卒業論文

題目

スーパースカラプロセッサを対象とする可変段数パイプラインの最適なパイプライン構成の研究

指導教員

佐々木　敬泰 助教

2015年

三重大学 工学部 情報工学科
コンピュータアーキテクチャ研究室

中村　悠暉 (411837)

内容梗概

近年，プロセッサの性能向上に伴う消費電力の増加が深刻な問題となっている．現在消費電力低減手法として，処理の負荷に対して周波数と電圧を動的に変更する Dynamic Voltage and Frequency Scaling (DVFS) が主流となっている．しかし，DVFS は電圧制御におけるオーバーヘッドが大きいために，その処理の負荷に対する制御が粗粒度である．そのため，消費電力を細粒度に制御する技術として，可変段数パイプライン (VSP) アーキテクチャが提案されている．VSP は実行するプログラムの振る舞いに応じてパイプライン段数を動的に変化させることで，消費電力の低減を実現している．VSP はパイプライン段数切替におけるオーバヘッドが小さいため，DVFS よりも細粒度な制御が可能である．しかし，VSP のスーパースカラに対する有効性は十分に示されていない．その理由として，スーパースカラのパイプライン構成は多種多様であるため，どの構成が最適であるか明らかにされていないことが挙げられる．本稿では，パイプライン構成を効率的に分析するため，実行結果を基にシミュレーションを行うトレースドリブンシミュレータを実装し，SimPoint という手法を用いた短時間で高精度な評価を行った．その結果から，最適なパイプライン構成を提案する．

Abstract

Increase of energy consumption caused by processor enhancement has recently become a serious problem. Dynamic voltage and frequency scaling (DVFS) which dynamically lowers the supply voltage and clock frequency is widely used to reduce energy consumption. However, it is difficult to deliver fine-grain energy optimization by using DVFS because a voltage regulator takes a long time for scaling the voltage. To reduce energy consumption at fine-grain interval, a variable stages pipeline (VSP) processor is proposed. VSP reduces energy consumption by dynamically varying the pipeline depth to suitable pipeline depth according to behavior of a running program. VSP can optimize energy at finer-grain than DVFS because pipeline scaling has a small overhead. However,effectiveness of VSP technique for superscalar processor was not shown enough. The reason is the most suitable pipeline structure is not clarified because of a variety of superscalar processor pipeline structures. In this paper, we fablicate Trace Driven Simulator which simulates by a simulation result and evaluate with SimPoint to obtain highy precise results in a short time for analyzing pipeline structures effectively. According to the results, We propose the most suitable pipeline structure.

目 次

図 目 次

表 目 次

1　はじめに

近年，プロセッサの性能向上に伴う消費電力の増加により，低消費電力と高性能の両立が要求されている．消費電力の増加は発熱量の増加やバッテリー持続時間の減少につながり，問題となる．そのため，プロセッサの負荷に応じてプロセッサの動作を切り換えることで，高性能かつ低消費電力を実現する手法が検討されている．その中の手法の一つに DVFS (Dynamic Valtage and Frequency Scaling)[1] と呼ばれる手法が提案されている．この手法は，電源電圧と周波数を動的に変更する手法であり，プロセッサの負荷が高い時には電圧と周波数を高くし，負荷が低いときには性能があまり必要でないため電圧と周波数を低くすることにより，高性能かつ低消費電力を実現する手法である．しかし，DVFS は電圧制御におけるオーバーヘッドが大きいために，その処理の負荷に対する制御が粗粒度である．そのため，要求されている性能を維持しつつ消費電力を細粒度に制御する技術が求められている．そこで，高性能と低消費電力を両立させる手法として可変段数パイプライン (VSP) アーキテクチャ[2] が提案されている．VSP はプロセッサに要求される性能に応じて動作周波数とパイプライン段数を動的に変化させることで，消費電力の低減を実現している．しかし，VSP のスーパースカラプロセッサにおける有効

性は十分に示されていない．スーパースカラプロセッサでは多くが命令実行順がプログラム記述順と異なるアウトオブオーダー実行となっており，段数切り替え後の動作の予測が困難である．そのため，この原因として最適なパイプライン構成が解明されていないことが挙げられる．スーパースカラプロセッサにおけるパイプライン構成は多種多様であり，実際の回路構成により，その全ての構成を実装し分析するのでは非効率であるため，VSP トレースドリブンシミュレータの実装を行った．また，従来の最初の数億サイクルのシミュレーションでは精度が低く，これに対し，プログラム全体をシミュレーションするのでは膨大な時間がかかり非効率であるため，短時間で高精度な評価を可能とする SimPoint という手法を用いた評価を行った．その結果から，最適なパイプライン構成を提案する．

