Vol.2021-ARC-244 No.35
Vol.2021-SLDM-194 No.35
Vol.2021-EMB-56 No.35
2021/3/26

発行キューの電力削減のための発行幅制御方式

後岡 瑞希[1,a)]　安藤 秀樹[1]

概要：現在のプロセッサでは, 微細化に伴い進行速度が温度に指数的に依存する信頼性の低下が問題になっている．特に，ホットスポットと呼ばれる電力密度が高い箇所では高温となるため問題が発生しやすい．ホットスポットの 1 つに，発行キュー (IQ: issue queue) がある．1 サイクルに発行される命令数は，発行幅を埋めるほど安定して多くはない．このため，実際に発行される命令数に適応させて，回路のゲーティングにより発行幅を制限すれば，性能を大きく落とさずに消費電力を削減できる．
本論文では，発行幅を制限するための制御アルゴリズムを提案する．定期的に実行サイクル数を測定し，性能が低下しない最小の発行幅を探索することで，4.7% の平均性能の 低下で発行幅を 49.1% 削減することができた..

1. はじめに

現在のプロセッサでは，微細化に伴い信頼性低下が問題になっている [1]．その原因には様々なものがあるが，いずれもその進行速度が温度に指数関数的に依存しているため [2–5]，高温ではデバイスの寿命が短くなる．また，プロセッサ上にはホットスポットと呼ばれる単位面積あたりの消費電力が大きい場所が多くあり，そうでない場所に比べ温度の上昇が大きい．このため，前述の故障を引き起こしやすい．したがって，ホットスポットの消費電力は削減する必要がある．

ホットスポットを生成する回路の 1 つに発行キュー (IQ:issue queue) がある．現在のプロセッサでは性能を向上させるために IQ の発行幅が広くなる傾向があるが，プログラムの命令レベル並列性は常に発行幅を埋めるほど安定して大きくないため，使用される発行幅は平均的にはあまり大きくなく，一部の回路は有効に使用されていない．

本研究では，性能を大きく低下させることなく IQ の消費電力を削減するため，発行論理のゲーティングにより発行幅を削減し，消費電力を削減することを目的としている．

本論文では，このために発行幅の制御方法を提案する．この手法では，実行中に定期的に実行サイクル数を測定し，最大発行幅での性能をぎりぎり満たす最小の発行幅を探索することによって発行幅を決定する．

以下本論文の構成について述べる．2 章では関連研究について述べる．3 章では発行幅の削減を IQ の電力削減につなげるための回路のゲーティングについて述べ，4 章で発行幅の制御方法を提案する．5 章では評価の方法と結果について述べる．そして，6 章で本論文をまとめる．

2. 関連研究

IQ の発行幅を狭める類似の研究として，Homayoun らによる CAM や SRAM を使うユニットのサイズを動的に変更することで消費電力を削減する研究がある [6]．この研究では，L2 キャッシュミスのあとや複数の L1 キャッシュミスが発生している期間では，プロセッサの性能が低下し，性能を維持するために必要な発行幅が大幅に小さくなる一方，リオーダバッファ (ROB:reorder buffer) やレジスタの必要数が増加する観測に基づき，発行幅及びウェイクアップ幅と，ロード/ストアキュー (LSQ:load-store queue)，ROB，そしてレジスタファイルのサイズを動的に変える．これによりわずかな性能低下で大幅に消費電力を削減する．

近い研究として，Hu らによる実行ユニットの静的消費電力をパワーゲーティングにより削減する研究がある [7]．この研究では，パワーゲーティングのトリガとして発行命令数が 0 であったサイクルが一定期間続くこと，及び分岐予測ミスの発生を用いている．

Ponomarev らは，IQ，LSQ，ROB のサイズを動的に変更することにより消費電力を削減する手法を提案した [8]．この手法では，それぞれのキュー/バッファを複数のブロックに区切り，ブロック単位で容量の増減を行う．容量は必要最低限となるよう，使用されている容量を目標として制御を行う．具体的には，まず一定期間ごとに使用容量をサンプリングする．そして，一定のサンプル数ごとに使用容

[1] 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University
[a)] atooka@ando.nuee.nagoya-u.ac.jp

Vol.2021-ARC-244 No.35
Vol.2021-SLDM-194 No.35
Vol.2021-EMB-56 No.35
2021/3/26

量の平均をとり，設定に応じて以下の 1．または 2．のいずれかの方針で縮小を行う．

(1) 平均使用容量が現在の容量より 1 ブロック分以上小さいならば，1 ブロック分容量を縮小する．
(2) 平均使用容量以上の容量となる最小のブロック数まで縮小させる．

また，容量が埋まっているためエントリを割り当てることができなかったサイクル数を数え，そのサイクル数が一定周期内で閾値を超えた時に容量の拡大を行う．

Folegnani らは，発行論理の有効に働かない部分をゲーティングすることにより消費電力を削減する手法を提案した [9]．これは，IQ の空のエントリやすでにレディであるウェイクアップ論理のエントリを非活性化するとともに，一定時間ごとにキューの最終エントリに命令が入った回数に応じて IQ のサイズを変更する．

3. 背景

本節では，発行幅を削減することで IQ の電力を削減するための回路のゲーティングについて説明する．

3.1 回路のゲーティング

IQ の構成と回路は，基本的に文献 [10–12] に記載のものを仮定する．具体的には、ウェイクアップ回路は CAM 構成であり，セレクト回路はプレフィクス・サム構成である．

回路を以下のようにゲーティングする．

- ウェイクアップ論理をゲーティングするために、比較器にイネーブル入力を設ける．これが 1 の時は比較器のプリチャージを行い，比較を行う，0 の時は，プリチャージを行わず，比較も行わず 0(不一致) を出力する．
- 選択論理をゲーティングするために，プリフィクス・サムを構成する各加算器にイネーブル入力を設ける．制限された発行幅に応じて加算器から発行幅を超える発行許可信号が出力されないよう制御する．回路のプリチャージも行わない．

これらのゲーティングによって，動的・静的エネルギーを削減することができる．

3.1.1 ウェイクアップ論理のゲーティング

図 1 に、ゲーティングを可能とするウェイクアップ論理に用いるイネーブル付き比較器のシンボルを示す。論理は、前述したとおりであるが、式で書くと以下のようになる．

```
if (e)
  match = a == b;
else
  match = false;
```

