Vol.2009-ARC-185 No.2
2009/10/26

CMP におけるオンチップルータのランタイムパワーゲーティングの評価

松谷宏紀[†1,†2] 鯉渕道紘[†3]
中村宏[†1] 天野英晴[†4]

共有メモリ型 CMP において，NoC のスタンバイ電力を削減するため，オンチップルータへの電力供給を細粒度に制御する．具体的には，各ルータにおいて，パケット転送が完了した仮想チャネルへの電力供給を止め，また次のパケットが接近してきたら電力供給を再開する．しかし，この方法では，スリープ中の回路への電力供給を再開してから実際に動作可能になるまでに一定の遅延がかかり，CMP の性能に影響が生じる．このようなウェイクアップ遅延を隠蔽するため，次のパケット到着を事前に検出し，電力供給を開始するための手法が必須である．本論文では，このためのウェイクアップ手法を 3 種類提示し，共有メモリ型 CMP のフルシステムシミュレータ上で SPLASH-2 ベンチマークを用いて評価する．その結果，ルータのリーク電力を最大 57.6% 減らすことができ，また，ウェイクアップ遅延によるアプリケーションの実行時間の悪化を大幅に抑えることができた．

Evaluations of Run-Time Power-Gating of On-Chip Routers for CMP

Hiroki Matsutani,[†1,†2] Michihiro Koibuchi,[†3]
Hiroshi Nakamura[†1] and Hideharu Amano[†4]

This paper discusses the fine grain power management techniques to reduce standby power of on-chip networks used in shared memory CMPs. The power supply to each virtual channel in a router is stopped after forwarding packets, while it is resumed when the next packet is approaching to the router. However, the application performance is sacrificed, since a certain wakeup latency is required to activate a sleeping virtual channel after resuming the power supply to the virtual channel. To mitigate the wakeup latency, the wakeup methods that detect the next packet arrival and activate a corresponding virtual channel in advance are essential. In this paper, we introduce three wakeup methods and evaluate their performance by using SPLASH-2 benchmark programs in a full system simulator of shared memory CMPs. The simulation results show that one of the wakeup methods reduces the router leakage power by 57.6%. It also considerably mitigates the performance degradation due to the wakeup latency compared to a naive wakeup method.

1. はじめに

半導体技術の微細化にともない 1 チップ上に複数のマイクロプロセッサを実装できるようになった．コンシューマ用途においても 2 コアや 4 コアの製品が広く普及しており，コアの数は今後も増え続けると予想される．コンシューマ用途のマルチコアでは，プログラミングの容易さから，すべてのコアが同一のメモリ空間を共有する共有メモリ型のチップマルチプロセッサ（CMP）が現実的と言える．ただし，複数のプロセッサが単一のキャッシュを共有するため，キャッシュアクセスに十分な帯域を確保しないとプロセッサ数に見合った性能向上は期待できない．そこで，キャッシュを複数のキャッシュバンクに分割して帯域を稼ぐアーキテクチャ（Non-Uniform Cache Architecture，NUCA）[1),2)] が有望視されている．NUCA ではプロセッサおよびキャッシュバンクを Network-on-Chip（NoC）[3)] で接続し，データ転送はオンチップルータを介したパケット転送によって行う．

このような CMP では，複数のプロセッサを用いて並列処理することで高いスループット性能を実現する．シングルコアに比べ，個々のプロセッサの動作周波数を低く抑えることができ，今後さらに深刻化するであろう消費電力の問題を緩和できる．言い換えれば，これは，コア数（物量）を増やすことで消費電力を抑えるというアプローチである．しかし，半導体技術の微細化にともないリーク電力が増加している昨今，単純に物量を増やすアプローチでは増やした面積の分だけリーク電力も増えてしまい低消費電力化の効果が失われてしまう．

リーク電力削減のためにいくつかの方法が実用化されてきたが，本研究ではパワーゲーティングに着目する．これはアイドル中の回路ブロックへの電力供給を遮断することでリーク電流を抑える技術であり，とりわけプロセッサやキャッシュなどに適用されてきた．一方，我々はオンチップルータを対象としたパワーゲーティングに着目してきた[4)]．オンチップルータの場合，コア間のデータ転送がなければ電力供給を遮断（スリープ）できるチャンスが多い．その一方で，スリープ中のルータをウェイクアップさせるには一定の遅延がかかり，これによってアプリケーションの性能が低下する．そこで，本論文では，オンチップルータにおけるウェイクアップ遅延を隠蔽する手法について検討を行う．

†1 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
†2 日本学術振興会特別研究員 (SPD)
Research Fellow of the Japan Society for the Promotion of Science (SPD)
†3 国立情報学研究所 / 総合研究大学院大学
National Institute of Informatics / The Graduate University for Advanced Studies
†4 慶應義塾大学大学院 理工学研究科
Graduate School of Science and Technology, Keio University

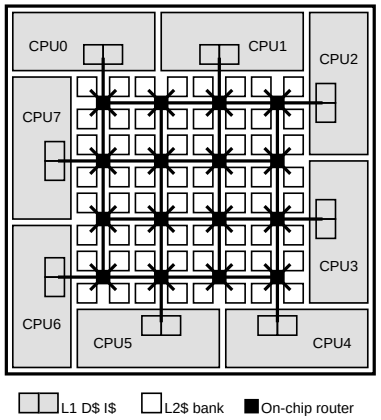


図 1 対象とする CMP の外観

本論文の構成は以下のとおりである．まず，2 章で，近年盛んに研究されている共有メモリ型の CMP アーキテクチャとその NoC について述べる．3 章では，オンチップルータにおける細粒度パワーゲーティングについて検討し，4 章でパワーゲーティングによるウェイクアップ遅延を隠蔽する手法を提示する．5 章では，これらのウェイクアップ制御手法を実際の CMP に適用し，アプリケーション性能とリーク電力の削減量について評価する．最後に 6 章で本論文をまとめる．

