社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－ARC－169 (2)
2006／7／31

# 命令フェッチ調節とサイズ可変なReservation Staionによる電力消費最適化

伊藤康宏† 菅原豊† 平木敬†

Out-Of-Order アーキテクチャに対する電力消費削減手法として最も効果的な手法の 1 つが Issue Logic に対する動的調節である。既存のものとして、スケジューリングウィンドウサイズの調節、と遅延発生時の Fetch Gating がある。本研究で提案する手法は上記の 2 手法を組み合わせ、より効率的な電力削減を可能とするものである。Fetch Gating は誤った分岐予測による不要命令とスケジューリングウィンドウの使用サイズを減らす事ができる。これによって従来よりも多くのスケジューリングウィンドウの invalid エントリの削減が可能になり、これが電力削減につながる。これらの調整機構自体の電力消費を最小化するため、単純なカウンタと比較のみのシンプルな回路を用いて実装される。評価は SPEC2000,95 ベンチマークを用いて行い、Go,Vpr などのベンチマークにおいて最大 5%の電力効率の最適化を達成した。

# FETCH ADAPTATION AND RE-SIZABLE RESERVATION STATION FOR ENERGY EFFICIENCY

YASUHIRO ITO ,† YUTAKA SUGAWARA † and KEI HIRAKI†

Dynamic resource adaptation is one of the methods of energy reduction in Out-Of-Order execution architecture. Size adaption of Reservation Station and fetch gating were already proposed as approaches for it. We propose that combining these two approaches in order to achieve more efficient energy reduction. Fetch gating reduce the useless instructions caused by miss branch prediction and the issue queue occupation. This enables the more reduction of invalid issue queue entry and reduces energy consumption. These adaption scheme are implemented by simple counters and comparators to minimize its energy consumption. We evaluated our scheme with SPEC2000,95 benchmarks and achieved 5% improvement in enegy efficiency at maximum

## 1. はじめに

消費電力の増加やそれに付随する発熱量の増加は著しく、パフォーマンス向上の妨げとなっている。このためプロセッサの省電力化は今後のプロセッサの発展の上での重要課題となっており、すでに数多くの研究がなされている。以下にその例を挙げる。

(1) 製造プロセスを小さくして駆動電圧を下げる
(2) トランジスタの改良または、サブストレートのバイアス電圧印加方法の改良によりリーク電流を下げる
(3) 実行する命令数を減らす (uOPs フュージョン)
(4) 資源の利用状況に基づいたクロック周波数の動的調節

プロセッサ内の資源で動的な調節が可能な箇所は

(1) 命令/データ Cache
(2) BTB(Branch Target Buffer)
(3) Issue Logic

等が存在し先行研究が多数発表されている。その中でも Issue Logic はスーパースカラプロセッサの並列度の上昇、パイプライン段数の増加に伴い消費電力に占める割合が増大し 35%に達する例も存在するため、Issue Logic の省電力化が強く求められている。

Issue Logic にはあらかじめ複数の命令をキャッシュから取得しておき、スケジューリングウィンドウに貯蔵し、命令間の依存関係を解決できた命令から実行ユニットに移動させ実行するという機構を備えている。これは命令レベル並列性を利用したプログラム実行の高速化をするためには不可欠ではあるが、同時に、Reservation Station,Re-Order Buffer といったハードウェア資源を必要とする。

これらの資源のサイズはプロセッサのピーク性能に合わせて決められるため、実行時間で平均すると大半のエントリが無効なエントリとなり、これらの存在は電力の過剰な消費を意味する。またキャッシュミス、分岐予測ミスが起こると、エントリが増大する。増加したエントリが先に実行され無い場合、これらは実行されるまで長くスケジューリングウィンドウに待機させられることになる。すると使用エントリ数が必要と

