名古屋大学大学院工学研究科博士前期課程
修士論文

ランダム発行キューにおける
タグ2段階比較による電力削減

令和 2年 3月
情報・通信工学専攻

松田康誉

概要

現在のスーパスカラプロセッサの発行キュー (IQ：Issue Queue) の消費エネルギーのうち，ウェイクアップ論理が占める割合は大きい．そしてその最大の原因はタグ比較の際の消費エネルギーである．

これに対してこれまで，タグ比較を 2 段階にすることで消費エネルギーを削減する手法が提案されていた．さらにタグ 2 段階比較によって IPC が低下する可能性が高い古い命令を保持するエントリのタグ比較を 1 段化する，先頭 1 段化方式が提案されていた．しかしこの IPC 低下抑制方式は，そのままでは現在の IQ の方式であるランダム・キューに対応できない．そこで本研究では，タグの 2 段階比較をランダム・キューに対応させるために，IQ 内の命令の古さを記録するプログラム・オーダー・キューを用いた方式を提案する．

本研究では，128 エントリ発行幅 6 の IQ について提案手法と従来手法のウェイクアップ論理の消費エネルギーを測定し，提案手法の従来手法に対するエネルギー削減率を評価した．その結果，ウェイクアップ論理の消費エネルギー削減率は 37.5%，IQ 全体では 6.8%であった．また IPC 低下は平均 0.72%に抑えられた．

目 次

第 1 章　はじめに

現在のプロセッサは，単位面積当たりの電力が大きい場所を多数持っている．そのような場所ではそうでない場所に比べ温度上昇が著しく，過渡故障や永久故障の原因となる [1]. このためそのような電力を場所を持つ回路は，電力を低下させる必要がある．

発行論理の構成要素であるウェイクアップ論理も，そのような電力消費が大きい場所である．文献 [2] によれば，ウェイクアップ論理の消費エネルギーはプロセッサ・コア全体の 16.3% を占めるとされている．

ウェイクアップ論理の消費エネルギーが大きい最大の要因は，タグ比較をするダイナミック・ロジックの回路がエネルギーを大きく消費するからである．ダイナミック・ロジック回路はスタティック・ロジック回路に比べて高速化と回路面積の低減が可能であるが，ウェイクアップ論理で使用される比較器ではタグ比較の結果が不一致の場合，プリチャージされた電荷がグラウンドに流れる．ウェイクアップ論理のタグ比較ではそのほとんどが不一致となるため，多くのエネルギーが消費される．

これに対し本研究では，**タグの 2 段階比較**によってエネルギー削減を試みる．この方法ではタグ比較を行う CAM 回路として，基本的に，2 段階で動作させる回路 [3] を使用している．この回路を使い，タグの下位ビットをまず先に比較し，一致した場合に限って上位ビットの比較も行う．1 段階目のタグ比較で一致した時のみ 2 段階目が動作するので，1 段階目でタグ比較の結果が不一致の際に消費エネルギーを削減できる．一方でタグ比較を 2 段階にすると，IQ の動作は 1 サイクルで終了しないので，依存する命令を連続するサイクルに発行できなくなり IPC が低下する．この IPC を抑制する方法として，これまで IPC に悪影響を与えると推測される少数のエントリのタグ比較を従来通り 1 段階で行うという方法が取られてきた [4].

しかしこの IPC 低下を抑制する方法についての研究は，IQ の方式として，現在では使用されていない方式を仮定しており，現在の**ランダム・キュー**と呼ばれる方式に対応できない．具体的には，古い方式では命令が IQ の中でプログラム順に並んでいたが，ランダム・キューではそうではないので，古い方式と同様の効果を期待できない．

そこで本研究では，**プログラム・オーダー・キュー (PQ : Program order Queue) 方式**という IPC 低下抑制策を提案する．この方式では，IQ 内の命令の古さを記録しておき，最も以前にディスパッチされたいくつかの命令のタグ比較を 1 段化する．

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 IQ を構成する各回路について述べる．4 章ではタグ 2 段階比較の動作とその回路について述べる．5 章ではこれまでに研究されてきたタグ 2 段階比較による IPC 低下の抑制手法を述べる．6 章では提案手法である PQ 方式による IPC 低下の抑制手法を述べる．7 章では測定に用いる回路の構成を述べる．8 章ではエネルギーの計算方法などの評価方法を述べる．9 章では遅延，IPC，エネルギーについての評価結果を述べる．10 章で本研究をまとめる．

第 2章　関連研究

最初に IQ に関連する研究について述べ，次にタグの 2 段階比較に関する研究について述べる．

2.1　IQ についての研究

Palacharla らは，命令発行幅と IQ のサイズを変化させた時の，ウェイクアップ論理と選択論理の遅延を評価した [5]．また，遅延を小さくするために，IQ を複数の FIFO バッファで構成し，依存する命令を同じ FIFO バッファに割り当てる依存ベースの IQ を提案した．この手法では，各バッファの先頭の命令のみ発行可能かチェックすれば良いので，回路が単純化され遅延が減少する．

Stark らは，IPC をほとんど低下させずに，ウェイクアップ論理と選択論理をパイプライン化する手法を提案した [6]．この手法では，投機的にウェイクアップを行うことで，依存する命令を連続するサイクルで発行できるようにした．

Michaud らは，IQ をシフト・レジスタと CAM で構成された発行バッファに分けることで，複雑さを低減する手法を提案した [7]．この手法では，デコード・ステージで命令の発行タイミングを予測し，発行までの遅延に応じてシフト・レジスタに挿入する．そして，シフト・レジスタより順に発行バッファに命令が送られ，発行バッファより命令は発行される．予測が十分正確であれば，発行バッファのサイズは発行幅まで近づけられる．

Folegnani らは，空のエントリの比較器や既にレディなオペランドを持つ比較器など，タグを比較する必要がない比較器を動作させないことで，消費エネルギーを削減する手法を提案した [2]．

Ponomarev らは，リソース要求に応じて IQ のサイズをリサイズすることで，消費エネ

ルギーを削減する手法を提案した [8].

Ernst らは，IQ に入ってくる命令のうちのほとんどが，はじめから少なくとも 1 つのソース・オペランドがレディであると指摘した [9]．そして IQ に，2 つのソース・オペランドを保持できるエントリに加えて，1 つのソース・オペランドのみ保持できるエントリと，ソース・オペランドを保持しないエントリを用意し，レディでないソース・オペランドの数に応じていずれかにディスパッチする手法を提案した．さらにこの手法を実現するために，命令の 2 つのオペランドの内，あとにレディになるオペランドを予測する手法も提案した．

五島らは，ウェイクアップ論理を従来の CAM ではなく，依存行列と呼ぶ RAM で構成する手法を提案した [10]．これによって比較器を用いずに依存する命令をウェイクアップすることが可能で，ウェイクアップの遅延を短縮できる．

Brown らは，発行する命令の選択を省略した IQ を提案した [11]．これは，命令がウェイクアップされたら，すべて投機的に即時発行する．発行された命令が実際に選択されたかどうかは後で検証する．これにより，ウェイクアップ論理と選択論理からなるクリティカルなループから選択論理が排除され，遅延を短縮できる．

Sassone らは，依存行列の遅延と電力をより小さくするための手法を提案した [12]．具体的には，従来はすべての命令について，その古さを完全に追跡していたのに対して，命令をグループ化してグループ単位で古いものを選択する．これにより，性能低下を最小限に抑えながら，回路の規模を小さくできる．

Lebeck らは．キャッシュ・ミスするロードのような長いレイテンシの命令に依存する命令を，IQ とは別の待機用バッファに入れ，その長いレイテンシの処理が完了するまで IQ に挿入しないという方式を提案した [13]．これによって，IQ が待機する命令で埋ることによって起こるストールの頻度が減り，性能が向上する．

Raasch らは，IQ をいくつかのセグメントに分割する方式を提案した [14]．この方式では，各命令の依存命令チェーンのレイテンシを元に割り当てるセグメントが決定される．そして，発行可能になる直前に最下位セグメントである発行バッファに命令を移動する．こ

の発行バッファでのみ発行を行うことで，すべてのエントリから発行できる通常の IQ と比較して遅延を短縮できる．

Brekelbaum らは，大きな低速のキューと小さいな高速のキューを使い分ける手法を提案した [15]．大きな低速のキューにはレイテンシが性能にあまり影響しない命令を入れる．低速のキューであるタイミングまでにレディにならなかった命令は，クリティカル命令として小さなキューに移される．小さなキューに入れられたクリティカルな命令は，高速に発行される．

Kim らは，レイテンシが互いに 1 サイクルの依存関係のある 2 つの命令をグループ化し，1 つの命令として IQ のエントリでスケジューリングすることで，依存グラフのエッジのレイテンシ短縮とキューの容量効率を上げる手法を提案した [16].

Gibson らは，依存する命令をポインタでつなぎ，ポインタをたどることでウェイクアップを行う手法を提案した [17]．この方式により CAM が不要になり，電力を削減できる．

Homayoun らは，キャッシュ・ミスの処理中に発行幅を半減させることで，IQ の消費電力を削減する手法を提案した [18]．発行幅半減中に元の発行幅の半分以上の命令が発行される場合，一時的にその命令を小さなバッファに移動させることで対応している．

2.2 タグの 2 段階比較についての研究

CAM による 2 段階比較を用いた研究は，数多く存在する [3] [19] [20]．しかし，発行キューのウェイクアップ論理のタグ比較に 2 段階比較を用いる手法は，先行研究である小林らの研究 [4] 以外では存在しない．小林らの研究を元にしたタグ 2 段階比較については，4 章で説明する．

第 3 章　IQ

本章では，研究対象である IQ の役割や構成，方式について述べる．

3.1　IQ の役割と構成

IQ は，リネームされた命令を保持し，実行順序をスケジューリングして，機能ユニットに発行するための回路である．実行順序のスケジューリングでは，命令間の依存関係を管理し，他の命令との依存関係が解消された命令のみに発行許可を出すという方法で行っている．この一連の動作を，ウェイクアップ論理，選択論理，タグ RAM，ペイロード RAM の 4 つの回路で行っている．以下各回路の役割と動作について述べる．

ウェイクアップ論理は命令間の依存関係を管理し，他の命令との依存関係が解消された命令に対して発行要求を出す回路である．各エントリは，対応する命令の 2 つのソース・オペランドに対応するタグ (ソース・タグ) と，それらの状態を表すレディ・フラグを保持している．CAM 回路で構成される比較器でソース・タグと発行された命令のディスティネーション・タグを比較し，一致すればレディ・フラグをセットする．2 つのフラグがセットされ，依存が解決したならば，発行要求信号を選択論理へ送る．

選択論理は，資源制約を考慮して，発行を要求された命令の中からそれを許可する命令を選択し，発行許可信号を出力する回路である．この選択においては，回路構成の単純化のために IQ の先頭エントリの命令をより優先する．

タグ RAM は発行待機中の命令のディスティネーション・タグを保持する回路で，選択論理から発行許可信号が送られると，対応する命令のタグを読み出し，それをウェイクアップ論理へ送る．

ペイロード RAM は発行待機中の命令のコードを保持する回路で，選択論理から発行許

可信号が送られると，対応する命令のコードを機能ユニットに送信する．

以上の 4 つの回路の他に，本研究ではエイジ論理という回路を用いる．これは，選択論理と並列に動作し，発行要求が出された命令の中で最も古い 1 命令を選ぶ．

3.2 IQの方式

一般に，レディな命令のうち，より古い命令を優先的に発行すれば性能はより高くなることが広く知られている．IQ 内で命令をプログラム順に並べておけば，レディな命令から発行する命令を選ぶ選択論理としては，単純な回路で，古い命令に高い優先度を与えるように実現できる．

そのような IQ としては主に，シフト・キューとサーキュラ・キューがある．シフト・キュー [21] は，命令を発行したエントリの空きを詰めるコンパクションを行うことで，高い容量効率を達成できるが，コンパクションには大きな電力を要し，回路が複雑になる．一方，サーキュラ・キューはコンパクションをしないので電力消費が小さくなる．しかし，空いたエントリの分だけ実効的な容量が低下する欠点がある．

コンパクションをせず，実行容量を低下させない方式として，ランダム・キューがある．ランダム・キューは，空いている任意のエントリに命令を挿入する．具体的には，IQ の空きエントリのインデックスを保持するフリーリストを用意し，ディスパッチ時にはフリーリストを参照して，空いたエントリに命令を挿入する．命令が発行されて IQ が無効化されると，そのインデックスをフリーリストに返す．

このようにランダム・キューは IQ の容量を無駄にすることがないため，Alpha 21464 [22]，AMD Bulldozer [23]，IBM POWER8 [24] といったプロセッサに用いられている．本研究でも，このランダム・キューを研究対象の IQ の方式とする．

第 4章　タグ2段階比較

本章では，ウェイクアップ論理におけるタグ 2 段階比較方式について述べる．

4.1　回路の概要と動作タイミング

図 4.1 に，タグ 2 段階比較の回路構成の概略図を示す．タグ 2 段階比較方式では，タグをある決められた比で低位ビットと高位ビットに分割する．低位ビットの比較が先に行われ，これが不一致ならば，高位ビットの比較は行われない．低位ビットの比較で一致した場合，高位ビットの比較が行われ，一致した場合のみタグ比較が一致したことになる．低位ビットの比較が不一致ならば，高位ビット比較器はプリチャージされないので，これはエネルギーを消費しない．

このように，タグ 2 段階比較は消費エネルギーを削減できるが，タグ比較を 2 段階にすることで遅延が大きくなりクロック・サイクル時間が増加する．それを防ぐために，2 段階比較の場合は IQ の動作を 2 サイクルで行う．従来の 1 段階比較と本研究の 2 段階比較の動作タイミングを図 4.2 に示す．通常，IQ の動作 (ウェイクアップ (WU)，選択 (SL)，タグ RAM 読み出し (TR)) は 1 サイクルで完了するが，2 段階比較方式ではウェイクアップに 1 サイクルをかけ，次サイクルで選択とタグ RAM 読み出しを行う．これらの動作は 1 サイクルを 3 分割したクロック ϕ_0〜ϕ_2 を使って制御される．以下では 2 段階比較タグ比較の動作について各フェーズごとに述べる．

まず ϕ_0 が立ち上がると，タグ RAM よりディスティネーション・タグが全エントリに出力される (図 4.1 の ⓪)．同時に低位ビットの比較器はプリチャージされる (①)．ϕ_0 が立ち下がると，低位ビットの比較器でタグ比較が行われる．

次に ϕ_1 が立ち上がると，低位ビット比較器でタグが一致した場合のみ高位ビットの比較

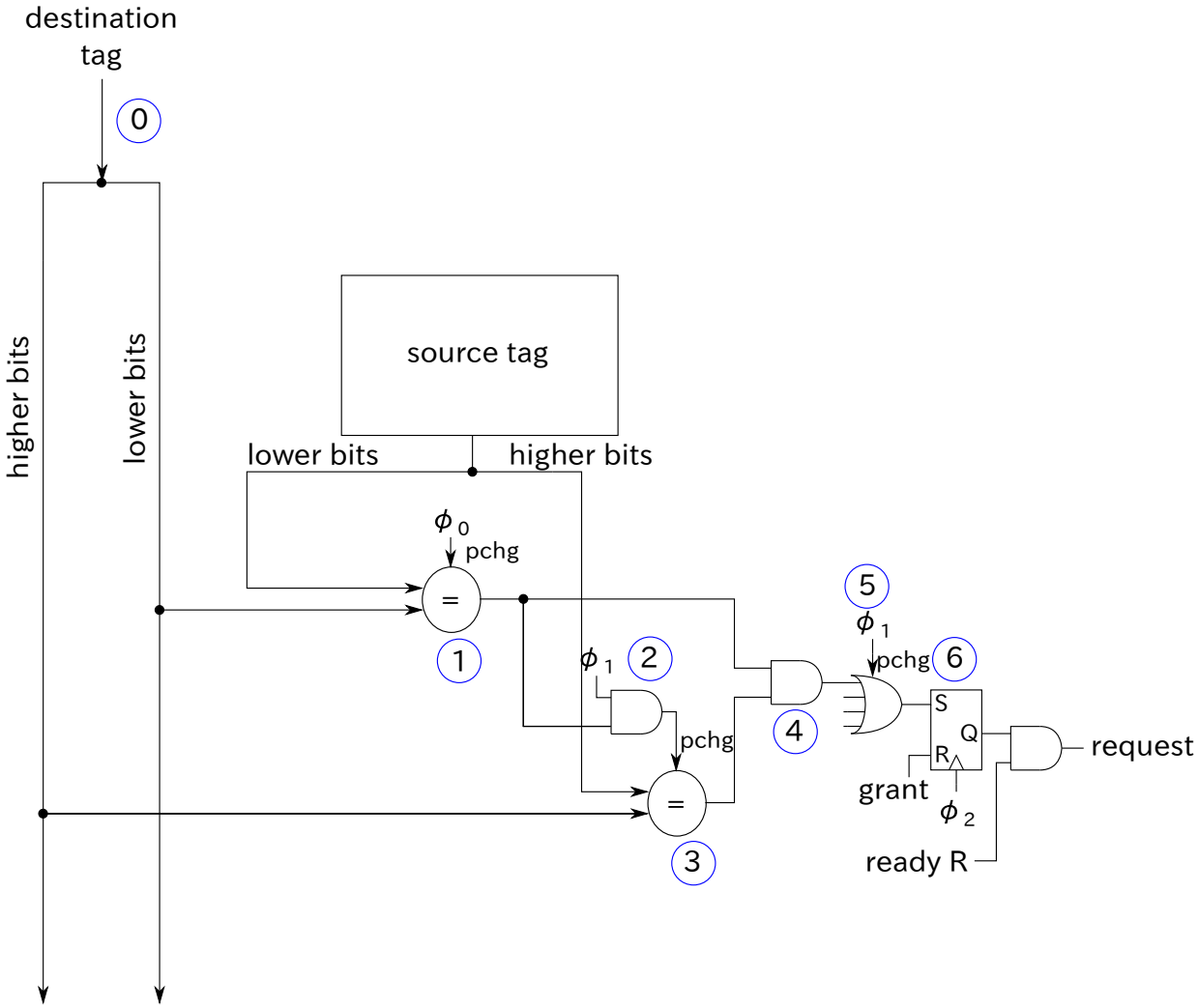


図 4.1: タグ比較を 2 段階で行うウェイクアップ論理の概略図

器がプリチャージされる (③)．つまり，高位ビット比較器のプリチャージ信号は低位ビット比較器のマッチ線によってゲーティングされており (②)，低位ビットのタグが不一致なら高位ビットの比較器はプリチャージされない．また，各マッチ線を入力とするダイナミック OR(マッチ OR) もプリチャージされる (⑤)．ϕ_1 が立ち下がると，高位ビットの比較器でタグ比較が行われる．そして，低位ビットのマッチ線と高位ビットのマッチ線の AND をとり (④)，マッチ OR に入力される．放送されたタグの内，1 つでも一致するタグがあればマッチ OR は H を出力する．

最後に ϕ_2 が立ち上がると，SR ラッチによってレディ・ビットが更新される (⑥)．

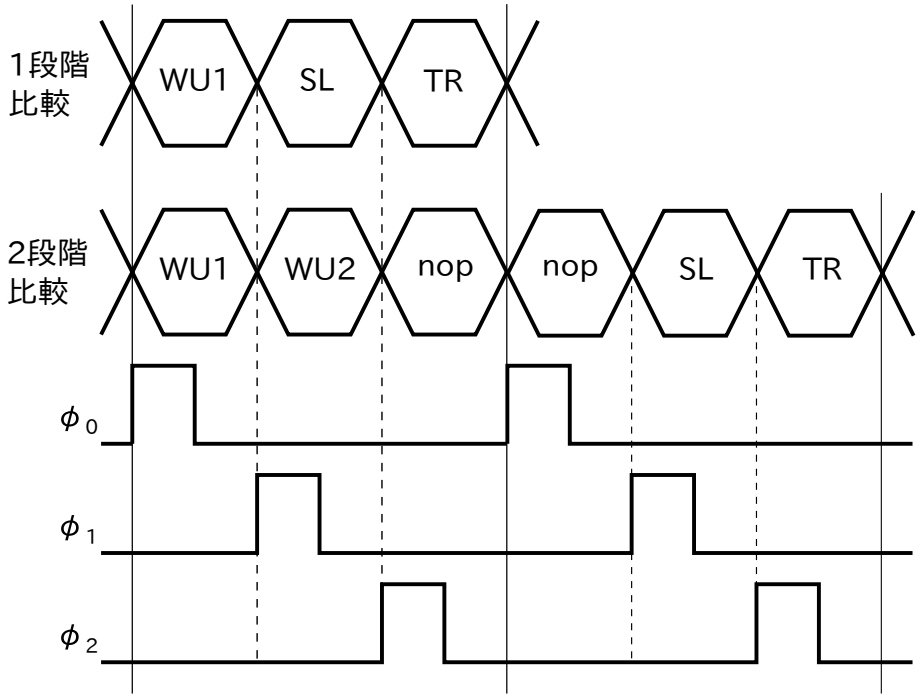


図 4.2: タグ比較の動作タイミングとクロック

4.2 タグ比較 1 段化の回路

ウェイクアップ論理のタグの 2 段階比較は消費エネルギーを削減できるが，タグが一致した場合にウェイクアップに 1 サイクルを要するため，IPC が低下する．そこで本研究では，実行時間に影響を与えるクリティカル・パスの上にあると推測される命令のエントリでのみ，従来通り 1 段階でタグ比較を行う．

どのエントリのタグ比較を 1 段階にするかについては次章以降で述べる．ここでは 1 段階比較と 2 段階比較の切り替え方について述べる．

図 4.1 において，タグ比較を 1 段階に切り替えた際に変更が必要な箇所は以下の点である．

1. 高位ビットの比較器のプリチャージのクロックを ϕ_0 とする (③).
2. マッチ OR のプリチャージのクロックを ϕ_0 とする (⑤).
3. レディ・ビットを保持する SR-FF のクロックを ϕ_1 とする (⑥).

これらの制御信号の切り替えは，トランスミッション・ゲートを用いた MUX によって行われる．

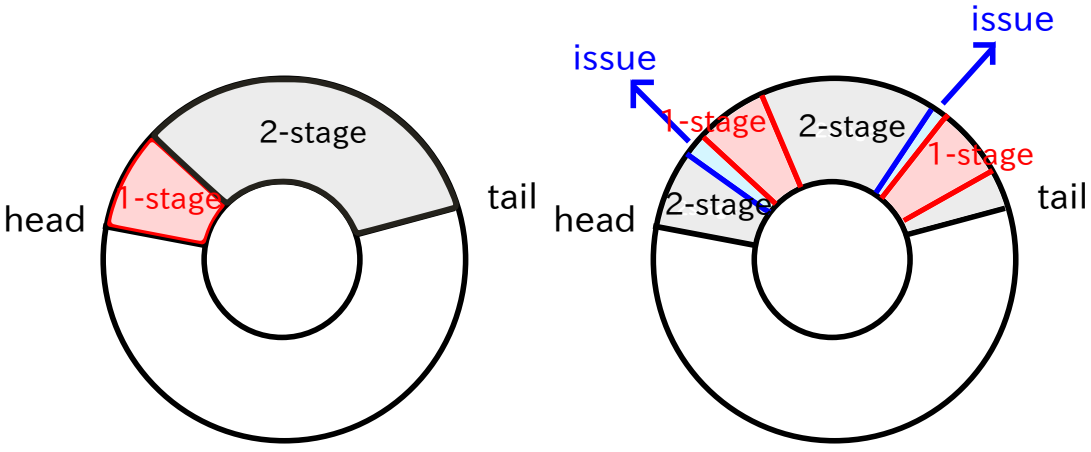
第 5 章　これまでの IPC 低下抑制手法

小林らは，サーキュラ・キューにおいて，IPC をほとんど低下させずにタグの 2 段階比較によってエネルギーを削減する手法を提案した [4]．具体的には，先頭 1 段化方式と後方 1 段化方式という，2 種類の IPC 低下抑制方式を提案した．図 5.1 にその概略図を示す．これらの方式では，IPC に悪影響を与える可能性の高い少数のエントリのみ従来通り 1 段階でタグ比較を行う．

以下では先頭 1 段化方式及び後方 1 段化方式について個別に詳しく説明する．

5.1　先頭 1 段化方式

ある命令がクリティカル・パス上にある場合，データフローの後方には多くの命令があるはずである．そのため，このような命令は IQ に長く滞在する．したがって，IQ の中の命令が古ければ古いほど，クリティカル・パス上にある可能性が高いと推測される．このようなクリティカル・パス上の命令は，ウェイクアップに要する遅延を小さくして，実行



(a) 先頭 1 段化方式　　(b) 後方 1 段化方式

図 5.1: サーキュラ・キューにおける IPC 低下を抑える手法

時間に影響を与えないようにする必要がある．

そこでこの方式では，図 5.1(a) に示すように，IQ の先頭の一定数のエントリのタグ比較を 1 段階にする．1 段化したエントリではエネルギーは削減できないが，他の多くのエントリで 2 段階比較を行うことで全体としてはエネルギーを削減できる．

5.2 後方 1 段化方式

多くの場合，ある命令が依存する命令はプログラム順で近い位置にある．したがって，ある命令が発行された時，そのすぐ後方の命令は依存関係が解消されて次に発行可能になる可能性が高いと推測される．

そこでこの方式では，図 5.1(b) に示すように，発行された命令の後方の命令一定数を 1 段化する．先頭 1 段化方式と同様，他の多くのエントリでは 2 段階比較を行うので，全体としてはエネルギーを削減できる．

5.3 問題点

小林らのタグ 2 段階比較及びその IPC 低下抑制の方式は有効であったが，サーキュラ・キュー自体に大きな問題がある．それは，発行されて空きとなったエントリに命令がディスパッチされず容量効率が悪いため，サーキュラ・キューが現在のプロセッサには使われていないことである．

そこで本研究では，小林らの研究をベースとして，現在のプロセッサで使用されているランダム・キューに対応した IPC 低下抑制方式を提案する．小林らの評価では，先頭 1 段化方式のほうが後方 1 段化方式よりも消費エネルギーと性能の両面で高い効果を示していたので，本研究では先頭 1 段化方式を適用する．

第 6 章　PQ 方式

本章では，提案手法であるプログラム・オーダー・キュー (PQ:Program Order Queue) 方式について述べる．

6.1　概要

5 章で述べた先頭 1 段化方式では，古い命令のタグ比較を 1 段化するため，IQ 内の命令の古さがわからなければならない．しかし，ランダム・キューではプログラム順に命令が並んでいないので，命令の古さがわからない．そこで本研究では，PQ という IQ のエントリのインデックスをプログラム順に保持する FIFO バッファを用いて，ランダム・キューの先頭 1 段化方式への適応を可能にする．

6.2　PQ の動作

PQ では，命令が IQ にディスパッチされた時に，そのディスパッチされた IQ のエントリのインデックスを末尾に挿入する．命令が発行された時には，その IQ のエントリのインデックスが書き込まれていた PQ のエントリも無効化される．そして，ウェイクアップのタグ比較時には，PQ の先頭の読み出しポート数分のエントリを読み，読み出したインデックスに対応する IQ のエントリのタグ比較を 1 段化する．PQ を読み出したら，次に読み出す有効なエントリまでヘッドポインタを移動させる．

本研究では，常に 1 段化するエントリの数と PQ の読み出しポート数を一致させる．そのために次のようにする．まず，1 段化しているエントリ数を表すカウンタを用意する．そして，カウンタの値が読み出しポート数と一致するまで，PQ の先頭から有効エントリを読み出して 1 段化する．1 段化されていたエントリが発行されたら，カウンタの値をダウ

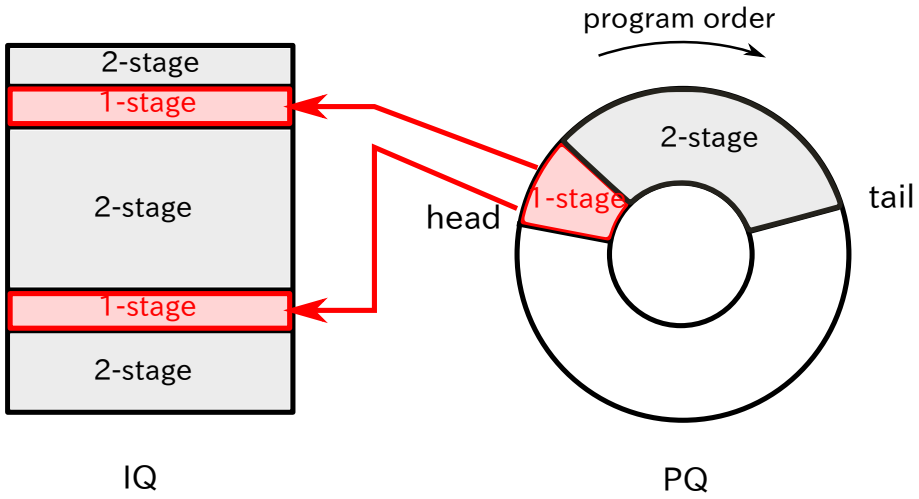


図 6.1: PQ によるタグ比較 1 段化 (読み出しポート数が 2 の場合)

ンし，その後また読み出しポート数と一致するまで PQ を読み出す (つまり，1 段化されているエントリ数と読み出しポート数が一致している間は，PQ の読み出しはしない).