2. 共有メモリ型 CMP アーキテクチャ

共有メモリ型 CMP アーキテクチャとその NoC について述べる．5 章の評価では，ここで示した CMP 向け NoC に対し，オンチップルータのパワーゲーティングを適用する．

2.1 全体構成

図 1 に本論文が対象とする共有メモリ型 CMP のチップレイアウトを示す．これは文献 2) で紹介されている「2010 年の CMP」をもとに，L2 キャッシュバンクの数など一部パラメータを修正したものである．

図に示すようにチップ内にプロセッサコア（CPU）を 8 個持つ．各プロセッサコアは非共有の L1 データキャッシュ（L1 D$），L1 命令キャッシュ（L1 I$）を持つ．L2 キャッシュ（L2$）はすべてのプロセッサ間で共有する．キャッシュアクセスを高速化するため，キャッシュの構成は SNUCA（statically mapped, non-uniform cache architecture）[1] とする．具体的には，L2 キャッシュを多数のキャッシュバンクに分割し，ブロックインデックスの下位数ビットをもとに割り当てるキャッシュバンクを決める[⋆1]．メインメモリおよびディレクトリコントローラ（Dir）はチップ外にあると仮定する．

次節では，メモリシステムのコヒーレンス制御について説明する．

2.2 キャッシュコヒーレンスプロトコル

CMP におけるキャッシュのコヒーレンス制御のために token coherence protocol[5] を使用する．これは snooping protocol のようなブロードキャストベースのコヒーレンスプロトコルであるが，directory-based protocol のようにパケットの到着順を気にする必要はない．

token coherence protocol ではキャッシュブロックごとにプロセッサと同数の token が存在し，メッセージ転送によってノード間を移動する．token のうち 1 個は owner token と呼ばれ，有効なデータとともに移動する．キャッシュブロックを追い出すときは token をメインメモリ（Dir）に返還する．プロセッサは 1 個以上の token を持つときそのキャッシュブロックを read でき（MOESI の S または O[⋆2]），全部の token を持つとき write できる（MOESI の M）．

コヒーレンス制御のように要求と応答に依存関係がある場合，end-to-end の protocol deadlock を防ぐため，要求と応答を異なる仮想チャネルに割り振る必要がある．マルチプロセッサシミュレータ GEMS[6] の実装では，仮想チャネルを以下のように割り振っている．

- ローカル L1$から L2$バンクへの要求（**VC0**）
- L2$バンクからローカル L1$への要求（**VC0**）
- L2$バンクから Dir への要求（**VC1**）
- Dir から L2$バンクへの要求（**VC1**）
- ローカル L1$または Dir から L2$バンクへの応答（**VC2**）
- L2$バンクからローカル L1$または Dir への応答（**VC2**）
- ローカル L1$からの persistent 要求（**VC3**）

persistent 要求は，プロセッサ（ローカル L1$）はキャッシュミスが一定時間が経過しても解決されないときに発動する．これを受信したノードは，このプロセッサに対し，ただちに token を転送しなければならない．starvation を防ぐため persistent 要求にも独立した仮想チャネルを用いる．

以上のように，メッセージタイプで仮想チャネルを割り当てているため，仮想チャネルごとの負荷はまちまちである．例えば，キャッシュヒットによりメインメモリへのアクセスが少なくて済めば VC1 の負荷は軽い．また，そもそも persistent 要求が必要になるケースは稀（全要求のうち 0.19%[5]）であるため VC3 の負荷も軽い．

次節では，上記のメッセージを転送するオンチップネットワークについて説明する．

2.3 Network-on-Chip（NoC）

プロセッサ（ローカル L1$ を含む）と L2$バンクを接続するために NoC を用いる．図 1 の

⋆1 頻繁に使われるキャッシュブロックをプロセッサの近隣に動的に移動させることもできる（DNUCA，dynamically mapped, non-uniform cache architecture[1]）．しかし，CMP では，あるプロセッサの近くにあるバンクは別のプロセッサからは遠くなってしまうため，結果的に高い効果は期待できないと言われている[2]．

⋆2 手元に owner token が含まれていれば O，含まれていなければ S.

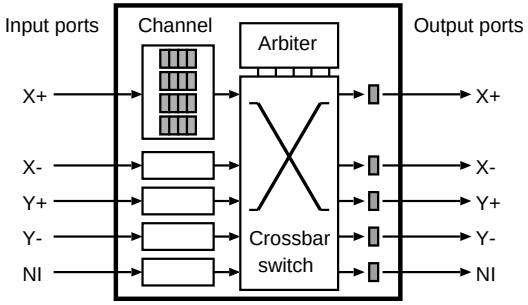


図 2 オンチップルータ（5-port，4-VC）

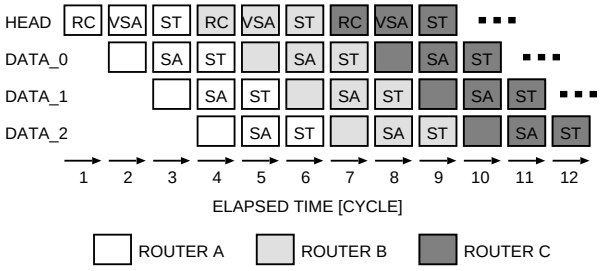


図 3 ルータのパイプライン（RC，VSA，ST）

例では，黒い四角がオンチップルータであり，オンチップルータが 4×4 のメッシュ状に相互接続されている．パケットルーティングとして，メッシュにおいて最もシンプルかつ一般的な次元順ルーティングを用いる．2.2 節で述べたとおり，コヒーレンス制御を実現するために 4 本の仮想チャネルを用いる[⋆1]．