† 東京大学
University of Tokyo

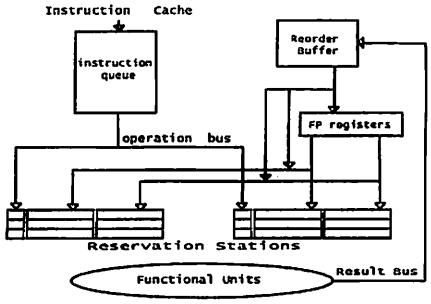


図1 Issue Logic の構成

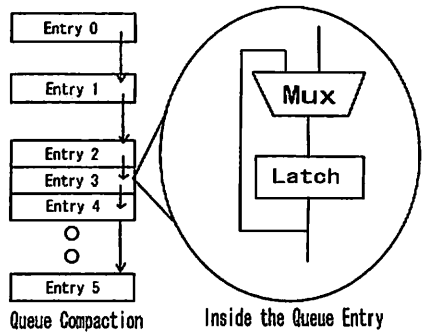


図2 スケジューリングウィンドウの構成

されている数よりも多く見積もられてしまうため、スケジューリングウィンドウのサイズ調整機構が有効に働かないという問題につながる。

本論文では、この問題を解決するため、命令 Fetch Gating とスケジューリングウィンドウサイズ調節による動作モデルを提案する。このモデルは遅延発生時に命令のフェッチを停止する機構を命令フェッチ部に追加し、さらに実行中の使用サイズからスケジューリングウィンドウサイズの調節を行う機構と組みあわせるものである。これによりスケジューリングウィンドウの要求サイズを低く保ち、より多くの不要エントリの削減を可能にする。

本稿の構成について説明する。以降第2章では Issue Logic における電力消費削減方法についての既存手法について概説し、第3章で本論文で実装している協調調節アルゴリズムについて解説する。第4章では実験とその結果について考察し、第5章に関連研究を挙げ、第6章でまとめている。

## 2. Issue Logic における電力削減

### 2.1 Issue Logic の構成と問題

まずは我々の提案手法が対象とする Out-Of-Order アーキテクチャの Issue Logic 部分について解説する。図1は典型的な Out-Of-Order アーキテクチャの Issue Logic を表している。

Issue Logic の流れを簡潔に説明すると

(1) 命令キャッシュからフェッチされた命令はスケジューリングウィンドウに蓄積される。
(2) Reservation Station では命令はそのオペランドがそろうまで待機させられる。
(3) 依存関係が解決しオペランドがそろった命令は各機能ユニットで実行される。

である。

### 2.2 サイズ調整可能スケジューリングウィンドウ

従来のスケジューリングウィンドウの構成を図3にあげる。すべてのエントリはラッチと比較演算、セレクタで構成された FIFO 構造になっている。命令をスケジューリングウィンドウから実行ユニット群に投入する度に、すべてのエントリがひとつずつ出口の方向にシフトする。結果としてすべてのエントリのラッチすべてに電力を供給せねばならず、シフトの数が大きくなるほど大量の電力消費を必要としてしまう問題がある。また Issue Logic の流れにおいて、スケジューリングウィンドウ のエントリはプログラムの実行中常にすべてが有効なエントリである事は無く、いくつかのエントリはただ電力を浪費するのみとなってしまう。

スケジューリングウィンドウの電力効率を改善する手法として Non-Compactic Latch- Based Issue Queue[10),11)] 等が存在する。Non-Compactic アプローチは Queue から出す時に FIFO のシフトを行わず、各エントリにソート用の通し番号を使い、それが古い順から Queue から出すものである。詳細は[11)] を参照されたい。ただしこのアプローチは $N$ 個のスケジューリングウィンドウのエントリが存在した場合、各エントリに $\log N$bit の長さの buffer を付加せねばならず、余分なテーブル参照が追加されるため、それ自身の消費電力と CPI の低下が消費電力の削減量と Trade Off となる問題がある。

もうひとつの改善手法として CAM/RAM based Issue Queue[9),10)] がある。従来の CAM 構造の Reservation Station による Select-Wakeup システムはスケーラブルではなく、クリティカルパスとなってしまうためクロック向上の妨げとなる。そこでオペランドの番号のみを CAM に格納し、他の命令情報を RAM に格納する事によって、Issue Logic の構造の簡略化と省電力化を達成することができる。

### 2.3 命令フェッチ動的調節

上記の Issue Logic の機構からすると、キャッシュミスなど命令実行に遅延が起こった場合、スケジューリングウィンドウから機能ユニットへの命令の流量が命令キャッシュからスケジューリングウィンドウへの命令の流量を下回ることになる、すると命令がスケジューリングウィンドウに停滞して無駄なエントリとなって

しまう。また分岐予測などが失敗した場合、それまでに投機的にフェッチされた命令はすべてフラッシュされてしまい、それまでにスケジューリングウィンドウに格納されたエントリも無効化されるためここにも無駄が生じる。また、その後実行される命令の待ち時間が長くなる。

命令のフェッチ量を動的に変更させることにより電力消費を削減する手法は Fetch Gating として既にいくつも提案されている。コンパイル段階で IPC を予測しそれにしたがってフェッチ量を調整する手法[3]、in-flight にできる命令の上限に達した場合フェッチを停止する手法[4] 等が存在する。