図 6.1 に PQ によるタグ比較 1 段化の概略図を示す.

6.3 PQ の容量

PQ はできるだけ小さな消費エネルギーで動作させたいため，先頭と末尾の間のエントリが無効化されても圧縮しない．このため，PQ に ”穴” が空き，容量効率が低下するので，PQ は IQ のサイズよりも大きくする必要がある．それでももし PQ の容量が足りず，新たに挿入するエントリがなくなってしまった場合は，2 通りの方法が考えられる．ひとつは PQ にディスパッチできるエントリができるまで，ディスパッチをストールさせる方法である．もうひとつは PQ に IQ のインデックスを挿入できなかった命令はすべて 2 段階比較する方法である．PQ のサイズや PQ の容量不足時の対応策の違いによる性能，消費エネルギーの違いは，9 章で評価する.

第 7章　測定に用いた回路の構成

本章では，遅延とエネルギーの評価で用いた各回路の設計やそのクリティカル・パスについて述べる．

7.1　IQ全体のクリティカル・パス

図 7.1 に提案手法の 1 段階比較と 2 段階比較の切り替え可能な IQ での各回路のタイミング図を示す．各回路の部位の名前の意味については以下のとおりである．

- broadcast(b.c.) to WU : ディスティネーション・タグがウェイクアップ論理の最上位から放送されて，一番下のエントリにタグが到達するまでの時間.
- LWU，HWU，WU : それぞれ順に，ウェイクアップ論理における 2 段階比較の低位ビットの評価期間，2 段階比較の高位ビットの評価期間，1 段階比較の評価期間.
- SR latch : タグ比較後，レディ・フラグを保持する SR ラッチがセットされ，リクエスト信号を生成する AND よりリクエスト信号が生成されるまでの時間.
- SL : 選択論理が動作している時間.
- TR : タグ RAM が動作している時間.
- 各回路の precharge はそれぞれのプリチャージ期間.

IQ のクリティカル・パスは図 7.1 の 1 段階比較の時のパスである．2 段階比較時の高位ビットのプリチャージと評価の時間は，それより遅延の大きい選択論理の動作と並列に行われるためクリティカル・パスにはならない.

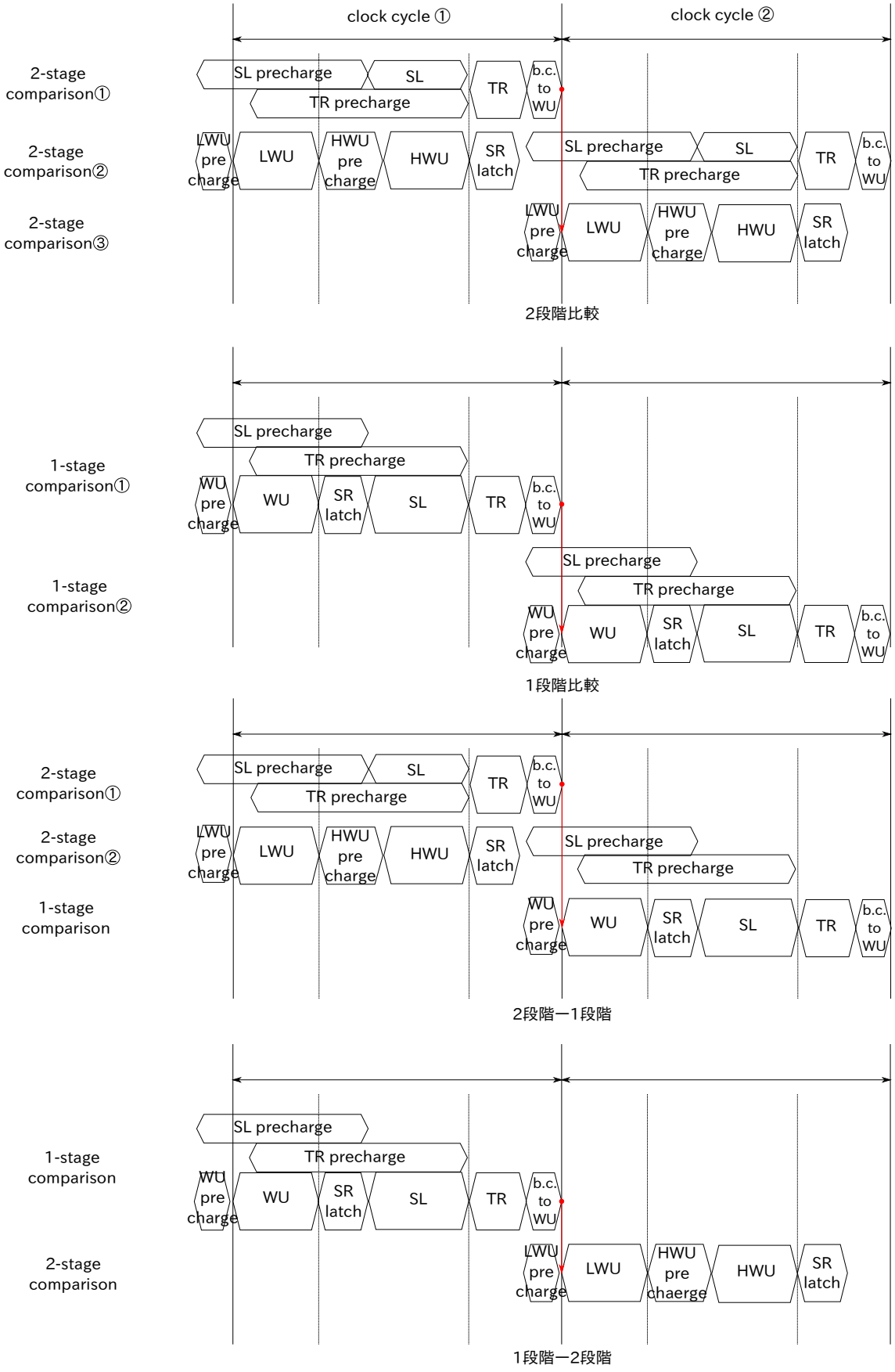


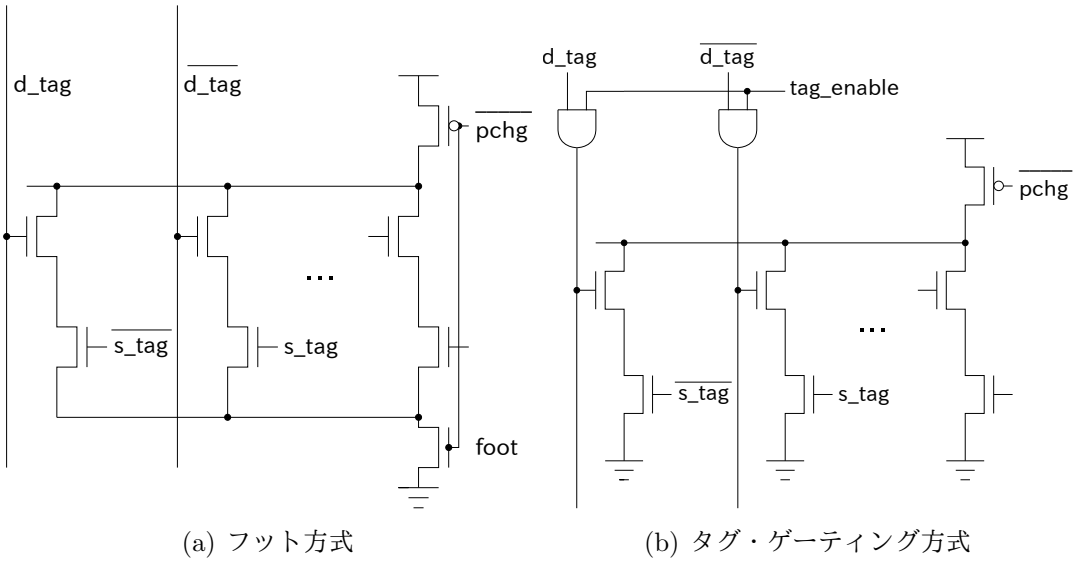
図 7.1: IQ の各回路の動作タイミング

7.2 ウェイクアップ論理

7.2.1 設計方式

ウェイクアップ論理では，タグRAMから送られてきたデスティネーション･タグと，ウェイクアップ論理が保持するソース・タグとを比較する．そのため，ウェイクアップ論理はCAMで構成される．

CAMの比較器はダイナミック・ロジックで構成されるため，毎サイクルマッチ線をプリチャージしてから比較が行われる．プリチャージ中はそのチャージされた電荷が抜けないようにする必要があるが，その方法として2種類の方法が考えられる．それらの方法を図7.2に示す．一つは図7.2(a)のフット方式である．この方式では，比較器の全ビットのプルダウン・トランジスタの下にフットと呼ばれるNMOSを1つ配置し，プリチャージ中オフにすることでチャージした電荷がグラウンドに抜けないようにする．もう一つの方法は図7.2(b)のタグ･ゲーティング方式である．この方式では，プリチャージ中タグ線の正転線と反転線の両方をLにすることで，チャージした電荷がグラウンドに抜けないようにする．



(a) フット方式　　(b) タグ・ゲーティング方式

図 7.2: CAMの比較器の方式

これらの2方式の遅延と消費エネルギーの違いについて，以下で述べる．まず，遅延の

大きさでこの 2 方式を比較すると，タグ・ゲーティング方式のほうが小さいと言える．この理由はクリティカル・パスに追加される回路の違いによるものである．タグ・ゲーティング方式ではタグ線に AND ゲートを 1 つ追加するだけであるが，フット方式では全ビットのプルダウン・トランジスタのソース側を束ねてフットを配置するので，プルダウン・トランジスタの総段数と配線容量が増加して遅延が大きくなる．

次に，消費エネルギーで比べると，タグ・ゲーティング方式はタグの放送で消費するエネルギーがより大きくなり，フット方式はマッチ線のプリチャージで消費するエネルギーがより大きくなると言える．タグ・ゲーティング方式でタグの放送に消費するエネルギーが大きくなる理由は，タグ線が無駄に遷移するからである．例えばフット方式ではタグ線が H→H と遷移しない場合でも，タグ・ゲーティング方式では H→L→H と一度 L にゲーティングされてから H に遷移するのでエネルギーを消費する．一方で，フット方式でマッチ線のプリチャージに消費するエネルギーが大きくなる理由は，プリチャージされる電荷の量の違いによるものである．タグ・ゲーティング方式は，プリチャージ中プルダウン・トランジスタの上から 1 段目が必ずオフになるので，それより上の寄生容量分のみのチャージ量で済む．一方でフット方式は 1 段目と 2 段目のプルダウン・トランジスタがオンである可能性があるので，その場合 1 段目と 2 段目とその下の配線の寄生容量分までチャージされてしまう．

遅延が小さいことを踏まえると，通常の 1 段階比較の比較器ではタグ・ゲーティング方式のほうが望ましい．しかし提案手法では，フット方式を採用する．これは，高位ビットのプリチャージのタイミングがエントリごとに異なるためである．提案手法において高位ビットのプリチャージは，1 段階比較時は低位ビットと同じタイミングで行われ，2 段階比較時は低位ビットのタグ比較の後に行われる．この混在する 2 パターンのプリチャージのタイミングにタグ・ゲーティング方式で対応しようとするには，それぞれのタイミングに合わせてタグ線を 2 度ゲーティングする方法と，エントリごとにタグをゲーティングする方法が考えられる．しかし，前者はタグの遷移回数が増加することによって，後者はエン

トリごとに AND ゲートが必要であることによって，いずれも消費エネルギーが大きく増加してしまう．一方フット方式では，各エントリでフットをオフにするタイミングを MUX で変えるだけなので，消費エネルギーは 1 段階比較のみの時とほとんど変わらない．このため，本研究では比較対象も含めすべてフット方式で比較器を設計する．

7.2.2 回路の構成

図 7.3 に評価に用いたウェイクアップ論理の回路図 ($IW = 4$ の例) を，図 7.4 に 1 サイクルのクロック・タイミング図を示す．図 7.3 のトランジスタのそばにある数字はそのゲート幅 (λ) を表し，論理ゲートのそばに付いている数字は (NMOS のゲート幅 (λ)/PMOS のゲート幅 (λ)) を表す．図の赤字のゲート幅は，サイズ 128 発行幅 6 の IQ において遅延が最小になるように調整したゲート幅である．ゲート幅を大きくしてもほとんど遅延が変わらない場合 (10λ 大きくしても 1ps も短くならないような場合) は，消費エネルギーへの影響を考えてなるべく小さいものを選んだ．また，図では省略されているが，タグ線に 1 つリピータが配置されており，タグ線のドライバやリピータは簡単のためゲート幅をすべて同じ構成とした．

図 7.3 に“F”と書かれた NAND ゲートがあるが，これは図 7.5 に示す構成となっている．フットが配置されている理由は，入力のマッチ線はリーク電流が多く電位が大きく低下し，通常の NAND ゲートでは貫通電流が多く発生してしまい，多くの電力を消費するのでこのような設計になっている．

図 7.4 に示す各クロックの役割は以下の通りである．

- ϕ_0 : 1 段階比較の時は低位ビットと高位ビットのマッチ線をプリチャージする．2 段階比較の時は低位ビットのマッチ線をプリチャージする．

- ϕ_1 : 1 段階比較の時はレディ信号の SR ラッチのイネーブル信号となる．2 段階比較の時は低位ビットの比較が一致した時のみ高位ビットのマッチ線をプリチャージする．

- ϕ_{01}：2 段階比較の時の高位ビット比較器のフットの制御信号である．タグが放送される低位ビットのプリチャージのタイミングからフットでゲーティングすることで，新たに放送されたタグによって高位ビットのマッチ線に残っていたチャージが抜けるのを防ぐ．

- ϕ_2：2 段階比較の時の SR ラッチのイネーブル信号である．

- ϕ_3：1 段階比較の時の NAND“ F ”のフットの制御信号である．

- ϕ_4：2 段階比較の時の NAND“ F ”のフットの制御信号である．

レイアウトは，酒井氏の修論 [25] に記されたものをもとにした．ただし，レイアウトにはフットや提案手法の追加の回路のサイズ分は含まれていない．

7.2.3 回路のクリティカル・パス

本研究では，ウェイクアップ論理のクリティカル・パスの遅延を 1．タグ線，2．比較器，3．SR ラッチに 3 分割して考える．以下それぞれの求め方を述べる．

1. タグ線のクリティカル・パス：タグ線が H→L に遷移した時，一番下のエントリに信号が到達するまでの遅延をタグ線のクリティカル・パスの遅延とする．L→H ではなく H→L である理由を説明する．タグの放送と比較器のプリチャージは同時に行われるが，その際タグ線にゲートリークにより一部プリチャージの電荷が漏れる．これにより，H→L 遷移時にタグ線の電荷が抜けるのが遅れる．よって H→L 遷移時がクリティカル・パスの遅延となる．

2. 比較器のクリティカル・パス：1 番下のエントリにタグが放送されてから，1 ビットのみ不一致であるときの，SR ラッチに L が伝えられるまでの遅延を比較器のクリティカル・パスの遅延とする．提案手法については，1 段階比較の時に低位ビットで 1 ビッ

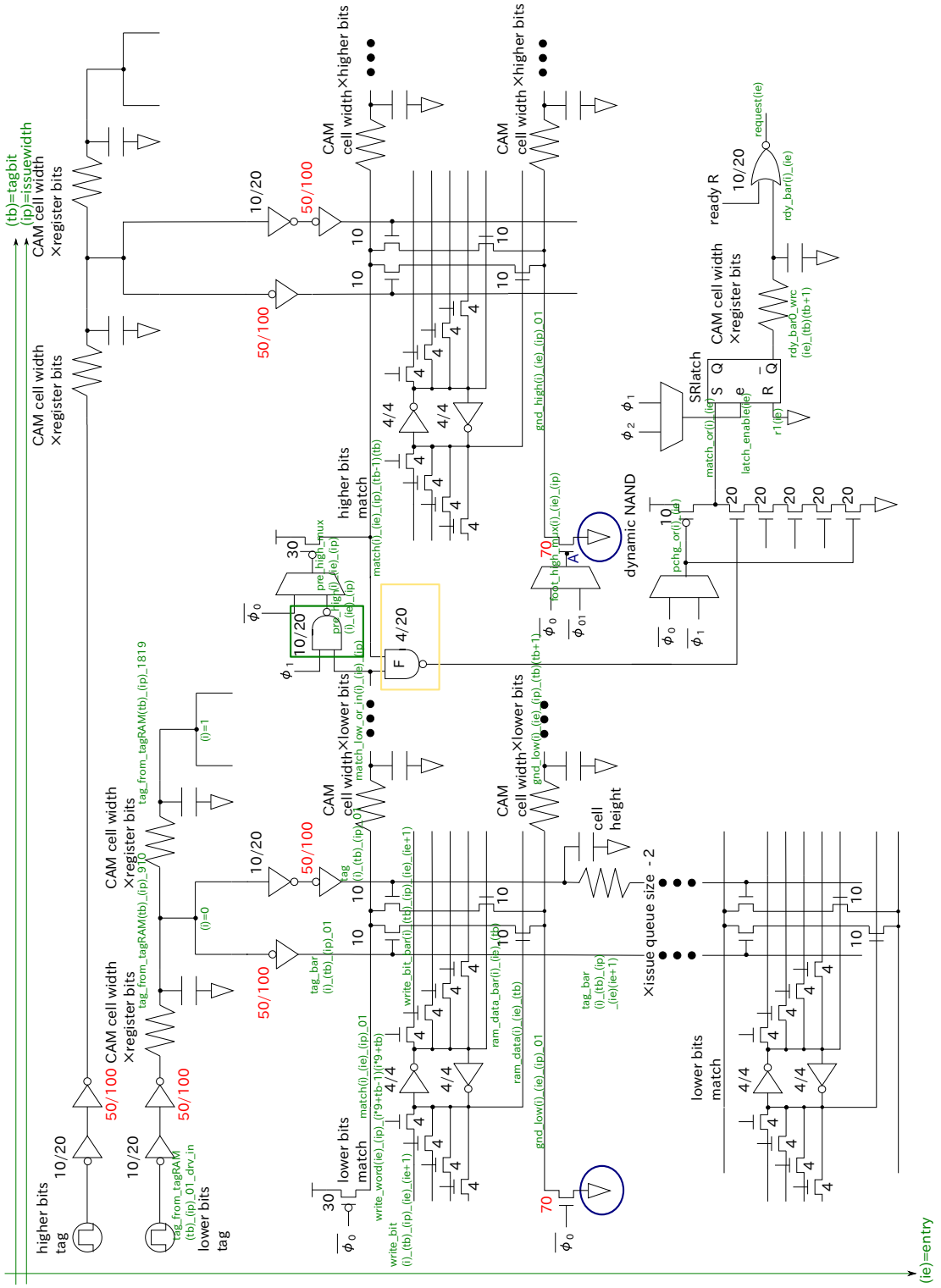


図 7.3: ウェイクアップ論理の回路 (IW = 4)

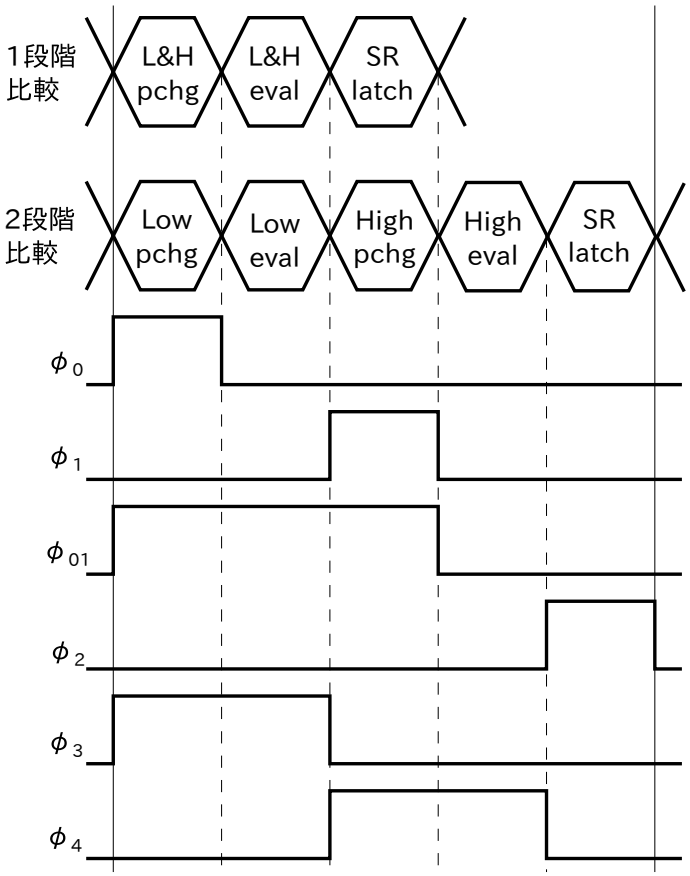


図 7.4: ウェイクアップ論理のクロック

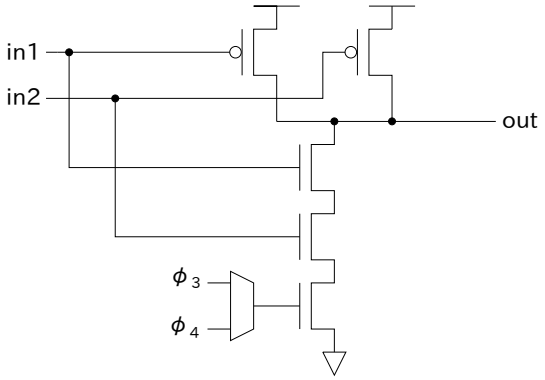


図 7.5: フットつき NAND ゲート ”F ”

ト，もしくは高位ビットで 1 ビットが不一致である時の遅延のうち，長い方をクリティカル・パスの遅延とする．

3. SR ラッチのイネーブル信号が立ってから，図 7.3 右端の NOR の出力が遷移するまでの遅延を SR ラッチのクリティカル・パスの遅延とする．

7.3 選択論理

7.3.1 設計方式

選択論理の方式として，ツリー・アービタ方式 [5] とプレフィック・サム方式 [26] が存在する．このうち，プレフィック・サム方式のほうがツリー・アービタ方式よりも遅延が短いとされているため [26]，本研究ではこれを採用する．