2 先行研究

2.1 DVFS

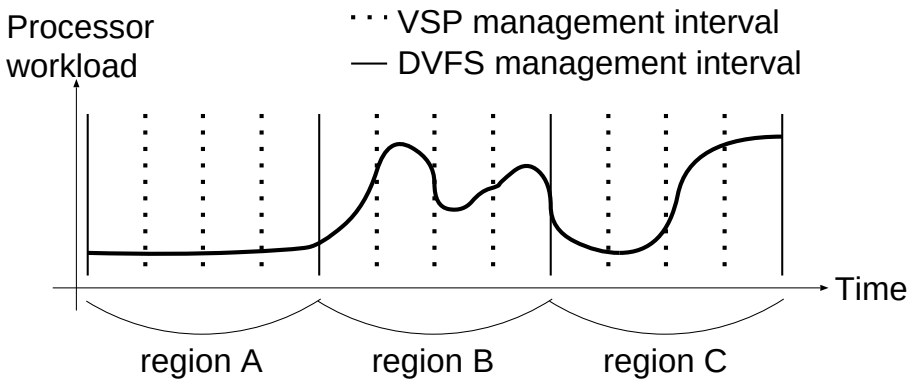

図 2.1: DVFS と VSP のスケジューリング間隔

DVFS は現在消費電力低減に対する手法として主流となっている．処理の負荷が小さい時には周波数と電圧を下げ，逆に負荷が大きい時にはそれらを上げる．このように動的に調整することにより消費電力低減を実現している．しかし，DVFS は将来的に消費エネルギー削減効率の低下が予想されている．なぜなら近年 CMOS の電源電圧は低下の一途をたどっており電源電圧の下げ幅は小さくなっているためである．さらに，DVFS は電圧制御におけるオーバーヘッドが大きい (10ms 以上) ために，その処理の負荷に対する制御が粗粒度であるという問題点がある．図 2.1 に示す DVFS と VSP のスケジューリング間隔の通り，電源電圧を変化させない VSP の方がより細粒度な制御が可能である．また，インテルによ

りナノセカンドオーダの電圧制御システムが開発されているが，現在それは一般的な ASIC フローや他社で用いることはできない．

2.2 VSP

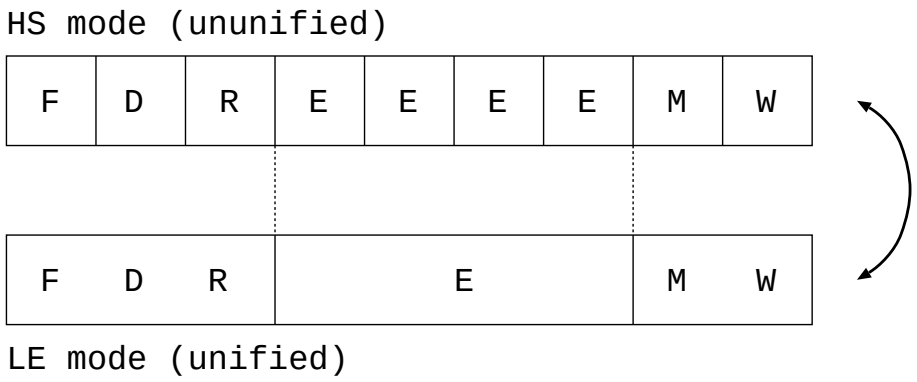

図 2.2: Variable Stages Pipeline

VSP アーキテクチャの構成の例を図 2.2 に示す．VSP アーキテクチャは高性能である High-Speed (HS) モードと低消費電力である Low-Energy (LE) モードの二つのモードを持つ．プロセッサにかかる負荷が大きく高い性能を要求されている場合，多段のパイプラインプロセッサとして高周波数で動作する HS モードに切り換える．一方，プロセッサにかかる負荷が小さく高い性能を要求されていない場合，パイプラインステージを統合し少段のパイプラインプロセッサとして低周波数で動作する LE モードに切り換える．

この二つのモードの切り替えは，各ステージの節目であるパイプライ

ンレジスタと呼ばれる D フリップフロップを用いて行う．パイプラインレジスタを有効化することによりステージ間を分離し HS モードに，無効化することによりステージ間を統合し LE モードに切り換える．

一般にパイプラインレジスタへのクロック供給のためのエネルギーは膨大であるため，LE モード時にパイプラインレジスタを無効化することは，低消費エネルギーの実現に大きく貢献している．さらに，パイプライン段数が少段になることによりデータ依存や分岐による待ちサイクルが低減されるため，分岐予測器など一部の回路が不必要となる．このような回路へのクロック供給を止めることでも低消費エネルギーを実現している．そして，分岐予測ミスペナルティとデータ依存による待ちサイクルの削減は，低消費エネルギーだけでなく，Instruction Per Cycle (IPC) が増加するという利点もある．IPC とは、1 サイクルあたりに実行できる命令数で、この数字が大きいほど計算速度が速いことを意味する。しかし，近年主流となっている高性能プロセッサであるスーパースカラプロセッサに対する有効性は示されていない．

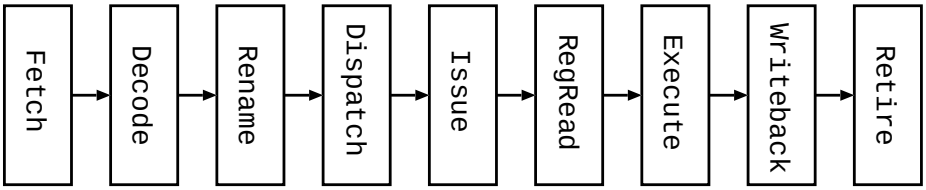

図 2.3: FabScalar

2.3 FabScalar

ノースカロライナ州立大学により，スーパースカラプロセッサ自動生成ツール FabScalar[3] が提案されている．スーパースカラプロセッサとは同時に複数の命令をフェッチし複数の実行ユニットを並列に動作させるプロセッサのことで，FabScalar はノースカロライナ州立大学により提案されているスーパースカラプロセッサを自動設計するツールセットである．フェッチ幅，パイプライン段数，各ユニット数等のパラメータを与えることで様々な構成にすることができる．FabScalar により設計されるプロセッサの最も基本的な構成を図 2.3 に示す．

