースカラ版MIPSCORE（MIPSCORE-SS）をVerilogHDLで記述し，FPGAへの実装を行った．本稿では，実装したプロセッサのアーキテクチャについて述べ，ハードウェア規模と性能について評価する．

2　MIPSCORE

MIPSCOREは，ソフトウェアシミュレータであるSimMips[1]をベースに開発されたソフトプロセッサであり，MIPS32に準拠した命令セットを持つマルチサイクルプロセッサである．

本稿では，MIPSCOREの命令数を削減し，5段のパイプライン化を施したもの（MIPSCORE-pipe)をベースとして，スーパースカラプロセッサを実装する．

3　スーパスカラ版MISPCOREのアーキテクチャ

プロセッサのスーパースカラ化は，1サイクルに複数の命令を実行することで，性能向上を目指す手法である．

図1にスーパスカラ版MIPSCORE(MIPSCORE-SS)のパイプライン構成を示す．MIPSCORE-SSは，最大で2命令を同時実行可能な2-wayのスーパスカラプロセッサである．パイプラインは，命令フェッチ（IF），デコード・スケジュール（ID），実行（EX)，メモリアクセス（MA），ライトバック（WB）の5つのステージからなる．命令の実行，およびレジスタファイルへのライトバックはin-orderに行う．また，フォワーディングにより，EX・MAステージで生成された結果は，ライトバックを待たずに使用できる．

3.1　IFステージ

IFステージでは，1サイクルに最大2命令をフェッチする．分岐予測を実装していないため，分岐命令は常に分岐不成立と仮定して命令フェッチを行う．また，MIPSCORE-SSでは，遅延分岐をサポートしない．命令キャッシュは，2命令境界に整列したアクセスのみをサポートする．そのため，分岐命令の飛び先が，2命令境界に整列していないアドレスである場合，1命令をNOPに置き換える．

Implementation and Evaluation of Super Scalar MIPSCORE
Yoshito Sakaguchi†, Takayuki Matsumura †, Kenji Kise†
†Tokyo Institute of Technology Graduate School of Information Science and Engineering

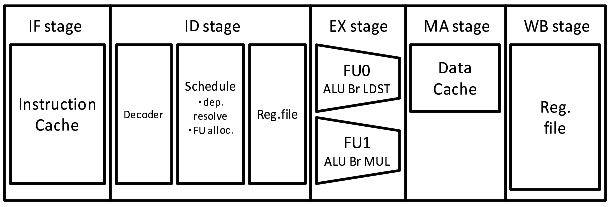

図 1: MIPSCORE-SSのパイプライン構成

3.2　IDステージ

IDステージでは，IFステージでフェッチした2命令をデコードし, 命令スケジューリング，レジスタファイルの読み出しを行う．

スケジューリングでは，実行ユニットの調停および，パイプライン内に存在する命令間の依存関係を解決する．乗算器，ロードストアユニットはそれぞれ1つずつである．フェッチした命令が2つとも乗算命令，もしくは，ロードストア命令である場合，調停が必要になる．このような場合，先行する命令に実行ユニットを割り当てる．依存関係の解決は，EX・MAステージで実行されている命令のデスティネーションレジスタを調べることで行う．パイプライン内で先行する命令と依存関係が無い，もしくは，フォワーディングによって次サイクルで結果が利用可能であれば，命令は実行可能である．例えば，同時にフェッチした2命令の内，先行する命令Aの結果を，もう一方の命令であるBが利用する場合がある．この場合，フォワーディングではAの結果をBに供給出来ない．そのため，Aのみ発行され，BはIDステージで1サイクルストールされる．また，先行する命令を発行できない場合，後続の命令も発行されない．IDステージに発行出来ない命令が存在する間，命令フェッチは停止する．

3.3　EXステージ

EXステージには2つの実行ユニットFU0/FU1を持つ．ロード・ストア命令はFU0でのみ，乗算はFU1でのみ実行できる．算術・論理・分岐命令は，FU0・FU1両方で実行できる．実装されている全ての命令は1サイクルで完了する．

3.4　MAステージ

MAステージでは，FU0で計算したアドレスを使用してメモリアクセスを行う．FU1で実行された命令は，このサイクルでは何もせず，次のサイクルでレジスタファ

表 1: FPGA への実装に要するハードウェア量と主なモジュール毎の内訳

	LUT	FF
instruction decoder	115	0
FU0	433	0
FU1	857	0
Sch+forwarding logic	376	0
register file	2464	1024
MIPSCORE-SS total	5493	1764
MIPSCORE-pipe	2672	634

イルに結果を格納する.