7.3.2 回路の構成

回路，レイアウトは，末永の卒論 [27] に記されたものをもとにした．2 ビット同士 (a1,a0),(b1,b0) を加算する加算器の回路図を図 7.6 に示す．図の赤字は遅延を短くするために調整したゲート幅である．末永の卒論の回路では，各加算器でプリチャージされた電荷が漏れてしまい電力が増加する．そこで，加算器のセルの横幅を超えない範囲の幅のフットを配置した．これにより，例えば発行要求数が 0 の時の消費エネルギーを 32%削減できる．

図 7.7 には選択論理全体の回路図を示す (ただしクリティカル・パスに属する部分のみ図示されている)．図の中の adder に続く数字の羅列は，各加算器のプルダウン・トランジスタの列の数である．

7.3.3 回路のクリティカル・パス

選択論理のクリティカル・パスは図 7.7 に示すように，先頭から 2 番目の発行要求が全段の加算器を通り末尾から 2 番目のプレフィックス・サムを出力し，末尾の発行許可を出して選択論理を横断するまでのパスである．

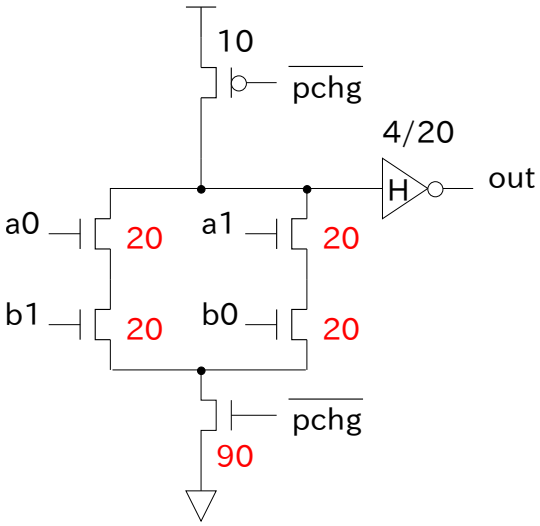


図 7.6: 選択論理の加算器の回路構成

7.4 エイジ論理

7.4.1 回路の構成

回路，レイアウトは，酒井の修論 [25] に記されたものをもとにした．この回路は，Alpha21464 [22] で用いられたものが，もとになっている．ただし，コンフリクト線のリピータにはフットを追加した．

図 7.8 に発行幅が 1 のときのエイジ論理と，選択論理から出力される発行許可との調停を行うアービタの回路図を示す．赤字のゲート幅は，遅延が最小になるように調整されたゲート幅である．また，図では示されていないが，リクエスト線とコンフリクト線にはリピータが配置されている．リクエスト線のリピータは後述するクリティカル・パスに等間隔に縦横合わせて 3 つ配置されるようになっており，クリティカル・パス以外では，それと同じ間隔で配置されている．コンフリクト線のリピータは各線の中央に 1 つ配置されている．

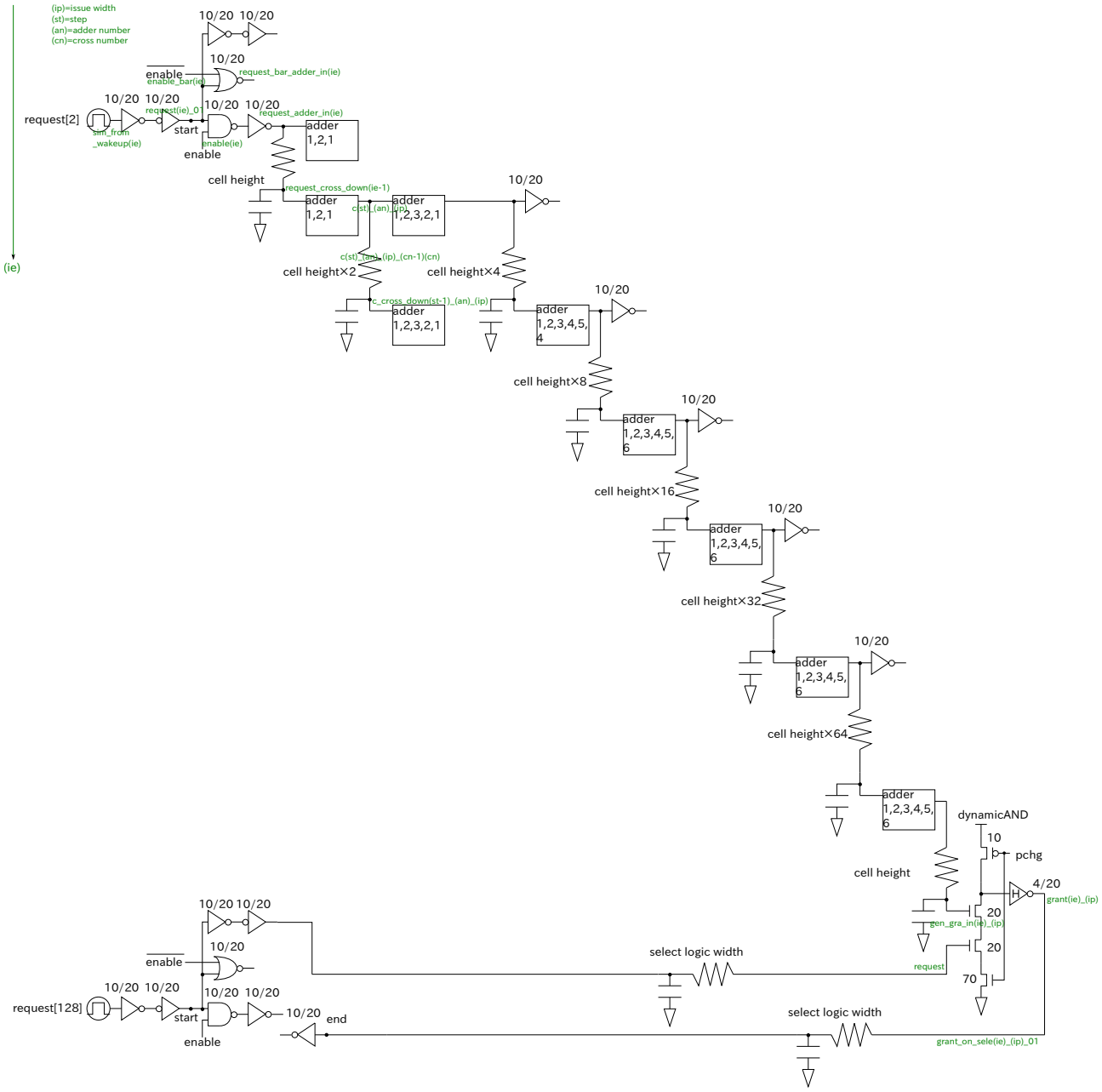


図 7.7: 選択論理の回路構成 (クリティカル・パスに属する部分のみ図示)

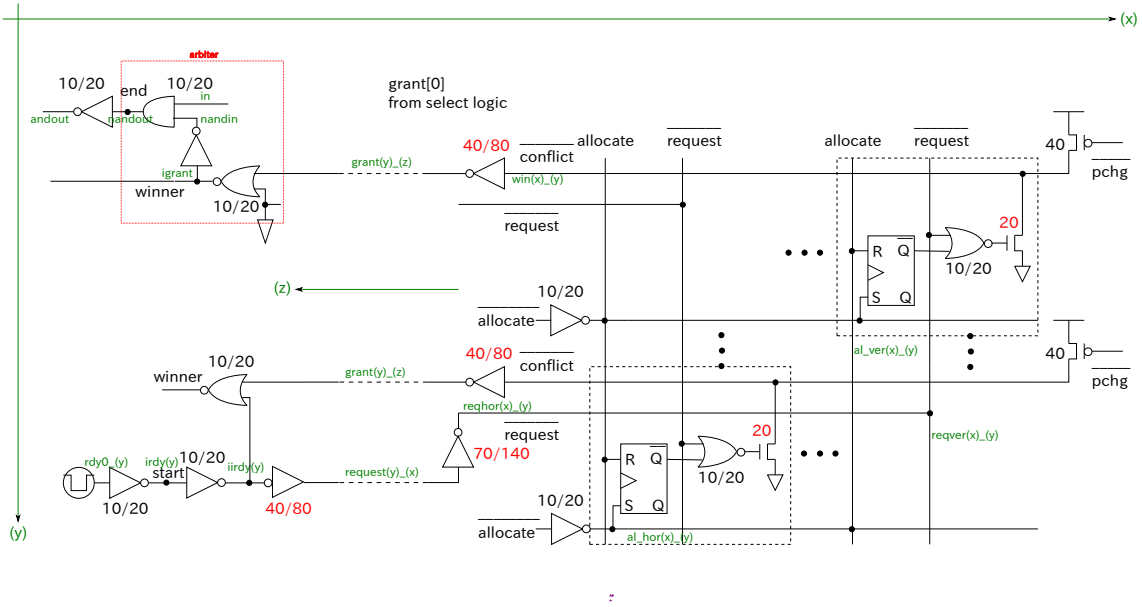


図 7.8: エイジ論理とアービタの回路

7.4.2 回路のクリティカル・パス

エイジ論理の回路のクリティカル・パスは次のとおりである．まず，発行要求がエイジ論理の一番下のエントリに入力されるところを起点として，一番下の行のリクエスト線を横断し，一番右の列を縦断する．そして，プルダウン・トランジスタを通してコンフリクト線をディスチャージする．そして，ウィナー出力が得られたら，選択論理から得られた発行許可と調停を行い，選択論理からの発行許可が打ち消される点を終点とする．

7.5 タグ RAM・ペイロード RAM・PQ

7.5.1 回路の構成

RAM の回路は 8T デュアルポート SRAM [28] で構成され，そのレイアウトはすべて末永の卒論 [27] をもとにした．読み出し回路はシングル・エンドである．図 7.9 にタグ RAM で使用した読み出し回路を示す．

タグ RAM は IQ のクリティカル・パス上にあるので，できるだけ遅延を小さくする必要がある．そのため，バンクサイズ 8 でバンク化された回路を使用し，遅延が小さくなるよ

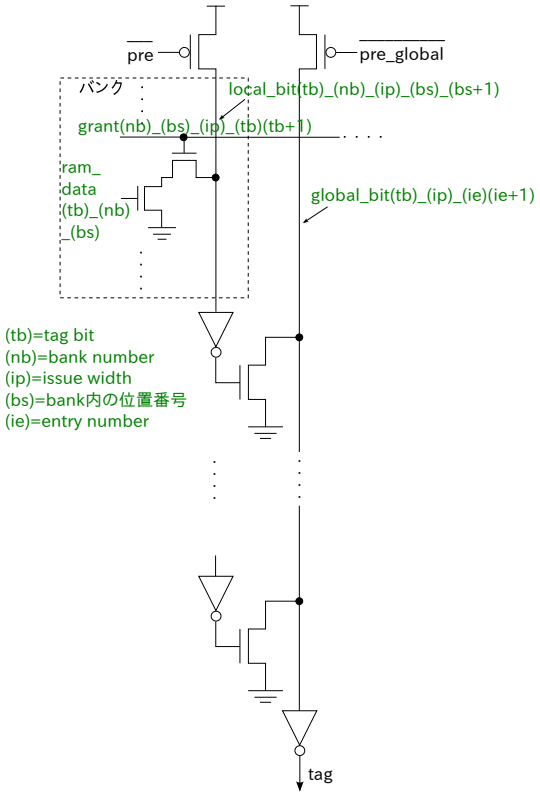


図 7.9: RAM 読み出し回路

うに MOS のゲート幅やリピータの個数を調整した．一方でペイロード RAM，PQ はバンク化せず，ビット線のリピータは 128 エントリごとに配置した．

エントリのビット数は各回路で異なる．タグ RAM は 9 ビットである．ペイロード RAM は，本研究では命令長を 32 ビット，論理レジスタの幅を 5 ビット，物理レジスタの幅を 9 ビットと仮定しているので，$32 - 5 \times 3 + 9 \times 3 = 41$ ビットとする．PQ は，128 エントリの IQ のインデックスを保持するので，7 ビットである．

7.5.2 回路のクリティカル・パス

タグ RAM のクリティカル・パスは，発行許可信号が一番上のエントリのワード線を右端から左端まで横断し，タグの左端のビットが読み出されて一番下で出力されるパスである．本研究ではタグ RAM の遅延測定の起点を，選択論理の発行許可が出力される点として，その後のウェイクアップ論理の上を発行許可信号が横断する遅延分を含んでいる．ペイロード RAM や PQ の遅延については，IQ のクリティカル・パスにならないと考え，測定は行わない．

7.6 ディスパッチ

7.6.1 回路の構成

図 7.10 にディスパッチのための RAM 書き込み回路を示す．書き込み回路はダブル・エンドであり，ウェイクアップ，タグ RAM，ペイロード RAM ，PQ ですべて同じ構成ある．末永の卒論 [27] では書き込み回路がダイナミック・ロジックで構成されていたが，本研究ではスタティック・ロジックで構成する．この変更の理由は，ディスパッチが IQ のクリティカル・パス上にないので高速化が必要ないことと，消費エネルギーの削減のためである．

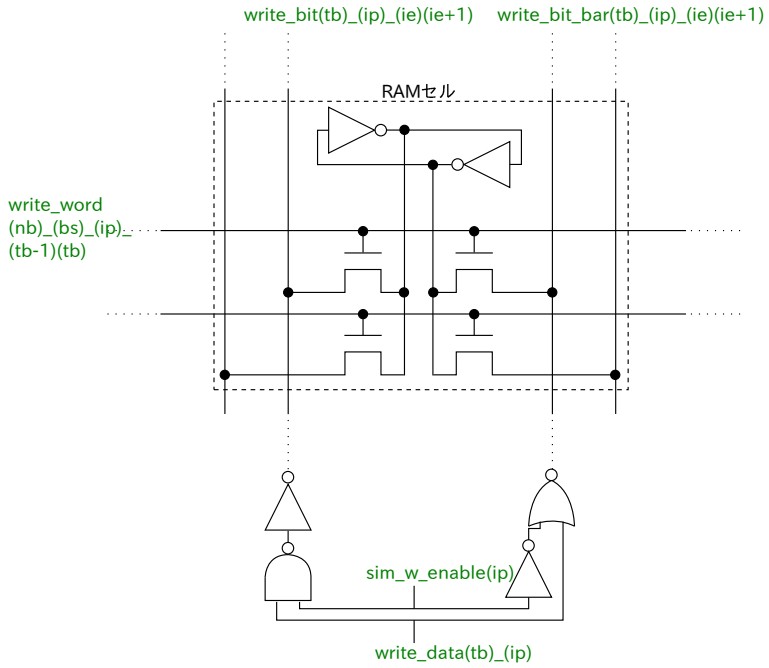


図 7.10: RAM 書き込み回路

第 8 章　評価方法

本章では，評価の方法について述べる．

8.1　回路の遅延・消費エネルギー評価評価

回路の遅延と消費エネルギーは，HSPICE による回路シミュレーションにより測定した．16nm LSI プロセスを仮定し，トランジスタ・モデルとして，アリゾナ州立大学の Predictive Technology Model(PTM) [29] を使用した．International Technology Roadmap for Semconductors(ITRS) [30] により公開されているデータより，単位長さあたりの配線抵抗は 46.96MΩ/m，配線容量は 0.165nF/m とした．

8.2　IPC・動作回数の評価環境

提案手法による IPC 低下抑制効果と発行キューの各回路の動作回数は，SimpleScalar Tool Set Version 3.0a [31] をベースにしたプロセッサ・シミュレータで評価した．ベンチマーク・プログラムは，SPEC CPU2006 から正しく動作しない wrf を除いた 28 本を採用した．プログラムは gcc 4.5.3 でオプション-O3 を用いてコンパイルした．プログラムの入力には ref データ・セットを用いた．各プログラムを 16B 命令をスキップして，その後の 100M 命令を実行した．表 8.1 に基準プロセッサの構成を示す．

8.3　消費エネルギーの導出方法

本研究では，回路シミュレーションによる消費エネルギーの評価結果とプロセッサ・シミュレーションによる回路の動作回数の評価から，発行キューの消費エネルギーを求める．本節では，それを導出するための計算方法について述べる．

表 8.1: プロセッサの構成

Pipeline width	6-instruction wide for each of fetch, decode, issue and commit,
Reorder buffer	256 entries
Issue queue	128 entries
Load/store queue	128 entries
Physical registers	int and fp 256 registers each
Tag bit width	9 bits
Branch prediction	12-bit history, 4K-entry PHT gshare, 2K-set, 4-way BTB 10-cycle misprediction penalty
Function unit	3 iALU, 1 iMULT/DIV, 2 Ld/St, 2 fpALU
L1 I-cache	16KB, 8-way, 64B line
L1 D-cache	16KB, 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
Data prefetcher	stream-based, 32-stream tracked, 16-line distance, 2-line degree, prefetch to L2 cache

8.3.1 発行キューの消費エネルギー

発行キューの消費エネルギー E_{IQ} は以下の式で表される．

$$E_{IQ} = E_{WU} + E_{SL} + E_{AG} + E_{TR} + E_{PR} + E_{PQ} + E_{DP} \quad (8.1)$$

ここで，E_{WU} はウェイクアップの消費エネルギー，E_{SL} は選択論理の消費エネルギー，E_{AG} はエイジ論理の消費エネルギー，E_{TR} はタグ RAM の消費エネルギー，E_{PR} はペイロード RAM の消費エネルギー，E_{PQ} は PQ の消費エネルギー，E_{DP} はディスパッチ時の消費エネルギーである．これらはすべて動的消費エネルギーである．

静的消費エネルギー E_{static} についてはどの回路も以下の式で求める．

$$E_{static} = E_{static\_per\_cycle} \times sim\_cycle \quad (8.2)$$

ここで，$E_{static\_per\_cycle}$ はサイクルあたりの静的消費エネルギー，　sim_cycle は SimpleScalar によるシミュレーションの実行サイクル数である．$E_{static\_per\_cycle}$ は，各回路のクロック・サイクル中で動的電力の変化がほとんどなくなったタイミングの電力と，クロック・サイクル時間の積である．この方法によって測定される静的消費エネルギーは，スタティック・ロジック回路についてのみである．ダイナミック・ロジック回路については，プリチャージ語のリーク電流を測定する必要があるが，測定が困難なため，本研究では動的消費エネルギーに含めている．

8.3.2 ウェイクアップ論理の消費エネルギー

ウェイクアップ論理の消費エネルギー E_{WU} は以下の式で表される．

$$E_{WU} = E_{tag\_br} + E_{comp} + E_{other} \quad (8.3)$$

ここで，E_{tag_br} はタグ放送の消費エネルギー，E_{comp} は比較器のエネルギー，E_{other} はその他の消費エネルギーである．

E_{tag_br}の導出

E_{tag_br} は以下のように求める．

$$E_{tag\_br} = E_{tag\_br\_per\_tag} \times (N_{issue\_tag} + N_{diff\_tag}) \tag{8.4}$$

ここで，$E_{tag\_br\_per\_tag}$ は一つのタグを WU に放送するために要するエネルギーである．N_{issue_tag} は放送されるタグの個数である．N_{diff_tag} は前サイクルに放送されたタグの数より現サイクルに放送されたタグの数が少ない場合における，その差である．N_{diff_tag} を考慮する理由は，前サイクルでタグが放送されていたタグ線に現サイクルでタグが放送されない場合，それらのタグ線の全ビットに 0 が放送されるため (タグ RAM から出力される無効なタグの値は回路構成上 0 となる)，エネルギーを消費するからである．

タグの放送では，正転値については各ビットが $0 \rightarrow 1$ に遷移するとき，反転値については $1 \rightarrow 0$ に遷移するときに消費エネルギーが消費される．また，$0 \rightarrow 1$ と $1 \rightarrow 0$ の遷移で消費エネルギーは等しいとする．本研究では，タグ放送について長期的に見るとタグの値はランダムなので，タグ内で遷移するビットの割合は 1/2 と考え，$E_{tag\_br\_per\_tag}$ をタグ内の全ビットが遷移した時の半分の消費エネルギーとして定義して測定する．

E_{comp} の導出

E_{comp} の求め方を説明する．まず，従来の比較段数を変化させられない比較器について考える．今回は無効化されているエントリの比較器，および既にレディなオペランドの比較器では，比較器を無効化してプリチャージを行わない方式を想定し，これらの場合はエネルギーを消費しないと考える．後者については，レディ線の反転線と pchg 制御線の論理積を新たな pchg 制御線にすることで実装可能である ($new\_pchg = pchg \wedge \overline{rdy}$)．また，ソース・オペランドが一つしかない命令の場合も，エントリの片方の比較器ではプリチャージを行わないものとする．

一般にタグ比較において一致した場合にはそのサイクルは消費エネルギーを消費しない．

この根拠は，その一致した比較器では次のサイクルにプリチャージが必要ないからである．しかし本研究ではその消費エネルギーの評価方法上，タグ比較が一致した時もタグ比較が一致しなかった時と同様のエネルギーを消費するものとして計算する．その理由を以下に説明する．

> **タグ比較が一致した時もタグ比較が一致しなかった時の消費エネルギーで計算する理由**：比較が一致したサイクルではマッチ線はディスチャージされないが，その直後のサイクルにレディとなった状態で比較が行われることでマッチ線がディスチャージされ，エネルギーを消費する．それ以降は，レディであるためにマッチ線はプリチャージはされないのでディスチャージもされない．つまり，比較が一致した時は，その後一度だけマッチ線がディスチャージされるので，結果的に比較が一致しなかった時と同じエネルギーを消費する．

上記の例外として，レディとなった直後にその命令が発行され，そのエントリにすぐ別の命令が挿入される場合がある．この場合，新たなソース・タグとの比較が行われ，これも 1 回の比較として消費エネルギーに加算されるため，二重で比較器の消費エネルギーを加算してしまうことになる．しかし，このような場合が起こる割合は，ほとんどのベンチマークでは非常に少ない (図 8.1 参照)．そのため，タグ比較が一致した時すべてを，タグ比較が一致しなかった時の消費エネルギーで計算する．

以上より，従来手法の比較器の総消費エネルギー E_{comp} は，1 つの比較器が 1 回比較を行い不一致であった時の消費エネルギー E_{not_match} と，比較回数 N_{comp} の積で近似して求める．

$$E_{comp} \approx E_{not\_match} \times N_{comp} \tag{8.5}$$

ここで言う比較回数とは，1 つのディスティネーション・タグと 1 つのソース・タグの比較を 1 回と数えた時の回数である．

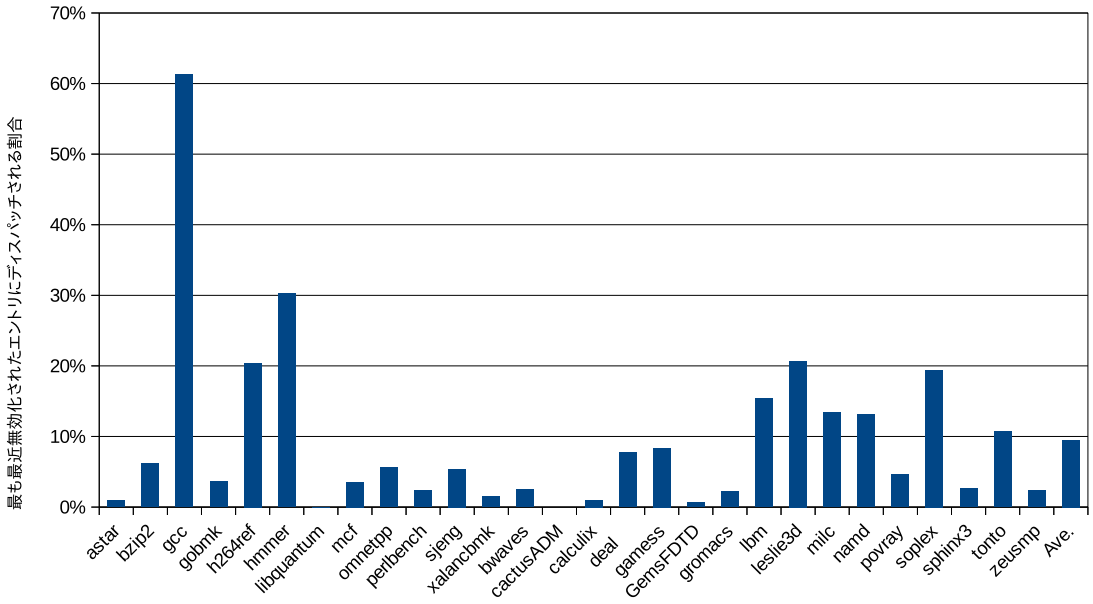


図 8.1: 全てディスパッチされた命令中の，最も最近にレディになったエントリにディスパッチされた命令の割合

ここまでは従来手法の比較器について考えたので，ここからは比較段数を変えられる提案手法の比較器について考える．提案手法の比較は，1 段階と 2 段階が切り替えられ，1 段階比較では低位ビットと高位ビット両方不一致，両方一致，低位ビットのみ一致，高位ビットのみ一致，の 4 つに場合分けできる．2 段階比較では，低位ビットが不一致 (高位ビットが一致か不一致かによらない)，両ビット一致，低位ビットのみ一致，の 3 つに場合分けできる．