図 2 にイネーブル付き比較器の回路を示す．通常の比較器と最も異なる点は，プリチャージ・トランジスタの制御にイネーブル信号が入力されている点である．イネーブル

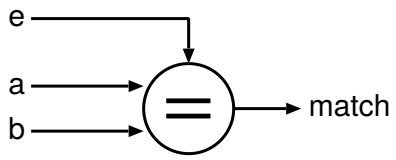


図 1 イネーブル付き比較器のシンボル

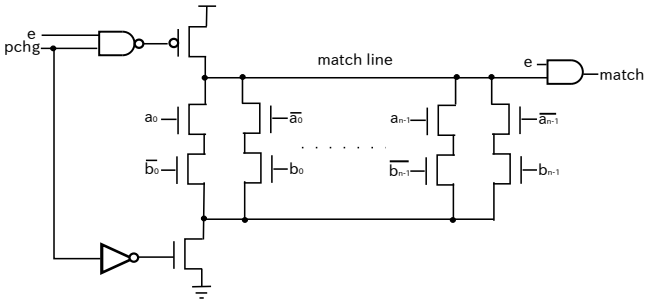


図 2 イネーブル付き比較器の回路

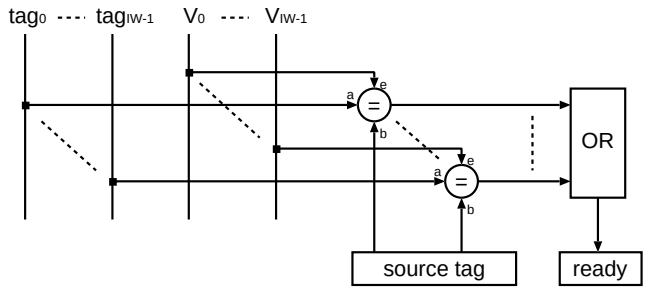


図 3 イネーブル付き比較器を使ったウェイクアップ論理のタグ比較回路

が H の場合，通常の比較器として動作するが，L の場合，マッチ線はプリチャージせず，L を出力する．L を出力するために，マッチ線の値がイネーブルが H のとき有効となるようその出力に AND ゲートを挿入している．

イネーブル付き比較器を使ったウェイクアップ論理のタグ比較回路を図 3 に示す．V_n を比較器のイネーブル端子に接続し，発行幅の制限に応じて設定する．現在の発行幅を iw に制限している場合、セレクト回路は iw 個を超える発行要求を許可しないので (3.1.2 節参照)，$V_0, \cdots, V_{iw-1} = 1$, $V_{iw}, \cdots, V_{IW-1} = 0$ とする．ここで，IW は，ハードウェアが保有している最大発行幅である．

3.1.2 選択論理のゲーティング

表 1 に，$IW = 4$ の場合の選択論理に使用する加算器の入出力の数のエンコーディングを示す．図 4 に，イネーブル付き加算器の入出力を表すシンボルを示す．イネーブル信号 $e_n = 1$ のとき，出力 $c_0 \cdots c_{n-1}$ が有効となり，$c_n \cdots c_{IW-1}$ が強制的に 0 となる．これを使い，定められた発行幅 iw について，加算器は和が iw 未満のとき，通常通りエンコードされた和を出力し，iw 以上の時、オール 0 を出力する．これにより，発行要求許可は高々 iw 個となることが保証される．表 2 に，発行幅とイネーブル信号の関係を示す．

Vol.2021-ARC-244 No.35
Vol.2021-SLDM-194 No.35
Vol.2021-EMB-56 No.35
2021/3/26

value	0	1	2	3	≥ 4
encoding	1000	0100	0010	0001	0000

表 1 選択論理に用いる加算器の入出力の数のエンコーディング ($IW = 4$).

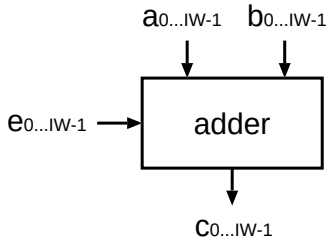


図 4 イネーブル付き加算器のシンボル

iw	e_0	e_1	e_2	e_3
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1

表 2 発行幅とイネーブル信号の関係 ($IW = 4$).

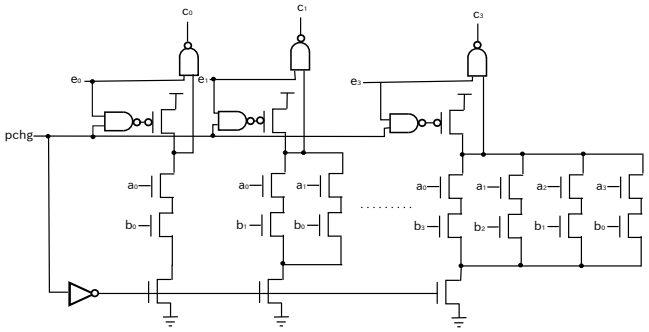


図 5 イネーブル付き加算器の回路 ($IW = 4$)

図 5 に，イネーブル付き加算器の回路図を示す．

4. 提案手法

本論文では，IQ の動作を削減するための手法として，その発行幅を制限するための制御手法を提案する．本節では，まず提案手法の概要を説明した後，その詳細を説明する．そして，基本方式の問題点を挙げ解決法を説明する．

4.1 提案方式の概要

本研究では性能をできるだけ低下させずに IQ の消費電力を低下させることを目的としている．そこで提案手法では，一定の命令数 (インターバルと呼ぶ) ごとに実行サイクル数を計測する．そして，最大発行幅で実行した場合に対するサイクル数の誤差 ER が予め定めた定数 ϵ に対して $|ER| < \epsilon$ となるような最小の発行幅 iw を探索し，発行幅を iw に制限する．

