6G-09

CPUパイプラインのおける命令発行部の多重化による ステージ利用の効率化

田中 勇気† 中野 秀洋† 宮内 新†

†東京都市大学大学院 工学研究科

1 まえがき

近年，モバイルシステムや組み込みシステムが普及しCPU利用環境が多様化している．それに伴いCPUの更なる高速化が求められている．高速化の主な手法としてはパイプライン (Pipeline)[1]，スーパースカラ (Super scalar)[1, 2]，VLIW (Very Long Instruction Word)[1]，Fill Unit[3]，分岐予測・先行実行などがある．これらの手法はCPUパイプラインのスロットの有効利用に着目した技術である．しかし, 更なるパイプラインの利用効率向上を実現するためには，各ステージ単位での有効利用が必要となる．そのために分岐命令のステージ利用状況に着目したロスの少ない実行手法を示す．またスーパースカラなどの先行技術と共存可能で且つ，複雑なネットワーク環境においてコンピュータ間での互換性がある技術を提案する．

2 提案手法

ヘネシーとパターソンが著書 [1] で述べているようにパイプラインにおいて分岐命令は機能ユニットがあるEXステージ以降を使用しないで処理可能である．提案手法では分岐命令と同時に次の命令を呼び出し，分岐命令で未使用のEXステージ以降を後続命令の処理に利用することによりステージの利用効率を向上させる．

図1は本手法のフロントエンドのブロックダイアグラムである．本手法の実現には命令の同時呼び出し実現のためにIFステージやIDステージの配線の2重化が必要となる．しかし回路量の大きい機能ユニットの多重化は避けることができる．これにより回路量及び消費電力のオーバーヘッドを抑えた高速化が可能である．

図2は提案手法の動作である．各図の右下に実行命令列を示す．図に示すようにフロントエンド (IF, IDステージ) のパイプラインはA, Bの2ラインある．最初

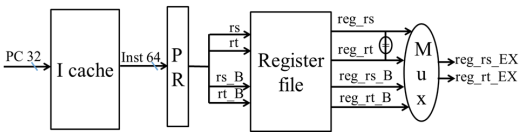

図 1: **提案手法 フロントエンド ブロック図**

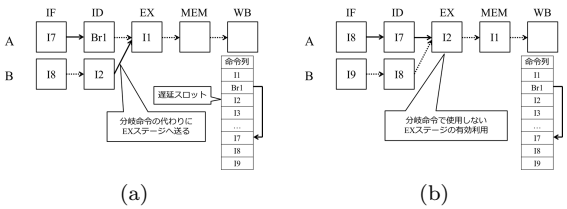

図 2: **提案手法動作** 各図の右下は実行命令列，I1-I9は通常命令, Br1は分岐命令で分岐先はI7である．

のサイクルでラインAにI1をフェッチし，ラインBにI1の一つ先のアドレスにあるBr1をフェッチする．3サイクル目にはラインAはEXステージにI1，IDステージにBr1，IFステージには分岐予測によりI7がある．ラインBはIDステージに遅延スロットにあったI2，IFステージにはI8がある．ここでラインAのIDステージに分岐命令があるので次のサイクルではEXステージにラインAのBr1ではなくラインBのI2がEXステージに送られる (図2a)．4サイクル目でラインAのEXステージにI2が移り分岐命令で使用されないEXステージが有効利用される (図2b)．

2.1 実験結果及び考察

本研究ではソフトウェアシミュレータであるSimpleScalar v.30[1] を使用し，命令発行部を多重化し分岐命令をIDステージで処理が完了するように変更を加えた．テストデータにSPEC95[2] ，Dhrystone[3] 及びCoremark[4] を用いてスループットを計測した．さらにverilogHDLにて提案手法のCPUパイプライン回路を

Improvement of stage utilization efficiency by instruction issue unit multiplexing in the CPU pipeline

Yuki TANAKA†, Hidehiro NAKANO†and Arata MIYAUCHI†
†Computer Engineering, Tokyo City University, Tamadutsumi 1-28-1, Setagaya-ku, Tokyo, 158–8557 Japan
{g1481530, hnakano}@tcu.ac.jp, miyauchi@ic.cs.tcu.ac.jp

[1] SimpleScalar LLC
[2] The Standard Performance Evaluation Corporation
[3] A Synthetic Systems Programming Benchmark
[4] An EEMBC benchmark