以上のような場合分けで，提案手法の比較器の総消費エネルギー E_{comp} は，比較段数を n，比較結果を $result$ として，比較器 1 個の消費エネルギーを $E_{(n,result)}$，その比較回数を $N_{(n,result)}$ とすると，次のように求められる (比較結果 $result$ については，$both\_match$ は低高両ビット一致，$neither\_match$ は両ビットとも不一致，low_only_match は低位ビットのみ一致，$high\_only$ は高位ビットのみ一致，low_not_match は低位ビットが不一致であることを表す)．

$$\begin{aligned} E_{comp} &\approx E_{(1,neither\_match)} + E_{(1,both\_match)} + E_{(1,low\_only\_match)} + E_{(1,high\_only\_match)} \\ &+ E_{(2,low\_not\_match)} + E_{(2,both\_match)} + E_{(2,low\_only\_match)} \end{aligned} \tag{8.6}$$

以下では，それぞれの場合での消費エネルギー計算方法を説明する．

- 1段階比較両ビット不一致時 $E_{(1,neither\_match)}$：この場合，1段階比較の両ビット不一致時の比較器あたりの消費エネルギー $E_{(1,neither\_match)\_per\_comp}$ とその回数 $N_{(1,neither\_match)}$ の積で求められる．

$$E_{(1,neither\_match)} = E_{(1,neither\_match)\_per\_comp} \times N_{(1,neither\_match)} \tag{8.7}$$

- 1段階比較両ビット一致 $E_{(1,both\_match)}$：前述した「タグ比較が一致した時もタグ比較が一致しなかった時の消費エネルギーで計算する理由」により，両ビット一致時の消費エネルギーは，不一致時の比較器当たりのエネルギー $E_{(1,neither\_match)\_per\_comp)}$ とその回数 $N_{(1,both\_match)}$ の積で求められる．

$$E_{(1,both\_match)} = E_{(1,neither\_match)\_per\_comp} \times N_{(1,both\_match)} \tag{8.8}$$

- 1段階比較低位ビットのみ一致 $E_{(1,low\_only\_match)}$：両ビット一致時は次サイクルでエントリが無効化されるが，低位ビットのみ一致の場合は次サイクルは有効エントリのままである．したがって，次サイクルも消費エネルギーとして測定上加算されるので，現サイクルの消費エネルギーだけを考えれば良い．よって低位ビットではマッチ線がディスチャージされず，高位ビットのみでディスチャージされる場合の比較器1つあたりの消費エネルギー $E_{(1,low\_only\_match)\_per\_comp}$ とその回数 $N_{(1,low\_only\_match)}$ の積で表される．

$$E_{(1,low\_only\_match)} = E_{(1,low\_only\_match)\_per\_comp} \times N_{(1,low\_only\_match)} \tag{8.9}$$

- 1 段階比較高位ビットのみ一致 $E_{(1,high\_only\_match)}$：高位ビットではマッチ線がディスチャージされず，低位ビットのみでディスチャージされる場合の比較器 1 つあたりの消費エネルギーの比較器 1 つあたりの消費エネルギー $E_{(1,high\_only\_match)\_per\_comp}$ とその回数 $N_{(1,high\_only\_match)}$ の積で表される．理由は 1 段階比較の低位ビットのみ一致の時と同じである．

$$E_{(1,high\_only\_match)} = E_{(1,high\_only\_match)\_per\_comp} \times N_{(1,high\_only\_match)} \tag{8.10}$$

- 2 段階比較低位ビット不一致 $E_{(2,low\_not\_match)}$：この場合の比較器 1 つあたりの消費エネルギー $E_{(2,low\_not\_match)\_per\_comp}$ とその回数 $N_{(2,low\_not\_match)}$ の積で表される．高位ビットがプリチャージされないので，消費エネルギーを削減できる．

$$E_{(2,low\_not\_match)} = E_{(2,low\_not\_match)\_per\_comp} \times N_{(2,low\_not\_match)} \tag{8.11}$$

- 2 段階比較両ビット一致 $E_{(2,both\_match)}$：前述した「タグ比較が一致した時もタグ比較が一致しなかった時の消費エネルギーで計算する理由」により，不一致の時の消費エネルギーを用いて計算する．ただし，両ビットともプリチャージされた状態で次サイクルにディスチャージされるので，$E_{(2,low\_not\_match)}$ ではなく $E_{(1,neither\_match)}$ を用いる．

$$E_{(2,both\_match)} = E_{(1,neither\_match)} \times N_{(2,both\_match)} \tag{8.12}$$

- 2 段階比較低位ビットのみ一致 $E_{(2,low\_only\_match)}$：低位ビットではマッチ線がディスチャージされず，高位ビットのみでディスチャージされる場合の比較器 1 つあたりの消費エネルギー $E_{(2,low\_only\_match)\_per\_comp}$ とその回数 $N_{(2,low\_only\_match)}$ の積で表される．理由は 1 段階比較の低位ビットのみ一致の時と同じである．

$$E_{(2,low\_only\_match)} = E_{(2,low\_only\_match)\_per\_comp} \times N_{(2,low\_only\_match)} \tag{8.13}$$

E_{other} の導出

E_{other} の求め方を説明する．E_{other} には，各比較器マッチ線を入力とする OR 回路などの消費エネルギーが含まれる．この OR 回路はダイナミック・ロジックで構成されており，タグ比較が一致した場合，そのサイクルではエネルギーを消費しない．しかし，前述した「タグ比較が一致した時もタグ比較が一致しなかった時の消費エネルギーで計算する理由」と同様の理由で次サイクルにはエネルギーを消費するため，不一致の時の消費エネルギーを用いて計算する．1 つのソース・タグに割り当てられた部分の比較器とタグ線以外の消費エネルギーを $E_{other\_not\_match}$，有効なエントリのレディでないソース・オペランドに対応するタグの数を N_{other} とすると，E_{other} は

$$E_{other} \approx E_{other\_not\_match} \times N_{other} \tag{8.14}$$

と求められる．$E_{other\_not\_match}$ は，すべての比較器で比較が不一致であった時の消費エネルギー $E_{wu\_not\_match}$ から全比較器のエネルギー $E_{comp\_not\_match}$ とタグの放送の全消費エネルギー $E_{tag\_br\_not\_match}$ を引き，全ソース・タグの個数 (IQ のサイズ (IQS) の 2 倍) で割ったものである．

$$E_{other\_not\_match} = \frac{E_{wu\_not\_match} - E_{comp\_not\_match} - E_{tag\_br\_not\_match}}{2 \times IQS} \tag{8.15}$$

8.3.3 選択論理の消費エネルギー

プレフィックス・サム方式の選択論理では，1 番目のエントリから下に向かって発行のリクエストを加算していくが，その際各加算器からワンホット表現されたビット列が出力される．そして，合計リクエスト数が発行幅の数に達すると，それ以後の計算は不要なのでビット列は 0 で埋められる．つまり，発行幅の数に達した位置が発行キューの上の方か下の方かで，0 を出力する加算器の数が変わってくる．加算器が 0 を出力する時，エネルギーを消費しないので，発行幅の数に達したエントリが上の方であればあるほど，選択論理の消費エネルギーは小さくなる．

実際に発行幅 6 で，選択論理全体の 1 サイクル分の消費エネルギーを測定した結果を図 8.2 に示す．横軸の数字の列は 1 サイクルにリクエストが出されたエントリ番号を表している．この図からわかるように，リクエスト数が発行幅未満の場合，ほぼ一定の消費エネルギーを消費する．リクエスト数が発行幅以上の場合は，6 番目のリクエストが入力されたエントリ番号が小さいほど，消費エネルギーが小さくなっていることがわかる．{1, 2, 3, 4, 5, 6} は，6 番目のリクエストが最も上から入力されるパターンなので，消費エネルギーは最小となる．

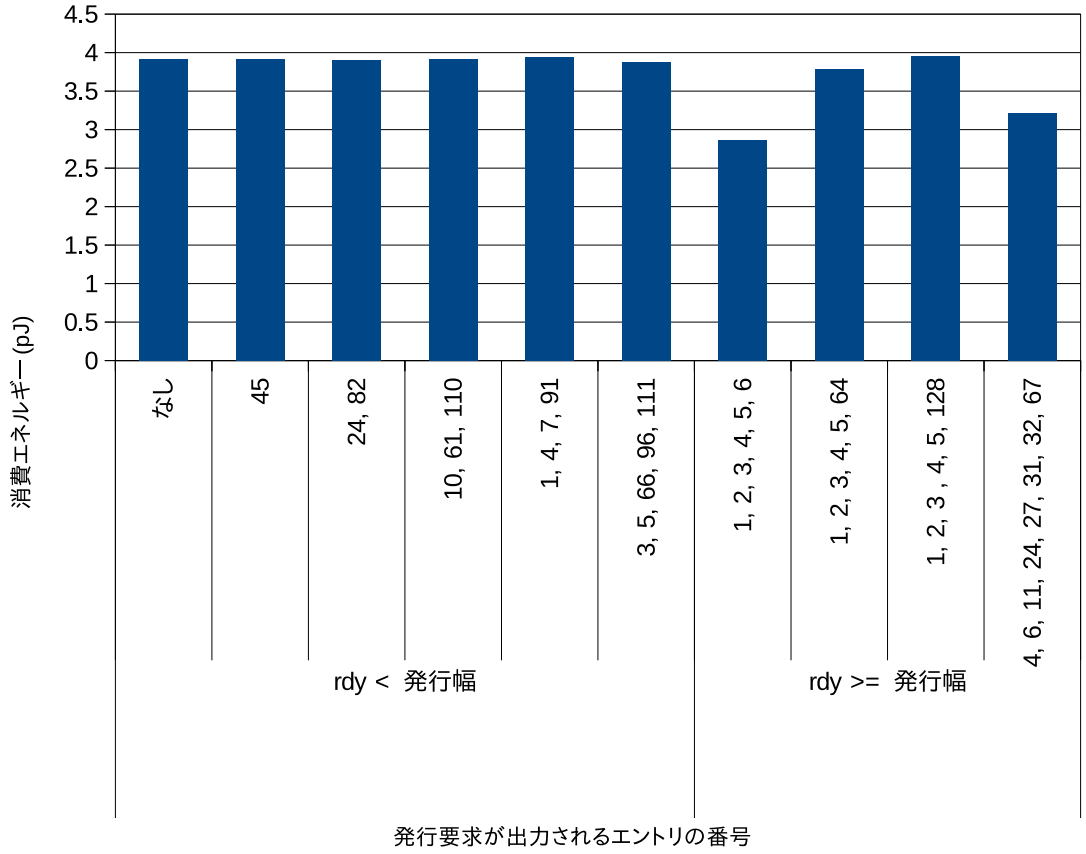


図 8.2: リクエストが入力されるエントリ番号のパターンによる，選択論理の消費エネルギー

このような消費エネルギーとリクエスト数の関係を考慮して，E_{SL} は以下のように求める．

$$E_{SL} = E_{rdy\_num<iw} \times N_{rdy\_num<iw} + E_{rdy\_num\geq iw} \times N_{rdy\_num\geq iw} \tag{8.16}$$

ここで，$E_{rdy\_num<iw}$ はリクエスト数が発行幅未満の時の消費エネルギー，$N_{rdy\_num<iw}$ はその回数，$E_{rdy\_num\geq iw}$ はリクエスト数が発行幅以上の時の消費エネルギー，$N_{rdy\_num\geq iw}$ はその回数である．$E_{rdy\_num<iw}$ については，どの入力パターンでもほぼ一定なので，任意のパターンで測定すれば良い．本研究ではリクエストがないときの消費エネルギーを適用する．$E_{rdy\_num\geq iw}$ は，$issue\_width$ 番目のリクエストがどのエントリから入力されたかに依存するので，パターン $\{1, 2, 3, 4, 5, n\}(6 \leq n \leq 128)$ の消費エネルギーを測定し，それらの平均とする．

8.3.4 エイジ論理の消費エネルギー

エイジ論理の消費エネルギー E_{AG} は以下のようにして求められる．

$$E_{AG} = E_{req} + E_{conf} + E_{nor} \tag{8.17}$$

ここで，E_{req} はリクエスト線を信号が伝達されるときに消費される消費エネルギー，E_{conf} はコンフリクト線の消費エネルギー，E_{nor} はエイジ行列の各セルに 1 つ配置されている NOR ゲートの消費エネルギーである．調停回路については，動的消費エネルギーは小さいので無視し，静的消費エネルギーのみ評価する．SR ラッチの消費エネルギーも静的消費エネルギーのみ評価する．以下でそれぞれの求め方について説明する．

E_{req} の導出

E_{req} の求め方を説明する．E_{req} はリクエストの数に依存する．したがって，1 つのリクエストあたりの消費エネルギー $E_{req\_per\_req}$ とリクエストの総数 N_{req} の積で求められる．

$$E_{req} = E_{req\_per\_req} \times N_{req} \tag{8.18}$$

ここで $E_{req\_per\_req}$ は，エントリごとに違いがある．リクエスト線のエネルギー消費の原因は主に配線とリピータの寄生容量である．図 8.3 にリクエストが入力されたエントリ番号ごとのリクエスト線の消費エネルギーを示す．エントリ番号が増えるに従ってリクエスト線

が長くなるので，消費エネルギーが増えている．また，急激に増えたり減ったりしている部分があるが，これは配置されているリピータの個数が増えたり減ったりしたためである．

このようにリクエスト線は入力されるエントリによって消費エネルギーが異なるが，ランダムキューなのでどのエントリでも同確率でリクエストが入力されるため，その平均を用いる．

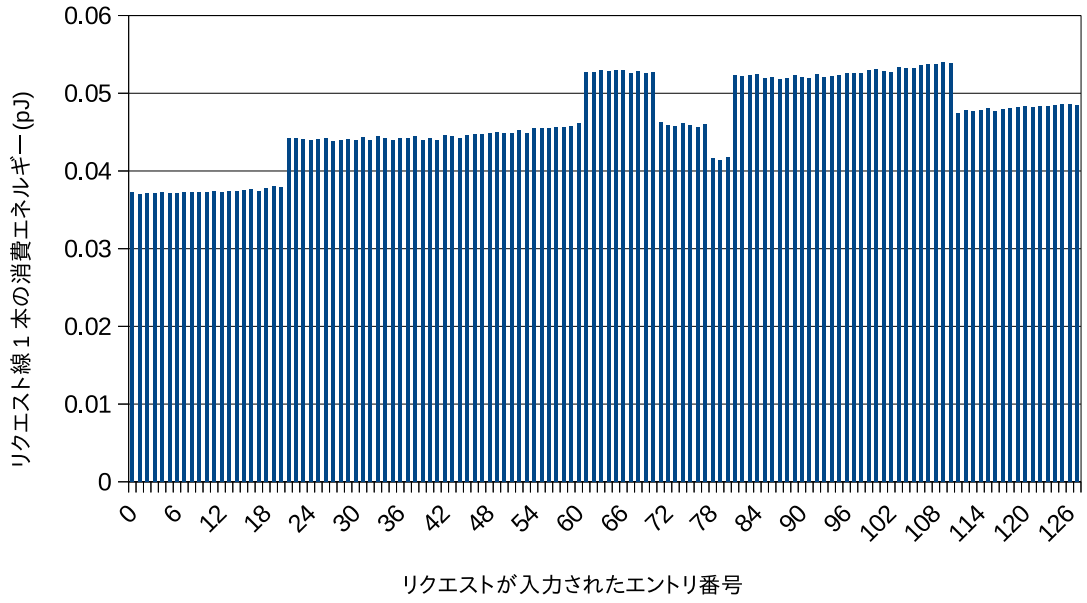


図 8.3: 入力されるエントリに対するリクエスト線の消費エネルギー

E_{conf} の導出

E_{conf} の求め方を説明する．E_{conf} は，理論的にはコンフリクト線がディスチャージされるときに消費される消費エネルギーである．ディスチャージされるコンフリクト線は，エイジ論理で選ばれ発行される命令より新しい命令の有効なエントリのコンフリクト線である．ただし，ディスチャージされない場合も，回路がダイナミック・ロジックなためにリーク電流が大きく，その消費エネルギーも少なくない．ディスチャージされない時とは，エントリがエイジ論理で選ばれ発行される命令より古い命令を保持している時と，ウェイク

アップ論理からのリクエストが一つもない時である．一方で無効なエントリは，プリチャージされず消費エネルギーを全く消費しないと考える．

以上より E_{conf} は，1 本のコンフリクト線がディスチャージされる時の消費エネルギーを $E_{conf\_discharge}$，ディスチャージされない時の消費エネルギーを $E_{conf\_nondischarge}$，エイジ論理で選ばれる命令より新しい命令のエントリの数を N_{newer}，エイジ論理で選ばれる命令自身とそれより古い命令のエントリの数を N_{older}，リクエストが 0 の時の有効なエントリの数を N_{zero_req} とすると以下のように求められる．

$$E_{conf} = E_{conf\_discharge} \times N_{newer} + E_{conf\_nondischarge} \times (N_{older} + N_{zero\_req}) \tag{8.19}$$

ここで $E_{conf\_discharge}$ は，測定した回路ではダイナミック・ロジックの配線のちょうど中央に 1 つリピータが配置されているため，リピータの左右どちらでディスチャージが起こるかでその消費エネルギーが異なる．例えば図 8.4 に示すように，リピータの左側でディスチャージが起こっても，右側ではディスチャージされず，そのコンフリクト線にチャージされた電荷の $\frac{1}{2}$ しかディスチャージされない．一方でリピータの右側でディスチャージが起こると，リピータのプルダウン・トランジスタによって左側もディスチャージされるので，そのコンフリクト線の全電荷がディスチャージされることになる．

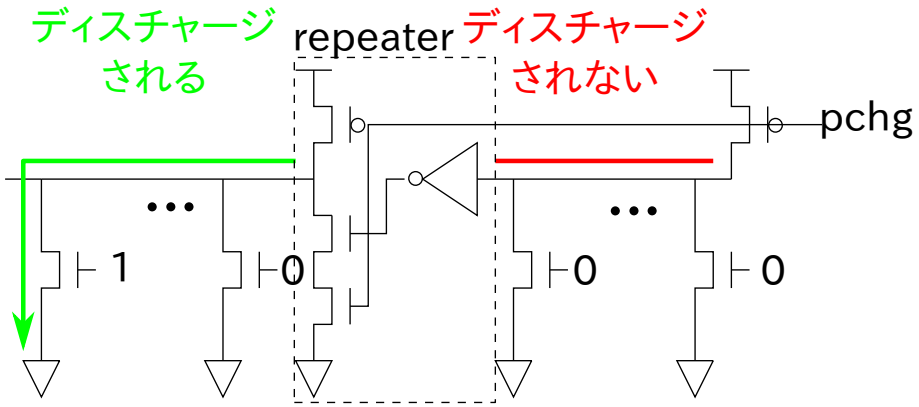


図 8.4: リピータの左側のみでディスチャージが起こると，右側ではディスチャージされない

しかし本研究では，コンフリクト線のディスチャージの消費エネルギーは，全電荷がディスチャージされる時の消費エネルギーであると近似し，左側のみでディスチャージされる時の消費エネルギーはエネルギー計算に用いない．これは，左側のみでディスチャージされる場合が非常に少ないためである．この根拠について以下で述べる．

まず前提として，あるエントリのコンフリクト線の左側だけがディスチャージされる場合というのは，そのエントリの命令より古い命令のリクエスト信号がすべて発行キューの上半分のエントリに入力される場合である．この理由は図 8.5 で示すように，上半分のエントリに入力されたリクエスト信号はコンフリクト線のリピータの左側をディスチャージし，下半分のエントリに出力されたリクエスト線はリピータの右側をディスチャージするためである．

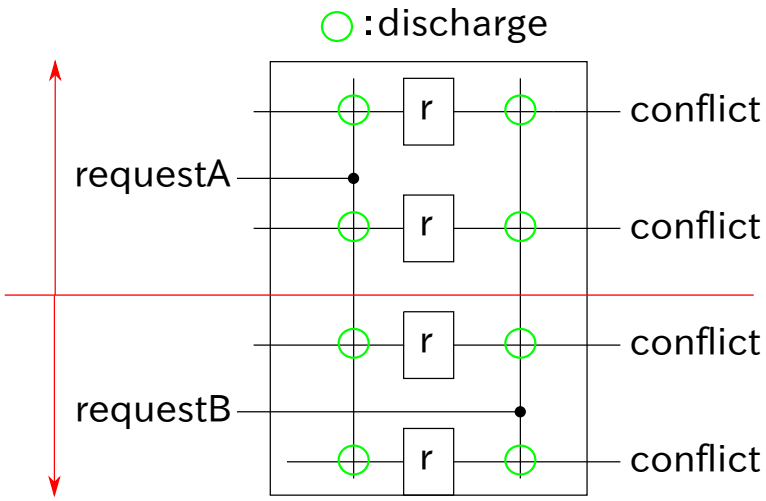


図 8.5: 上半分のリクエスト信号はリピータの左側をディスチャージし，下半分のリクエスト信号は右側をディスチャージする

次に，コンフリクト線の左側だけがディスチャージされる場合が起こる確率は，そのエントリの命令より古い命令のリクエスト信号の数によって変化し，その数が多い場合は確率は非常に小さくなる．発行キューのサイズを IQS，自身より古い命令のリクエストの数を r とすると，この確率は $\frac{{}_{\frac{IQS}{2}}C_r}{{}_{IQS}C_r}$ と表せる．図 8.6 に，自身より古い命令のリクエスト信号

の数に対するこの確率分布を示す．図より，自身より古い命令のリクエスト信号が数個あれば，コンフリクト線の左側のみでディスチャージされる確率は非常に小さくなり，全電荷がディスチャージされるということがわかる．

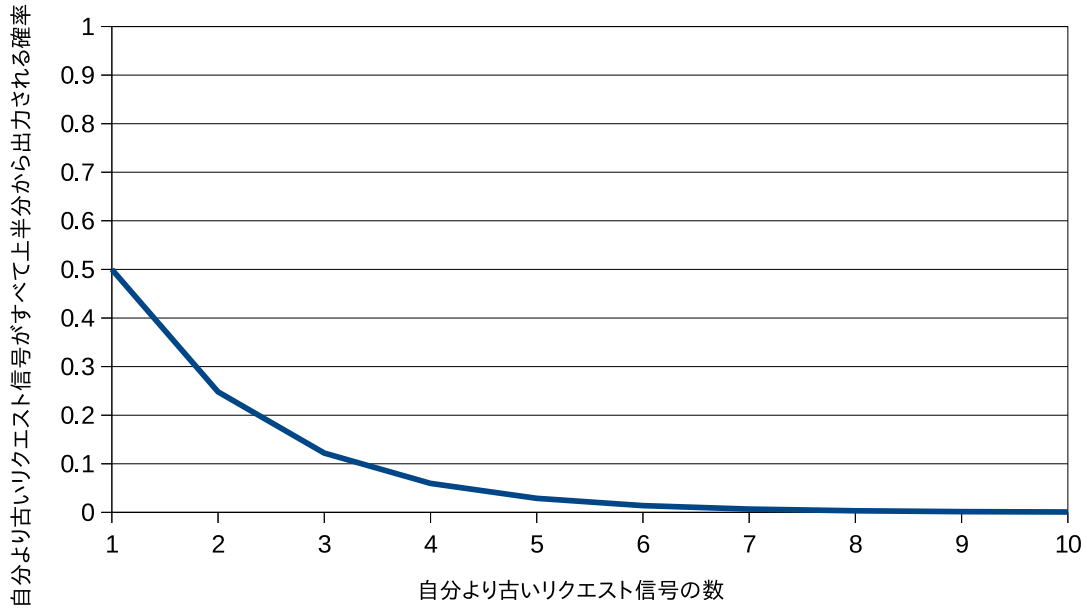


図 8.6: 自身より古い命令のリクエスト信号の数に対して，これらが全て上半分のエントリからの入力である確率

そして，この自身より古い命令のリクエスト信号の数は多い場合が多い．その根拠として，図 8.7 に 1 サイクルに入力されるリクエスト信号の数の分布を示す．一般に発行キュー内の古い命令がよりリクエストを出しやすいことを考えると，各エントリの命令自身より古い命令のリクエスト信号の数は，この 1 サイクルに入力されるリクエスト信号の数に近似できる．つまり図 8.7 が示すように，リクエスト信号の数は広く分布しているため，自身より古い命令のリクエスト信号の数も広く分布していると考えられる．そのため，図 8.6 のリクエスト信号がすべて上半分のエントリに入力される確率もほぼ 0 になる．よって，コンフリクト線のリピータの左側だけがディスチャージされる確率は非常に小さいので，コンフリクト線がディスチャージされる時の消費エネルギー $E_{conf\_discharge}$ はコンフリクト線

の全電荷がディスチャージされる時の消費エネルギーとする．

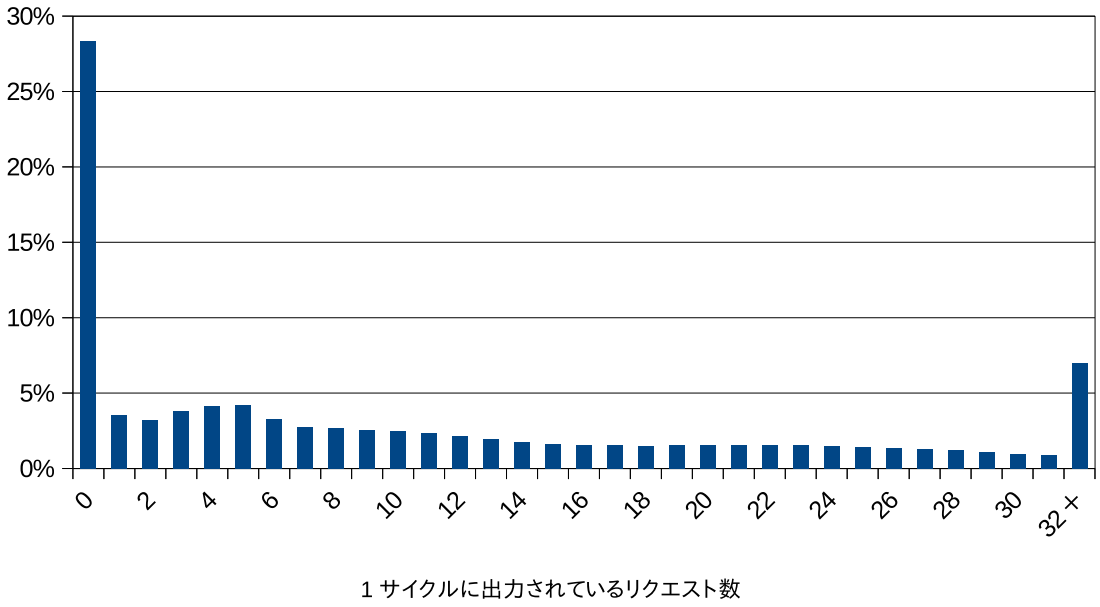


図 8.7: リクエスト数の分布

E_{nor} の導出

E_{nor} の求め方を説明する．E_{nor} は，あるリクエストに対して，それより新しい命令を保持するエントリの NOR ゲートの出力が L→H に遷移することで消費する消費エネルギーである．これは，次の式で求められる．

$$E_{nor} = E_{nor\_per\_nor} \times N_{nor} \tag{8.20}$$

ここで，$E_{nor\_per\_nor}$ は NOR1 個の 1 回の出力の遷移 (L→H) あたりの消費エネルギー，N_{nor} は出力が遷移した NOR の総個数である．