オンチップルータとして典型的なワームホールルータを用いる．**図 2** は 5 個の物理チャネルを持つルータの例である．仮想チャネルごとに独立した入力バッファを持つ．各仮想チャネルに入力されたパケットは，出力チャネルを計算（routing computation，RC），出力仮想チャネルの割り当て（virtual channel allocation，VA）とクロスバのアービトレーション（switch allocation，SA）を行う．VA および SA のアービトレーションに勝つと，クロスバ上を 1 フリットずつ転送する（switch traversal，ST）．

図 3 に典型的な 3 サイクルルータのパイプライン処理の流れを示す．VA と SA は並列に実行可能であるため[7]，VA および SA ステージをまとめて VSA ステージとしている．ルータのパイプライン段数（通信遅延）はアプリケーションの実行時間に大きな影響を与えるため，ルータにパワーゲーティングを適用する際は，電源の On/Off に伴う通信遅延の増加を極力抑えなければならない．

次章では，パワーゲーティング技術について述べた後，オンチップルータにおけるパワーゲーティングの粒度とウェイクアップ遅延について議論する．

3. オンチップルータにおけるパワーゲーティング

パワーゲーティングでは，スリープ対象の回路と GND ラインの間，もしくは，スリープ対象の回路と Vdd ラインの間にスリープトランジスタを挿入し，このスリープトランジスタをオンオフすることで回路への電力供給を制御する．前者をフッタ型，後者をヘッダ型のパワーゲーティングと呼ぶ．本研究では前者をオンチップルータに適用する．

パワーゲーティングでは，何らかの方法で対象回路のアイドル状態を検出し，電力供給を遮断する（スリープ制御）．加えて，スリープ中の対象回路が使われることを検出し電力供給を再開（ウェイクアップ制御）できなければならない．スリープ制御およびウェイクアップ制御には一定の遅延がかかる．例えば，文献 8) では FPMAC にパワーゲーティングを適用しており，電流スパイクを抑えるためウェイクアップ処理に 6 サイクルを要している．

ウェイクアップ遅延はパワーゲーティングの対象回路の規模に大きく依存する．以降では，オンチップルータにパワーゲーティングを施すに際し，まず，パワーゲーティングの粒度とウェイクアップ遅延について議論する．

3.1 パワーゲーティングの粒度

想定しているパワーゲーティングの粒度として，粒度の粗い順に，1) 物理チャネル単位，2) 仮想チャネル単位，3) 入力バッファのフリット単位となる．

図 2 のオンチップルータでは，コヒーレンスプロトコルをサポートするため，物理チャネルごとに 4 本の仮想チャネルを持っている．つまり，物理チャネル単位のパワーゲーティングでは，全部の仮想チャネルが使われていないときのみ電力供給を止めることができる．仮想チャネルごとに異なるタイプのメッセージが流れ，負荷がまちまちであることを考えると，仮想チャネル単位のパワーゲーティングのほうがスリープできるチャンスが多いと言える．

さらに粒度の細かいパワーゲーティングとして，仮想チャネルバッファを部分的にパワーゲーティングする手法が考えられる．実際，図 2 に示すとおり仮想チャネルごとに数フリット分のバッファを持つため，バッファ全体のうち本当に使われる部分のみに電力を供給すれば，最小限のリーク電力でパケットを転送できる．ただし，この場合，仮想チャネル内のバッファ以外の部分（ルーティング計算や制御ロジック）は常に稼働できる状態にしておかなければならず，仮想チャネルを丸ごとスリープさせる手法に比べて，削減できるリークの量は少なくなる．

⋆1 仮想チャネルの代わりに，メッセージ毎に独立した物理ネットワークを用いることも可能であるが，負荷が非常に小さい仮想チャネル（VC1 や VC3 など）を敢えて独立した物理チャネルで実現するのは効率的ではない．

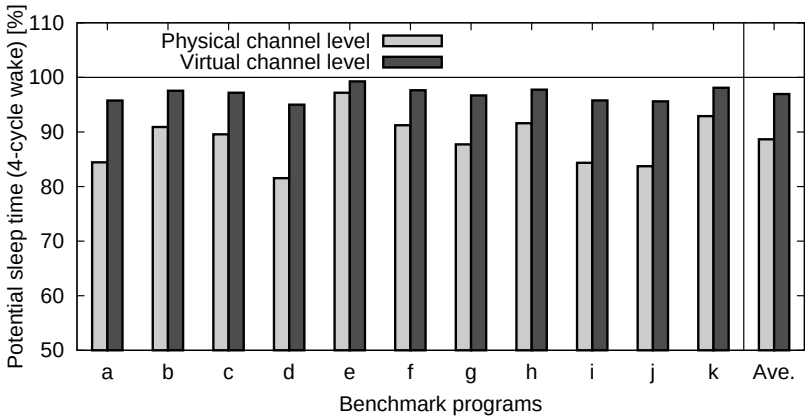


図 4 SPLASH-2 ベンチマークにおける，物理チャネル単位および仮想チャネル単位のパワーゲーティングのスリープ率（ウェイクアップ遅延は 4 サイクル）．プログラムは (a) radix，(b) lu，(c) fft，(d) barnes，(e) radiosity，(f) ocean，(j) raytrace，(h) volrend，(i) water-nsquared，(j) water-spatial，(k) fmm の 11 種類．