FabScalar を使用する事により，様々な構成のスーパースカラプロセッサを短時間で設計する事が可能となる．そのため，この FabScalar に VSP 構造を組み込み評価を行うことで，スーパースカラプロセッサに対する有効性を示すことができる．本研究では一つの構成のスーパースカラコアに VSP アーキテクチャを適用するにとどまらず，多種多様なスーパー

スカラアーキテクチャ上における VSP の効果を比較検討し，VSP の有効性を示すことを目標としている．

3 スーパスカラへの VSP 適用時の実装と評価方法における問題点

3.1 最適なパイプライン構成における問題

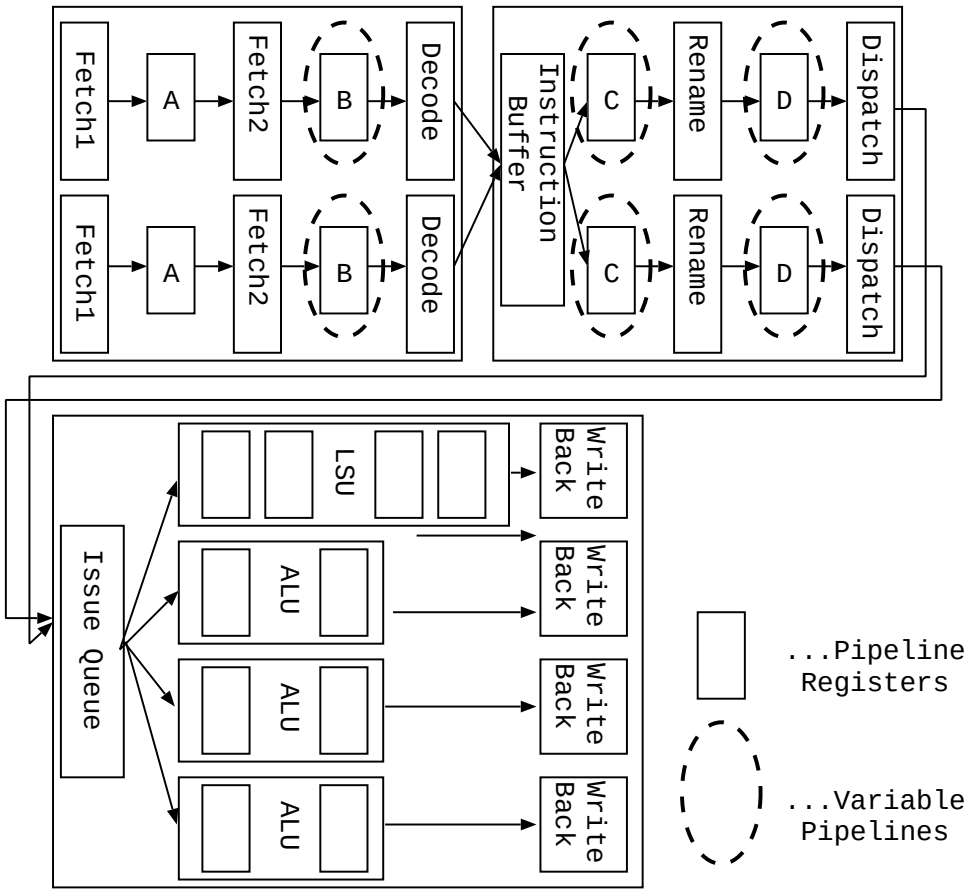

図 3.4: 従来の FabScalar-VSP

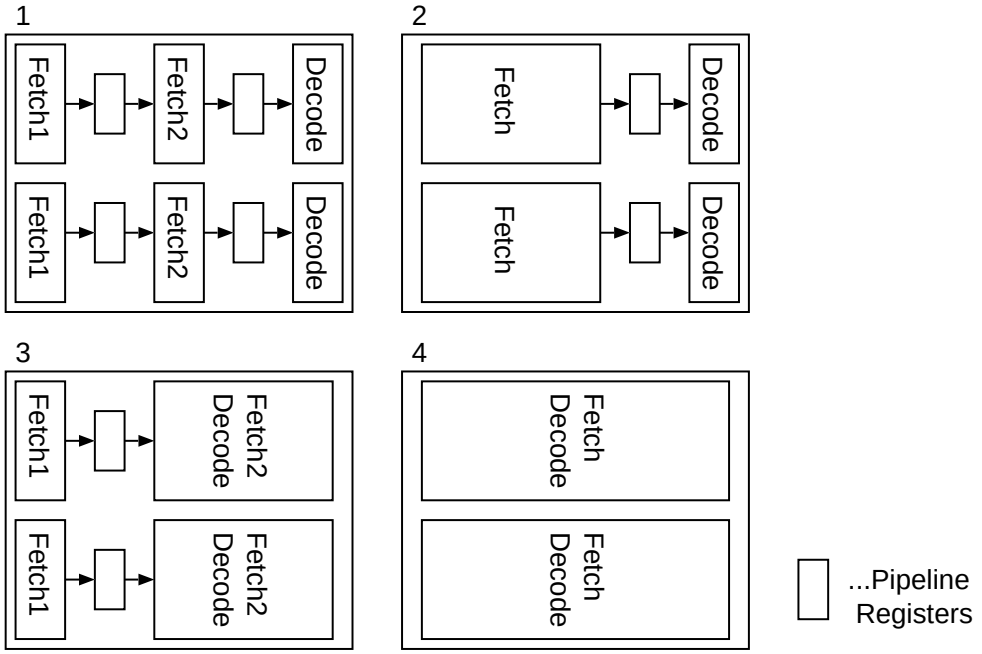

図 3.5: パイプライン構成例

シングルパイプラインプロセッサにおける VSP の利点を以下にまとめる．

- パイプライン段数と周波数の細粒度な制御とパイプラインレジスタへのクロック供給を停止することにより，消費電力低減を実現している．

- パイプライン段数減少時，分岐予測ミスペナルティとデータ依存による待ちサイクルが削減される．そのため，IPC が増加する．

前述の通り，様々なスーパースカラコアにおける VSP の有効性，特に

上記のような利点を示すため，FabScalar に VSP を導入することを予定している．従来の FabScalar-VSP では図 3.4 のようなパイプライン構成で実装されている．なお，ここでは説明のため簡略化した図を用いるが，実際は Issue Queue まで 4 並列のスーパースカラプロセッサである．このパイプライン構成で評価を行ったが，評価結果では IPC が低下していた．FabScalar は命令実行順がプログラム記述順と異なるアウトオブオーダー実行のスーパースカラプロセッサであるため，LE モード時の動作の予測が困難である．例えば，スーパースカラプロセッサは複数ユニットを並列動作させるため，LE モード時に，あるステージは使用頻度が高いが別のステージは使用頻度が低いという現象が生じる．この場合使用頻度の低いパイプラインステージの統合は無駄である．そして，アウトオブオーダー実行であるため，シングルパイプラインプロセッサにおいては VSP が有効であったプログラムについて，FabScalar においても有効であるとは限らない．そのような理由から，現状図 3.4 のようなパイプライン構成となっているが，最適なパイプライン構成について分析する必要がある．