## 3. 方　　式

実行において何らかの遅延が生じたときに、遅延時のスケジューリングウィンドウのエントリが増加する。これが既存のスケジューリングウィンドウのサイズ調整効率を悪化させている。この問題を解決するため、我々はウィンドウのリサイズと命令のフェッチの一時停止させる事を協調して動作させる機構を提案する。性能低下、実装コストの増加による余分な電力の増加を避けるため、これらの機構は加算機、比較機などの少数の単純な回路によって実装される。このため既存手法にあるような予測的な振る舞いは不可能で、遅延が発生してから動作が始まる動的な調整機構になっている。

### 3.1 スケジューリングウィンドウサイズ調整機構

本研究で対象とするのは、大きなスケジューリングウィンドウを持つ Way 数の大きなスーパースカラアーキテクチャを想定した。また、さらに Queue をいくつかのブロックに分割し、使用率によってブロックの一部のクロック供給を調整できる機構を実装した。これによりプログラム実行の状況に応じた動的なスケジューリングウィンドウのサイズ変更が可能となった。

サイズ調節のアルゴリズムは以下のとおりである。

(1) ブロック 1 つあたりのエントリ数を B とする
(2) N サイクル周期ごとに周期内での最大エントリ数をとり M とする
(3) $M < B \times K$ となる最小の K を求め、次の周期は K 個のブロックのみを有効にする

我々の実装したサイズ可変なスケジューリングウィンドウの図 3 を以下に示す。1 ブロックのサイズを 16、全体で 64 エントリの構成のスケジューリングウィンドウを図示した。

サイズ調整時のデータの退避などによるオーバーヘッドを回避するため、ある周期での使用サイズを超える最小の数に調整する手法をとった。FIFO は従来の compaction をする Queue であることを仮定した。ブロック間、ブロック内部でも compaction は発生する。N と B を小さくすることによって、より細かい

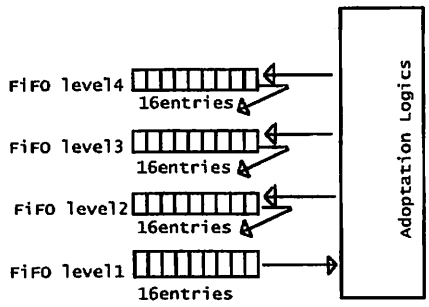


図 3　サイズ可変スケジューリングウィンドウの構成

粒度でのスケジューリングウィンドウの使用サイズ変化に適応することができるようになるため、理想的な電力効率に近づく。

### 3.2 フェッチ調節アルゴリズム

我々の実装したアルゴリズムは前章に挙げた既存の Fetch Gating とは違うアプローチをとっており、スケジューリングウィンドウのエントリ数と実行ユニットに発生した遅延を使用する。スケジューリングウィンドウエントリ数に閾値を設け、Fetch Gating をするかしないかを決めている。これはスケジュールウィンドウサイズが小さいうちに Fetch Gating をすることにより ILP を過剰に損なうことを避けるためである。スケジューリングウィンドウサイズの調節機構と同じく、アルゴリズム自体を最小限シンプルに押さえることでそれ自体が消費する電力消費の最小化を目的としている。

アルゴリズムのステップは以下のようになっている

(1) 毎クロックごとの commit された命令数を $c$ とし,issue された命令数を $d$ とする
(2) スケジューリングウィンドウの有効エントリ数が閾値 M 以上で、かつ $c < 2 \times d$　ならば次のクロックでのフェッチは停止される

このフェッチ調節アルゴリズムについて以下の図 4 に示す。

## 4. 評　　価

### 4.1 評価方法

提案手法を SimpleScalar シミュレータとその電力消費評価モデル Wattch[1] 上に実装した上で既存の手法を単体で実装した場合とで総電力消費量、実行サイクル数などを比較をした。評価に用いたベンチマークは SPEC CPU2000,95 の中から vpr,go など 6 個を選択した。シミュレータのパラメータは表 1 のとおりである。スケジューリングウィンドウサイズは 512 と

