セル・アロケーション・キャッシュのハードウェア設計及び近似除算器によるセル制御回路の面積削減手法の提案と評価

情報科学科　碇　知朗　　指導教員：佐々木　敬泰

1　はじめに

高性能，低消費電力を実現する手段の一つであるマルチコアプロセッサでは，コア間でのデータ競合に起因するキャッシュミスが増加し，性能が低下する問題がある．その問題を解決する手法の一つとして，キャッシュ・パーティショニング [1] を応用したセル・アロケーション・キャッシュ [2] (以下「CAC」と呼ぶ) が提案されている．しかしながら，CAC はこれまでモデルベースでの評価しかされておらず，詳細なハードウェア設計及び面積評価は行われていない．そこで本研究では CAC のハードウェア設計，及び近似除算器を用いた回路面積削減手法の提案，及び評価を行った．その結果，CAC の面積増加率は通常のキャッシュと比較し 0.01% 未満であり，提案する 2 つの近似除算器は僅かな回路規模で実現できることが明らかとなった．

2　セル・アロケーション・キャッシュ

2.1　セル・アロケーション・キャッシュの概要

キャッシュ・パーティショニングは図 1 に示す通り，キャッシュ領域をウェイ単位に分割しコアに割り当てを行う．コアに割り当てられるウェイ数は割り当てた際のキャッシュミス減少数で判断され，最適な割り当てを実現しデータの競合を防ぐ．しかしウェイは粗粒度であるため，参照の局所性に起因した未使用領域やコアが必要とするキャッシュ領域に過不足が生じやすく，キャッシュを効率的に活用できない．そこで CAC では図 2 に示す通り，キャッシュ領域をウェイよりも細かいセルと呼ばれる単位に分割しコアに割り当てる．コアに細粒度であるセルを割り当てることで，使用頻度の低い領域を別のコアへと割り当てることが可能となり，キャッシュの利用効率を向上させることができる．

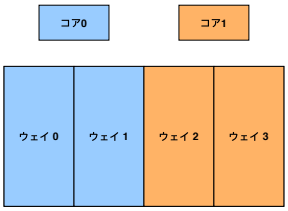

図 1　キャッシュ・パーティショニングの概念図

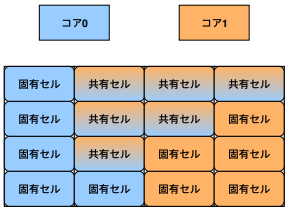

図 2　セル・アロケーション・キャッシュの概念図

2.2　セルの管理手法

CAC はコア間でのデータ共有を可能にするために，キャッシュミス時のリプレース対象を割り当てられたコアのみに限定した固有セルと，通常のキャッシュ同様にアクセスが可能な共有セルの 2 種類のセルを実装する．セルはデータの共有率を基に一定期間ごとに切り替えられる．共有率を算出するために，セルごとに 3 種類のカウンター，すなわち，キャッシュへのアクセス数をカウントする Access Counter，全てのキャッシュヒット数をカウントする Hit Counter，割り当てられていないコアからのキャッシュヒット数をカウントする Hit Other Counter を実装する．データの共有率 Shared Ratio は式 (1) から求められる．

$$\text{Shared Ratio} = \frac{\text{Hit Count of other core}}{\text{Total Hit Count}} \tag{1}$$

3　CAC 実装手法の提案と面積評価

3.1　セルへの追加要素

文献 [2] では実装に追加カウンタが必要であることしか述べられておらず，詳細なハードウェア構成は明らかにされていない．そこで，本研究ではまず CAC を実現するのハードウェア構成を提案する．図 3 は提案するセル制御回路のデータパスである．3 種類のカウンタは事前に定義する期間が経過するまで値を保持する．今回，セルへのアクセス数が 300,000 回と期間を定義したため，カウンタは 19bit となる．また各カウンタをインクリメント，リセットするために論理素子や比較器，加算器が必要となる．

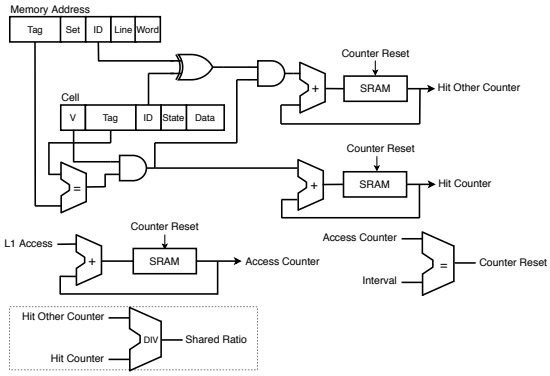

図 3　セルのデータパス

3.2　評価結果

前述の 3 種類のカウンタを実装した場合の CAC と通常のキャッシュを比較した際の面積増加率を評価する．CACTI6.0 を使用し，8MB，8 ウェイ，128 セルの CAC の面積増加率を算出した結果，0.01% 未満となった．この結果より，カウンタを実装した場合の影響は僅かだとわかった．

4　近似除算器によるセル制御回路の面積削減手法の提案と評価

CAC では共有率の計算に除算を用いるが，一般に除算器は回路規模が大きいという問題がある．そこで CAC の実装面積削減のために共有率の算出に使用する除算器に着目する．

4.1　提案手法

除算は除数が 2 の累乗である時，シフト演算で代用できる性質を生かしセルの共有率をシフト演算で算出する．本研究では除数と算出する期間を近似の対象として 2 つの手法 [3] を提案する．除数近似法は共有率を算出するタイミングで除数を 2 の累乗に近似する．正確なタイミングで計算が可能だが，除数を近似するため，共有率の誤差が発生する．期間近似法は除数が 2 の累乗となったタイミングで共有率を算出する．この手法は共有率を算出するインターバルを近似の対象としている．近似による誤差は実行するプログラムのメモリアクセス数に影響される．

4.2　評価結果

本研究では，提案する 2 つの近似除算器を SystemVerilog を用いて詳細設計し，論理合成し面積評価を行った．その結果，NAND 換算で除数近似法は 933 ゲート，期間近似法は 431 ゲートと僅かな回路規模で実現できることがわかった．また，アーキテクチャシミュレータ Gem5 を使用し共有率の誤差による性能への影響を検証した結果，2 手法とも実行サイクル数に変化は見られなかった．よって性能は変わらないため，回路規模の少ない期間近似法を用いることが良いとわかった．

5　おわりに

本研究では CAC のハードウェア構成を提案し，近似除算器を用いた回路規模の削減手法を提案した．結果としてカウンタの影響は僅かであり，提案する近似除算器を用いることで回路規模を低減することができた．今後は OS や複雑なベンチマークなど実環境に近いシミュレーション環境の構築を課題とする．

参考文献

[1] G. E. Sue, et al., “Dynamic Partitioning of Shared Cache Memory,” Journal of Supercomputing, vol.28, no.1, pp.7–26, 2004.

[2] M. Kito, et al., “Performance Evaluation of Dynamic Cell Allocation Cache using Cycle Accurate Simulator,” Proceedings of the International Symposium on Computing and Networking, pp.1–3, 2018.

[3] T. Ikari, et al., “Hardware Design and Evaluation of Cell Allocation Cache,” Proceedings of the International Workshop on Networking, Computing, Systems, and Software, pp.1–4, 2019.