Vol.2009-ARC-183 No.8
Vol.2009-OS-111 No.8
2009/4/22

3次元DRAM-プロセッサ積層実装を対象としたオンチップ・メモリ・アーキテクチャの提案と評価

橋口慎哉[†1]　小野貴継[†1]
井上弘士[†2]　村上和彰[†2]

本稿では，3 次元積層 DRAM の利用を前提とし，大幅なチップ面積の増加を伴うことなく高いメモリ性能を達成可能な新しいキャッシュ・アーキテクチャを提案する．3 次元実装された DRAM を大容量キャッシュとして活用することで，オフチップメモリ参照回数の劇的な削減が期待できる．しかしながら，その反面，キャッシュの大容量化はアクセス時間の増加を招くため，場合によっては性能が低下する．この問題を解決するため，提案方式では，実行対象プログラムのワーキングセット・サイズに応じて 3 次元積層 DRAM キャッシュを選択的に活用する．ベンチマーク・プログラムを用いた定量的評価を行った結果，提案方式の静的制御方式で平均 35%，動的制御方式で平均 43%の性能向上を達成した．

On-Chip Memory Architecture for DRAM Stacking Microprocessors

SHINYA HASHIGUCHI,[†1] TAKATSUGU ONO,[†1]
KOJI INOUE[†2] and KAZUAKI MURAKAMI[†2]

In this paper, we propose a new architecture that can achieve high memory performance without large footprint overhead for DRAM-stacked processors . 3D stacked DRAM caches can dramatically reduce off chip memory accesses. However, this approach degrades performance in some cases because increasing cache size makes access time longer to solve this problems. Our approach selectively leverages the stacked DRAM cache based on the valiation of working set sizes. The results of our quantitative evaluation showed that the proposed approach achieves 35% of memory performance gain in static control method and 43% in dynamic control method.

1. はじめに

新しい半導体チップの実現法として 3 次元実装が注目されている．これまでの 2 次元実装 LSI においては，回路の大規模化に伴いブロック間接続のための配線が長くなり，ひいては，動作周波数の低下や消費電力の増大を招くといった問題があった．これに対し，3 次元実装 LSI では，3 次元方向へ回路を集積することで配線長を維持しつつ，回路を大規模化できるといった利点がある．また，たとえば DRAM とロジックのように異なる製造プロセスを得て作成した複数のダイを積層する事も比較的容易になる．

このような背景の中，3 次元実装デバイスを前提としたプロセッサ構成法に関する研究が行われるようになってきた．その中でも特に，個別の製造プロセスを得て完成した大容量 DRAM ダイとプロセッサ・ダイを積層し，これらの間を TSV （Through Silicon Via）で接続するアプローチが大きな注目を集めている[2)4)5)]．プロセッサ・コアと大容量メモリを 1 個の LSI チップに混載することで大容量オンチップ・メモリを実現すると共に，TSV による高オンチップ・メモリバンド幅も利用可能となる．これにより，深刻化の一途を辿るメモリウォール問題の抜本的解決策として期待できる．

文献 2) では，積層した DRAM を L2 キャッシュとして活用する．このような DRAM キャッシュ・スタック法（以降，DRAM スタック法と略す）では，高速なタグ検索を実現するために，SRAM 実装されたタグメモリをプロセッサ・コアと同一レイヤに有する．L1 キャッシュ・ミスが発生した際，当該タグメモリを参照すると同時に，積層された大容量 DRAM のアクセスを開始する．本方式では，32〜64MB 程度の DRAM をラストレベル・キャッシュとして使用するため，オフチップ・アクセス回数を劇的に削減できる．しかしながら，その反面，キャッシュの大容量化はアクセス時間の増加を招くため，場合によっては性能が低下するといった問題が生じる．

そこで本稿では，従来の DRAM スタック法が有する問題点を解決し，さらなる高性能化を実現する新しいメモリ構成法として**ハイブリッド・キャッシュ・アーキテクチャ**を提案する．また，ある理想条件下において定量的評価を行い，提案方式による潜在的な性能向上の

†1 九州大学大学院システム情報科学府 〒 819-0395 福岡市西区元岡 744 番地 Graduate School of Information Science and Electrical Engineering, Kyushu University 744 Motooka Nishi-ku Fukuoka 819-0395 JAPAN
Email: s-hashiguchi,ono@c.csce.kyush-u.ac.jp
†2 九州大学大学院システム情報科学研究院 〒 819-0395 福岡市西区元岡 744 番地 Faculty of Information Science and Electrical Engineering, Kyushu University 744 Motooka Nishi-ku Fukuoka 819-0395 JAPAN
Email: inoue,murakami@i.csce.kyush-u.ac.jp