N_{nor} を典型的な場合を想定した近似により求める．以下では近似の方法について述べる．この近似では，古い命令を保持するエントリから順番にリクエストが入力されるものと仮定する．例えば，r 個のリクエストが入力された時は，1 から r 番目までに古い命令を保持

するエントリからリクエストが入力されたと考える．そして，そのうちの k 番目に古い命令のリクエストは，$(IQS-k)$ 個の自身より新しい命令のエントリの NOR の出力を遷移させる．この個数の 1 から r 番目までの総和が，あるサイクルに r 個のリクエストが入力された時に出力が遷移する NOR の個数 $N_{nor\_per\_cycle}$ であり，以下のように求められる．

$$\begin{aligned} N_{nor\_per\_cycle} &\approx \sum_{k=1}^{r}(IQS-k) \\ &= IQS \times r - \frac{1}{2} \times r \times (r+1) \end{aligned} \tag{8.21}$$

(8.21) 式はさらに簡単な式に近似できる．図 8.8 に (8.21) 式と (8.21) 式から第 2 項以下 $-\frac{1}{2} \times r \times (r+1)$ を無視した場合の，リクエスト数に対する出力遷移する NOR の個数を示す．図 8.8 が示すように，第 2 項以下を無視しても，特にリクエスト数が小さい場合ではほとんど出力遷移する NOR の個数は変わらない．リクエスト数は図 8.7 が示すように小さい場合が多いので，第 2 項以下は無視して良いと考えられる．そのため $N_{nor\_per\_cycle}$ をさらに以下のように近似する．

$$N_{nor\_per\_cycle} \approx IQS \times r \tag{8.22}$$

最後に，出力が遷移した NOR の総個数 N_{nor} を求める．N_{nor} は各サイクルの $N_{nor\_per\_cycle}$ の総和なので，次のように近似できる．

$$\begin{aligned} N_{nor} &\approx \sum_{k=1}^{sim\_cycle} N_{nor\_per\_cycle(k)} \\ &= IQS \times \sum_{k=1}^{sim\_cycle} r_k \\ &= IQS \times N_{req} \end{aligned} \tag{8.23}$$

ここで，sim_cycle はシミュレーションの実行サイクル数，$N_{nor\_per\_cycle(n)}$ は n サイクル目で出力遷移した NOR の個数，r_n は n サイクル目で入力されたリクエスト数，N_{req} はリクエストの総数である．

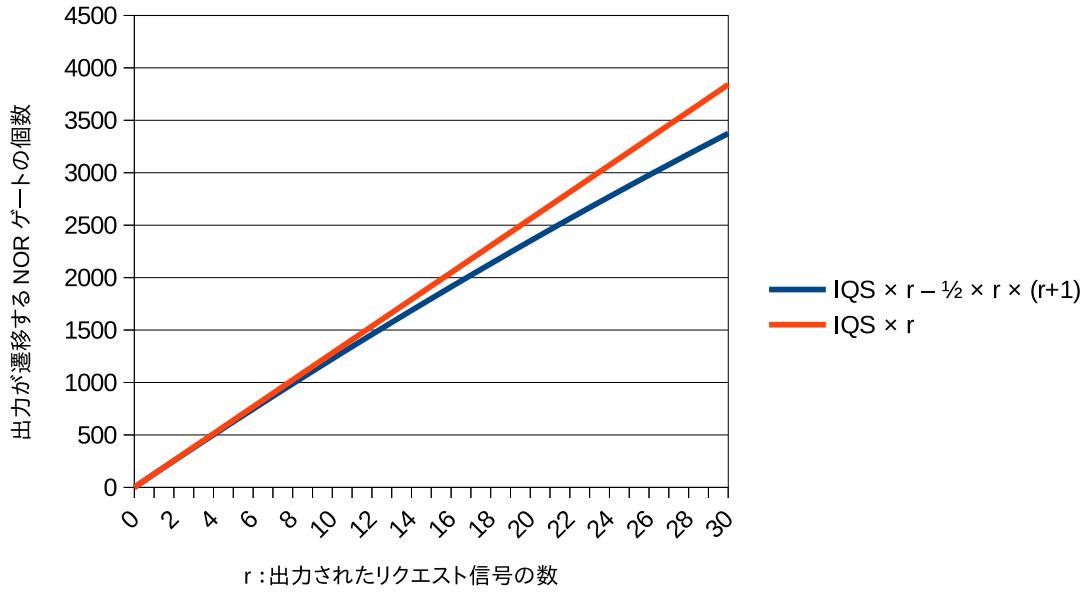


図 8.8: 入力されたリクエストの数に対する出力遷移する NOR の個数

8.3.5 タグ RAM・ペイロード RAM・PQ の消費エネルギー

タグ RAM，ペイロード RAM，PQ のそれぞれの読み出し消費エネルギー E_{read} の求め方を説明する．長いビット線の読み出し回路がダイナミック・ロジックで構成されており，プリチャージされた電荷のリークが大きい．これにより，値を読み出さない場合でも再度プリチャージする必要があるため，エネルギーを消費する．そのため，消費エネルギーは以下のように，リークによって常時消費するエネルギー ($E_{read\_constant}$) と読み出す値の個数に比例して消費するエネルギー ($E_{read\_proportional}$) に分けて計算する．

$$E_{read} = E_{read\_constant} + E_{read\_proportional} \tag{8.24}$$

$$E_{read\_constant} = E_{read\_per\_cycle(0)} \times sim\_cycle \tag{8.25}$$

$$E_{read\_proportional} = \frac{E_{read\_per\_cycle(read\_port)} - E_{read\_per\_cycle(0)}}{read\_port} \times N_{read} \tag{8.26}$$

ここで，$E_{read\_per\_cycle(n)}$ は 1 サイクルに n 個の値を読みだす時の読み出し回路全体の消費エネルギー，sim_cycle はシミュレーションの実行サイクル数，$read\_port$ は読み出しポー

ト数，N_{read} は読み出す値の総個数である．第 1 項がリークによって常時消費するエネルギーであり，第 2 項が読み出す値の個数に比例して消費するエネルギーである．

$E_{read\_per\_cycle(read\_port)}$ の求め方を説明する．読み出し回路はシングル・エンド方式であるので，0 のビットを読む時と 1 のビットを読むときでは大きく消費エネルギーが異なる．そこで，読み出される値は，平均するとそれぞれのビットは 1/2 ずつ出現すると考え，$E_{read\_per\_cycle(read\_port)}$ はすべて 1 のビットを読む時の消費エネルギーとすべて 0 のビットを読む時の消費エネルギーの平均とする．また，エイジ論理について図 8.5 で示した場合と同じく，ビット線のリピータと読み出すエントリの位置関係によって，消費エネルギーが異なる．読み出されるエントリの位置はランダムなので，リピータで区切られたビット線の各区間でそれぞれ 1 つずつ読み出しエネルギーを測定し，それらの平均をとる．

$E_{read\_per\_cycle(0)}$ の求め方を説明する．$E_{read\_per\_cycle(0)}$ はエントリの各ビットで 1 を保持しているか 0 を保持しているかでプルダウン・トランジスタがオンかオフが異なるので，リーク電荷の量も変わってくる．そのため，すべてのエントリで 1 を保持している時の消費エネルギーと 0 を保持している時の消費エネルギーの平均をとる．

8.3.6 ディスパッチの消費エネルギー

ディスパッチ時にウェイクアップ，タグ RAM，ペイロード RAM，PQ のそれぞれで消費される書き込み消費エネルギー E_{write} は，1 サイクルに n 個の値を書き込む時の書き込み回路全体の消費エネルギー $E_{write\_per\_cycle(n)}$，書き込みポート数を $write\_port$，書き込む値の総個数を N_{write} とすると，

$$E_{write} = \frac{E_{write\_per\_cycle(write\_port)}}{write\_port} \times N_{write} \tag{8.27}$$

と求められる．書き込みはダブル・エンド方式で行われるので，前サイクルで書き込んだビットと異なるビットを書き込む時にエネルギーを消費する．前サイクルからのビットの遷移の仕方は，L→L，L→H，H→H，H→L の 4 パターン存在するが，これらが全て同じ確

率で起こると考え，これらの平均を $E_{write\_per\_cycle(write\_port)}$ とする．また，書き込み回路はすべてスタティック・ロジックで構成されているので，書き込む値がない場合はエネルギーを消費しない．よって読み出しエネルギーの計算とは異なり，常時消費するエネルギーは考慮しなくて良い．

第 9 章　評価結果

9.1　遅延

各回路のクリティカル・パスの遅延を評価した結果を表 9.1〜9.3 に示す.

表 9.1: 従来手法 (base) のクリティカル・パスの遅延 (ps)

タグの放送	比較器	SR ラッチ	エイジ論理＋配線	タグ RAM	合計
88	63	37	195	90	473

表 9.2: 提案手法の比較器の遅延 (ps)

低位ビット：高位ビット	1：8	2：7	3：6	4：5	5：4	6：3	7：2	8：1
低位ビットでディスチャージ	37	41	45	49	52	56	59	62
高位ビットでディスチャージ	65	61	58	54	50	46	43	39

表 9.3: 選択論理とエイジ論理の遅延比較 (ps)

選択論理へのパスがエイジ論理を横断	エイジ論理へのパスが選択論理を横断
235	195

ウェイクアップ論理の比較器の遅延は，従来の 1 段階比較のみの回路 (base) と提案手法の 2 段階比較と 1 段階比較を切り替えられる回路の両方を評価した．提案手法については，表 9.2 に示すように，低位ビットと高位ビットの比 1:8〜8:1 のそれぞれについて 1 段階比較の時に低位ビットで 1 ビット，もしくは高位ビットで 1 ビットが不一致であった時の遅延を評価し，長い方をクリティカル・パスとした．タグの放送の遅延と SR ラッチ以降の遅延は，従来の 1 段比較のみの回路で評価している．提案手法のモデルでは，SR ラッチのイネーブル線には MUX が接続されるが，それはクリティカル・パスにはないと考えた.

選択論理とエイジ論理の遅延は，それの回路の配置によって，配線遅延が異なる．そこで，選択論理へのパスがエイジ論理を横断する場合 (図 9.1) とエイジ論理へのパスが選択

論理を横断する場合 (図 9.2) の 2 通りを測定した．表 9.3 が示すように，エイジ論理へのパスが選択論理を横断する場合の方が短かったので，本研究ではこの配置とクリティカル・パスを採用する．

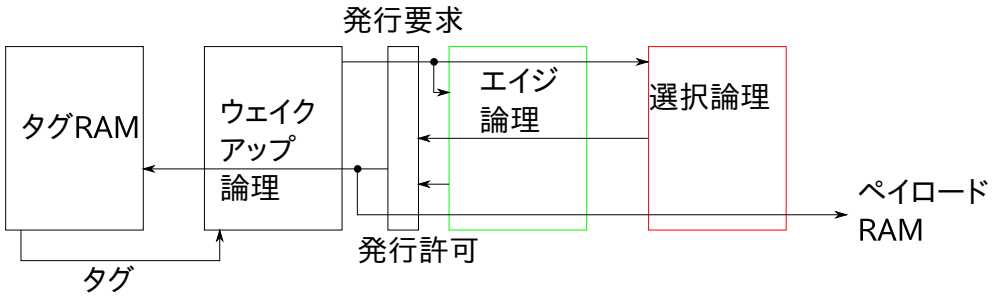


図 9.1: 選択論理へのパスがエイジ論理を横断

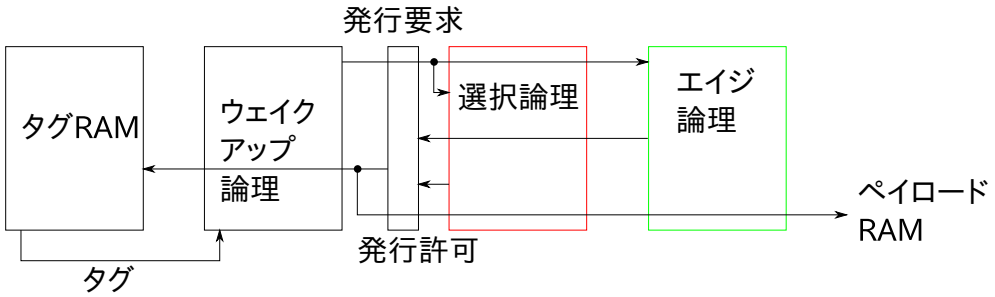


図 9.2: エイジ論理へのパスが選択論理を横断

9.2 IPC

本節では，PQ のサイズや PQ の読み出しポート数など，いくつかのパラメータによる提案手法の性能の違いについて，従来の 1 段階のタグ比較のみの発行キューと IPC で比較して評価する．

9.2.1 PQ のサイズ・PQ の容量不足時の対応による IPC の違い

PQ のサイズによる性能の違いと，PQ がいっぱいの時にディスパッチをストールさせるか (stall)，ストールさせずに PQ に入らなかった命令については 2 段階比較をするか (not

stall) の 2 種類の方法による性能の違いを評価する．図 9.3 に読み出しポート数が 6 の時の，従来手法 (base) に対する IPC の平均の低下率を示す．リオーダ・バッファのサイズがが 256 なので，存在するインフライト命令も 256 個であるため，PQ のサイズは多くても 256 でよい．また，読み出しポート数を 6 にしたのは，発行幅が 6 なので，毎サイクル古い命令が発行されると考えるという理想的な場合を想定すると，6 が最適と考えられるからである．読み出しポート数による性能の違いについては，次の節で述べる．

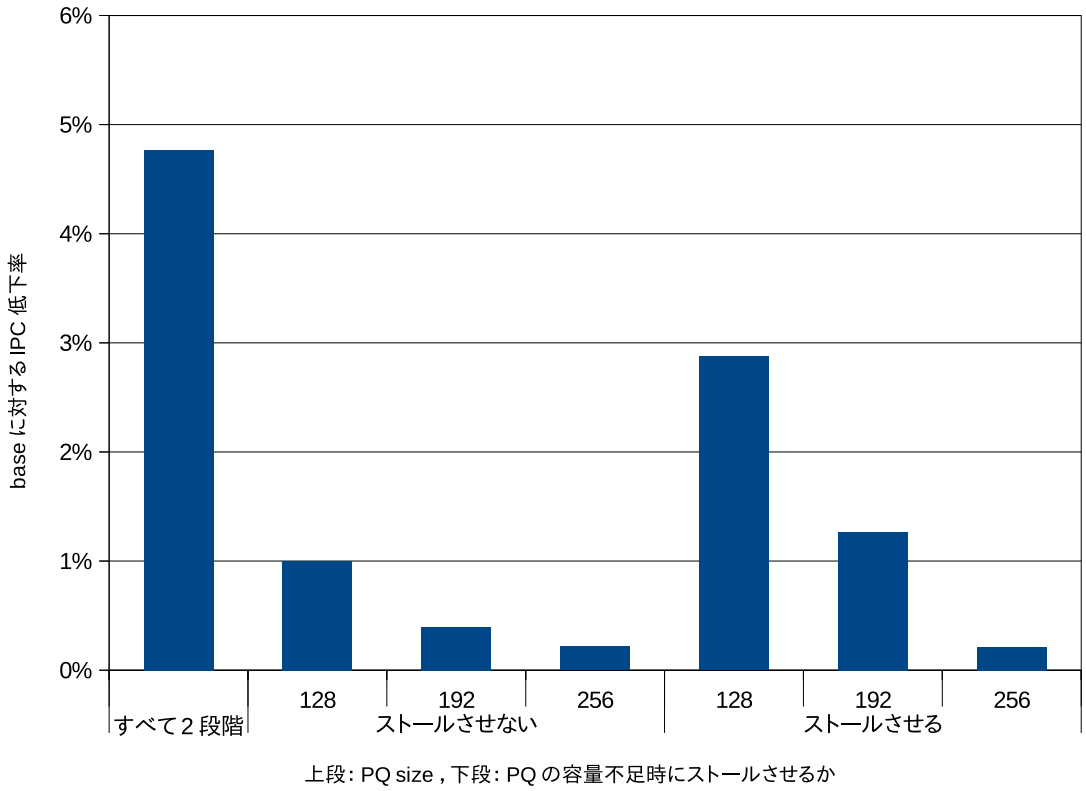


図 9.3: PQ のサイズ，PQ の容量不足時の対応による，IPC 低下率 (PQ read ports : 6)

図 9.3 を見ると，PQ が容量不足の時は明らかにストールさせない方が IPC 低下は小さくなるとわかる．よって，PQ が容量不足の時はストールさせずに PQ に入らなかった命令については 2 段階比較を行うという方法を以下の評価では適用する．

PQ のサイズについては，以下でより詳細な評価を行う．図 9.4 に PQ の容量不足時にストールさせない場合の，各ベンチマークの提案手法の IPC 低下率を示す．凡例は PQ のサ

イズを示している．図を見ると，ほとんどのベンチマークでは，PQ のサイズが 128 でも 3%以下になっている．*gcc*，*bwaves*，*lbm* は PQ のサイズが 128 では IPC が大きく低下している．これは，小さな PQ では容量不足のために，本来 1 段化すべき比較を 2 段階でしてしまっていたためと考えられる．*namd* は PQ のサイズを大きくしても IPC 低下が大きいことから，タグ比較を 1 段化すべき命令がより新しい命令にも多いということが考えられ，提案手法があまり効果的ではないプログラムと言える．

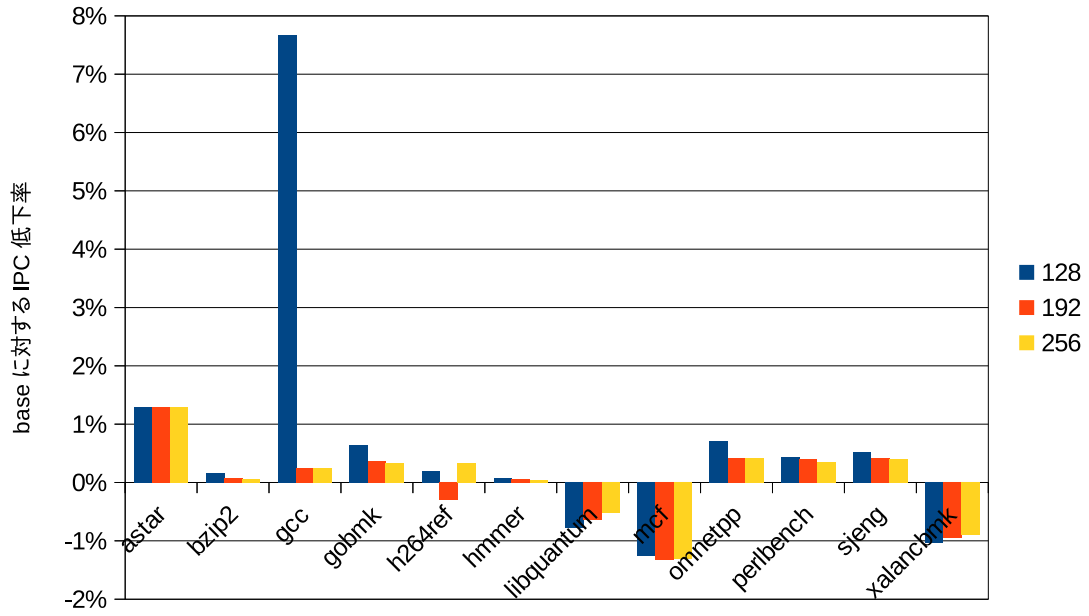
以上より，PQ のサイズが 128 では大きく性能低下するベンチマークが存在するので，消費エネルギーを評価する際の PQ のサイズは 192 か 256 とする．

9.2.2 PQ の読み出しポート数による IPC の違い

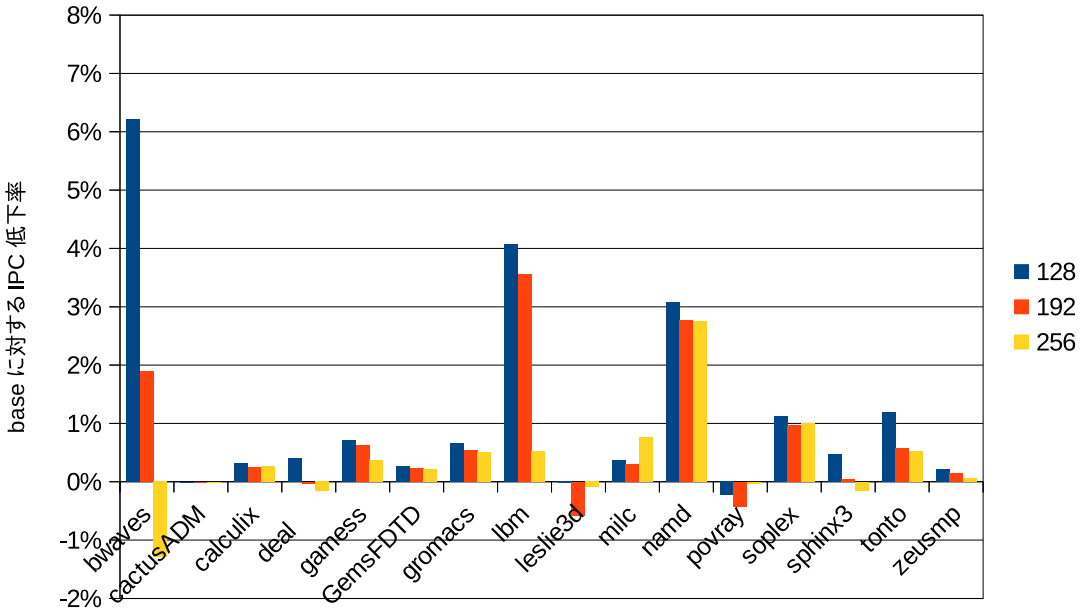
PQ の読み出しポート数に対する性能の違いを評価する．図 9.5 に PQ のサイズが 128，192，256 でポート数を変化させた時，従来手法 (base) に対する IPC の平均の低下率を示す．凡例は読み出しポート数を示している．

図 9.5 より，読み出しポート数が増えると，IPC 低下率は小さくなるとわかる．これは，読み出しポート数が増えると 1 段階比較の割合が増えるので，当然の結果と言える．

図 9.6 には，PQ のサイズを 256 としてタグ比較をすべて 2 段階で行った場合の，各ベンチマークの提案手法による IPC 低下率を示す．PQ のサイズは読み出しポート数のみの影響を評価するため最大の 256 にしている．ほとんどのベンチマークでは，読み出しポート数を増やすごとに IPC 低下率が微減している．ただし，*gcc* では 2 から 4 の間で大きく減少している．よって，消費エネルギーを評価する際の PQ の読み出しポート数は 4 か 6 とする．また，前節の評価と同様に *namd* では IPC 低下の抑制効果が小さいと言える．



(a) int



(b) fp

図 9.4: PQ のサイズに対する IPC 低下率 (PQ read ports：6)