また，実際の回路設計により全てのパイプライン構成について分析するのでは莫大な時間がかかり非効率的である．例えば，図 3.5 のように，フェッチデコードステージの構成パターンについてだけでも 4 パターン存

在する．他のパイプラインステージについても同様である．つまり，パイプラインレジスタの数を N とすると，2^N 通り存在することになり，構成パターンは莫大となる．そして，これはある一種のスーパースカラプロセッサについてである．実際のスーパースカラプロセッサの構成は多種多様であるため，様々な構成について評価を行うことができる環境が必要となる．

3.2 評価方法における問題

例えば，MCF の場合では，プログラム全体の命令数は 580 億命令であるが，その全てを用いて電力評価するのはシミュレーション時間が掛かりすぎるため非現実的である．そこで，従来は最初の数億命令を用いて評価をしていた．しかし，一般にプログラムの最初の部分は初期化等のルーチンが占めており，プログラムの特徴を考慮した評価であるとは言い難く，特に，VSP はプログラムの特徴に合わせて動作するので，この問題の影響を受けやすい．そのため，より精度が高くシミュレーション時間も現実的である手法が求められる．

4 高精度なパイプライン構成評価方法の提案

4.1 VSP トレースドリブンシミュレータの実装

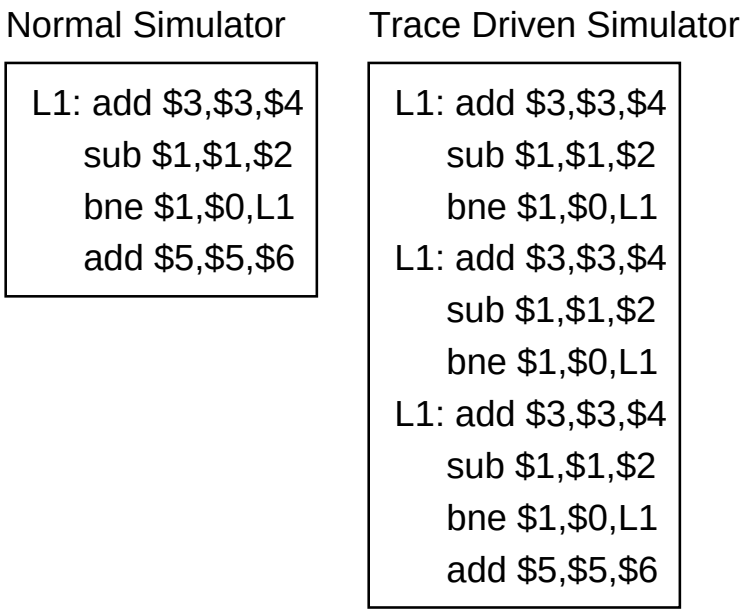

図 4.6: 通常のシミュレータとトレースドリブンシミュレータの入力データ例

パイプライン構成の効率的な分析のため，VSP トレースドリブンシミュレータの実装を行う．トレースドリブンシミュレータとは，プロセッサやシミュレータの実行結果を基にシミュレーションを行うシミュレータである．図 4.6 の左に通常のシミュレータ，右にトレースドリブンシミュレータの入力するデータの違いの例を示す．通常のシミュレータの場合，分岐命令について値を確認し，分岐先を特定する必要がある．しかし，トレースドリブンシミュレータの場合，分岐先が次にフェッチされるため，

その値を確認する必要がない．図 4.6 の例では 3 つ目の bne 命令について，二回 L1 にジャンプし，3 回目はジャンプしない例を示している．このように，トレースドリブンシミュレータの場合はレジスタの値について保持や計算の必要がないため，レジスタがレディ状態であるかどうかのみを確認できるようにすれば良い．そして，分岐先が特定されているという特徴によりプログラム途中からの実行が容易となる．さらに，C 言語を用いたプログラミングであるため，より自由な変更が可能であり，様々なパイプライン構成を分析するのに最適である．