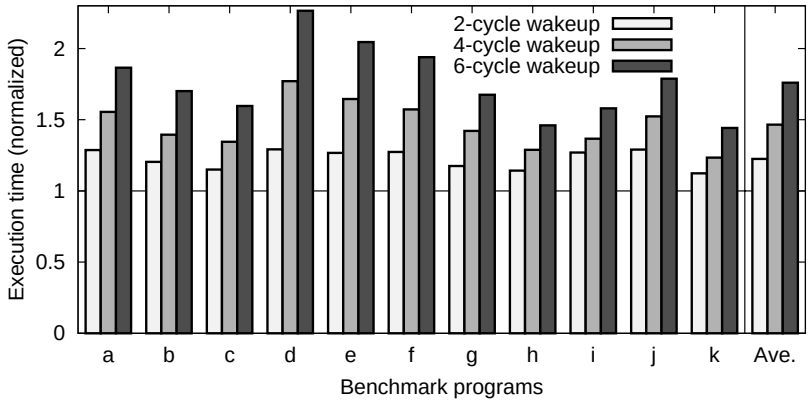


図 5 ウェイクアップ遅延と SPLASH-2 ベンチマークの実行時間（ウェイクアップ遅延が 0 サイクルのときの実行時間を 1 として正規化）．プログラムは (a) radix，(b) lu，(c) fft，(d) barnes，(e) radiosity，(f) ocean，(j) raytrace，(h) volrend，(i) water-nsquared，(j) water-spatial，(k) fmm の 11 種類．

ここでは予備評価として，物理チャネル単位および仮想チャネル単位のパワーゲーティングにおいて，どれだけの期間スリープできるかをシミュレーションにより求めた．評価のために CMP 環境上で SPLASH-2 ベンチマークプログラムを走らせた[⋆1]．評価結果を**図 4** に示す．ここでは電力供給をオンしてから対象回路が実際に動作可能になるまでの遅延（ウェイクアップ遅延）を 4 サイクルとした．このグラフより，物理チャネル単位で平均 88.7%，仮想チャネル単位で平均 96.4% の期間スリープできたことが分かる．

このように，仮想チャネル単位のほうがスリープできる期間が長いため，本論文では仮想チャネル単位のパワーゲーティングに着目する．

3.2 ウェイクアップ遅延

本章の冒頭で述べたとおり，パワーゲーティングではスリープ（電源オフ）中の回路ブロックに電源を供給し，実際に使用可能になるまでに一定の遅延が必要である．とりわけ，コヒーレンス制御ではキャッシュブロックの read や write 権限を得るためにプロセッサやメモリ間で制御メッセージのやりとりが頻発する．したがって，このようなシステムでは，オンチップルータのウェイクアップ遅延によってアプリケーションの実行性能が大幅に悪化する．

通信遅延とアプリケーション性能の関係を示す一例として，仮想チャネルバッファのウェイクアップ遅延を 2 サイクル，4 サイクル，6 サイクルとしたときの SPLASH-2 ベンチマークプログラムの実行時間を **図 5** に示す．ウェイクアップ遅延が 0 サイクルのときの実行時間を 1 として正規化してある．このグラフより，アプリケーションの実行時間はウェイクアップ遅延が 2，4，6 サイクルのとき平均で 22.5%，46.5%，76.0% 延びてしまっている．オンチップルータにパワーゲーティングを適用することでアプリケーションの実行時間が延びてしまっては，リーク電力は削減できてもトータルの消費電力は逆に増えてしまう．したがって，パワーゲーティングにともなうウェイクアップ遅延を隠蔽する仕組みが必要不可欠である．

⋆1 評価環境およびパラメータの詳細は 5 章で述べる．

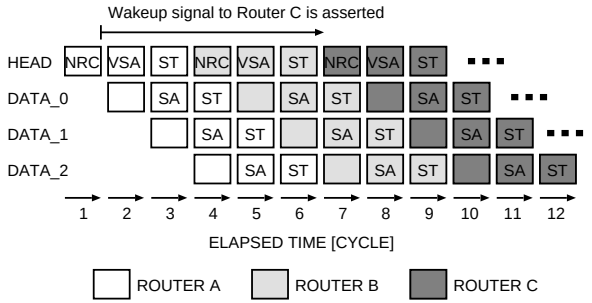
4. ウェイクアップ制御方式

本章では，ウェイクアップ遅延を隠蔽する手法を 3 つ示し，それぞれのメリットとデメリットについて議論する．

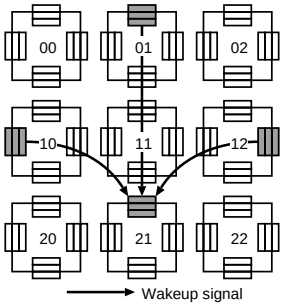
4.1 Look-Ahead 方式

look-ahead 方式では，各ホップにおいて 2 ホップ先で使われるルータの入力チャネルを検出し，その入力チャネル（もしくは仮想チャネル）に対して事前にウェイクアップ信号を送る[4]．ウェイクアップ対象の入力チャネルでは，ウェイクアップを受信してから実際にパケットが到着するまで数サイクルの余裕があるため，その間にウェイクアップ作業を完了できれば，ウェイクアップ遅延を完全に隠蔽できる．

2 ホップ先で使われるルータの入力チャネルを検出するために look-ahead ルーティングを応用する．look-ahead ルーティングでは，**図 6(a)** の NRC（next routing computation）が示すとおり，$i+1$ ホップ目の出力チャネルを i ホップ目のルータが計算する．$i+1$ ホップ目の出力チャネルは $i+2$ ホップ目の入力チャネルに直結しているため，i ホップ目のルータにおいて 2 ホップ先で使われる入力チャネルを正確に検出できる．