可能性を明らかにする．ハイブリット・キャッシュは，通常のキャッシュメモリ，ならびに，大容量 DRAM 用タグメモリとしての動作が可能であり，実行対象プログラムのワーキングセット・サイズに応じて適応的に動作モードを変更する．具体的には，ワーキングセット・サイズが小さい場合には通常の L2 キャッシュメモリとして動作し，DRAM キャッシュへのアクセスは禁止する．一方，より大きな L2 キャッシュが必要だと判断した場合には DRAM キャッシュ用タグメモリとして動作する．このように，選択的に大容量 DRAM を活用することで，高速アクセスとキャッシュミス率の改善の両立を可能にする．

以下，第 2 節では文献 2) で提案された DRAM スタック法を紹介し，その問題点を整理する．次に，第 3 節でハイブリッド・キャッシュを提案し，そのマイクロアーキテクチャならびに制御方式の詳細を説明する．第 4 節ではベンチマーク・プログラムを定量的評価を行い，提案方式の有効性を明らかにする．最後に第 5 節で簡単にまとめる．

2. DRAM スタック法

2.1 メモリ・アーキテクチャ

従来の 2 次元実装プロセッサ（以降，ベースプロセッサと呼ぶ），ならびに，文献 2) で提案された DRAM スタック法の構成を**図 1** に示す．以降，本稿では，プロセッサ・コアが実装されたダイを**下層ダイ**，DRAM が実装されたダイを**上層ダイ**と呼ぶ．ベースプロセッサは下層ダイのみで構成され，1 個のプロセッサ・コアと L2 キャッシュが搭載されていると仮定する．一方，DRAM スタック法では，3 次元実装技術により上層ダイとして DRAM を積載し（以降，上層 DRAM と呼ぶ），これを L2 キャッシュのデータ領域として使用する．L2 キャッシュのタグ情報に関しては，高速なキャッシュ検索を可能とするために SRAM を用いて下層ダイに実装される[⋆1]．このタグメモリのサイズは，プロセッサのアドレスビット長とキャッシュ構成に異存する．たとえば，上層 DRAM の容量が 64MB，アドレス長 64 ビット，ブロックサイズ 64B，連想度 16 の場合，そのタグ領域には約 5MB の容量が必要となる．これは，デュアルコアプロセッサ・チップに搭載される L2 キャッシュ容量とほぼ同程度の実装面積を要する．したがって，DRAM スタック法では，ベースプロセッサにおいて L2 キャッシュとして使用していた SRAM 領域を利用してタグメモリを搭載する．

2.2 性能向上の条件

第 2.1 節で説明した DRAM スタック法における最大の利点は，ラストレベル・キャッシュ

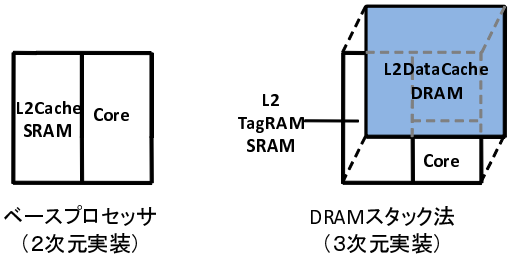

図 1 ベースプロセッサと DRAM スタック法の構成

の大容量化に伴うオフチップ・アクセス回数の大幅な削減である．その一方，一般にキャッシュ・ヒット時間はキャッシュ容量に比例して長くなる傾向にある．そのため，ベースプロセッサにおける下層 L2 キャッシュへのアクセス時間と比較して，DRAM スタック法での上層 DRAM アクセス時間は長くなる．その結果，L2 キャッシュのヒット時間が増大し，ひいては，性能低下をもたらす恐れがある．そこで本節では，DRAM スタック法により性能向上を達成するための条件を整理する．

ベースプロセッサと DRAM スタック法に関するメモリ性能の優劣を解析するため，本節では以下の式で表される *AMAT*（Average Memory Access Time：平均メモリアクセス時間）を用いる．

$$AMAT[\text{cycles}] = HT_{L1} + MR_{L1} \times (HT_{L2} + MR_{L2} \times MMAT) \quad (1)$$