4.2 SimPoint による評価

シミュレーションの高速化手法の一つとして SimPoint[4] が提案されている．プログラム中では繰返し実行される部分が存在するため，大凡の全体の振る舞いを表す部分が存在する．その部分を文献 [4] ではシミュレーションポイントと呼んでいる．このシミュレーションポイントについて評価を行うことで，高い精度の評価が短時間で得られる．SimPoint は現在そのシミュレーションポイントの選択手法として主流となっているが，プログラム途中からのシミュレーションの開始を要するため，従来のプロセッサモデルでは導入が不可能であった．しかし，今回はトレースドリブンシミュレータを用いるため，この SimPoint を用いることができる．

5　シミュレーション結果の解析と最適なパイプライン構成の提案

5.1　実装

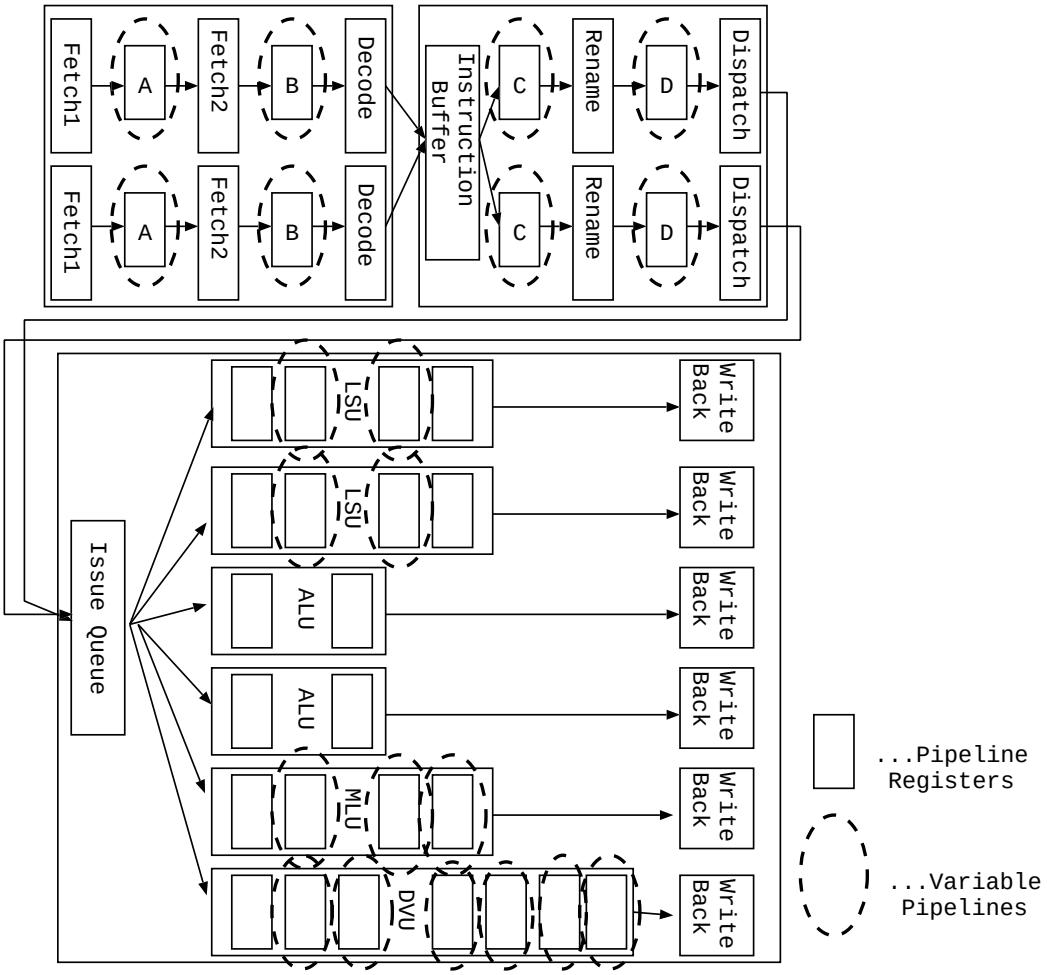

図 5.7: VSP トレースドリブンシミュレータのパイプライン構成

VSP トレースドリブンシミュレータを図 5.7 のような構成で実装した．

従来の FabScalar-VSP と同様に，図 5.7 では 2 並列となっているが，実際

は4並列である．また，FabScalarは一般のプロセッサとは異なる構成の箇所があるため，一部従来のFabScalar-VSPとは異なる構成となっている．本シミュレータは，Variable Stagesの有効，無効を切り替えるかどうかを自由に変更することができる．なお，いくつかのステージが統合できないのは，ハードウェア上の制約によるものである．Instruction BufferではDecodeステージにより1つの命令が2つの命令に分かれることがあるため，増えた命令を保存するステージが必要となるため，Decodeステージと Instruction Bufferを統合することは困難である．同様に，Issue Queueでも命令依存により待機する命令を保存する必要があるため，Dispatchステージと Issue Queueステージを統合することはできない．また，それぞれの実行ステージの最初のパイプラインレジスタが可変でないのは，レジスタの値を物理レジスタから受け取るのに最低でも1サイクル必要であるからである．同様に，LSUの最後のパイプラインレジスタが可変でないのは，キャッシュからデータを受け取るのに最低でも1サイクルは必要であるからである．