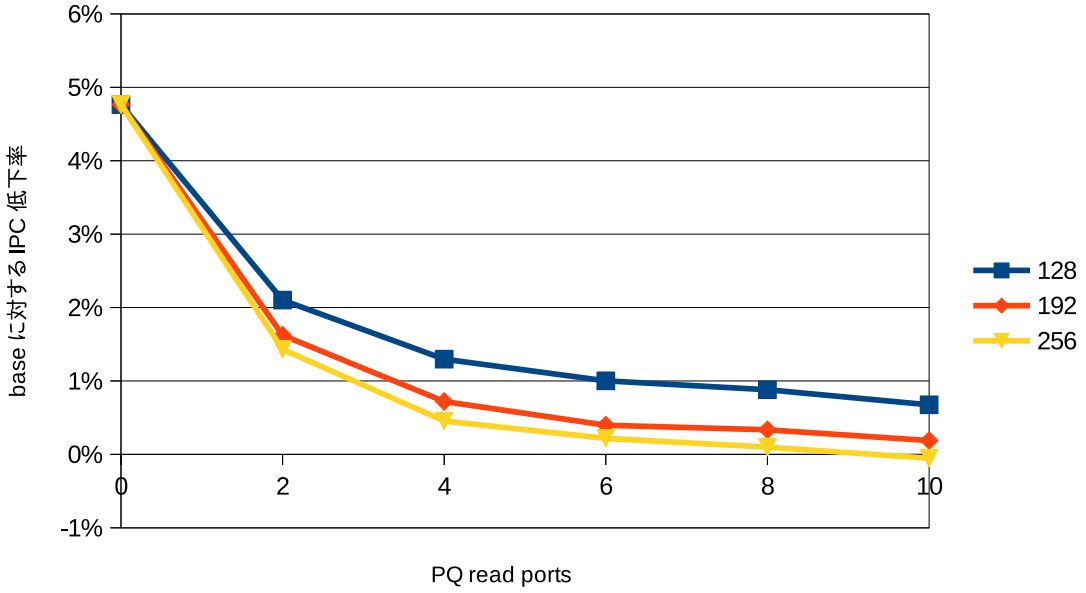
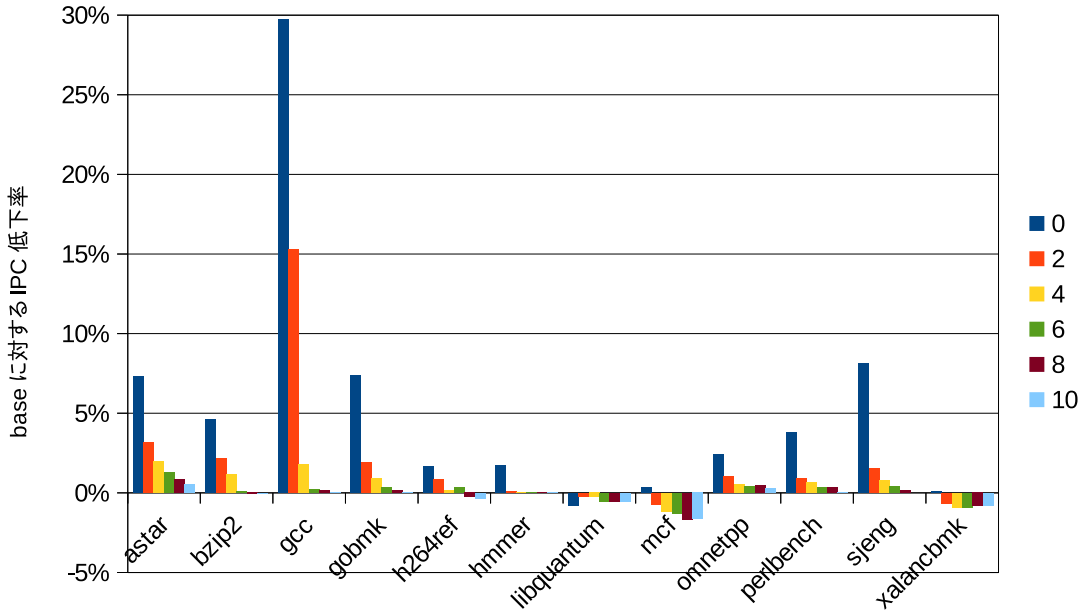
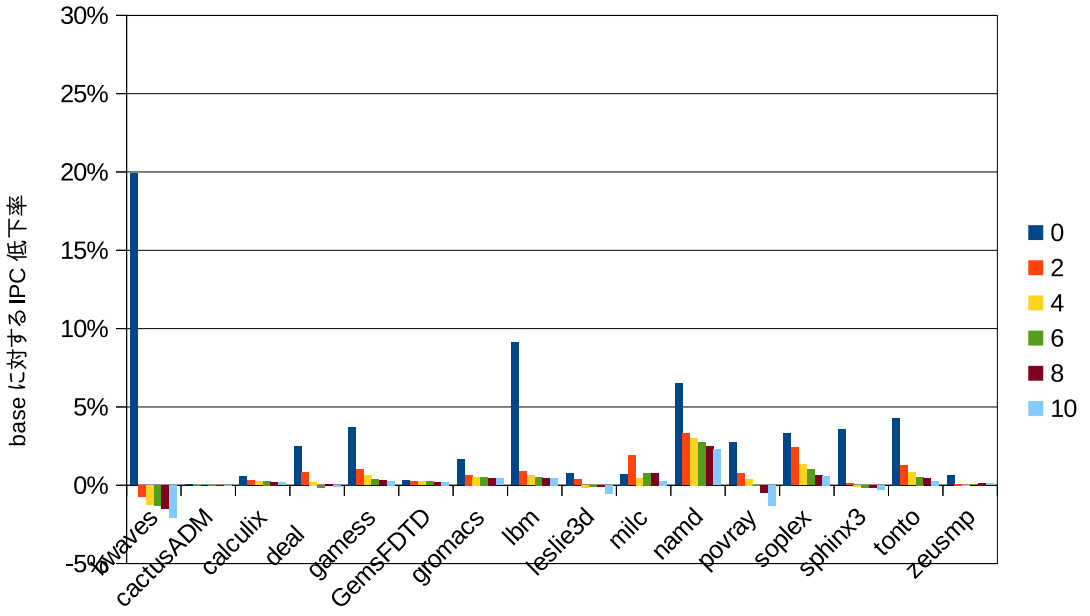


図 9.5: PQ の読み出しポート数に対する IPC 低下率



(a) int



(b) fp

図 9.6: PQ の読み出しポート数に対する IPC 低下率 (PQ size：256)

9.3 消費エネルギー

本節では，まず各回路の動的な消費エネルギーを求め，最後に発行キュー全体の消費エネルギーを求めて，従来手法に対する PQ 方式の消費エネルギー削減効果を評価する．静的な消費エネルギーについては最後に述べる．

各回路の消費エネルギーの評価を次のように行った．まず HSPICE によるシミュレーションにより 1 サイクルあたりの消費エネルギーを求める．そして，8.3 節で示した消費エネルギーの計算式を用いて，SimpleScalar をベースにしたシミュレータで評価された各回路の動作回数を乗じ，各ベンチマーク・プログラムごとの総消費エネルギーを求める．

評価するのは，従来手法の 1 段階のタグ比較のみのモデル，2 段階のタグ比較のみのモデル，PQ 方式を用いたモデル (PQ のサイズ : 128，192 または 256，読み出しポート数 : 6 または 16) とする．従来手法の 1 段階のタグ比較のみのモデルを C(1)，2 段階比較のみのモデルを C(2)，PQ のサイズ X，読み出しポート数を Y とする PQ 方式を用いたモデルを P(X,Y) と表記することとする．

HSPICE によるシミュレーションで用いるクロック・サイクル時間はウェイクアップ論理，エイジ論理，タグ RAM の遅延の合計である．この内ウェイクアップ論理の遅延は，表 9.2 に示したように低位ビットと高位ビットの比によって異なる．そのため，ウェイクアップ論理の消費エネルギーの測定では，その比によって異なるクロック・サイクル時間を用いる．それ以外の回路の消費エネルギーの測定では．従来手法のウェイクアップ論理を適用した時のクロック・サイクル時間を用いる．

9.3.1 ウェイクアップ論理

1 サイクルあたりの消費エネルギー

図 9.7 から図 9.11 に提案手法において，種々の比較段数と比較結果の場合の 1 サイクルあたりのウェイクアップ論理の消費エネルギーを示す．これらの評価は発行幅分のタグ

が放送され，すべてのエントリのすべてのソース・タグでタグ比較が行われた時の消費エネルギーである．横軸の比は低位ビットと高位ビットの比を表している．各グラフ左端の conventional は従来の 1 段階のタグ比較しかできないモデルである．棒グラフは 5 つの部分に分かれており，以下に各部分について説明する．

- discharge：比較器のマッチ線のディスチャージ (図 7.3 の青で囲われた部分で測定)．
- tag line ： タグを全エントリに放送するタグ線
- NAND(1)：マッチ線の出力側に接続された NAND または NOT ゲート (図 7.3 の黄で囲われた部分)．
- NAND(2)：2 段階比較で高位ビットのプリチャージ・クロックをゲーティングする NAND ゲート (図 7.3 の緑で囲われた部分)．

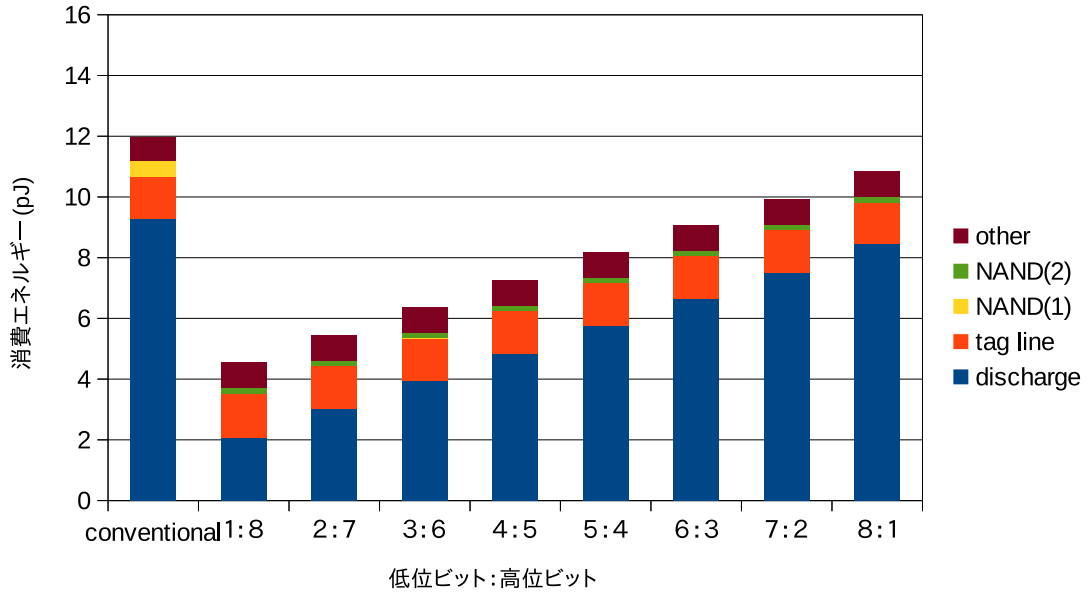


図 9.7: 2 段階比較・低位ビットが不一致の時の消費エネルギー

以下では，図 9.7 から図 9.11 の結果について考察する．特に，個別に消費エネルギーを測った回路の部位のうち，モデルごとの変化があった discharge，NAND(1)，NAND(2) の

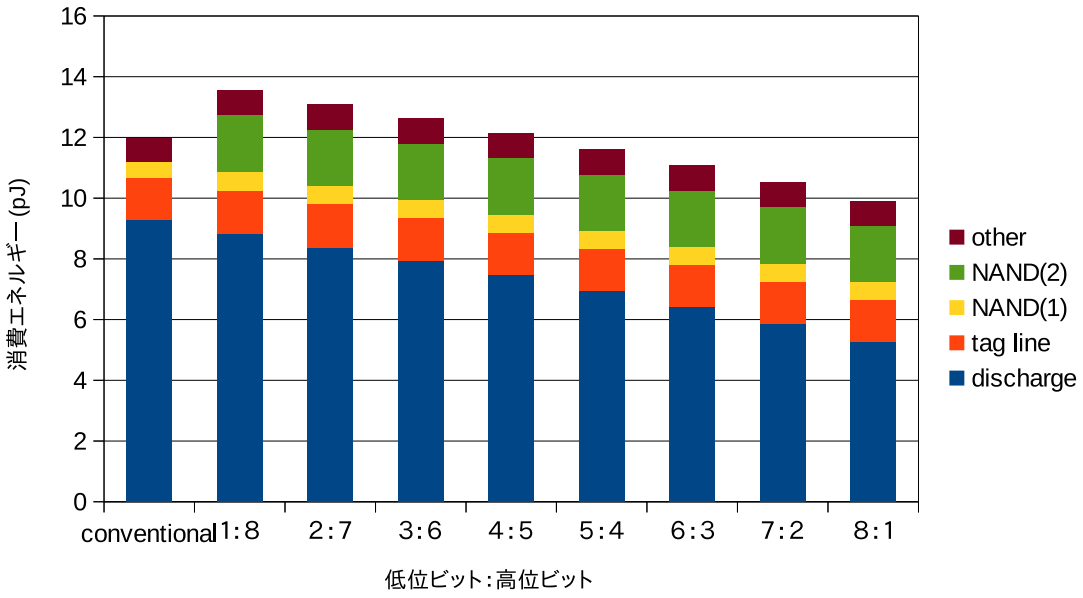


図 9.8: 2 段階比較・低位ビットのみ一致した時の消費エネルギー

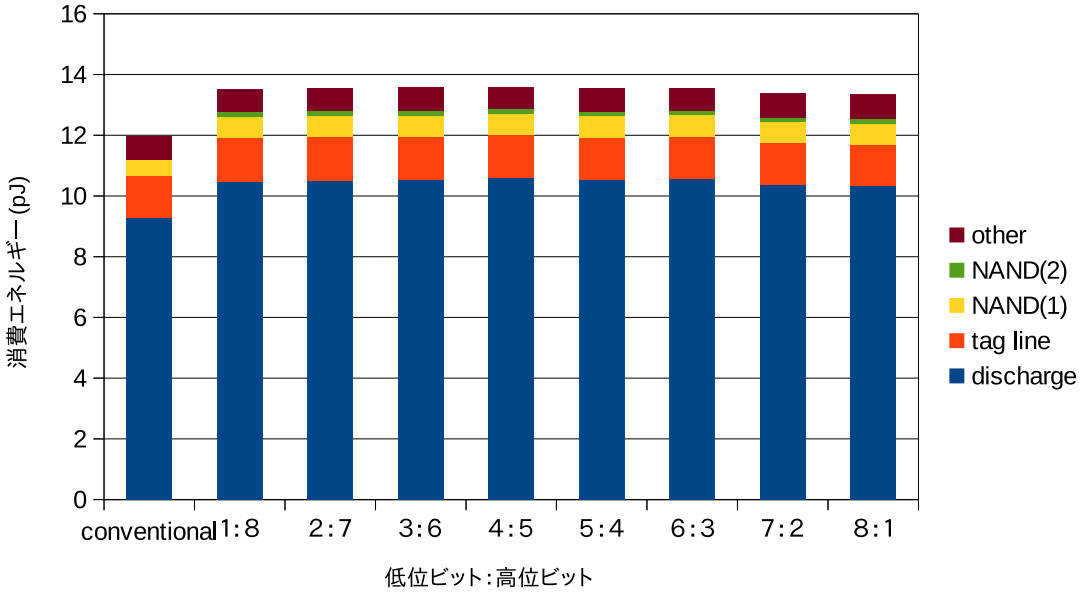


図 9.9: 1 段階比較・低高両方のビットで不一致の時の消費エネルギー

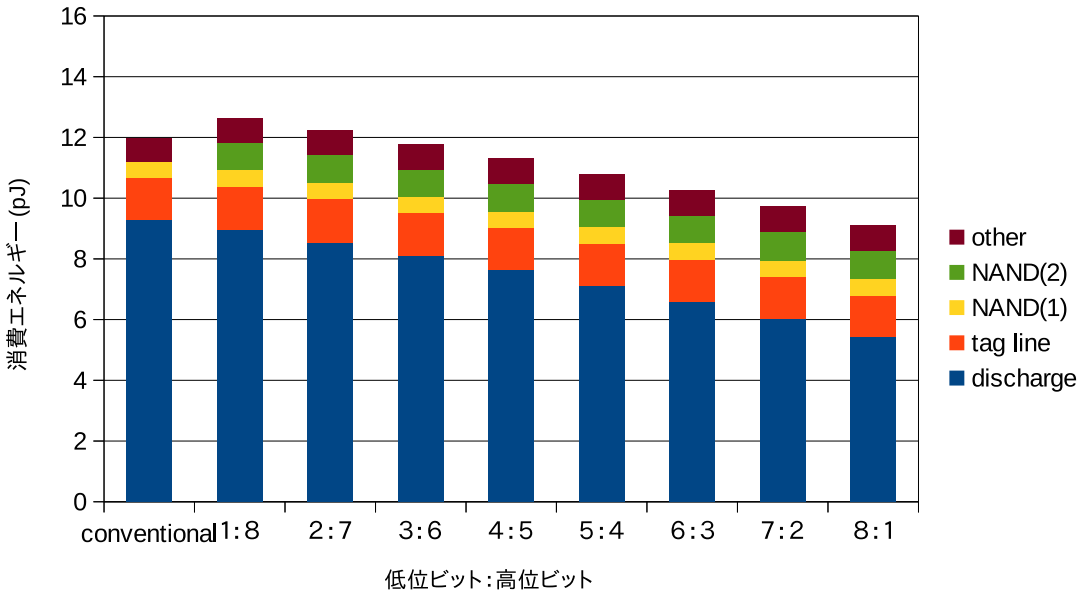


図 9.10: 1 段階比較・低位ビットのみ一致した時の消費エネルギー

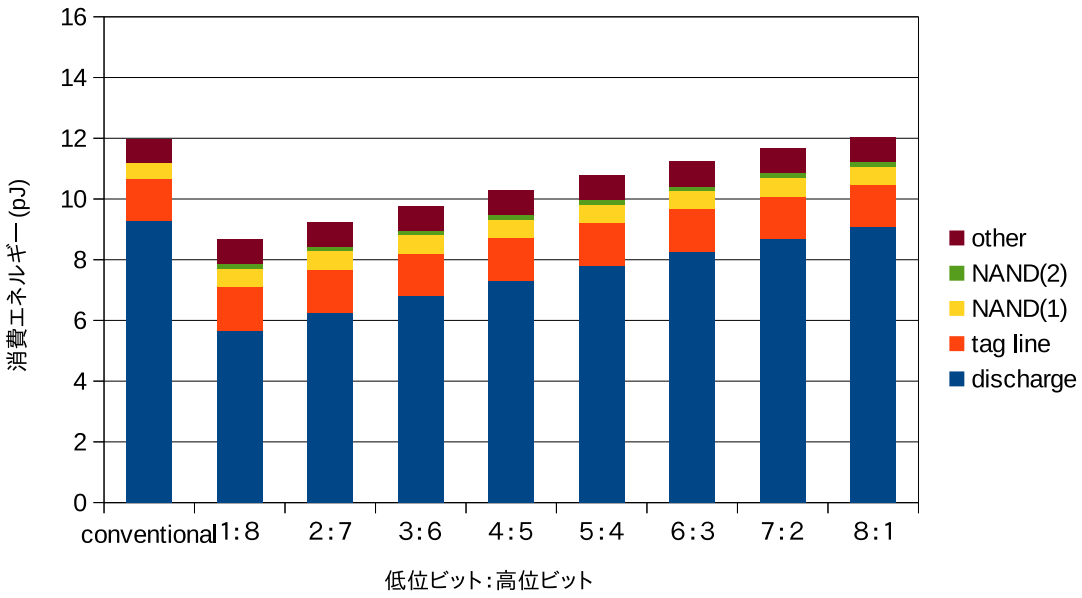


図 9.11: 1 段階比較・高位ビットのみ一致した時の消費エネルギー

3 つの部位について述べる．

- discharge

  図 9.7 より，2 段階比較でタグの低位ビットが不一致の場合は，低位ビットのビット数が少なく高位ビットのビット数が多いほど，消費エネルギーをより多く削減できていることがわかる．これは，タグの 2 段階比較では低位ビットが不一致の場合，比較器の低位ビット部分ではエネルギーが消費されるが，高位ビット部分では比較は行われないので，エネルギーが消費されないためである．一方，比較を 1 段化した時は，図 9.9 が示すように，高位ビットの比較器でもエネルギーが消費されるため，従来手法との消費エネルギーの差はほとんどない．

  図 9.8，9.10，9.11 では一致した方のビットの割合が大きいほど，消費エネルギーが小さくなっている．これは，一致した方のマッチ線はディスチャージされないため，エネルギーを消費しないからである．

- NAND(1)

  図 9.7 の場合においてそれ以外の場合よりこの消費エネルギーが小さいのは，2 段階比較で低位ビットが不一致の場合高位ビットマッチ線はプリチャージされないため，NAND(1) の出力は High のままであるためである．一方他の場合は，高位ビットもプリチャージされるので，NAND(1) の出力が L に遷移してまた H に戻ることでエネルギーを消費する．

- NAND(2)

  図 9.7，9.9，9.11 の場合では，この消費エネルギーはほとんど消費されていない．この理由は低位ビットが不一致ならば出力は遷移しないからである．

  図 9.8 の場合では，この消費エネルギーは大きい．この理由はこのゲートの出力が遷移し，かつその出力信号が MUX を通って高位ビットのプリチャージ・クロックとな

るためである．図 9.10 の場合がこれより小さいのは，ゲートの出力は遷移するが，その出力信号は高位ビットのプリチャージ・クロックにはならずに MUX で遮断されるためである．

消費エネルギーの計算

8.3 節で示した式を用いて，消費エネルギーを計算する．図 9.7 から図 9.11 に示した tag line の消費エネルギーを発行幅 6 で除したものを 1 本のタグ線の消費エネルギー，discharge，NAND(1)，NAND(2) の消費エネルギーを加えて比較器の個数 1536(発行幅 × キューサイズ ×2) で除したものを 1 個の比較器の消費エネルギー，other の消費エネルギーをソース・タグの数 256 で割ったものをその他の 1 エントリあたりの消費エネルギーとして，各ベンチマーク・プログラムでの動作回数を乗じて計算する．

図 9.12 に，従来の 1 段化のみの手法 (C(1)) に対する，ウェイクアップ論理の全ベンチマーク平均の消費エネルギー削減率を示す．平均では，どのモデルにおいても低位ビットと高位ビットの比が 3 : 6 の時が最も削減率が高く，その削減率は 37%程度である．提案手法のモデルにおいて，読み出しポート数が 6 よりも 4 の時のほうが 2 段階でタグ比較をすることが増えるので，消費エネルギーをより削減することができている．

9.3.2 選択論理

1 サイクルあたりの消費エネルギー

選択論理は 8.3 節で述べたように，発行要求の数が発行幅未満か発行幅以上かで場合分けをして消費エネルギーを評価する．図 9.13 に 1，2，3，4，5，6 番目のエントリから発行要求が出力されていて，残り 1 つが 6〜128 番目のエントリから出力される時の消費エネルギーを示す．左端のグラフは発行要求がない時の消費エネルギーである．

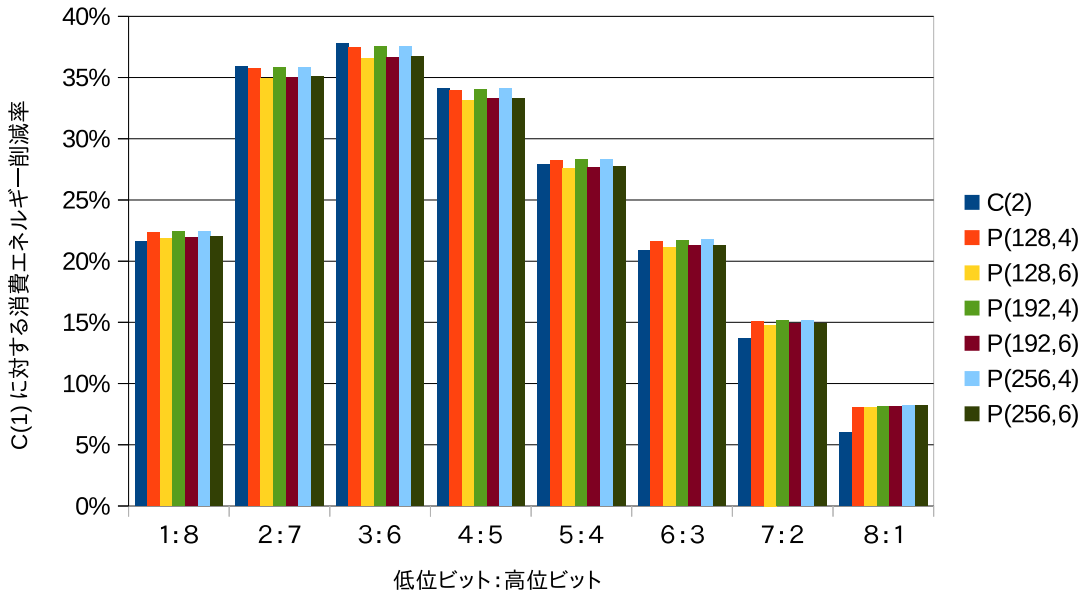


図 9.12: C(1) に対するウェイクアップ論理の平均消費エネルギー削減率

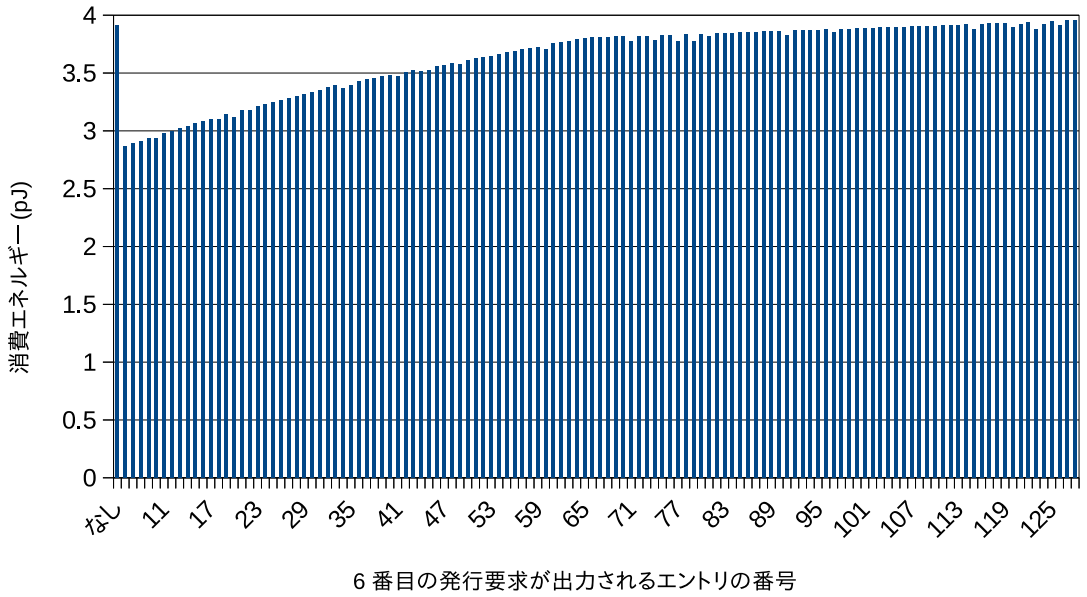


図 9.13: 1〜5 番目のエントリで発行要求が出力され，6 番目の発行要求が 6〜128 番目のエントリから出力される時の選択論理の消費エネルギー

消費エネルギーの計算

8.3 節で示した式を用いて，消費エネルギーを計算する．発行要求が 0 のときの消費エネルギーを発行要求が発行幅未満の時の消費エネルギーとして，図 9.13 で示した各値の平均を発行幅以上の時の消費エネルギーとして計算に用いる．