ここで，誤差 ER は発行幅 X での実行サイクル数を C_X，最大発行幅を IW として以下の式で与えられる．

$$ER = \frac{C_{iw} - C_{IW}}{C_{IW}} \quad (1)$$

4.2 基本方式

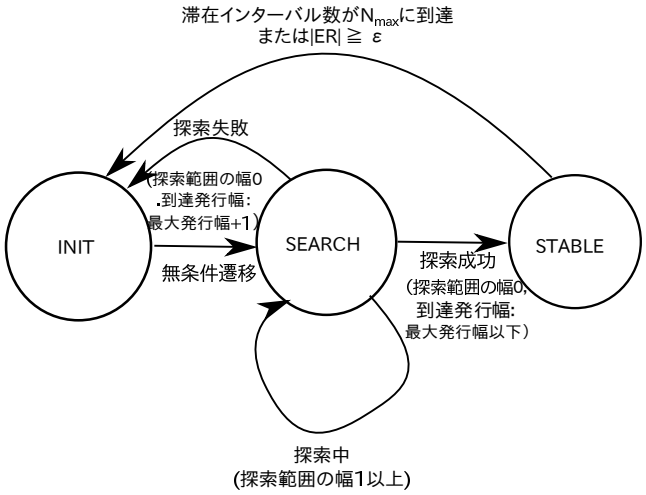


図 6 状態遷移

本節では，提案手法での基本動作について説明する．

提案手法での制御は，INIT，SEARCH，STABLE の 3 状態で表される有限状態機械を用いて行う．状態遷移を**図 6** に示す．まず状態 INIT において最大発行幅で実行し，その時のサイクル数を計測する．これを目標実行サイクル数とする．そして，状態 SEARCH へ移行する．状態 SEARCH で発行幅を変動させて $|ER| < \gamma$ を満たす最小の発行幅 iw を探す．見つかった場合は，状態 STABLE に移行し，以下のように発行幅を制御する．

- $|ER| < \epsilon$ の場合，状態が安定しているとし，何もせずに状態 STABLE にとどまる．
- $|ER| \geq \epsilon$ となった場合，誤差が大きいと判断し，状態 INIT に戻る．

また，状態 STABLE にとどまるインターバル数には，上限 N_{max} を設ける．この理由は，状態 INIT で目標実行サイクル数を取得したときと最大発行幅での性能が大きく変わっている可能性があり，新たな目標サイクル数を取得する必要があるためである．以下の節で，状態ごとに詳しく説明する．

4.2.1 INIT

この状態は，最大発行幅での実行サイクル数を取得するための状態である．この状態での実行サイクル数を目標実行サイクル数としている．最大発行幅で実行し，C_{IW} を得て，次のインターバルで状態 SEARCH に遷移する．

4.2.2 SEARCH

この状態は最大発行幅での実行時に十分近い性能となる最小の発行幅 iw を探すことを目的としている．状態 INIT で取得した最大発行幅での実行時の IPC（整数に切り下げる）を初期値とし，2 分探索により iw を見つける．

Vol.2021-ARC-244 No.35
Vol.2021-SLDM-194 No.35
Vol.2021-EMB-56 No.35
2021/3/26

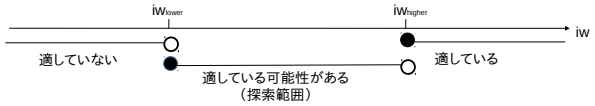


図 7　発行幅の探索範囲とその意味

図 7 に示すように，適している可能性のある最小の発行幅を iw_{lower}, 性能低下が十分小さいとわかっている最小の発行幅を iw_{higher} とし，iw_{higher} を候補としつつ $iw_{lower} \leq iw < iw_{higher}$ の範囲でより適した発行幅を探索する．

iw_{lower} と iw_{higher} の初期値を，以下のように定める．

- iw_{lower}：状態 INIT で取得した IPC の小数点以下を切り捨てた $max(\lfloor IPC \rfloor, 1)$．これを下限とする理由は，適した発行幅が IPC 未満であることはなく，発行幅を IPC より小さくすると必ず性能が低下するためである．
- iw_{higher}：最大発行幅 IW に 1 を加えた $IW + 1$. +1 としているのは，状態 INIT で実行した時と適した発行幅が変わってることにより，適した発行幅を発見できなかった場合にそれを検出するための番兵として働くためである．

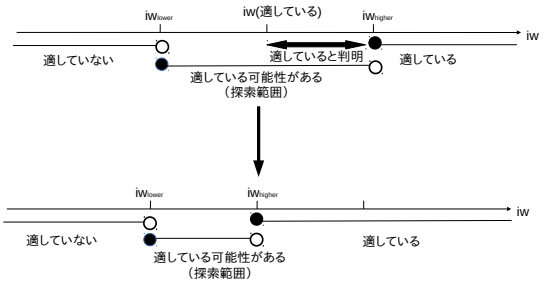
以下のように，2 分探索により最適な発行幅を探す．

まず，探索範囲の中間となる $\lfloor \frac{iw_{lower}+iw_{higher}}{2} \rfloor$ を発行幅 iw として実行する．そして，インターバルの終了時に ER を計算し，$|ER| < \gamma$ を満たしているかをチェックする．それに応じて以下のように iw_{lower} と iw_{higher} を更新する（図 8 参照）．

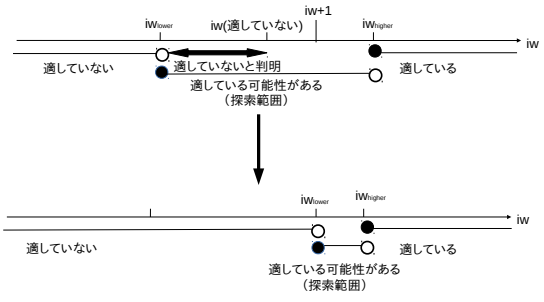
- $|ER| < \gamma$ の場合
  iw での性能低下が十分小さいと判断して，図 8(a) に示すように iw を新たな iw_{higher} とする．
- $|ER| \geq \gamma$ の場合
  iw が適した発行幅ではないと判断して，図 8(b) に示すように $iw+1$ を適している可能性のある最小の発行幅とする．すなわち，$iw+1$ を新たな iw_{lower} とする．

iw_{lower} と iw_{higher} の更新後，それらの値に応じて，以下のように次のインターバルの状態を決定する．

- $iw_{lower} \neq iw_{higher}$ の場合
  探索が完了していないため，状態 SEARCH にとどまり，前述したように iw_{lower} または iw_{higher} を更新する．
- $iw_{lower} = iw_{higher}$ の場合
  探索が完了したため，$iw_{lower}(= iw_{higher})$ の値に応じて以下のように状態を遷移する．
  - $iw_{lower} \leq IW$ の場合
    適した発行幅が見つかったと判断して，iw_{lower} を発行幅として状態 STABLE に遷移する．
  - $iw_{lower} > IW$ の場合
    適した発行幅が見つからなかったと判断して，状態 INIT に遷移する．