時間の都合上，分岐予測器やキャッシュについては実際の動作を模擬するのではなく，既存の研究で発表されているベンチマーク毎に応じたキャッシュミス率や分岐予測ミス率を，パラメータとして与えることで，

確率的にキャッシュミス，分岐予測ミスの発生の有無を決定している．

また，前述の通り C 言語によるシミュレータであり，様々な構成に変更ができるよう実装を行った．そのため，今回の評価では FabScalar に VSP 構造を実装することを考慮しこのシミュレータを実装したので，主に FabScalar においてのパイプライン構成について評価を行ったが，このシミュレータを簡単な変更により，様々な構成のスーパースカラプロセッサについて評価を行うことができる．

しかし，時間の都合上，HS モードと LE モードを動的に切り替えるコントローラーが未実装である．そのため，VSP として動作させた時の全体の性能や電力削減率を評価することはできないが，HS モード時と比較した LE モード時の IPC や，電力削減量について評価することができ，その結果から LE モード時においての最適なパイプラインを分析することができる．

5.2 評価方法

ベンチマークには SPEC2000CINT のうち，今回はシミュレータ上で動作した GZIP，BZIP2，MCF，PARSER，TWOLF の 5 つを用いた．その他のものについては，本シミュレータの入力データの生成に使用した

シミュレータとツールが原因で動作しなかったため，今回の評価からは外した．

また，評価条件として，キャッシュについては周波数を下げないように設定している．すなわち，パイプライン統合時は統合前と比べ，プロセッサ側から見るとキャッシュの速度が上がっている．例えば，プロセッサ側の周波数が 2 分の 1 となった場合，プロセッサ側から見るとキャシュ側の周波数は 2 倍となっているように見える．このようにする理由は，性能を重視した評価を行うため，周波数を低下させた際の性能維持として，キャッシュミスペナルティを減らすためである．

また，前述の通り，ベンチマーク毎にキャッシュミス率と分岐予測ミス率を変更することでプログラムの特性を模倣している．実験で用いたパラメータを表 5.1 に示す．分岐予測ミス率は文献 [5] を，キャッシュミス率は文献 [6] を，キャッシュミスペナルティは文献 [7] を参考に決定した．

なお，レジスタ復元サイクル数とは分岐予測ミス時に起こるレジスタの復元にかかるサイクル数のことである．MIPS R10000 ではリネームマップテーブル情報を保存しておき，分岐予測ミス時にそれをコピーし復元するフラッシュコピー方式が用いられている．そのため，フラッシュコピー方式ではレジスタの復元を 1 サイクルで行うことができる．しかし，

このフラッシュコピー方式にはハードウェア規模が大きく増加するという問題があるため，IPC の大きな向上を見込めない場合，アーキテクチャマップテーブルから演算器を用いて復元する通常の方式を行った方が良い．この場合，レジスタ 32 ビットについて演算器を用いて復元するため，図 5.7 の通り，演算器 6 個により 6 サイクルかけて全てのレジスタについて復元することができる．

ただし，トレースドリブンシミュレータの場合，分岐先の正しい命令が既にフェッチされているため，その分岐先の命令がフェッチされてから Issue Queue に到達するまでのサイクル数を考える必要がある．新しい命令のフェッチとレジスタの復元は同時に行うため，分岐先の命令がフェッチされてから Issue Queue に到達するまでのサイクル数とレジスタ復元サイクル数のうち大きい方が分岐ミスペナルティとなる．Issue Queue に到達するまでのサイクル数とは，すなわち Issue Queue までのパイプライン段数である．そのため，図 5.7 の通り，フラッシュコピー方式の場合，Issue Queue までのパイプライン段数が分岐予測ミスペナルティとなり，アーキテクチャマップテーブルから復元する通常の方式の場合，レジスタ復元サイクル数が分岐予測ミスペナルティとなる．

今回，Issue Queue までのパイプライン段数を変更するため，まずはそ

の変更が反映されるフラッシュコピー方式で評価を行う.

表 5.1: 実験で用いたパラメータ

	GZIP	BZIP2	MCF	PARSER	TWOLF
分岐予測ミス率 (%)	1.1	0.8	0.8	0.8	1.8
キャッシュミス率 (%)	2.030	11.547	0.822	1.302	2.497
レジスタ復元サイクル数 (サイクル)	1	1	1	1	1
キャッシュミスペナルティ (サイクル)	10	10	10	10	10

5.3 評価結果

表 5.2 に評価結果を示す．今回，評価を行ったパイプライン構成のパターンが多く，性能を重視した評価を行ったため，ベンチマーク毎の IPC の上位 5 つの構成パターンを表として示す.

最上位の項目がベンチマークの種類で，最左端の項目が IPC の順位である．また，図 5.7 の構成を元に，ステージ数について，

(Issue Queue まで，LSU，MLU，DVU)

で表している.