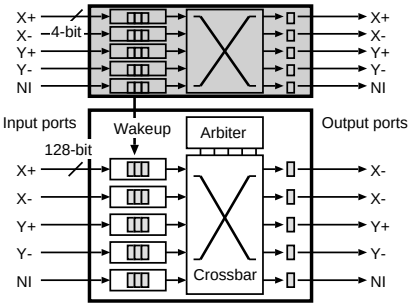


(a) Look-ahead 向けパイプライン（NRC，VSA，ST）

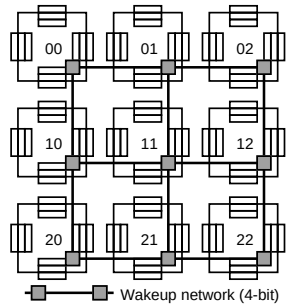


(b) ウェイクアップ信号

図 6 Look-ahead 方式．**図 6(b)** はルータ 21 の Y+ポートをウェイクアップさせるためのウェイクアップ信号．



(a) Wakeup packet 方式向けルータ



(b) ウェイクアップネットワーク

図 7 Wakeup packet 方式．スリープ不可のウェイクアップネットワーク（4-bit）を持つ．

2 ホップ先で使われる入力チャネルを検出できたら，その入力チャネルに対しウェイクアップ信号をアサートする．図 6(b) に示すように，各入力チャネルから到達可能な 2 ホップ先の入力チャネルに対しウェイクアップ信号を張る必要がある．

ただし，look-ahead 方式でウェイクアップ遅延を隠蔽できるのはパケット転送の 2 ホップ目からである．ネットワークインタフェイスにおけるパケット生成時に，1 ホップ目のルータに対してウェイクアップを要求したとしても，1 サイクル分の遅延しか隠蔽できない．

したがって，i ホップ目のルータは以下の式で与えられる $T^i_{recover}$ サイクル分のウェイクアップ遅延を隠蔽できる．

$$T^i_{recover} = \begin{cases} 2n - T_{wire} - 1 & i \geq 2 \\ 1 & i = 1 \end{cases} \quad (1)$$

ただし，n をルータのパイプライン段数，T_{wire} をウェイクアップ信号の配線遅延とする．例えば，$n = 3$ かつ $T_{wire} = 1$ のとき，2 ホップ目以降のルータは最大 4 サイクル分のウェイクアップ遅延を隠蔽できる．

4.2 Wakeup Packet 方式

wakeup packet 方式では，実際のパケット転送より 1 サイクル先行して wakeup packet と呼ばれる制御パケットを宛先に送信する．wakeup packet を中継したルータは，すぐに対応する入力チャネル（仮想チャネル）のウェイクアップを開始する．経路上の各ルータにおいて wakeup packet の到着が実際のパケット到着より十分に早ければ，ウェイクアップ遅延を隠蔽できる．

wakeup packet を転送するために，**図 7(b)** に示すウェイクアップネットワークが必要となる．通常のルータとウェイクアップネットワークのルータは **図 7(a)** に示すような形で接続される．なお，ウェイクアップネットワーク自体をスリープさせることはできないため，多少リーク電力が増えることになる．それでも，wakeup packet には宛先アドレスのみ含まれていれば良く，ウェイクアップネットワークのデータ幅は高々数ビットであるため，ウェイクアップネットワークによって消費されるリーク電力は小さい．

wakeup packet は宛先ルータで破棄されるため end-to-end の protocol deadlock を起こすことはなく，仮想チャネルは不要である．加えてデータ幅が非常に小さく，クロスバも非常に単純なため，ウェイクアップネットワークでは 1 サイクルルータが利用できる．

したがって，i ホップ目のルータは以下の式で与えられる $T^i_{recover}$ サイクル分のウェイクアップ遅延を隠蔽できる．

$$T^i_{recover} = (n-1)(i-1) + 1 \quad (2)$$

ただし，n をルータのパイプライン段数とする．例えば，$n = 3$ のとき，3 ホップ目（$i = 3$）のルータは最大 5 サイクル分のウェイクアップ遅延を隠蔽できる．

4.3 Active Buffer Window 方式

active buffer window 方式は，文献 9) においてルータバッファの低消費電力化のために提案された．active buffer window 方式では，次に書き込まれる数フリット分のバッファを事前にウェイクアップさせておく．事前にウェイクアップさせておくバッファ量を active window サイズと呼ぶ．**図 8** の例では，write pointer から 3 フリット分のバッファをウェイクアップさせている（active window サイズは 3 フリット）．active buffer window 方式では，1 フリット書き込まれる度に次の 1 フリット分のバッファをウェイクアップさせることで，常に active window サイズ分のバッファを利用可能にする．

したがって，パケット長が active window サイズ以下であれば，ウェイクアップ遅延を完全

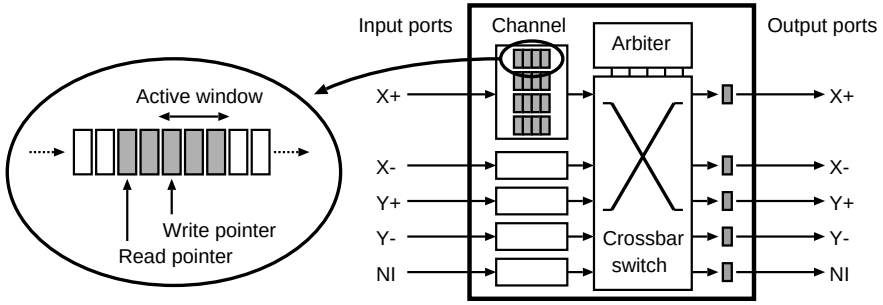


図 8 Active buffer window 方式．次に書き込まれる数フリット分のバッファを事前にウェイクアップさせる．

表 1 CMP の評価パラメータ

Processor	UltraSPARC-III
L1 I-cache size	16 KB (line:16B)
L1 D-cache size	16 KB (line:16B)
# of processors	8
L1 cache latency	1 cycle
L2 cache size	256 KB (assoc:4)
# of L2 cache banks	64
L2 cache latency	6 cycle
Memory size	4 GB
Memory latency	160 cycle