(a) 適した発行幅と判断した場合



(b) 適した発行幅でないと判断した場合

図 8　状態 SEARCH での発行幅の 2 分探索

4.2.3 STABLE

この状態では，最大発行幅での実行時に近い性能を保ちつつ低い発行幅を保つことを目的とする．このために，以下のように動作する．

- $|ER| < \epsilon$ である場合，発行幅が適切と判断して，状態 SEARCH にとどまる．
- $|ER| \geq \epsilon$ である場合，発行幅の制御を失敗していると判断して，状態 INIT に遷移する．
- 例外的な措置として， 状態 STABLE にとどまっているインターバル数が，閾値 N_{MAX} に達した場合，INIT で定めた目標性能が変化している可能性があるとして，INIT に戻る．

4.3 基本方式における問題

4.2 節で述べた基本方式を予備評価したところ，性能低下は小さく抑えられるが，発行幅削減率が小さいという問題があることがわかった．これは，以下の 2 つのことが原因と考えられる．

- 状態 STABLE での発行幅削減率が大きな性能低下を起こさない最小の固定発行幅で実行した場合ほど大きくできていない．
- IPC の短時間の大きな変動が頻発し，状態 STABLE にとどまることができない

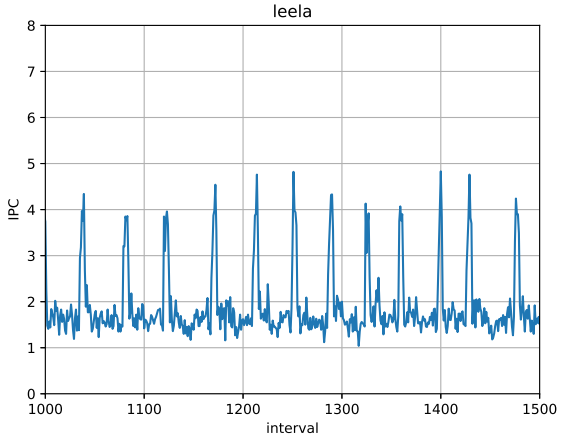
後者については以下の項で詳しく述べる．

Vol.2021-ARC-244 No.35
Vol.2021-SLDM-194 No.35
Vol.2021-EMB-56 No.35
2021/3/26

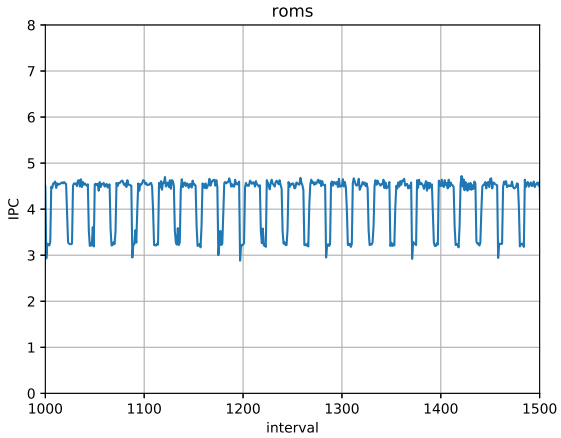
4.3.1 IPC の短時間の大きな変動による性能低下

インターバルごとの IPC の変動を計測したところ，一部のベンチマークで短い周期で一時的に大きな IPC の変動を起こしていた．高頻度で一時的な IPC 変動が明確に現れた leela と roms における IPC の変動を図 **9** に示す．この変動は 10 万命令単位の周期のループの一部の IPC が，他の部分と異なっていることによる．

このような一時的な変動に対して，発行幅制御が失敗していると判断し，すぐに状態 INIT に戻るのではなく，状態 STABLE にとどまり，変動が元に戻るまで待つことが有益である．基本方式では即座に状態 INIT に戻り最大発行幅で実行するので，発行幅削減率の低下を引き起こす．また，状態 SEARCH では適した発行幅の探索の過程で，適した発行幅が $max(\lfloor IPC_{INIT} \rfloor, 1)$ でない限り 1 度以上小さすぎる発行幅で実行するため，これにより性能低下も引き起こす．ここで，IPC_{INIT} は INIT での IPC である．



(a) leela



(b) roms

図 9 短期的な IPC の変動

4.4 基本方式における問題への対処

基本方式における発行幅を大きく削減できないという問題を解決するため，以下の 2 つの制御を導入した．

- 状態 STABLE において，投機的に発行幅を削減する
- 状態 STABLE において，誤差率が一定インターバル数連続してしきい値を超えた場合に状態 INIT に戻る．

以下でそれぞれについて述べる．

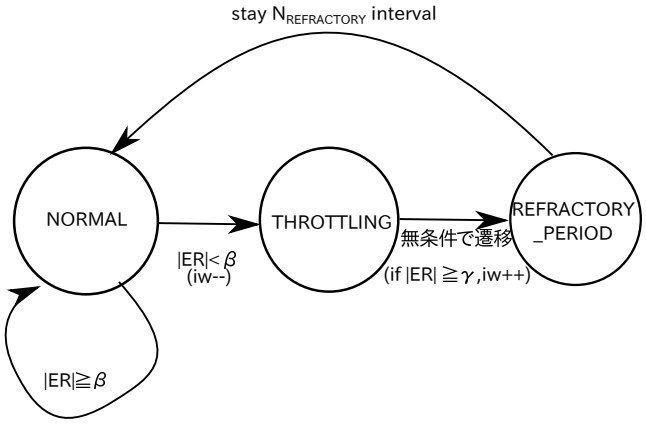


図 10 内部有限状態機械の状態遷移

4.4.1 投機発行幅削減

基本方式において，発行幅削減率を十分大きくするためには状態 STABLE での発行幅を削減する必要があるが，ER の値を用いて単純に削減しようとすると，以下の 2 つの要因により，状態 STABLE から INIT に遷移しやすくなってしまう．