また，ベンチマークのうち GZIP,BZIP2,MCF の 3 つにおいては乗除算命令が存在しなかったため，MLU，DVU においては全く影響がない．そのため，その 3 つのベンチマークの MLU,DVU については x で表して

いる．

表 5.2: 各パイプライン構成毎の IPC

IPC RANK	GZIP	BZIP2	MCF	PARSER	TWOLF
1	(2,2,x,x) 2.707	(2,2,x,x) 3.106	(2,2,x,x) 2.727	(2,2,2,3) 2.981	(2,2,2,3) 2.568
2	(4,2,x,x) 2.692	(4,2,x,x) 3.100	(4,2,x,x) 2.721	(2,2,3,3) 2.981	(2,2,2,4) 2.567
3	(6,2,x,x) 2.677	(6,2,x,x) 3.695	(6,2,x,x) 2.714	(2,2,4,3) 2.981	(2,2,3,3) 2.565
4	(2,3,x,x) 2.508	(2,3,x,x) 2.907	(2,3,x,x) 2.446	(2,2,2,4) 2.979	(2,2,2,7) 2.564
5	(4,3,x,x) 2.495	(4,3,x,x) 2.903	(4,3,x,x) 2.441	(2,2,3,4) 2.979	(2,2,4,3) 2.560

5.4 考察

Issue Queue までのパイプライン統合と実行パイプラインステージのパイプライン統合による，分岐予測ミスペナルティとデータ依存による待ちサイクルの削減，そして LSU のパイプイライン統合によるキャッシュミスペナルティの削減により，(2,2,2,3) のパイプライン構成の IPC が最大になったと考えられる．また，どの結果においても LSU の統合が他のパイプライン統合と比べて IPC への影響が大きい．これはプログラムの特性として，ロードおよびストア命令が頻出することが原因だと考えられる．以上より，IPC においては (2,2,2,3) のパイプライン構成が最適で

ある．

しかし，前述の通り，フラッシュコピー方式にはハードウェア規模が増大する問題点があり，図 5.3 に示す Issue Queue までのパイプライン段数に着目した評価結果の通り，Issue Queue までの段数を 6 から 2 へと変更しても，IPC は最大でもたった 0.04 しか増加せず，ハードウェア規模の増大に見合った効果があるとは言えない．そのため，電力削減量についても効果が見込めないのであれば，Issue Queue までの段数を少なくする必要はないと言える．

そこで，大まかな電力削減量について見積もる．

一般にチップ全体において消費電力の約 30 %をクロックが占め，クロックは D フリップフロップ (DFF) に用いられる．そこで，FabScalar 上での DFF の個数の概算を行った．図 5.7 に示されている Issue Queue までのパイプラインレジスタ A，B，C，D，および LSU における DFF の個数を表 5.4 に示す．

FabScalar では乗除算ユニットが存在しないため，MLU と DVU は無視している．しかし，MLU と DVU の電力削減量を無視してでも十分な電力削減量であることが示せれば，MLU と DVU の削減が加わった際にはより電力削減可能なため，有効な見積もりだと言える．

チップ全体の消費電力に対するクロックの消費電力が約 30 %であること，および表 5.4 の DFF の個数より，

$$30 \times \frac{(429 + 511 + 553 + 565)}{8962} \simeq 6.889 \qquad (1)$$

となり，プロセッサ全体で約 6.889%の電力を削減できるため，電力削減量としては Issue Queue までのパイプライン段数の少段化の効果はあると言える．

よって，パイプラインレジスタは無効化するが，フラッシュコピー方式を用いずにアーキテクチャマップテーブルから復元する通常の方式を用いる．この場合，前述の通り，分岐予測ミス時のペナルティは常に 6 サイクルとなる．すなわち，IPC としては，Issue Queue までのパイプライン段数が 6 の時の結果と等しくなる．

表 5.3: Issue Queue までのパイプライン段数 2 と 6 の IPC 比較

GZIP	BZIP2	MCF	PARSER	TWOLF
(2,2,x,x)	(2,2,x,x)	(2,2,x,x)	(2,2,2,3)	(2,2,2,3)
2.707	3.106	2.727	2.981	2.568
(6,2,x,x)	(6,2,x,x)	(6,2,x,x)	(6,2,2,3)	(6,2,2,3)
2.677	3.095	2.714	2.956	2.527