4 評価

ここでは，実際に MIPSCORE-SS を FPGA 上で動作させ，評価を行う．FPGA は，Xilinx 社の Spartan-6 シリーズの XC6SLX16 を対象とする．論理合成･bitstream 生成には ISE12.3 webpack を使用する.

4.1 ハードウェア規模

表 1 に，MIPSCORE-SS の主要なモジュール別（命令デコーダ，FU0，FU1，スケジューラとフォワーディング論理），コア全体，比較対象として MIPSCORE-pipe のコア全体の占有リソースを示す.

表から，使用リソースの約半分がレジスタファイルの実装に費やされていることが分かる．MIPSCORE-pipe と MIPSCORE-SS の使用リソース差の大部分はレジスタファイルによるものである．2-way のスーパースカラでは，4R2W すなわち 6 ポートのレジスタファイルが必要である．FPGA で使用可能なメモリブロックは最大 2 ポートであり，そのままではレジスタファイルとして使用できない．この実装では BlockRAM を使用せず，FF と LUT を使用してレジスタファイルを実装した．その結果，このようにリソースを消費する結果となった.

4.2 性能

ここでは，MieruPC 用に移植した Dhrystone-2.1 を使用して性能評価を行う.

MIPSCORE-pipe，MIPSCORE-SS 両者で，命令・データキャッシュは，共に 4KB のダイレクトマップキャッシュである.

表 2 に MIPSCORE-pipe および MIPSCORE-SS の dhrystone-2.1 の実行結果を示す．動作周波数は，ともに 40MHz とした．同じ周波数で動作した場合，スーパースカラ化により約 27%の性能向上が得られている.

ハードウェア規模が約 2 倍であることを考えると，それに見合う性能向上とは言い難い．ただし，ハードウェア増加の大部分はレジスタファイルによるものである．レジスタファイルの実装に BlockRAM を使用できるようにすることで，ハードウェア量当りの性能は改善出来る可能性がある.

表 2: Dhrystone-2.1 の実行結果. コアは 40MHz で駆動

	MIPSCORE-pipe	MIPSCORE-SS
DMIPS	22.354	28.175

図 2: Dhrystone 実行時の 1 サイクル当り実行命令数

図 2 に MIPSCORE-SS の実行サイクルの詳細ついて示す.（1）キャッシュミスによるストールサイクル数 (stall)，（2）2 つの命令を実行したサイクル数 (dual)，（3）1 つの命令を実行したサイクル数 (single)，（4）命令を実行していないサイクル数 (no)，を計測した．命令実行数に NOP 命令はカウントしていない.

Dhrystone では，命令・データ共に 4KB のキャッシュに必要なデータがほぼ収まるため，ミスが発生せず，(1) は極めて小さい．また，(2) のケースも多いとは言えない．グラフから 2 つの実行ユニットが同時に利用されるのは．実行サイクルの 15%以下にすぎないことが分かる.

（4）のケースも多く，これは，分岐命令によるものと考えられる．現在の MIPSCORE-SS には分岐予測器が実装されていない．そのため，成立する分岐につき，2cycle のペナルティが発生する．Dhrystone において分岐が成立する確率は約 57%で，(4) のケースの約 81%が分岐ミスによるものであった．このことから，分岐ミスペナルティが 2cycle 程度であっても，分岐予測器の実装が必要ということが分かる.

5 まとめ

in-order 発行・完了の 2-way スーパースカラプロセッサを VerilogHDL で記述し，ハードウェア規模と動作の確認を行った．その結果，現状ではハードウェア量増加に対して，性能向上が見合わないと言える．この問題は，レジスタファイルの実装方法を変更することで改善出来る可能性がある.

本研究の一部は, 科学技術振興機構・戦略的創造研究推進事業 (CREST)「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による.

参考文献

[1] 藤枝他．教育・研究に有用な MIPS システムシミュレータ SimMips 情報処理学会論文誌, Vol.50, No.11, pp.2665-2676