- 必要最小限の発行幅より大きい発行幅で実行しても性能はほぼ変化せず，必要最小限の発行幅より大きい発行幅となっているかの判別が困難である
- 必要最低限の発行幅を下回ると性能が大きく低下し，状態 STABLE から外れる

これらの問題の発生を避けて IPC の低下を抑えつつ発行幅を削減するために，$|ER|$ が小さい場合に投機的に発行幅を削減し，性能低下が起こってしまった場合は発行幅を戻すという方法を導入する．

具体的な実装方法としては,$|ER|$ が十分小さい場合に投機的に発行幅の削減を試みるための内部的な有限状態機械を追加して発行幅を削減できるようにする．

この有限状態機械の状態は次のようなものである. 状態遷移を図 **10** に示す.

- NORMAL:平常状態．$|ER| < \beta$ なら誤差が小さく，発行幅を削減しても性能が低下しない可能性があると判断し，発行幅を 1 削減して状態 THROTTLING に移行する．
- THROTTLING:投機的に発行幅を削減している状態．$|ER| < \gamma$ なら投機的発行幅削減に成功したものとして発行幅をそのままとし，そうでないなら失敗したものとして発行幅を元に戻す．また，どちらの場合でも状態 REFRACTORY_PERIOD に移行する．
- REFRACTORY_RERIOD：発行幅削減をしばらく行

わない状態．この状態に入ってから規定のインターバル数が経過すると状態 NORMAL に遷移する．

なお，上位状態 STABLE の内部状態としての初期状態は REFRACTORY_RERIOD とした．これは，状態 SEARCH で探った適した発行幅からすぐに変更することが適していないと考えたためである．

4.4.2 状態 STABLE での一定インターバルの停留

4.3.1 節で述べたように，一時的な IPC 変動に対して基本方式では，状態 STABLE にとどまれば性能低下を抑えつつ発行幅削減ができたにもかかわらず状態 STABLE から状態 INIT に出てしまう．

そこで $|ER| \geq \epsilon$ となり誤差が大きくなってしまっていると判断しても，その状態であるインターバルが一定インターバル数連続しない限り一時的な変動によるものである可能性があると判断し状態 INIT に遷移せず状態 STABLE にとどまることとする．ただし，投機的発行幅削減を行っている場合であって，内部状態が THROTTLING の場合は上述した一定インターバル数には含めない．この状態は，投機的に発行幅を削減している状態であるため，適した発行幅とは必ずしも言えないからである．この制御は具体的には以下のようなものである．

$|ER| \geq \epsilon$ である場合，発行幅の制御が適切でない可能性があると判断して，このような誤差を持つ連続したインターバル数をカウントする．このカウント数に応じて以下の判断を行う．

- カウント数が閾値 N_{hold} 未満の場合，一時的な IPC 変動によるものである可能性があると判断して，状態 STABLE にとどまる．
- カウント数が閾値 N_{hold} 以上の場合，一時的な IPC 変動によるものではなく発行幅制御が適切でないと判断して，状態 INIT に遷移する．

5. 評価

5.1 評価環境

提案手法を SimpleScalar ver.3.0 [13] をベースに修正したシミュレータを用いて評価した．ベンチマークに使用した命令セットは Alpha である．ベンチマークには SPEC CPU 2017 を用いた．入力には refspeed データセットを用い，先頭の 16B 命令をスキップしてその後の 100M 命令を評価した．プロセッサ構成は Intel Skylake をベースとした構成とした．**表 3** にプロセッサ構成を示す．

5.2 評価パラメータ

許容できる性能低下の上限を 5%として，最適なパラメータを探索し，決定した．決定したパラメータを**表 4** に示す．

5.3 評価モデル

評価した 4 種類のプロセッサ・モデルを示す．

- **BASE**: 提案手法を用いないモデル
- **提案手法**: 提案手法を用いるモデル
- **FIXED_SEPARATE**: 固定した発行幅に制限するモデル．ただし，各ベンチマークについて，性能低下が 5%以下になる最小の発行幅に制限する．
- **FIXED_WHOLE**: 固定した発行幅に制限するモデル．ただし，全ベンチマーク平均での性能低下が 5%以下になる最小の発行幅である 4 に固定する．

5.4 評価結果

図 11 に各ベンチマークにおける最大発行幅での IPC に対する相対 IPC を，**図 12** に各ベンチマークにおける平均発行幅を，**図 14** にこれらをそれぞれの軸にとった散布図 (平均のみ) を示す．そして，**図 13** に各ベンチマークでの提案手法での実行時の各状態にとどまったサイクル数の割合を示す．

図 11 よりわかるように，提案手法では，cam4 や lbm では性能低下が 10%前後と大きくなっているものの，平均での性能低下は 4.8%と抑えられている．また，図 12 からわかるように，ほとんどのベンチマークで発行幅を約 4 にまで削減できている (平均削減率：49.5%).

FIXED_SEPARATE と比較すると，平均で IPC 低下は 2.1%に対して 4.8%とわずかに劣っているものの，発行幅削減率は FIXED_SEPARATE で 48.7% に対して 49.5%とほぼ同等である．また，図 12 より FIXED_SEPARATE に対して xalancbmk と cam4，そして pop2 では発行幅削減率が大きくなっており，leela や omnetpp では発行幅削減率が比較的小さくなっている．このことより，cam4 での大きな性能低下は制御によって適した発行幅に到達できず発行幅を小さくしすぎたことが原因といえる．また，lbm については発行幅の大きな差がなかったことから制御がうまくいかず発行幅が適した発行幅の上下にほぼ均等な割合で変動し，発行幅の削減なしに性能低下が大きくなっていると考えられる．また，比較的発行幅削減率が小さかった leela や omnetpp に関しては，図 13 を見ると状態 STABLE の割合が小さくなっており，安定して状態 STABLE にとどまることができずに制御によるオーバーヘッドが大きくなっていたためと考えられる．