表 5.4: FabScalar における DFF の個数

	A	B	C	D	LSU	総数
DFF 個数	429	511	553	565	110	8962

5.5 VSP への見通し

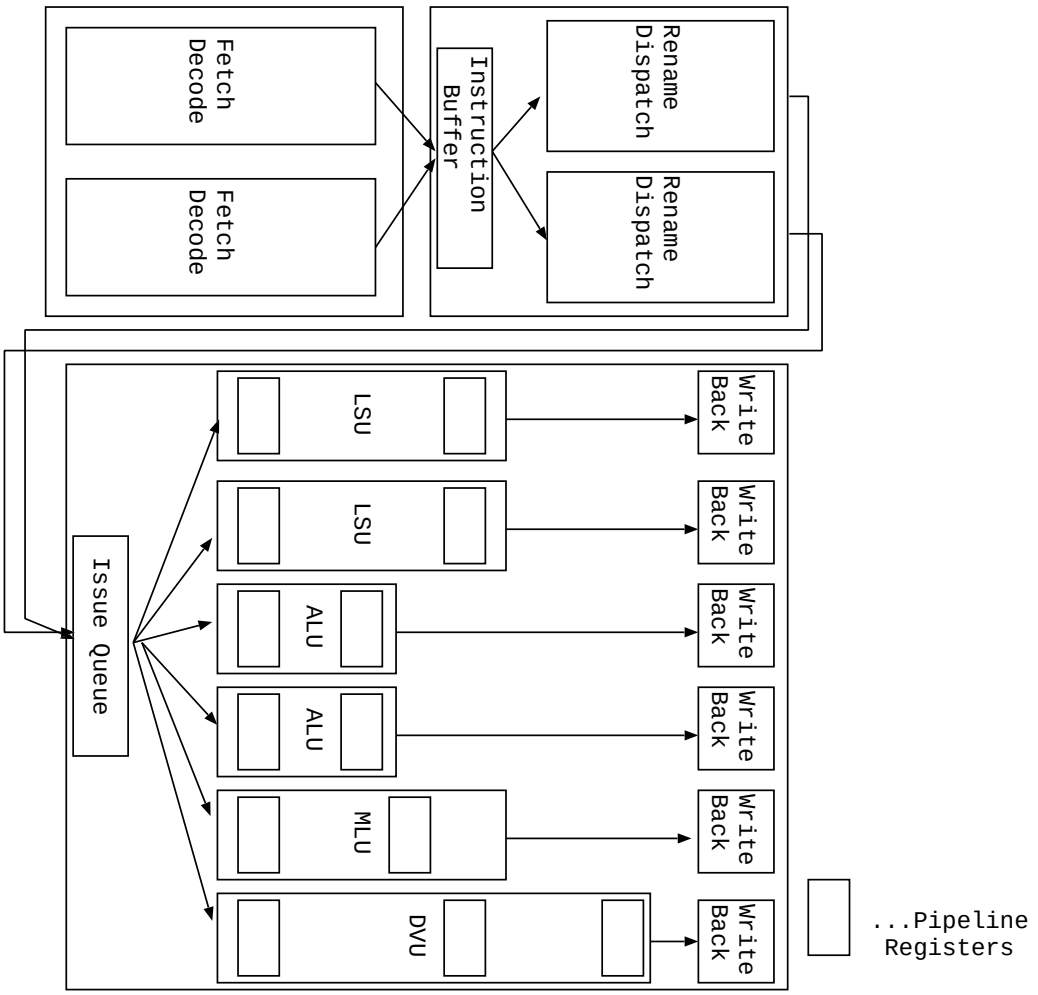

図 5.8: 提案する最適なパイプライン構成

以上を踏まえた提案する LE モード時の最適なパイプライン構成は図 5.8 のようになる．前任者のパイプライン構成 (3,4,2,2) による IPC の評価結果を表 5.5 に示す．提案パイプライン構成が従来の FabScalar-VSP のパイプライン構成と比べ，IPC において平均 0.37 向上することが分かった．

また，表 5.4 の DFF の個数より，提案パイプライン構成によるチップ全体に対する電力削減量は

$$30 \times \frac{(429 + 511 + 553 + 565)}{8962} \simeq 7.257 \qquad (2)$$

となり，提案パイプライン構成による LE モード時にはプロセッサ全体で約 7.257 %電力を削減できる．

表 5.5: 前任者のパイプライン構成による IPC

GZIP	BZIP2	MCF	PARSER	TWOLF
2.332	2.619	2.283	2.562	2.323

6 終わりに

本研究では，パラメータにより自由にパイプライン構成を変更可能な VSP トレースドリブンシミュレータの実装，SimPoint を利用した詳細な評価を行い，パイプライン構成毎の性能を分析した．その結果から，VSP 構造を持つスーパースカラプロセッサの最適なパイプライン構成を提案した．この提案構成では，従来の構成と比べて IPC において平均 0.279 の向上を確認し，LE モード時にはプロセッサ全体で約 7.257 %電力を削減できることが分かった．一方，キャッシュや分岐予測器，VSP コントローラが未実装であったため，それらの実装を行うことでよりプログラムの特性に応じた VSP の評価が可能となる．

今後はその構成を元にした FabScalar への VSP の適用を行い，スーパースカラに対する VSP の有用性を示していく．

謝辞

本研究の機会を与えて頂いた近藤利夫教授，並びにご指導，ご助言頂いた佐々木敬泰助教，深澤祐樹研究員に深く感謝いたします．また，様々な局面でご助力頂いたコンピュータアーキテクチャ研究室の皆様にも心より感謝いたします.

参考文献

[1] J. Pouwelse, K. Langendoen, H. Sips,“ Dynamic Voltage Scaling on a Low-Power Microprocessor ”, Proc. of The 7th ACM Int. Conf. on Mobile Computing and Net- working (Mobicom), pp .251-259, July 2001

[2] 三好 聖二, 他．FabScalar を用いた可変段数パイプライン構造を有するスーパースカラコアの詳細設計, 電子情報通信学会技術報告 CPSY, Vol.113, No.169, pp.103-108, 2013.

[3] N. K. Choudhary，et．al．”FabScalar: Com- posing Synthesizable RTL Designs of Arbitrary Cores within a Canonical Superscalar Template ”. ISCA-38, pp. 11-22, June 2011.

[4] E. Perelman, et.al. Picking Statistically Valid and Early Simulation Points , In the International Conference on Parallel Architectures and Compi- lation Techniques, September 2003.

[5] H. Gao and H. Zhou.“ Adaptive Information Processing: An Eective Way to Improve PerceptronBranch Predictors ”. Journal of Instruction-Level Parallelism , Vol 7. 2005.

[6] Jason F. Cantin and Mark D. Hill. “ Cache Performance for SPEC CPU2000 Benchmarks ”.2003-05. http://research.cs.wisc.edu/multifacet/misc/spec2000cache-data/ (accessed 2015-03)

[7] 有松 優 , 塩谷 亮太, 安藤 秀樹. L1 データ・キャッシュ・ミスに着目した命令発行キューの動的リサイジング, 電子情報通信学会技術研究報告. ICD, 集積回路 111(388), pp.47-53, 2012.

A　プログラムリスト

B　評価用データ