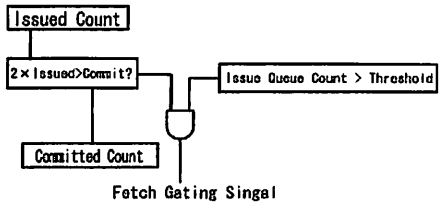


図 4 フェッチ調節アルゴリズムの構成

し、その他の各キューに関してはエントリ数は 2048 とした。また実装した機構についてもいくつかのパラメータの組を与え複数の実験を行っている。各ベンチマークを 2G 命令実行し、初期ミスなどによる影響の無い後半 1G の結果を採取した。

表 1 シミュレータ構成

| ISA | PISA |
|---|---|
| Branch predictor | Gshare: 256K entry |
| Branch mis-predict penalty | 80 cycles |
| Dispatch,Issue,Commit width | 8-ways |
| Re-Order Buffer | 2048 entries |
| Load Store Queue | 2048 entries |
| Maximum Issue Queue | 512 entries |
| L1 I-Cache | 64KB 8-way,3 cycles |
| L1 D-Cache | 64KB 4-ways,2 cycles |
| L2 Unified Cache | 4096KB 8-ways,40 cycles |
| Memory latency | 400 cycles |

### 4.2 評価結果

まず本アルゴリズムを導入することによる電力消費の変化と性能オーバーヘッドについて考察する。図 5 はスケジューリングウィンドウのリサイズを導入した場合と、さらに Fetch Gating と組み合わせた場合の総消費電力を各比較したものである。go では 5%,vpr では 10%の消費電力削減を下のに対し、perl では 1%そのほかでは 0.1%以下の削減にとどまった。go,vpr 以外のベンチマークで大きな消費電力削減ができなかった理由としては、

(1) 図 9 にあるように Fetch Gating が作動せずにスケジューリングウィンドウサイズが減らなかった (equake,ijpeg,mesa)
(2) 実行サイクルの増加による、他の部分の消費電力増加による相殺

が考えられる。

図 6 は図 5 と同様に実行サイクル数増加を比較したものである。equake,ijpeg,mesa では組み合わせたことによる実行サイクル数の増加は最大で 0.1%と無視できるほどだが、go,perl,vpr では 1%を超えるパフォーマンスの増加が見られた。go,perl,vpr は後に挙げるスケジューリングウィンドウの消費電力の評価等でよい結果を出していたため、このパフォーマンス低下は Fetch Gating が適切に働いた場合の副作用である。Fetch Gating が働くことにより一時的に投機実行が抑制され、命令実行のスループットが低下することによってこのパフォーマンス低下が現れたと考えられる。

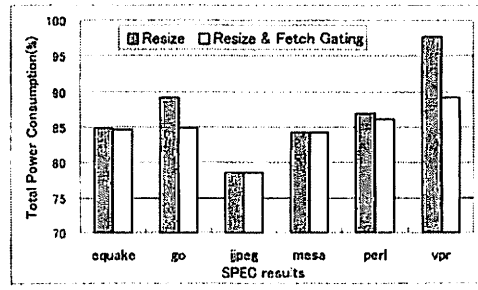


図 5 総電力消費比較

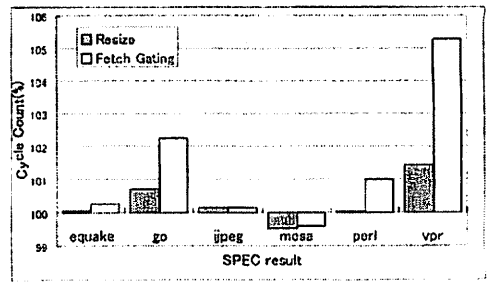


図 6 実行サイクル数比較

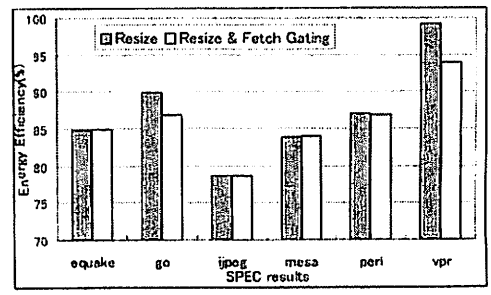


図 7 消費電力とサイクル数の評価

図 7 は Total 消費電力量と実行サイクル数に同じ重みをつけて掛け合わせ、これを電力効率として比較したものである。equake,ijpeg,mesa,perl においては二つを組み合わせた手法が約 0.1%リサイズのみを施したもの電力効率を下回る結果となったのに対し go,vpr では最大 5%電力効率を向上させる結果となった。perl においてはその性能オーバーヘッドの増加が 1%と大