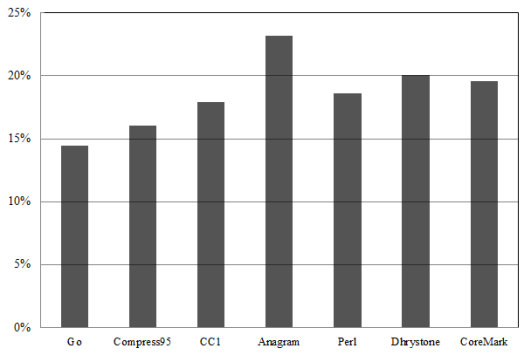

図 3: **Simplescalar による改善率の比較**

作成し Quartus II を用いて回路量の増加量を計測し，ModelSim にてスループットを測定し評価した．

図 3 は SimpleScalar で計測したサイクル数の改善率である．提案手法によるサイクル数の改善率は最も少ない go で約 14%，最も多い anagram で約 23%である．この差は各ベンチマークプログラムにおける命令比に因るものである．Dhrystone の改善率は約 20%，CoreMark の改善率も約 20%である．文献 [4] によると代表的ベンチマークプログラムである SpecMark の分岐命令は全体の約 20%である．提案手法の改善率は分岐命令数に依存しているので，Dhrystone と CoreMark の改善率が約 20%であることは妥当である．

次に実際に Quartus II にて作成した回路を用いて ModelSim に Dhrystone と Coremark のバイナリーコードを実行させた結果を表 1 に示す．

表 1: Dhrystone CoreMark 実行結果

	Dhrystone	CoreMark
サイクル数 (通常)	108,432	11,767
サイクル数 (提案手法)	90,963	9,522
改善サイクル数	17,469	2,245
改善率	16.11%	19.08%
分岐命令数	19,183	2,531
分岐命令率	17.69%	21.51%
分岐予測ヒット率	91.05%	88.70%
期待改善率	16.11%	19.08%
期待改善サイクル数	17,466	2,245
予測ミス	+3	0

Dhrystone を実行した場合，提案手法を適用しない回路では実行サイクルが 108,431 サイクル．適用した回路では 90,963 サイクルであった．改善率は 16.11%であり期待改善率と一致する．CoreMark を実行した結果は改善率 19.08%である．改善サイクル数と期待改善サイクル数は共に 2,245 で期待通りの高速化が確認できた．

3 結論

本研究では分岐命令における未使用のステージを次命令で利用することによりステージ利用効率の向上させる方法を提案した．また回路量の大きい機能ユニットの多重化を避け命令発行部のみの多重化により回路量及び消費電力のオーバーヘッドを抑えた高速化方法を提案した．実験から，MIPS アーキテクチャにおいて約 5%から 23%の速度向上を確認し本提案手法の有効性を示した．本研究で提案した構成を取り入れることによる回路量増加は約 16%から 28%となっており，高速化効果に対して十分に小さいオーバーヘッドであることを確認した．今後の課題としては，インテル CPU への対応が挙げられる．インテル CPU アーキテクチャ [5] は CISC と RISC の良い部分を取り込んだものになっている．さらに Fill unit 手法に類似した機構が実装されいる．また本提案手法と類似したマクロフュージョン手法も実装されている．しかしマクロフュージョンは分岐命令とその直前の命令を結合する手法であるが，本提案手法は分岐命令と後続命令を結合する点において根本的な違いがある．より複雑な次世代の CPU アーキテクチャへの適用について検討する必要がある．

参考文献

[1] John L. Hennessy and David A. Patterson. コンピュータアーキテクチャ 定量的アプローチ 第 4 版. 株式会社翔泳社, 2008.

[2] Mike Johnson. スーパースカラ・プロセッサ. 日経 BP 出版センター, 1994.

[3] Manoj Franklin and Mark Smotherman. A fill-unit approach to multiple instruction issue. In *Proceedings of the 27th Annual International Symposium on Microarchitecture*, MICRO 27, pp. 162–171, New York, NY, USA, 1994. ACM.

[4] Robert F. Cmelik, David R. Ditzel, Edmund J. Kelly, and Shing I. Kong. An analysis of mips and sparc instruction set utilization on the spec benchmarks. Technical report, Mountain View, CA, USA, 1993.

[5] Intel Corporation. , *Intel 64 and IA-31 Architectures Optimization Reference Manual*, Sep 2015.