表 2 NoC の評価パラメータ

Topology	4×4 mesh
Routing	dimension-order
Switching	wormhole
# of VCs	4
Buffer size	4 flit
Router pipeline	[RC][VSA][ST]
Flit size	128 bit
Control packet	2 flit
Data packet	5 flit

に隠蔽できる．一方で，active window サイズ分のバッファには常に電力が供給されるため，無負荷時においてもリーク電力を完全に減らすことはできない．

次章では，実際の CMP 環境を想定したシミュレーションにより上記の議論を検証する．

5. 評　　価

まず，本研究が対象としている CMP 向け NoC のシミュレーション環境について述べる．次に，3 種類のウェイクアップ制御手法をオンチップルータのパワーゲーティングに適用し，アプリケーションの実行時間とリーク電力の削減量について評価する．

5.1 評 価 環 境

2 章で述べた CMP アーキテクチャをシミュレーションする．キャッシュアーキテクチャは SNUCA とし，キャッシュコヒーレンス制御には token coherence protocol を使用する．**表 1** にプロセッサとメモリシステムの詳細，**表 2** に NoC のパラメータを示す．

OS まで含めた CMP のフルシステムシミュレーションのために GEMS[6] および Simics[10] を組み合わせて使用する．CMP の NoC 部分には，GEMS に含まれているネットワークモデル Garnet[11] を使用した．今回は Garnet の detailed network model を改変することで，仮想チャネル単位のパワーゲーティング，および，3 種類のウェイクアップ制御手法をモデリングできるようにした．

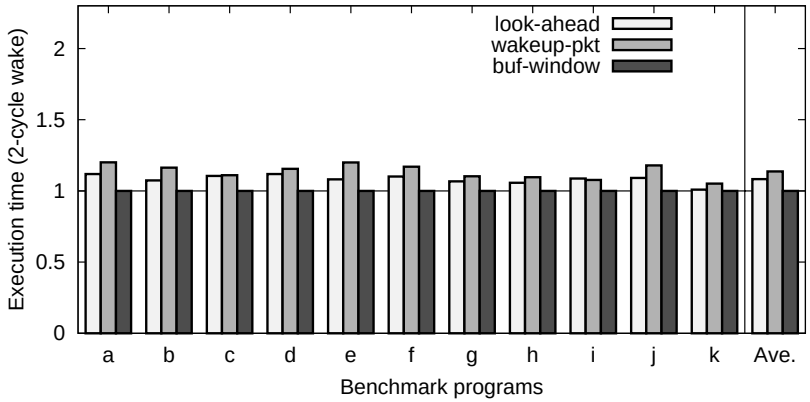


図 9 ウェイクアップ制御手法による SPLASH-2 ベンチマーク実行時間の改善（ウェイクアップ遅延 2 サイクル）．プログラムは (a) radix, (b) lu, (c) fft, (d) barnes, (e) radiosity, (f) ocean, (j) raytrace, (h) volrend, (i) water-nsquared, (j) water-spatial, (k) fmm の 11 種類．

ここでは，パイプライン段数 n は 3 段とし，look-ahead 方式におけるウェイクアップ信号の配線遅延 T_{wire} は 1 サイクルとした．また，active buffer window 方式の window サイズは 2 フリット分とした．

アプリケーションとして SPLASH-2 ベンチマーク[12] から 11 種類のプログラムを用いた．8 コアの CMP を想定したシミュレーション環境で Sun Solaris 9 を動作させ，そのうえで Sun Studio 12 の開発環境を用いて 11 種類のプログラムをコンパイルした．個々のプログラムは，スレッド数を 16 として Solaris 9 上で動作させた．各プログラムの実行時間として，処理のコア部分の実行サイクル数をカウントした．

5.2 ウェイクアップ遅延の影響

CMP 向け NoC において仮想チャネル単位のパワーゲーティングを行い，そのときの SPLASH-2 ベンチマークプログラムの実行時間を測定する．プログラムごとに，ウェイクアップ遅延を 0 サイクル，2 サイクル，4 サイクル，6 サイクルとしてシミュレーションを行った．そして，ウェイクアップ遅延が 0 サイクルのときの実行時間を基準に，2 サイクル，4 サイクル，6 サイクルのときの実行時間をグラフにした．

ここでは，ウェイクアップ制御手法として，look-ahead 方式，wakeup packet 方式，active buffer window 方式，ウェイクアップ遅延の隠蔽をいっさい行わない naive 方式の 4 種類を比

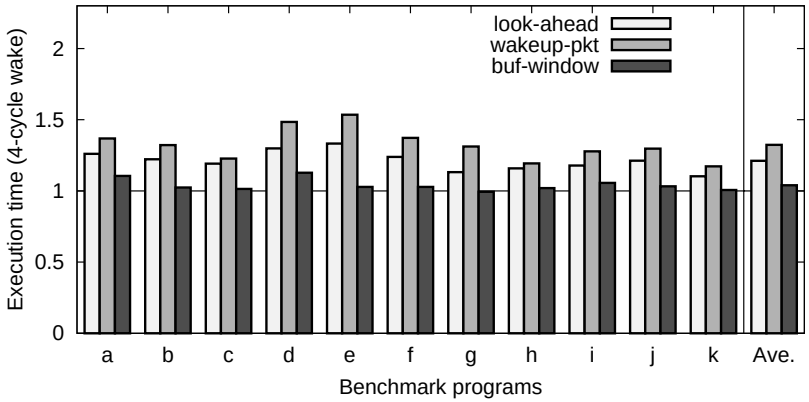


図 10 ウェイクアップ制御手法による SPLASH-2 ベンチマーク実行時間の改善（ウェイクアップ遅延 4 サイクル）．プログラムは (a) radix, (b) lu, (c) fft, (d) barnes, (e) radiosity, (f) ocean, (j) raytrace, (h) volrend, (i) water-nsquared, (j) water-spatial, (k) fmm の 11 種類.