きかったため、リサイズのみの電力効率を下回った。

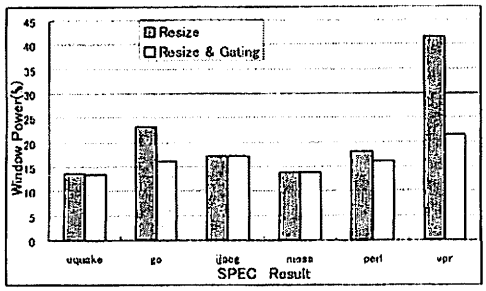


図 8 スケジューリングウィンドウの消費電力比較

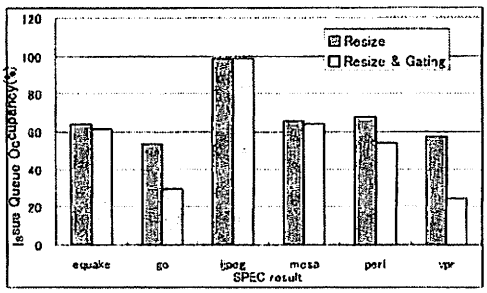


図 9 スケジューリングウィンドウの平均使用率比較

次にスケジューリングウィンドウに注目し Fetch Gating の効果を考察する。図 8 は各ベンチマーク結果におけるスケジューリングウィンドウの消費電力を比較したものである。go,vpr において Fetch Gating との組み合わせにより最大で組み合わせ前の 50%まで削減することができた。これはスケジューリングウィンドウの平均使用サイズが減少していることが原因であり、図 9 のスケジューリングウィンドウの平均サイズの比較のグラフにより確認できる。

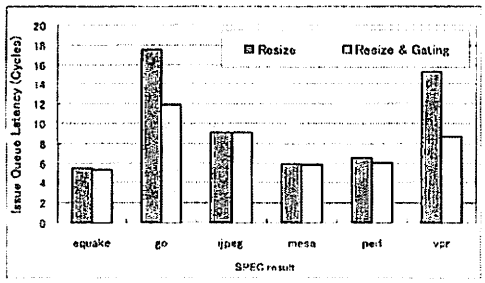


図 10 スケジューリングウィンドウの通過クロック数の比較

Issue Logic における不要なエントリとは、スケジューリングウィンドウに投入されてから実行ユニットに Issue されるまでのサイクル数が長いエントリである。

図 10 は各ベンチマークにおいて、命令がスケジューリングウィンドウの通過に要した平均サイクル数を比較したものである。電力消費削減効果があった go,vpr で平均通過サイクル数が削減されている事が見て取れる。

スケジューリングウィンドウの通過に要する時間の短縮化は、実行結果に関与しない命令を減少させる事で達成が可能である。これは通過に時間がかかるのは結果に関与しない投機的にフェッチされた命令が挿入されている事により、必要な命令がスケジューリングウィンドウでより長く待たされるのが原因だからである。

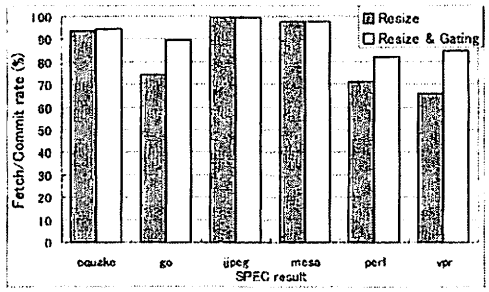


図 11 Commit/Fetch 比の比較

図 11 はフェッチされた命令数とコミットされた命令数の比を比較している。go,perl,vpr において Commit/Fetch 比の 10%以上の向上が見られる。Commit/Fetch が 1 に近づくほど不要命令の割合が少ない事を示すので、この 3 つにおいて不要命令の削減ができていることがわかる。

これらの結果から、go,vpr においては我々の提案手法はフェッチの動的調節により分岐予測ミスによるペナルティを低減させることによる、スケジューリングウィンドウの要求サイズの効率的削減に成功したといえる。

## 5. 関連研究

関連する研究は Fetch Gating に関するものと、スケジューリングウィンドウに関するものに分かれており、それぞれについて挙げる。Fetch Gating にはコンパイル段階で IPC を予測しそれにしたがってフェッチ量を調整する手法[3]、低信頼性だが高速な分岐予測機による間違った分岐の Fetch の回避[2] 等が存在する。コンパイラによる静的な予測はハードウェアへの実装の負担は無いがキャッシュミス、分岐ミスといった動的なイベントへの対処が不可能である。分岐予測機を二つ導入することによる Wrong Path のフェッチ