図 9.14 に従来の 1 段化のみの手法 (C(1)) で正規化した選択論理の全ベンチマーク平均の消費エネルギーを示す．より多くのタグ比較を 1 段化できるモデルほど実行時間が短くなるので，消費エネルギーは小さくなっている．

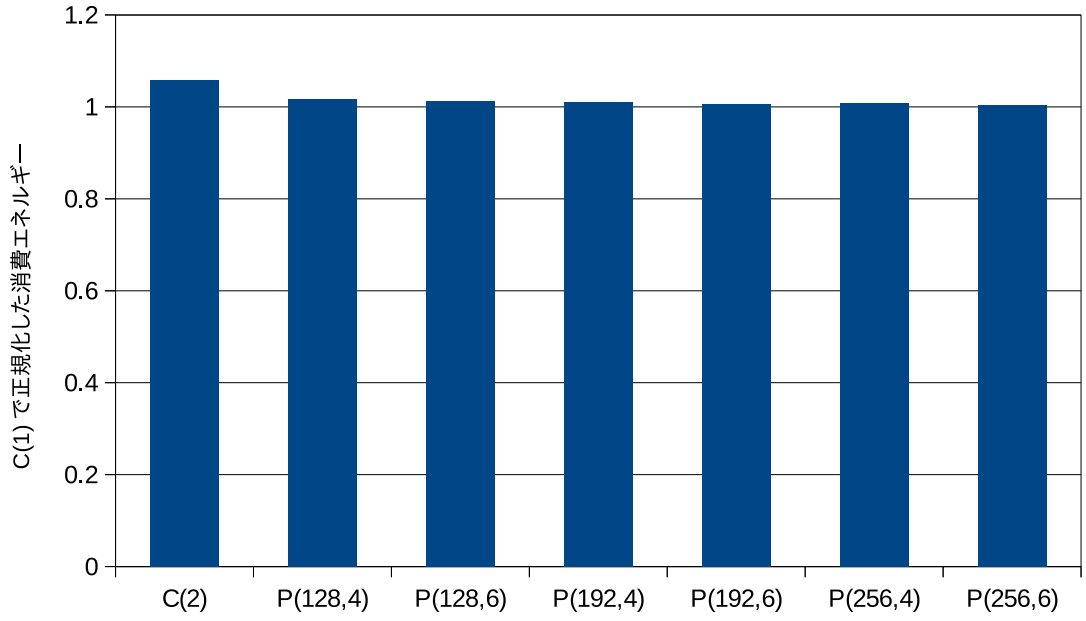


図 9.14: C(1) で正規化した選択論理の消費エネルギー

9.3.3 エイジ論理

1 サイクルあたりの消費エネルギー

エイジ論理は 8.3 節で述べたように，リクエスト線，コンフリクト線，NOR と分けて評価する．図 9.15 に消費エネルギーの計算に必要な各部分の単位量あたりの消費エネルギーを示す．リクエスト線，コンフリクト線は 1 エントリあたり，NOR は 1 個あたりの消費エネルギーである．

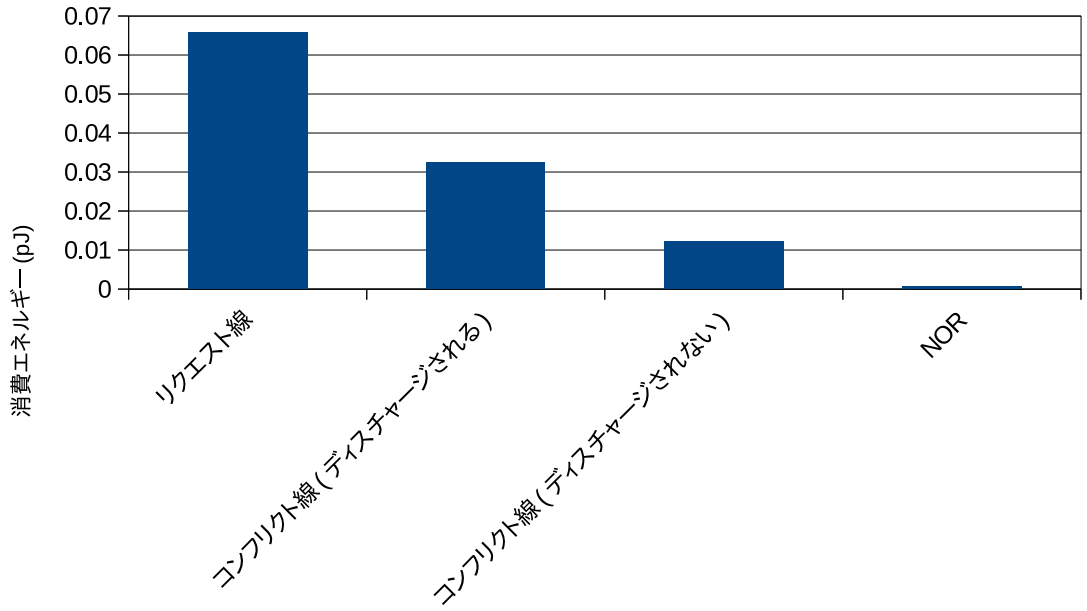


図 9.15: エイジ論理のエネルギー計算に用いる各部位の消費エネルギー

消費エネルギーの計算

8.3 節で示した式を用いて，消費エネルギーを計算する．図 9.16 に従来のタグ 1 段階比較のみのモデル (C(1)) で正規化したエイジ論理の全ベンチマーク平均の消費エネルギーを示す．モデルごとの消費エネルギーの違いはほとんど見られなかった．

9.3.4 タグ RAM・ペイロード RAM・PQ

1 サイクルあたりの消費エネルギー

1 サイクルあたりのタグ RAM とペイロード RAM の読み出しの消費エネルギーを図 9.17 に示す．タグ RAM のビット幅が 9 ビットであるのに対してペイロード RAM のビット幅は 44 なので，読み出す値 1 つあたりの消費エネルギーはおおよそそのビット幅に比例して大きくなっている．しかし，読み出す値がなくても毎サイクル消費するエネルギーは，ビット幅と比例していない．この理由は，タグ RAM のみバンク化されているために，そのようにされていない場合より配線長が長くなり，リーク電流が大きいからである．

PQ の読み出しの消費エネルギーを図 9.18 に示す．PQ のサイズが大きくなると，ビット

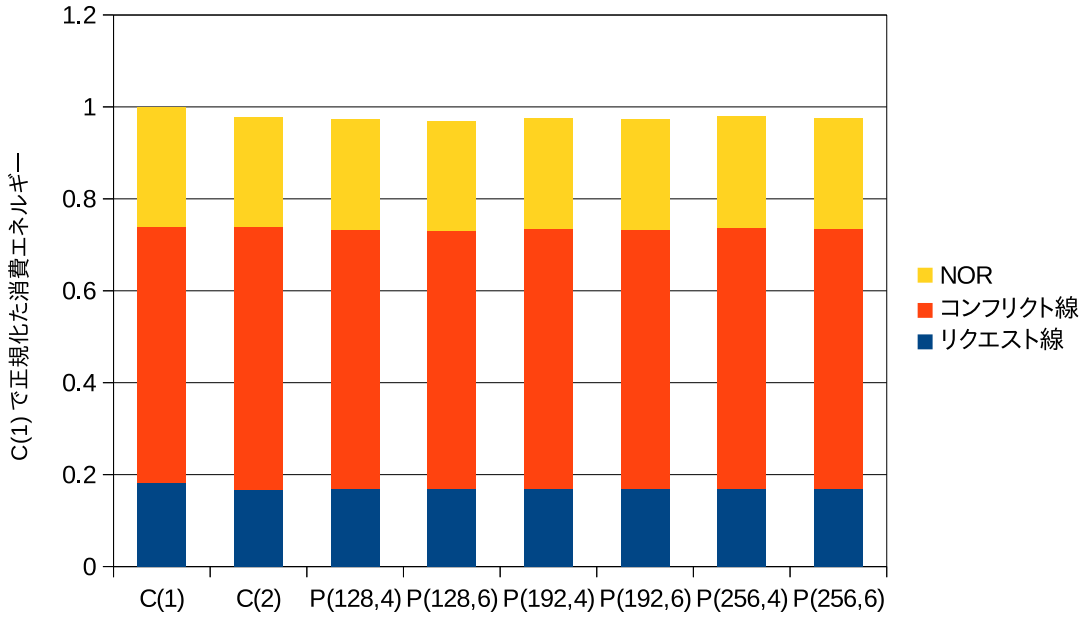


図 9.16: C(1) で正規化したエイジ論理の消費エネルギー

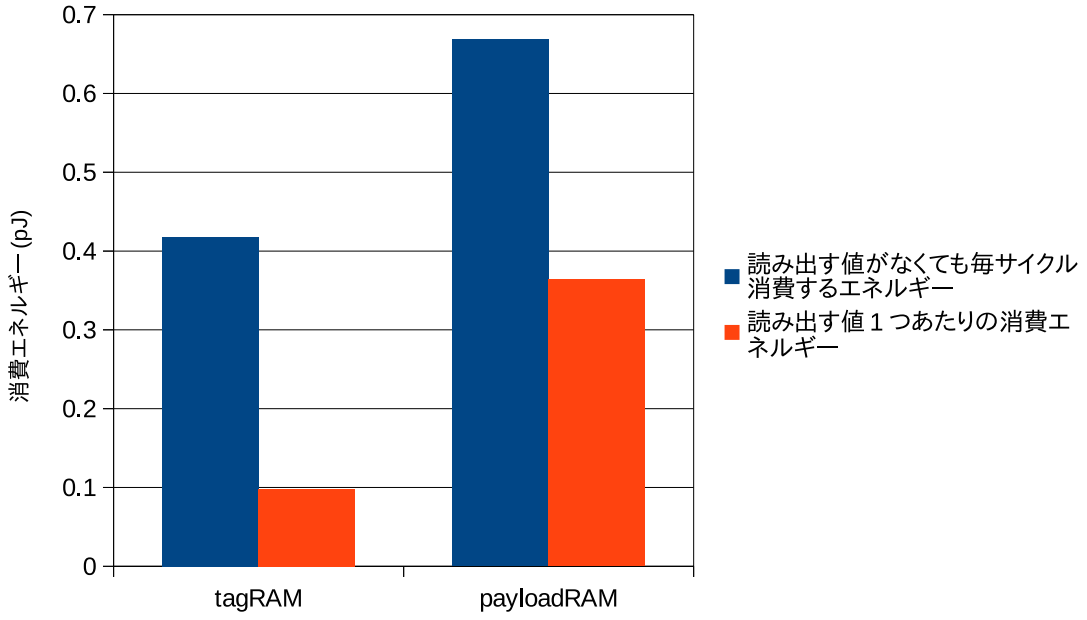


図 9.17: 1 サイクルあたりのタグ RAM，ペイロード RAM の読み出しの消費エネルギー

線が長くなる．そのため，リークによって毎サイクル消費するエネルギーも値を読み出す時の消費エネルギーも大きくなる．ポート数を増やすと，ビット先の本数が増える．そのため，リークによる消費エネルギーが大きくなる．

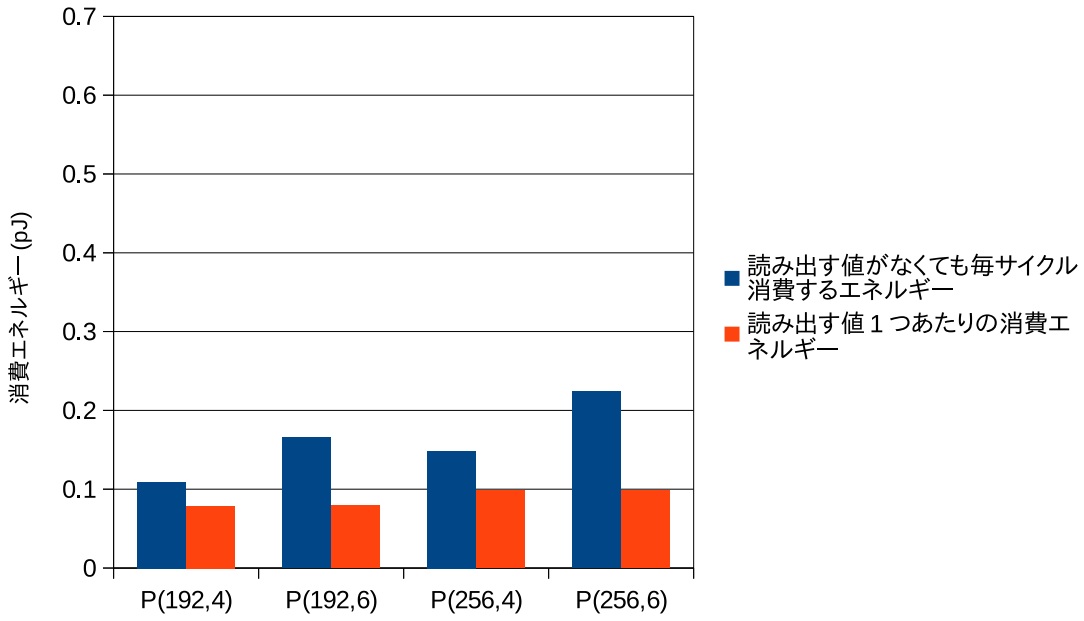


図 9.18: 1 サイクルあたりの PQ の読み出しの消費エネルギー

消費エネルギーの計算

8.3 節で示した式を用いて，消費エネルギーを計算する．読み出す値がなくても毎サイクル消費するエネルギーには実行サイクル数を，読み出す値 1 つあたりの消費エネルギーには読み出した値の個数を乗じる．

図 9.19 に従来のタグ 1 段階比較のみのモデル (C(1)) で正規化した，タグ RAM，ペイロード RAM，PQ の読み出し時の全ベンチマーク平均の消費エネルギーを示す．PQ によって，RAM の読み出しの消費エネルギーが大きく増加している．

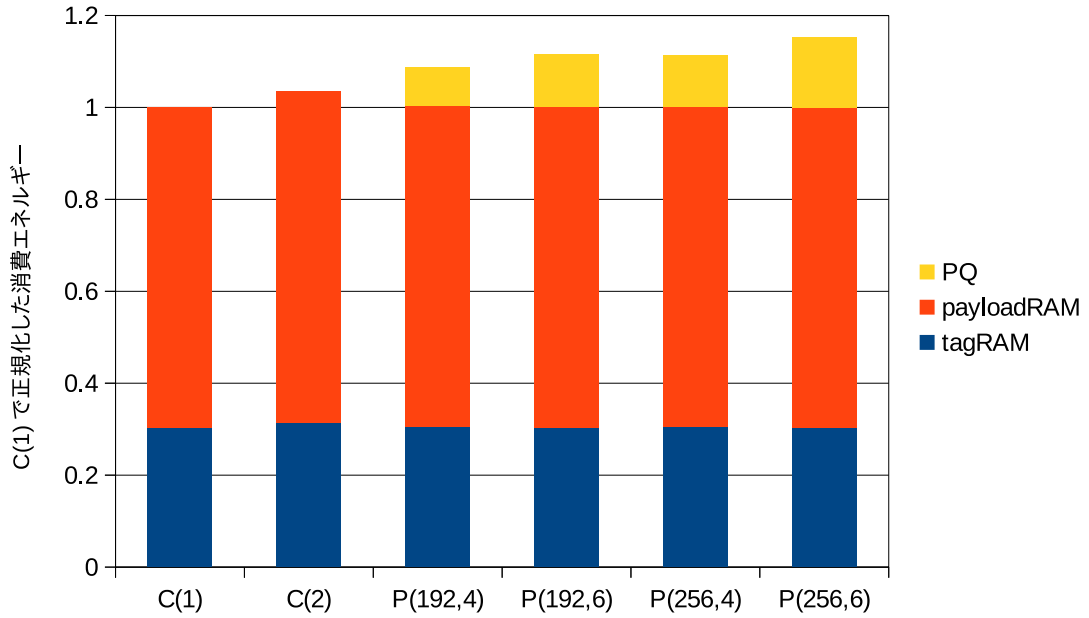


図 9.19: C(1) で正規化したタグ RAM，ペイロード RAM，PQ の読み出し時の消費エネルギー

9.3.5 ディスパッチ

1 サイクルあたりの消費エネルギー

1 サイクル・1 つの値の書き込みあたりの，ウェイクアップ論理，タグ RAM，ペイロード RAM のディスパッチの消費エネルギーを図 9.20 に，PQ のサイズ 192，256 での PQ のディスパッチの消費エネルギーを図 9.21 に示す．

消費エネルギーの計算

8.3 節で示した式を用いて，消費エネルギーを計算する．図 9.22 に従来のタグ 1 段階比較のみのモデル (C(1)) で正規化した，タグ RAM，ペイロード RAM，ウェイクアップ論理，PQ のディスパッチ時の全ベンチマーク平均の消費エネルギーを示す．

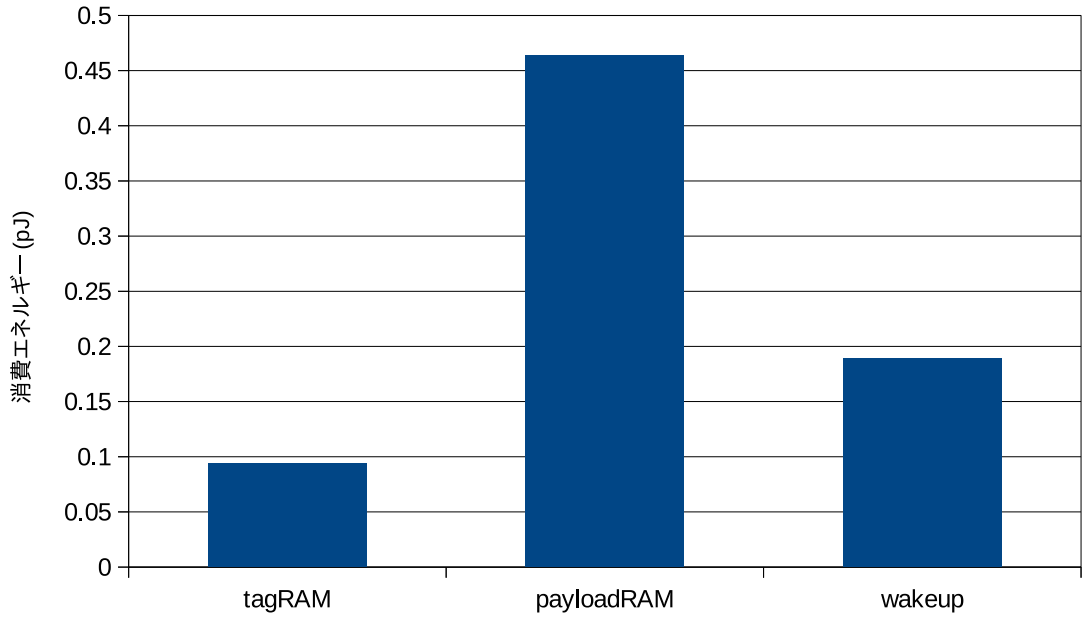


図 9.20: 1 サイクルあたりのディスパッチ時の消費エネルギー

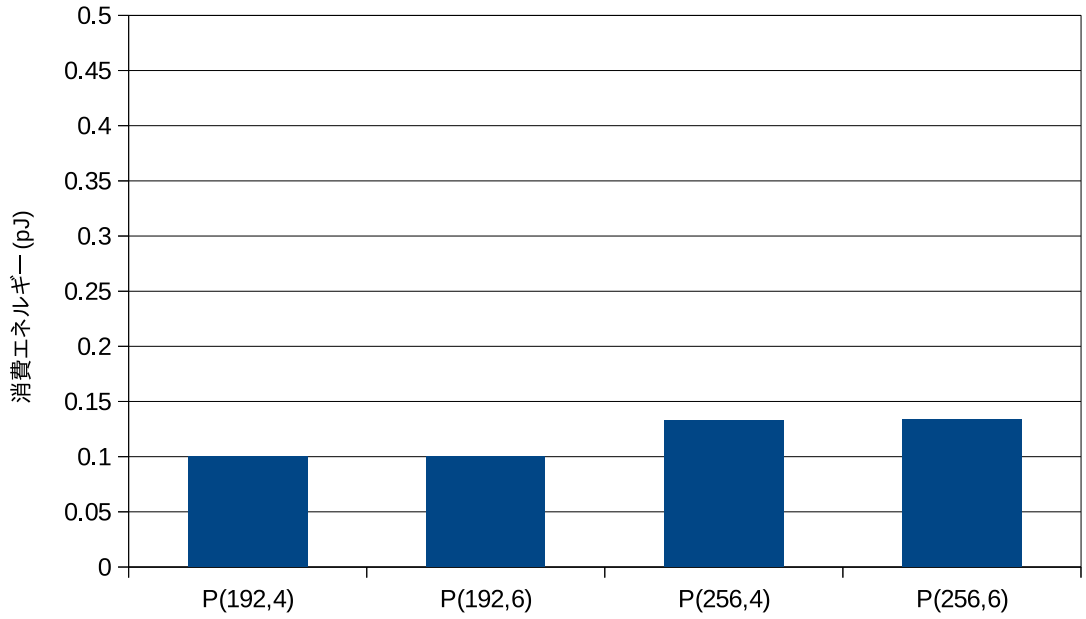


図 9.21: 1 サイクルあたりの PQ のディスパッチ時の消費エネルギー

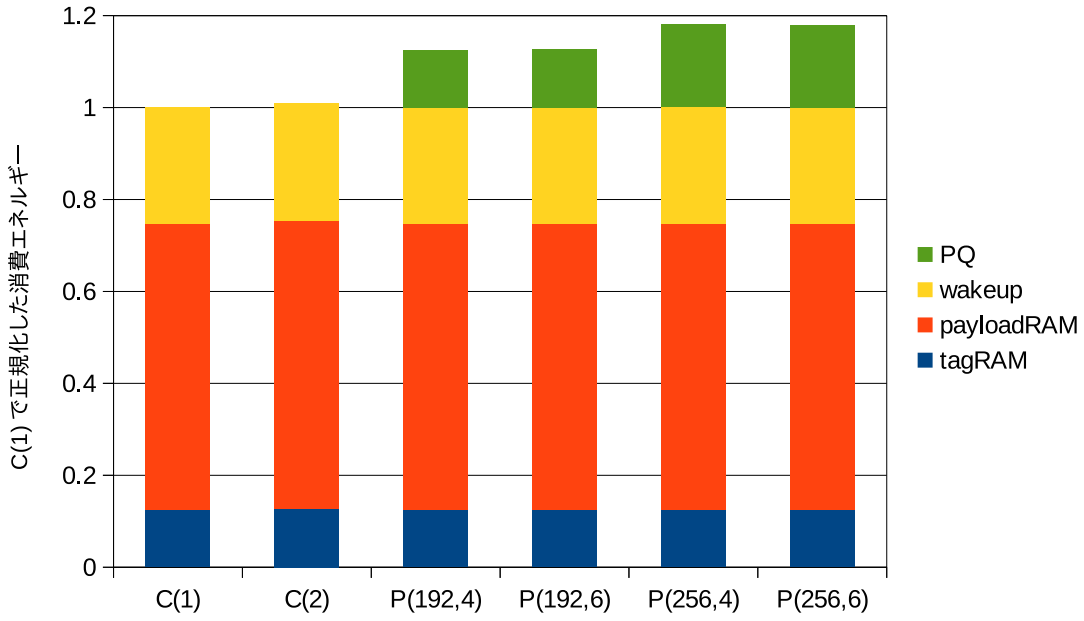


図 9.22: C(1) で正規化したディスパッチ時の消費エネルギー

9.3.6 発行キュー全体

図 9.23 に従来のタグ 1 段階比較のみのモデル (C(1)) で正規化された各回路の全ベンチマーク平均の動的な消費エネルギーを示す.