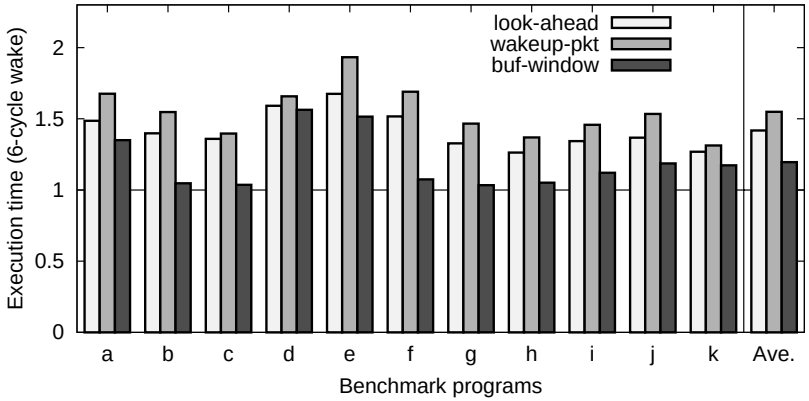


図 11 ウェイクアップ制御手法による SPLASH-2 ベンチマーク実行時間の改善（ウェイクアップ遅延 6 サイクル）．プログラムは (a) radix, (b) lu, (c) fft, (d) barnes, (e) radiosity, (f) ocean, (j) raytrace, (h) volrend, (i) water-nsquared, (j) water-spatial, (k) fmm の 11 種類.

較する．なお，naive 方式の結果はすでに 図 5 で示したとおりである．

図 9 にウェイクアップ遅延を 2 サイクルとしたときの 3 種類の方式の実行時間を示す．naive 方式では，2 サイクルのウェイクアップ遅延がかかると実行時間は平均 22.5% 延びた（図 5）．これが，look-ahead 方式では 8.3% 増，wakeup packet 方式では 13.7% 増，active buffer window 方式にいたっては完全に隠蔽できた（0% 増）．

表 3 ルータのリーク電力内訳 [uW].

Virtual channel x 20	552.0
Output latch x 5	23.6
Crossbar and arbiter	344.4
Total	920.0

表 4 仮想チャネルのリーク電力内訳 [uW].

FIFO buffer (4-flit)	22.8
Control	4.8
Total	27.6

図 10 はウェイクアップ遅延が 4 サイクルのときの結果である．naive 方式で平均 46.5% 実行時間が延びるところを，look-ahead では 21.2%増，wakeup packet では 32.4%増，active buffer window では 4.0% 増に抑えることができた．

さらに，**図 11** はウェイクアップ遅延が 6 サイクルのときの結果である．これまでの結果と同様，active buffer window 方式が最もウェイクアップ遅延の影響を抑えることができ，次いで look-ahead 方式，wakeup packet 方式の順となった．

5.3 リーク電力の削減量

look-ahead 方式，wakeup packet 方式，active buffer window 方式のそれぞれについて，ルータのリーク電力をどれだけ削減できたかを評価する．

まず，5.1 節の評価環境のところで紹介したオンチップルータ回路のリーク電力を見積もる．このために，オンチップルータ回路を Synopsys 社の Design Compiler を用いて合成した．ライブラリには電源電圧 1.10V の 45nm CMOS プロセスを用いた．このときのルータのリーク電力の内訳を **表 3** に示す．ここでは，各ルータは 5 本の物理チャネルを持ち，各物理チャネルが 4 本の仮想チャネルを持つと仮定しているため，仮想チャネルが合計 20 本ある．この仮想チャネル 1 本分のさらに詳細なリーク電力の内訳を**表 4** に示す．仮想チャネル内においては 4 フリット分の入力バッファがリーク電力の大部分を占めていることが分かる．このようなルータ回路に対し，look-ahead や wakeup packet 方式では仮想チャネル単位で電源制御を行う．一方，active buffer window 方式では window サイズを 2 フリットとして，バッファ 1 フリット単位で電源制御を行う．

次に，前節で行った SLPASH-2 ベンチマークのシミュレーション結果より，look-ahead, wakeup packet, active buffer window の各方式において，各仮想チャネルが平均してどれだけの期間スリープできたかを見積もる．ここではモデルを簡単にするため，ウェイクアップ過渡期[⋆1] は通常と同じだけのリーク電力を消費し，スリープ過渡期[⋆2] はリーク電力を消費しないと仮定する．また，ウェイクアップ遅延は 4 サイクルとする．

パワーゲーティングを一切行わないときのルータのリーク電力を 100% としたとき，各種方式によってルータのリーク電力をどれだけ削減できたかを計算した．**図 12** に評価結果を示す．このグラフより，パワーゲーティングを行わないときと比べ，look-ahead 方式では平均

⋆1 電源をオンしてから実際に回路が使えるようになるまでの期間.
⋆2 電源をオフしてから実際に回路の電位が下がりきるまでの期間.