また，FIXED_SEPARATE と比べて INT では性能低下が小さく，発行幅が大きくなっている場合が多く，FP では性能低下が大きく，発行幅が小さくなる場合が多いという傾向がある．この原因は判明していないものの，主に使用する機能ユニットの数とその種類の数が違うことにより発行幅を小さくした際の影響の出やすさが原因の 1 つとして考えられる．

また，FIXED_WHOLE と比較すると平均では発行幅削減率では FIXED_WHOLE の 50.0%に対して 49.5%とどちらもほぼ変わらず，IPC 低下も FIXED_WHOLE の 4.7%に

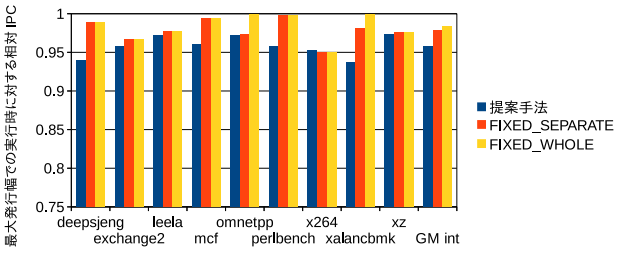
Pipeline width	8-instruction wide for each of fetch, decode, issue, and commit
Reorder buffer	224 entries
IQ	97 entries
Load/store queue	97 entries
Physical registers	224(int) + 224(fp)
Branch prediction	12-bit history 4K-entry PHT gshare, 2K-set 4-way BTB, 10-cycle misprediction penalty
Function unit	4 iALU, 2 iMULT/DIV, 2 Ld/St, 2 fpALU, 2 fpMULT
L1 I-cache	32KB, 8-way, 64B line
L1 D-cache	32KB, 8-way, 64B line, 2 ports, 2-cycle hit latency, non-blocking
L2 cache	2MB, 16-way, 64B line, 12-cycle hit latency
Main memory	300-cycle min. latency, 8B/cycle bandwidth
Prefetch	stream-based, 32-stream tracked, 16-line distance, 2-line degree, prefetch to L2 cache

表 3 プロセッサの構成

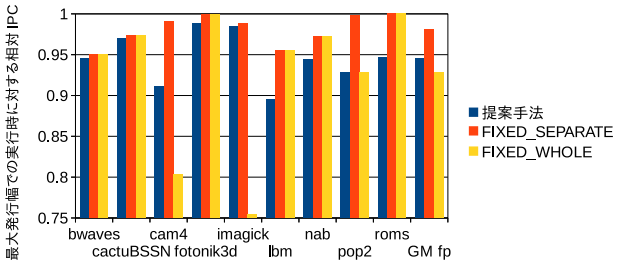
ϵ	状態 STABLE に発行幅を変動させずとどまる性能の誤差率の絶対値の上限	0.15
γ	状態 SEARCH で性能の誤差率が十分小さいとみなす誤差率の絶対値の上限	0.13
β	状態 STABLE で誤差が十分小さいとし投機的発行幅削減をする性能の誤差率の絶対値の上限	0.01
インターバルの命令数		10000
N_{MAX}	状態 STABLE にとどまる最大のインターバル数	1000
$N_{REFRACTORY}$	状態 STABLE での投機的実行を停止するインターバル数	10
N_{hold}	ER が ϵ で区切られた範囲から外れても状態 STABLE にとどまる連続したインターバル数の上限	5

表 4 パラメータ構成

対し 4.8%と変わらない．しかし，IPC 低下率が最大のベンチマークでは imagic の 24.6%に対し，lbm の 10.5%であり，大きく勝っている．このことから，FIXED_WHOLE では一部のベンチマークで大きく性能を低下させてしまう一方，提案手法ではオーバーヘッドこそあるものの事前情報なしに適切な発行幅を選択できていると言える．