図 9.23 に示すように，2 段階比較によってウェイクアップ論理の消費エネルギーは小さくなり，全体の消費エネルギーも小さくなることがわかる．例えば，PQ のサイズが 192，読み出しポート数が 4 の時 (P(192,4)) の場合，IQ 全体で削減率は 6.8%となった．

このように削減率が小さくなった理由は，従来手法でウェイクアップ論理の消費エネルギーが IQ の中で占める割合が小さいことである．小林の修論 [4] ではウェイクアップ論理は IQ の 36.8%を占めるとされているが，本研究の測定では 24.3%という結果になった．ウェイクアップ論理の消費エネルギーが占める割合が小さい最大の原因は，エイジ論理の追加である．エイジ論理は消費エネルギーが大きく IQ 全体の 24.1%を占めているが，小林の研究ではエイジ論理の使用が想定されていないので含まれていない．また，他の原因して選択論理が大きいということもあげれれる．本研究の評価では，選択論理はウェイクアッ

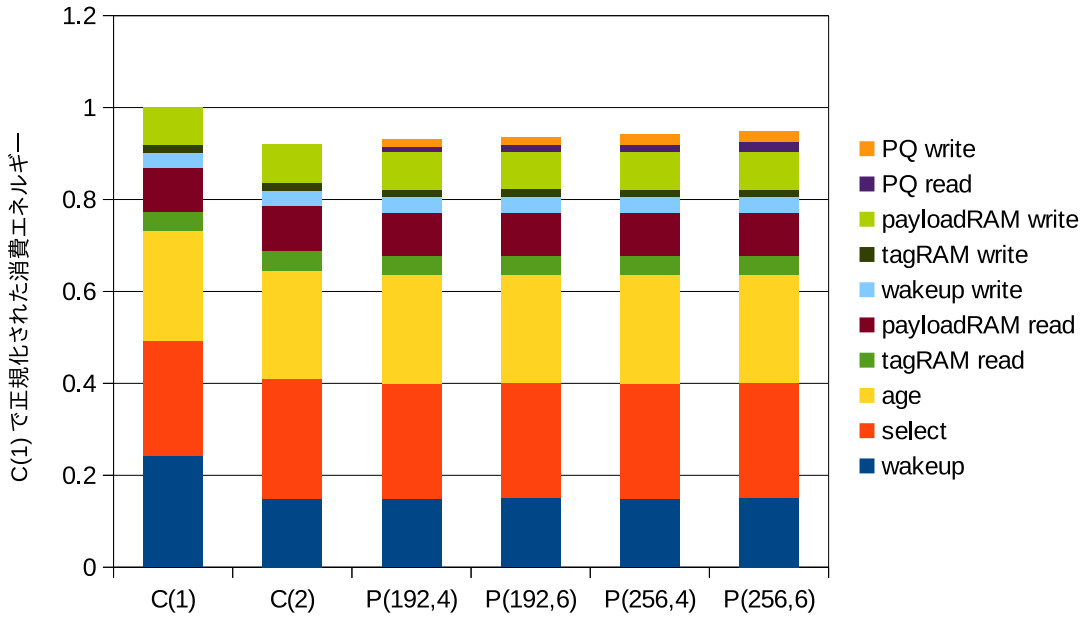


図 9.23: C(1) で正規化された発行キューの各回路の消費エネルギー

プ論理以上の 24.9%を占めているが，小林の評価ではウェイクアップ論理より 10%近く小さい 25.1%となっている．

これらの原因を解消する方策として，以下のようなものが考えられる．

- エイジ論理のコンフリクト線のプルダウン・トランジスタのサイズを縮小：これにより，コンフリクト線の配線の寄生容量が小さくなるのでプリチャージに必要なエネルギーが小さくなる．また，リーク電流も小さくなる．ただし，エイジ論理は IQ のクリティカル・パス上にあるので，遅延が増加する．

- 選択論理をスタティック・ロジックで構成：本研究の選択論理の加算器はすべてダイナミック・ロジックで構成されているので，ある加算器で前サイクルと今サイクルの出力結果が同じでも，プリチャージによってエネルギーを消費する場合がある．これを防ぐために，すべての加算器をスタティック・ロジックにすることが考えられる．ただし，これにより回路面積は増大する．

- IQ を 2 分割：Alpha21464 [22] では選択論理とエイジ論理について IQ を 2 分割し，総

発行幅 8 に対してそれぞれ 4 つずつ発行する命令を選択するような構成になっている．この方法を取り入れることで，エイジ論理のリクエスト線とコンフリクト線の配線が短くなり，消費エネルギーを小さくでき，遅延も短くなると考えられる．また，選択論理でも，加算器の総個数が減りプレフィックス・サムの出力桁数も減るので，エネルギーを小さくできると考えられる．ただし性能については，低下する可能性と向上する可能性がある．性能が低下する理由として考えられるのは，分割した IQ の片方の領域だけに発行可能命令が集中した場合，一度に発行される命令数が，分割していない IQ と比べて減少することである．性能が向上する理由として考えられるのは，最も古い発行可能命令をより多く選択できることである．

9.3.7 静的な消費エネルギー

以下では静的な消費エネルギーの評価について述べる．図 9.24 に各回路の 1 サイクルあたりの静的な消費エネルギーを示す．図 9.25 には PQ の 1 サイクルあたりの静的な消費エネルギーを示す．

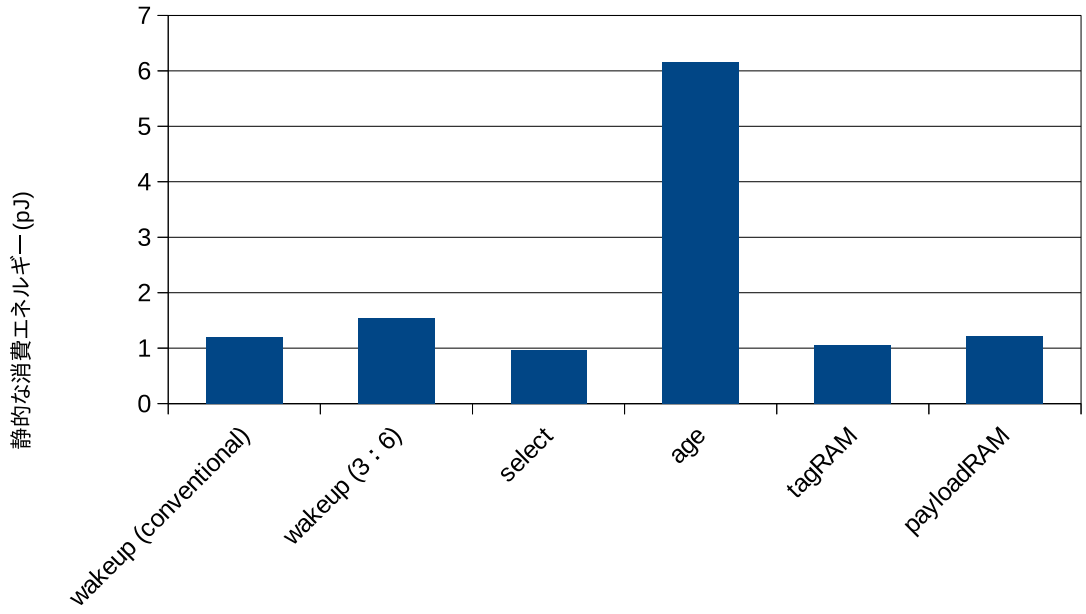


図 9.24: 1 サイクルあたりの発行キューの各回路の静的な消費エネルギー

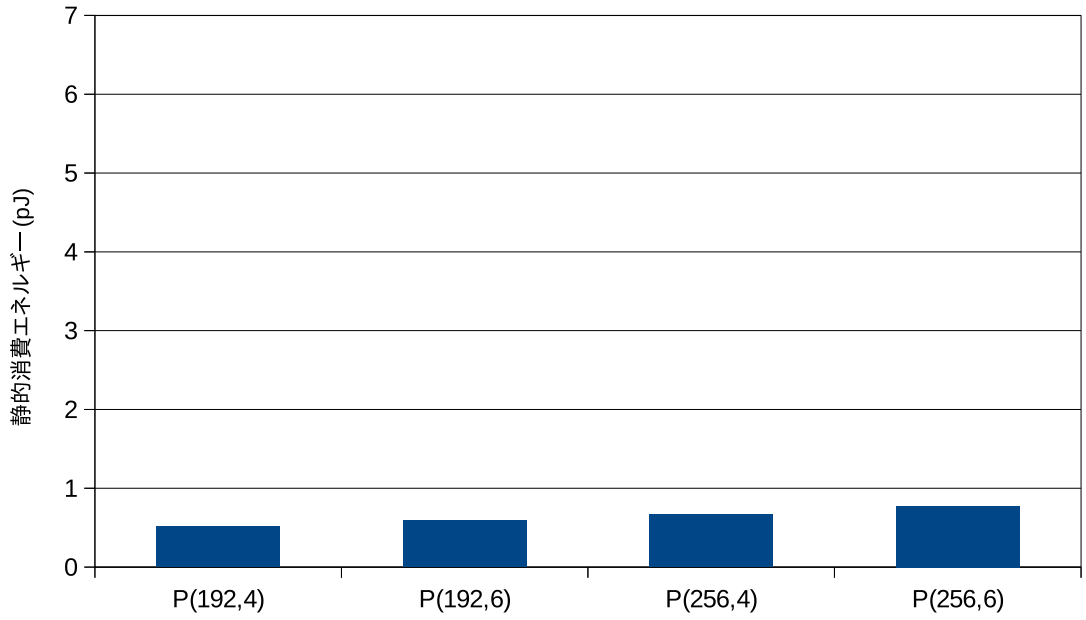


図 9.25: 1 サイクルあたりの PQ の静的な消費エネルギー

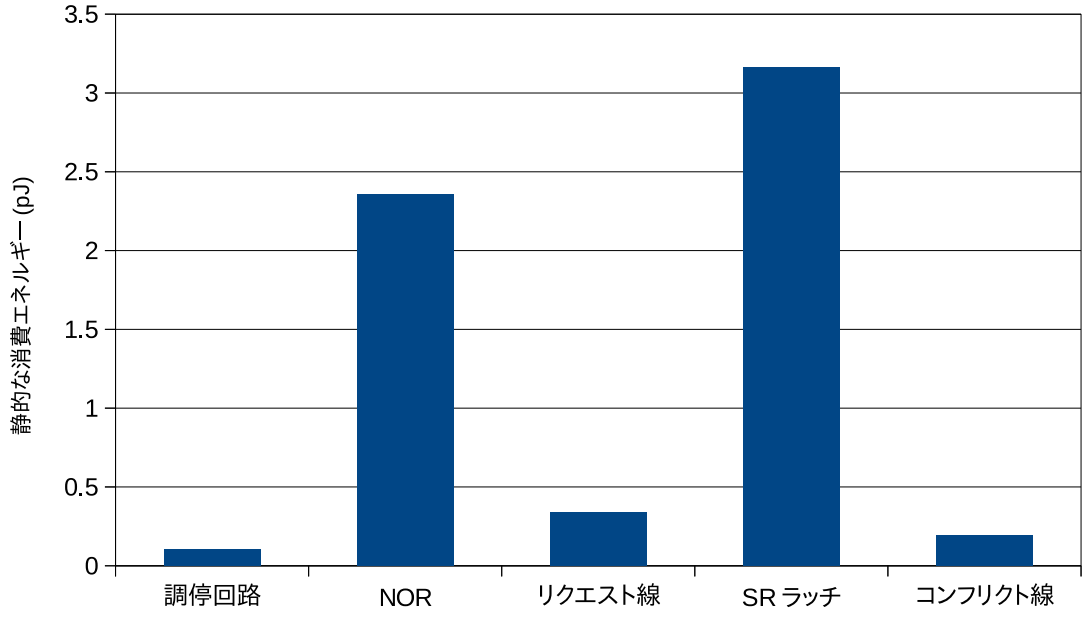


図 9.26: 1 サイクルあたりのエイジ論理の静的な消費エネルギー内訳

図 9.24，9.25 からわかるように明らかにエイジ論理の静的な消費エネルギーが大きいことがわかる．エイジ論理の静的消費エネルギーについてより詳細に述べる．図 9.26 にエイジ論理の静的な消費エネルギーの内訳を示す．図より，NOR と SR ラッチが大きいことがわかる．これらはエイジ論理の各セルに 1 個配置されているので，全体では 127×127 個と非常に多く存在する．この消費エネルギーを小さく方策としては，以下のようなものが考えられる．

- トランジスタ・サイズを小さくしてリーク電流を減らす．
- 前節で述べたように，IQ を 2 分割する．

図 9.27 には従来のタグ 1 段階比較のみのモデル (C(1)) で正規化された，各モデルの動的な消費エネルギーと静的な消費エネルギーの全ベンチマーク平均を示す．

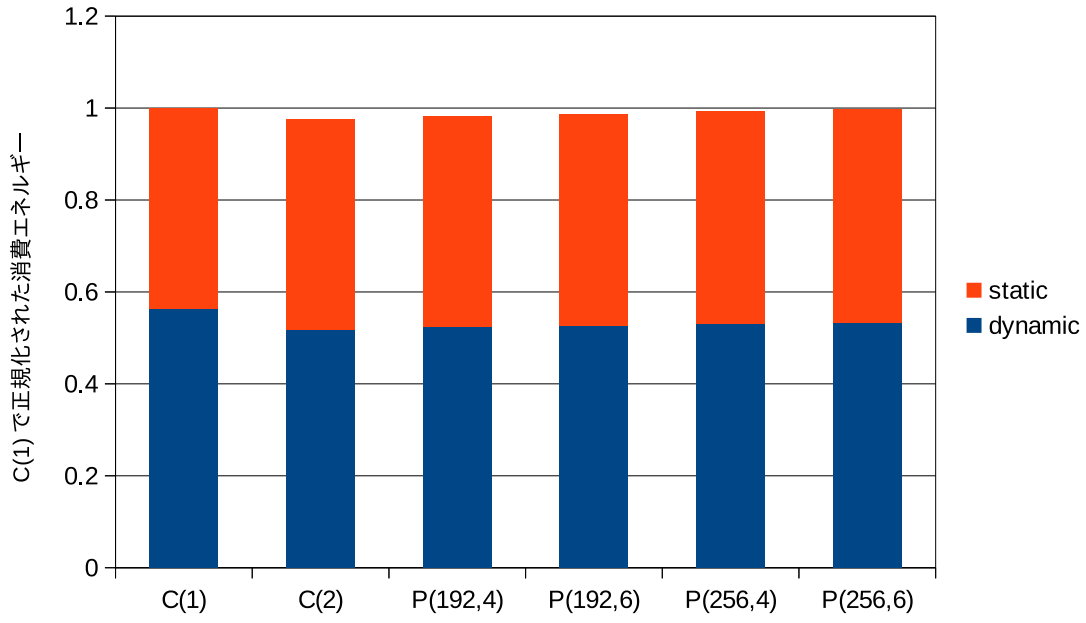


図 9.27: C(1) で正規化された発行キューの動的・静的な消費エネルギー

図 9.27 が示すように，従来手法 (C(1)) でも静的な消費エネルギーが 44%をも占める．この原因は，上述したようにエイジ論理の静的な消費エネルギーが大きいことである．また

PQ 方式を従来手法と比べると，例えば PQ のサイズが 192，読み出しポート数が 4 の場合，静的な消費エネルギー単独では 4.9%の増加し，動的 + 静的なエネルギーでは 1.8%削減された．

9.3.8 消費エネルギー未測定の回路部位

ディスパッチの時に命令を挿入するエントリのインデックスを計算するアドレスデコーダの消費エネルギーと，エイジ論理のディスパッチ時の消費エネルギーは今回は評価していない．どちらも無視できるほど小さくはないと考えられるので，今後測定する必要がある．

第 10 章　まとめ

IQ の中でウェイクアップ論理の消費エネルギーは大きく，特にタグ比較器の消費エネルギーが大きい．これに対してこれまで，タグ 2 段階比較という消費エネルギーを削減する手法と，それによる IPC 低下を抑制する手法が提案されていた．しかしこの IPC 低下抑制方式は，そのままでは現在の発行キューの方式であるランダム・キューに対応できない．そこで本研究ではランダム・キューに対応するために，PQ という命令のプログラム順を保持するキューを追加して，古い命令ののタグ比較を 1 段化する方式を提案した．そして HSPICE と SimpleScalar をベースにしたシミュレータによって，提案手法と従来手法の消費エネルギーと IPC を評価した．

評価の結果，提案手法によって，IPC をほとんど低下させずにウェイクアップ論理の消費エネルギーを削減できるとわかった．PQ のサイズが 192，PQ の読み出しポート数が 4 の時，ウェイクアップ論理の動的消費エネルギー削減率は 37.5%，IQ 全体では 6.8%であった．また IPC 低下は平均 0.72%で，すべてのベンチマーク・プログラムで 4%以下に抑えることができた．

付 録 A　計算に用いた消費エネルギーの値

消費エネルギーを求めるために 8.3 節の各計算式に代入した，1 サイクルあたりの回路の各部位の消費エネルギーの値をまとめる．表 A.1 にウェイクアップ論理の値，表 A.2 に選択論理の値，表 A.3 にエイジ論理の値，表 A.4 にタグ RAM，ペイロード RAM，ウェイクアップの RAM の読み書きの消費エネルギー，表 A.5 に PQ の読み書きの消費エネルギー，表 A.6 に静的消費エネルギーを示す．

表 A.1: ウェイクアップ論理の消費エネルギー (pJ)

低位ビット：高位ビット	$E_{tag\_br\_per\_tag}$	E_{not_match}	$E_{(1,neither\_match)\_per\_comp}$	$E_{(1,low\_only\_match)\_per\_comp}$
conventional	0.2283879083	0.0064032118		
1 : 8	0.2283879083		0.0073863371	0.0067681753
2 : 7	0.2283879083		0.0074072925	0.0065142314
3 : 6	0.2283879083		0.007410767	0.0062072467
4 : 5	0.2283879083		0.0074594428	0.0058999274
5 : 4	0.2283879083		0.0074053244	0.0055715961
6 : 3	0.2283879083		0.0074314346	0.0052324118
7 : 2	0.2283879083		0.007294736	0.0048870317
8 : 1	0.2283879083		0.0072616425	0.0044782397

$E_{(1,high\_only\_match)\_per\_comp}$	$E_{(2,low\_not\_match)\_per\_comp}$	$E_{(2,low\_only\_match)\_per\_comp}$	$E_{other\_not\_match}$
			0.0030887904
0.0041903453	0.0014819495	0.0073765527	0.0030887904
0.0045671816	0.0020872407	0.0070597168	0.0030887904
0.0049121813	0.0026773832	0.0067618075	0.0030887904
0.0052490623	0.0032658116	0.0064719696	0.0030887904
0.0055723758	0.0038701238	0.006101727	0.0030887904
0.0058763425	0.0044564289	0.0057580904	0.0030887904
0.0061595044	0.0050228415	0.005424432	0.0030887904
0.0064059589	0.0056109181	0.0050243335	0.0030887904

表 A.2: 選択論理の消費エネルギー (pJ)

$E_{rdy\_num<iw}$	$E_{rdy\_num\geq iw}$
3.91051	3.6327956369

表 A.3: エイジ論理の消費エネルギー (pJ)

$E_{req\_per\_req}$	$E_{conf\_discharge}$	$E_{conf\_nondischarge}$	$E_{nor\_per\_nor}$
0.0657669536	0.0323624432	0.0122530785	0.0007364243

表 A.4: RAM の読み書きの消費エネルギー (pJ)

	$E_{read\_per\_cycle(0)}$	$E_{read\_per\_cycle(read\_port)}$	$E_{write\_per\_cycle(write\_port)}$
tagRAM	0.4176686667	1.0066353333	0.5612150723
payloadRAM	0.6685664333	2.8531997667	2.782160625
wakeup			1.1339269647

表 A.5: PQ の読み書きの消費エネルギー (pJ)

PQ size	read port	$E_{read\_per\_cycle(0)}$	$E_{read\_per\_cycle(read\_port)}$	$E_{write\_per\_cycle(write\_port)}$
192	4	0.1091315333	0.4235648667	0.6011052311
192	6	0.1652498333	0.6380665	0.6033502098
256	4	0.1484155333	0.5410738667	0.7981112242
256	6	0.2240989333	0.8148406	0.8011445638

表 A.6: 静的消費エネルギー (pJ)

	$E_{static\_per\_cycle}$
C(1)	10.9070307227
P(192,4)	11.4194158561
P(192,6)	11.5011975561
P(256,4)	11.5732985227
P(256,6)	11.6824984561

謝辞

本研究をすすめるにあたり，多大なる御指導と御鞭撻を賜わりました名古屋大学大学院工学研究科情報・通信工学専攻　安藤秀樹教授，東京大学大学院情報理工学系研究科創造情報学専攻　塩谷亮太准教授に心より感謝いたします．また，本研究の遂行を支えてくださいました，名古屋大学大学院工学研究科情報・通信工学専攻安藤研究室の諸氏に深く感謝します.

参 考 文 献

[1] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, Temperature-aware microarchitecture: Modeling and implementation, *ACM Transactions on Architecture and Code Optimization*, Vol. 1, No. 1, pp. 94–125, March 2004.

[2] D. Folegnani and A. González, Energy-effective issue logic, In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pp. 230–239, June 2001.

[3] C. A. Zukowski and S.-Y. Wang, Use of selective precharge for low-power content-addressable memories, In *Proceedings of 1997 IEEE International Symposium on Circuits and Systems*, pp. 1788–1791, June 1997.

[4] 小林誠弥, 発行キューにおけるタグの 2 段階比較による消費エネルギー削減, 名古屋大学大学院工学研究科博士課程 (前期課程), 修士学位論文, 2015 年 3 月.

[5] S. Palacharla, N. P. Jouppi, and J. E. Smith, Complexity-effective superscalar processors, In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pp. 206–218, June 1997.

[6] J. Stark, M. D. Brown, and Y. N. Patt, On pipelining dynamic instruction scheduling logic, In *Proceedings of the 33rd Annual International Symposium on Microarchitecture*, pp. 57–66, December 2000.

[7] P. Michaud and A. Seznec, Data-flow prescheduling for large instruction windows in out-of-order processors, In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture*, pp. 27–36, January 2001.

[8] D. Ponomarev, G. Kucuk, and K. Ghose, Reducing power requirements of instruction scheduling through dynamic allocation of multiple datapath resources, In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pp. 90–101, December 2001.

[9] D. Ernst and T. Austin, Efficient dynamic scheduling through tag ellimination, In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pp. 37–46, May 2002.

[10] M. Goshima, K. Nishino, T. Kitamura, Y. Nakashima, S. Tomita, and S. Mori, A high-speed dynamic instruction scheduling scheme for superscalar processors, In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pp. 225–236, December 2001.

[11] M. D. Brown, J. Stark, and Y. N. Patt, Select-free instruction scheduling logic, In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, pp. 204–213, December 2001.

[12] P. G. Sassone, J. Rupley II, E. Brekelbaum, G. H. Loh, and B. Black, Matrix scheduler reloaded, In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pp. 335–346, June 2007.

[13] A. R. Lebeck, J. Koppanalil, T. Li, J. Patwardhan, and E. Rotenberg, A large, fast instruction window for tolerating cache misses, In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pp. 59–70, May 2002.

[14] S. E. Raasch, N. L. Binkert, and S. K. Reinhardt, A scalable instruction queue design using dependence chains, In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pp. 318–329, May 2002.

[15] E. Brekelbaum, J. Rupley, C. Wilkerson, and B. Black, Hierarchical scheduling windows, In *Proceedings of the 35th Annual International Symposium on Microarchitecture*, pp. 27–36, November 2002.

[16] I. Kim and M. H. Lipasti, Macro-op scheduling: Relaxing scheduling loop constraints, In *Proceedings of the 36th Annual International Symposium on Microarchitecture*, pp. 277–289, December 2003.

[17] D. Gibson and D. A. Wood, Forwardflow: A scalable core for power-constrained CMPs, In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, pp. 14–25, June 2010.

[18] H. Homayoun, A. Sasan, J. Gaudiot, and A. Veidenbaum, 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, Nov 2011.

[19] K. Pagiamtzis and A. Sheikholeslami, Content-addressable memory (CAM) circuits and architectures: A tutorial and survey, *IEEE Journal of Solid-State Circuits*, Vol. 41, No. 3, pp. 712–727, March 2006.

[20] C. Zhang, F. Vahid, J. Yang, and W. Najjar, A way-halting cache for low-energy high-performance systems, In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, pp. 126–131, August 2004.

[21] J. A. Farrell and T. C. Fischer, Issue logic for a 600-MHz out-of-order execution microprocessor, *Journal of Solid-State Circuits*, Vol. 33, No. 5, pp. 707–712, May 1998.

[22] R. P. Preston, R. W. Badeau, D. W. Bailey, S. L. Bell, L. L. Biro, W. J. Bowhill, D. E. Dever, S. Felix, R. Gammack, V. Germini, M. K. Gowan, P. Gronowski, D. B. Jackson, S. Mehta, S. V. Morton, J. D. Pickholtz, M. H. Reilly, and M. J. Smith, Design of an 8-wide superscalar RISC microprocessor with simultaneous multithreading, In *2002 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 334–472, February 2002.

[23] M. Golden, S. Arekapudi, and J. Vinh, 40-entry unified out-of-order scheduler and integer execution unit for the AMD Bulldozer x86-64 core, In *2011 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 80–82, February 2011.

[24] B. Sinharoy, J. A. Van Norstrand, R. J. Eickemeyer, H. Q. Le, J. Leenstra, D. Q. Nguyen, B. Konigsburg, K. Ward, M. D. Brown, J. E. Moreira, D. Levitan, S. Tung, D. Hrusecky, J. W. Bishop, M. Gschwind, M. Boersma, M. Kroener, M. Kaltenbacha, T. Karkhanis, and K. M. Fernsler, IBM POWER8 processor core microarchitecture, *IBM Journal of Research and Development*, Vol. 59, issue 1, pp. 2:1 – 2:21, January - February 2015.

[25] 酒井信二, ランダム並びの発行キューにおける命令再配置によるプロセッサの性能向上, 名古屋大学大学院工学研究科博士課程 (前期課程), 修士学位論文, 2017 年 3 月.

[26] M. Goshima, Research on high-speed instruction scheduling logic for out-of-order ilp processor, Ph.D. dissertation, Kyoto University, 2004.

[27] 末永泰士, 発行キューの遅延と消費エネルギー測定のための SPICE ネットリスト自動生成, 名古屋大学工学部電気電子・情報工学科，卒業論文, 2017 年 3 月.

[28] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective, fourth edition*, Addition Wesley, 2011.

[29] http://ptm.asu.edu/.

[30] International Technology Roadmap for Semiconductors, http://www.itrs2.net/.

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

[32] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures, In *Proceedings of the 42nd Annual International Symposium on Microarchitecture*, pp. 469–480, December 2009.