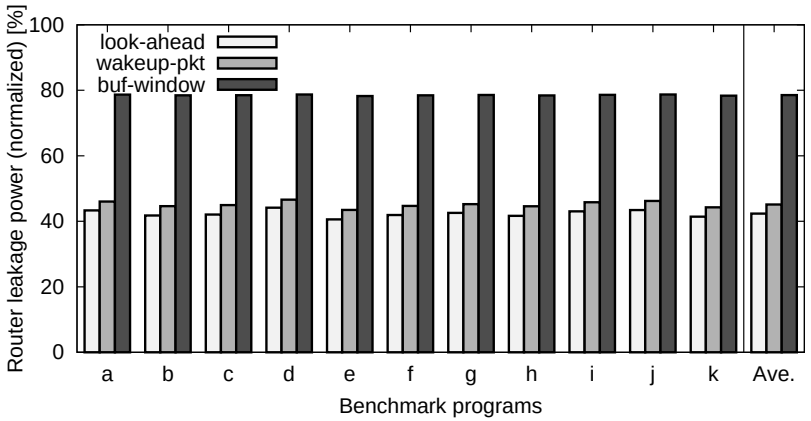


図 12 ウェイクアップ制御手法を用いたときのルータ 1 個当たりの平均リーク電力（パワーゲーティングを行わないときのリークを 100% として正規化）．プログラムは (a) radix，(b) lu，(c) fft，(d) barnes，(e) radiosity，(f) ocean，(j) raytrace，(h) volrend，(i) water-nsquared，(j) water-spatial，(k) fmm の 11 種類．

57.6%，wakeup packet 方式では 54.9%，active buffer window 方式では 21.5% リーク電力を削減できた．wakeup packet 方式は，look-ahead と比べ，4-bit 幅のウェイクアップネットワークが必要な分だけリーク電力が多くなった．active buffer window 方式はウェイクアップ遅延の影響を大幅に軽減できるものの，最低でも常に 2 フリット分のバッファ（仮想チャネルバッファの半分）がアクティブになるためリーク電力の削減量は小さくなった．

6. まとめと今後の課題

本論文では CMP 向けオンチップルータにおけるパワーゲーティングの粒度とウェイクアップ遅延を隠蔽する方式について議論した．仮想チャネル単位のパワーゲーティング向けのウェイクアップ手法として look-ahead 方式と wakeup packet 方式を評価した．また，仮想チャネルバッファのフリット単位のパワーゲーティング向けに active buffer window 方式を評価した．

評価の結果，look-ahead 方式によるパワーゲーティングによって平均 57.6% のリーク電力を減らすことができ，ウェイクアップ遅延によるアプリケーションの実行時間の悪化を最大 63.1% 抑えることができた．一方，active buffer window 方式は実行時間の悪化をほとんど抑えることができたが，リーク電力の削減量はかなり小さかった．

今後は，look-ahead，wakeup packet，active buffer window を組み合わせたウェイクアップ手法を検討する．例えば，大半のルータにはリーク削減量の多い look-ahead 方式を用いるが，トラフィック量の多い一部のルータにはウェイクアップ遅延をほぼ完全に隠蔽できる active buffer window 方式を採用することが考えられる．また，オンチップルータ回路にパワースイッチを挿入して，実際のウェイクアップ遅延を見積もることも考えている．

謝 辞 本研究は科学技術振興機構「JST」の戦略的創造研究推進事業「CREST」における研究領域「情報システムの超低消費電力化を目指した技術革新と統合化技術」の研究課題「革新的電源制御による次世代超低電力高性能システム LSI の研究」による．また，本研究は特別研究員奨励費の助成を受けて行われた．

参 考 文 献

1) Kim, C., Burger, D. and Keckler, S.W.: An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches, *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'02)*, pp. 211–222 (2002).
2) Beckmann, B.M. and Wood, D.A.: Managing Wire Delay in Large Chip-Multiprocessor Caches, *Proceedings of the International Symposium on Microarchitecture (MICRO'04)*, pp. 319–330 (2004).
3) Dally, W.J. and Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks, *Proceedings of the Design Automation Conference (DAC'01)*, pp.684–689 (2001).
4) Matsutani, H., Koibuchi, M., Wang, D. and Amano, H.: Run-Time Power Gating of On-Chip Routers Using Look-Ahead Routing, *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'08)* , pp.55–60 (2008).
5) Martin, M. M.K., Hill, M.D. and Wood, D.A.: Token Coherence: Decoupling Performance and Correctness, *Proceedings of the International Symposium on Computer Architecture (ISCA'03)*, pp.182–193 (2003).
6) Martin, M. M.K., Sorin, D.J., Beckmann, B.M., Marty, M.R., Xu, M., Alameldeen, A.R., Moore, K.E., Hill, M.D. and Wood, D.A.: Multifacet General Execution-driven Multiprocessor Simulator (GEMS) Toolset, *ACM SIGARCH Computer Architecture News (CAN'05)*, Vol.33, No.4, pp.92–99 (2005).
7) Dally, W.J. and Towles, B.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann (2004).
8) Vangal, S., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Iyer, P., Singh, A., Jacob, T., Jain, S., Venkataraman, S., Hoskote, Y. and Borkar, N.: An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS, *Proceedings of the International Solid-State Circuits Conference (ISSCC'07)* , pp.184–185 (2007).
9) Chen, X. and Peh, L.-S.: Leakage Power Modeling and Optimization in Interconnection Networks, *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'03)* , pp.90–95 (2003).
10) Magnusson, P.S. et al.: Simics: A Full System Simulation Platform, *IEEE Computer*, Vol.35, No.2, pp.50–58 (2002).
11) Agarwal, N., Peh, L.-S. and Jha, N.: Garnet: A Detailed Interconnection Network Model inside a Full-system Simulation Framework, Technical Report CE-P08-001, Princeton University (2008).
12) Woo, S.C., Ohara, M., Torrie, E., Singh, J.P. and Gupta, A.: SPLASH-2 Programs: Characterization and Methodological Considerations, *Proceedings of the International Symposium on Computer Architecture (ISCA'95)*, pp.24–36 (1995).