回避は分岐ミスペナルティの低減には成功するが、パイプラインフェーズの増加による命令実行のレイテンシ増加が無視できない。

スケジューリングウィンドウの調整では、ループ内で次第に最適な使用サイズに収束させる手法[7]等が存在する。この手法は映像を取り扱うようなベンチマークでは大変良好な結果を残したが、ループ内部での最適化のためループのサイズ等により、ベンチマークの種類により効果の差が激しい。

## 6. まとめ

本研究では実行時の遅延イベントから Fetch を調節する機構と動的調節可能なスケジューリングウィンドウを組み合わせて、効果的なスケジューリングウィンドウのリサイズを可能にする手法を提案した。その結果一部のベンチマークではスケジューリングウィンドウのみに省電力化モデルを適用したときと比較して最大 5%の電力効率の向上を達成した。よって動的な Fetch 調節がスケジューリングウィンドウの要求サイズ数削減に対し効果的であることを示した。しかしすべてのベンチマークプログラムにおいて良好な結果を示すことができたわけではなかった。スケジューリングウィンドウの無駄が少ないベンチマークでは効果を挙げることが不可能だった。逆に Fetch Gating により ILP が制限されパフォーマンスの低下を招いてしまった。今後の課題として我々のアルゴリズムを適用するかしないかを切り替えるなどの手段で、効果の出ないベンチマークについてのオーバーヘッドを軽減する事などが必要である。

現在プロセッサの進歩のアプローチはスーパースカラの複雑化、クロックの上昇からは離れている。しかし、コア単位でのパフォーマンス向上のためには、クロック上昇とそれに伴うパイプライン段数増加と同時実行する命令数の増加が不可欠である。だが、これらの資源に関わる無駄も増加するため、フロントエンドにおける電力消費の増加に直結する。今後プロセッサのコア単位での性能向上を目指すときには、今回我々の提案したようないくつかの調整機構が協調的に働くことによるハードウェア資源の節約が必須になると考えられる。

## 参考文献


[1] D. M. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimization. *In Proceedings of the 27th International Symposium on Computer Architecture* (2000)

[2] S. Manne, A. Klauser, and D. Grunwald. Pipeline Gating: Speculation Control for Energy Reduction. *In Proceedings of the 25th International Symposium on Computer Architecture* (1998)

[3] O. S. Unsal, I. Koren, C. M. Krishna, and C. A.Moritz. Cool-Fetch: Compiler-Enabled Power-Aware Fetch Throttling. *IEEE Computer Architecture Letters* (2002)

[4] T. Karkhanis, P. Bose, and J. E. Smith. Saving Energy with Just in Time Instruction Delivery. *In Proceedings of the International Symposium on Low Power Electronics and Design* (2002)

[5] S. Dropsho, A. Buyuktosunoglu, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, G. Semeraro, G. Magklis, and M. L. Scott. Integrating Adaptive On-Chip Storage Structures for Reduced Dynamic Power. *In Proceedings of the 11th International Conference on Parallel Architectures and Compilation Techniques*(2002)

[6] D. Folegnani and A. Gonzalez. Energy-Effective Issue Logic. *In Proceedings of the 28th International Symposium on Computer Architecture* (2001)

[7] Yu Bai and R. Iris Bahar Reducing Issue Queue Power for Multimedia Applications using a Feedback Control Algorithm *In Proceedings of the 22th International Conference on Computer Design* (2004)

[8] Timothy M. Jones, Michael F.P. O'Boyle,Jaume Abella and Antonio Gonzalez Software Directed Issue Queue Power Reduction *In Proceedings of the 11th International Symposium on High-Performance Computer Architecture* (2005)

[9] A. Buyuktosunoglu, S. Schuster, D. Brooks, P. Bose, P.Cook, D. Albonesi. An Adaptive Issue Queue for Reduced Power at High-Performance. *Springer-Verlag Lecture Notes in Computer Science Vol. 2008: 25-40* (2000)

[10] Alper Buyuktosunoglu,David H. Albonesi,Pradip Bose, Peter W. Cook,Stanley E. Schuster Tradeoffs in PowerEfficient Issue Queue Design *In Proceedings of ISLPED* (2002)

[11] J. Leenstra, J. Pille, A. Mueller, W. Sauer, R. Sautter, D.Wendel. A 1.8GHz Instruction Window Buffer. *In Proceedings of ISSCC* (2001)