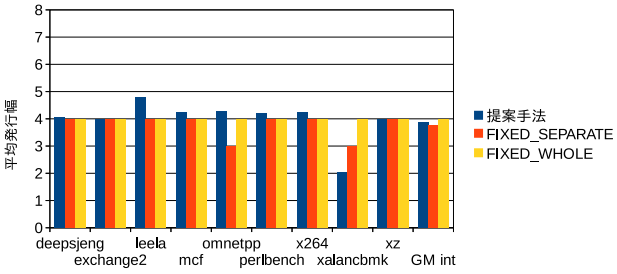


(a) INT

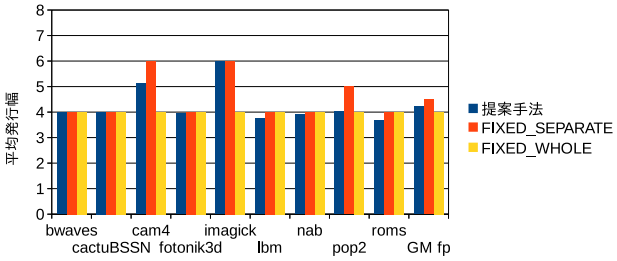


(b) FP

図 11 発行幅を削減しなかった場合に対する相対 IPC



(a) INT



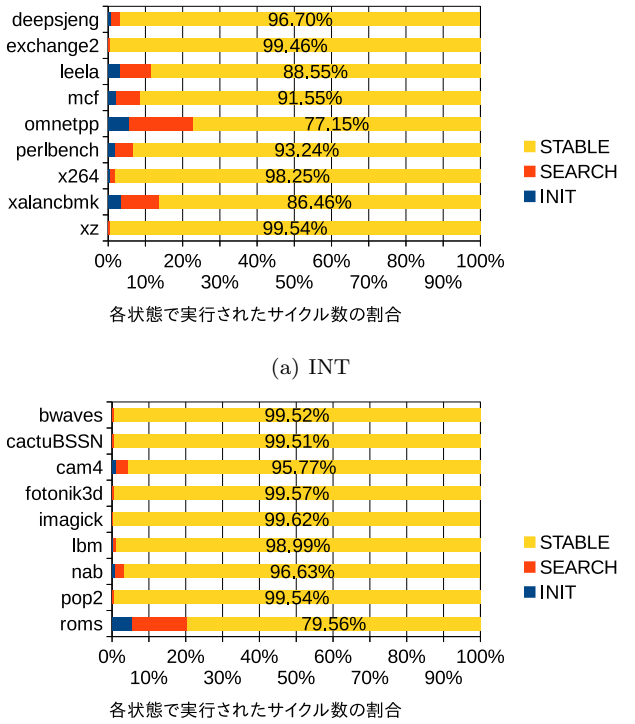
(b) FP

図 12 最大発行幅の平均

5.5 各改善手法の効果の評価

本節では各改善手法の効果を評価する．

Vol.2021-ARC-244 No.35
Vol.2021-SLDM-194 No.35
Vol.2021-EMB-56 No.35
2021/3/26



(a) INT

(b) FP

図 13 提案手法での各状態にとどまったサイクル数の割合

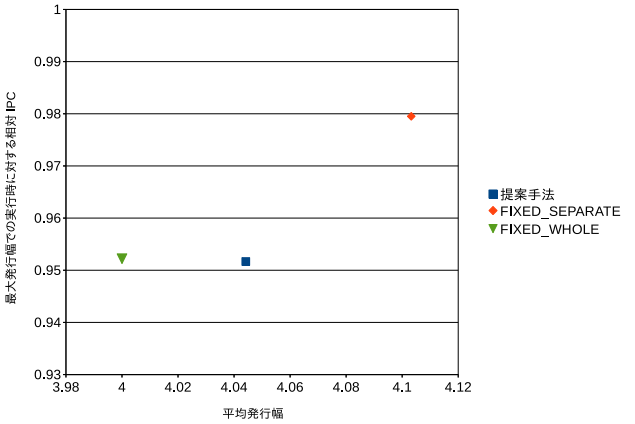


図 14 発行幅を削減しなかった場合に対する相対 IPC と最大発行幅の平均の散布図

以下の 4 モデルを評価した．

- **NONE**: 提案手法においてどの改善手法も用いないモデル
- **HOLD**: 状態 STABLE での一定インターバルの停留のみを用いるモデル
- **SPEC_REDUCTION**: 投機的発行幅削減のみを用いるモデル
- **FULL**: 改善手法を両方用いるモデル

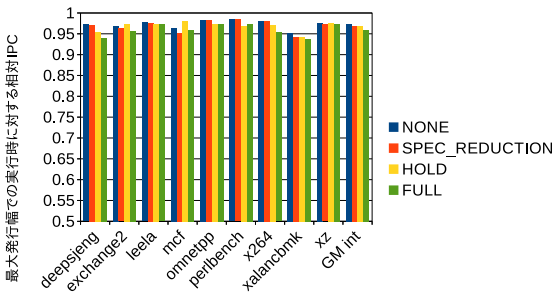
評価する各モデルについての最大発行幅での実行時に対する相対 IPC を図 **15** に，平均発行幅を図 **16** に，これら 2 つの関係を示す散布図を図 **18** に示す．また，各モデルそれぞれの各状態で実行したサイクル数の割合の平均を図 **17** に示す．

2 つの改善手法のうち，状態 STABLE での一定インターバルの停留を行うようにした HOLD では，NONE と比べて平均の発行幅削減率は 43.8%と変わらないものの性能低下は 4.8%から 3.0%と大きく抑えられている．これは図 17 よりわかるように状態 STABLE の割合が高くなっているからである．目的通り短い周期で状態 STABLE を離れることを防いでおり，それにより性能低下を抑えていると考えられる．また，これにより NONE では 31.4%と非常に大きな性能低下を起こしていた lbm についても 2.9%まで大幅に性能低下が抑えられている．また，最大の性能低下 (xalancbmk) も 5.7%に抑えられている．なお，4.3.1 節で短時間の大きな変動が明確に現れたとして例示した leela で，NONE でも性能低下を起こしておらず，HOLD での性能改善も起きていないのは，IPC の短時間の上昇が起こっていない時に合わせて発行幅が削減され，短時間の上昇が起こらなくなったことが原因と考えられる．

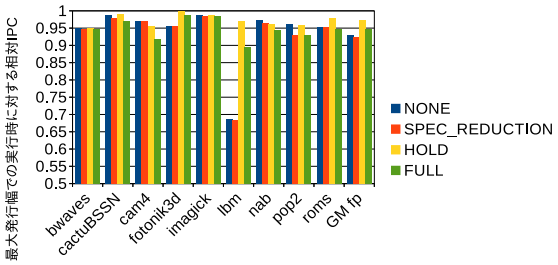
一方，投機的発行幅削減を行った SPEC_REDUCTION では，NONE と比べて発行幅削減率が 44.7%とほぼ向上していないにもかかわらず，性能低下は 5.3%と増加してしまっており，成功していない．ただし，同様に投機的発行幅削減を行っている FULL においては，HOLD と比べて性能低下が 3.0%から 4.7%と大きくなるものの発行幅削減率は 43.8%から 49.1%と拡大できている．この改善の理由は，FULL では，状態 STABLE に長くとどまることにあると思われる．図 17 よりわかるように HOLD や FULL では状態 STABLE の割合が高く長い間状態 STABLE にとどまっているが，NONE や SPEC_REDUCTION では STABLE の割合が低く，INIT の 10 倍程度しかないため短い周期で離れてしまっていることがわかる．

このことから，投機的発行幅削減は発行幅の削減につながるものの，状態 STABLE にとどまる長さが短いと削減の効果が得られる前に状態 STABLE を離れてしまうことになり，その結果，SPEC_REDUCTION では発行幅削減率が増加せず，性能低下というデメリットのみが現れてしまったと考えられる．

Vol.2021-ARC-244 No.35
Vol.2021-SLDM-194 No.35
Vol.2021-EMB-56 No.35
2021/3/26

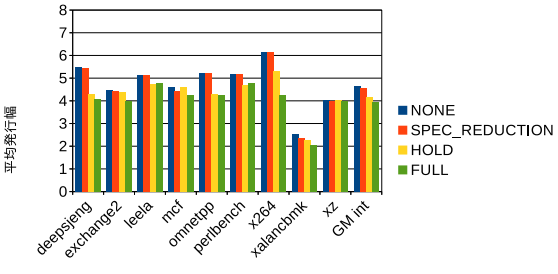


(a) INT

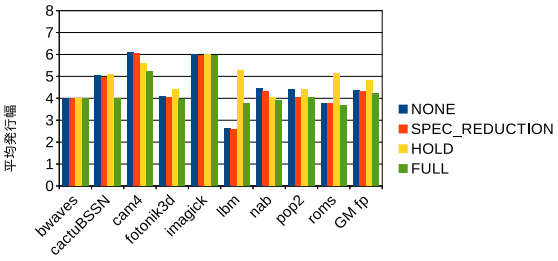


(b) FP

図 15 改善手法の適用段階ごとの最大発行幅での実行時に対する相対 IPC



(a) INT



(b) FP

図 16 改善手法の適用段階ごとの平均発行幅

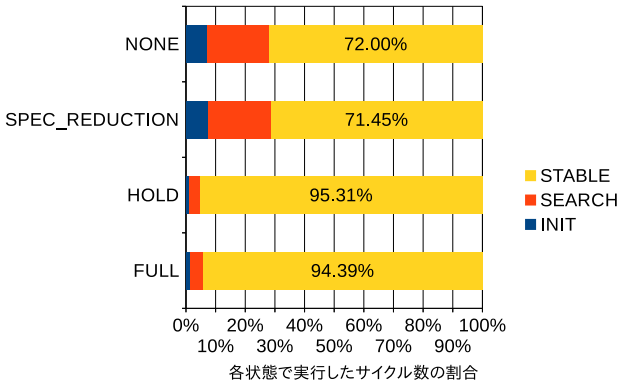


図 17 改善手法の適用段階ごとの各状態にとどまったサイクル数の割合

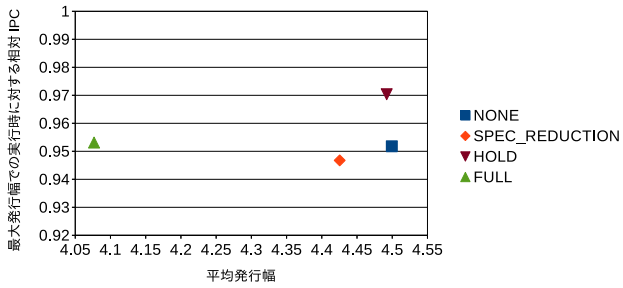


図 18 改善手法の適用についての平均発行幅と相対 IPC の関係

6. まとめ

LSI は，温度が高いほど劣化が起きやすくなる．電力密度が高い場所は，周辺より温度が高くなるため，故障を起こしやすくなる．このため，そのような場所の消費電力を削減する必要がある．

マイクロプロセッサにおいて，このような回路の 1 つに，IQ がある．現在のプロセッサでは性能を向上させるために IQ の発行幅が大きくなる傾向があるが，プログラムの命令レベル並列性は常に発行幅を埋めるほど安定して大きくないため，使用される発行幅は平均的にはあまり大きくない．そのため，発行幅を制限し，使用しない回路をゲーティングすることにより消費電力を削減できる．本研究では，そのための方法として，性能を大きく低下させることなく発行幅を制限するための制御法について提案した．

SimpleScalar をベースとしたシミュレータによって，提案手法による IPC への影響と発行幅の削減率を評価した．その結果，提案手法によって SPEC CPU 2017 ベンチマークにおいて平均 4.7%の性能低下で発行幅を 49.1%削減することができた．

参考文献

[1] Weste, N. H. E. and Harris, D. M.: *CMOS VLSI Design: A Circuits and Systems Perspective, fourth edition*, Addition Wesley (2010).
[2] Monsieur, F., Vincent, E., Roy, D., Bruyre, S.,

Pananakakis, G. and Ghibaudo, G.: Time to breakdown and voltage to breakdown modeling for ultra-thin oxides (Tox<32Å), *Proceedings of the 2001 IEEE International Integrated Reliability Workshop*, pp. 20–25, (2001).

[3] Khan, S. and Hamdioui, S.: Temperature Dependence of NBTI Induced Delay, *Proceedings of the 2010 IEEE 16th International On-Line Testing Symposium*, pp. 15–20 (2010).

[4] Black, J.: Electromigration–A brief survey and some recent results, *IEEE Transactions on Electron Devices*, Vol. ED-16, No. 4, pp. 338–347. (1969).

[5] Viswanath, R., Wakharkar, V., Watwe, A. and Lebonheur, V.: Thermal performance challenges from silicon to systems, *Intel Technology Journal*, Vol. 4, No. 3, p. 116 (2000).

[6] Homayoun, H., Sasan, A., Gaudiot, J. and Veidenbaum, A.: Reducing Power in All Major CAM and SRAM-Based Processor Units via Centralized, Dynamic Resource Size Management, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 19, No. 11, pp. 2081–2094 (2011).

[7] Hu, Z., Buyuktosunoglu, A., Srinivasan, V., Zyuban, V., Jacobson, H. and Bose, P.: Microarchitectural techniques for power gating of execution units, *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, pp. 32–37 (2004).

[8] Ponomarev, D., Kucuk, G. and Ghose, K.: Reducing Power Requirements of Instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources, *Proceedings of the 34th Annual ACM/IEEE International Symposium on Microarchitecture*, pp. 90–101 (2001).

[9] Folegnani, D. and González, A.: Energy-effective Issue Logic, *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pp. 230–239 (2001).

[10] Yamaguchi, K., Kora, Y. and Ando, H.: Evaluation of Issue Queue Delay: Banking Tag RAM and Identifying Correct Critical Path, *Proceedings of the 29th International Conference on Computer Design*, pp. 313–319 (2011).

[11] Yamaguchi, K., Kora, Y. and Ando, H.: Delay Evaluation of Issue Queue in Superscalar Processors with Banking Tag RAM and Correct Critical Path Identification, *IEICE Transactions on Information and Systems*, Vol. E95-D, No. 9, pp. 2235–2246 (2012).

[12] Goshima, M.: Research on High-Speed Instruction Scheduling Logic for Out-of-Order ILP Processor, PhD Thesis, Kyoto University (2004).

[13] http://www.simplescalar.com/.