- HT_{L1}/HT_{L2}：L1/L2 キャッシュ・ヒット時間 [cycles]
- MR_{L1}/MR_{L2}：L1/L2 キャッシュ・ミスの割合
- $MMAT$：主記憶のアクセス時間 [cycles]

ベースプロセッサと DRAM スタック法の違いは，メモリ階層の中での L2 キャッシュ部分にある．したがって，両者において，HT_{L1}，MR_{L1}，ならびに，$MMAT$ に差は生じない．ここで，HT_{L2} と MR_{L2} に関して，それぞれ，ベースプロセッサの場合を HT_{L2_BASE} と MR_{L2_BASE}，また，DRAM スタック法の場合を HT_{L2_DRAM} と MR_{L2_DRAM} で表記する．この場合，厳密には実行対象プログラムの特性に異存するが，多くの場合において以下の関係式が成り立つ．

$$HT_{L2\_BASE} \leq HT_{L2\_DRAM}, MR_{L2\_BASE} \geq HT_{L2\_DRAM} \quad (2)$$

したがって，ベースプロセッサに対して DRAM スタック法により性能向上を実現するための条件は式 (3) となる．

⋆1 一般に，DRAM と SRAM は製造プロセスが大きく異なる．そのため，同一層に DRAM データ領域と SRAM タグ領域を実装することは可能であるが，その場合には製造コストが増加する．

$$HT_{L2\_BASE} + MR_{L2\_BASE} \times MMAT > HT_{L2\_DRAM} + MR_{L2\_DRAM} \times MMAT \quad (3)$$

この式を変形すると，式 (4) が導き出される．

$$MR_{L2\_BASE} - MR_{L2\_DRAM} > \frac{HT_{L2\_DRAM} - HT_{L2\_BASE}}{MMAT} \quad (4)$$

左辺は DRAM スタック法の採用による L2 キャッシュミス率削減効果 $MR_{L2\_REDUCTION}$，右辺の $HT_{L2\_DRAM} - HT_{L2\_BASE}$ は L2 キャッシュアクセス時間オーバーヘッド $HT_{L2\_OVERHEAD}$ であり，単位は共にクロックサイクルである．この式より，DRAM スタック法により性能向上を実現するためには，L2 キャッシュアクセス時間オーバヘッドに対し，それを隠蔽できるだけの L2 キャッシュミス率削減効果が必要であることが分かる．

2.3 問題点と解決すべき課題

本節では，DRAM スタック法の問題点と解決すべき課題を整理する．特に断りが無い限り，ベースプロセッサは 2MB の L2 キャッシュ（ヒット時間は 12 クロックサイクル），DRAM スタック法は 32MB の L2 キャッシュ（ヒット時間は 60 クロックサイクル）の搭載を想定する．また，各キャッシュの連想度は 8，ラインサイズは 64B，主記憶アクセス時間 $MMAT$ は 300 クロックサイクルと仮定する．その他の実験環境の詳細については，第 4.1 節で示す内容と同一である．

DRAM スタック法では，L2 キャッシュは DRAM で構成されているため，L1 キャッシュミスが発生した場合，常に上層 DRAM へアクセスする必要がある．一般に，チップ製造後のキャッシュヒット時間は定数値となる．したがって，図 1 に示すように，ベースプロセッサならびに DRAM スタック法にてその構成が決定された場合には，$HT_{L2\_OVERHEAD}$ も同様に定数値となる．一方，$MR_{L2\_REDUCTION}$ に着目した場合，以下のような問題点が生じる．

- **L2 キャッシュミス率改善効果はプログラム間で異なるため，場合によっては DRAM スタック法の導入により性能が低下する．**複数ベンチマークプログラムの実行において，L2 キャッシュ容量を 2MB から 128MB まで変化させた場合のミス率を**図 2** ならびに**図 3** に示す．横軸は L2 キャッシュ容量，縦軸は L2 キャッシュミス率である．図 2 の *171.swim, 172.mgrid, 173.applu, 183.equake* や図 3 の *LU, FMM* などでは，L2 キャッシュサイズを 32MB に拡大することで大幅なミス率削減を達成している．これに対し，図 3 の *FFT* においては，32MB の L2 キャッシュ容量に対してワーキングセット・サイズが十分に大きいため，L2 キャッシュミス率の改善は極めて低い．また，図 2 の *164.gzip* や図 3 の *Raytrace* などでは，2MB の L2 キャッシュ容量に対してワーキングセット・サイズが十分に小さいため，DRAM 積層による恩恵を受けることができない．このように，L2 キャッシュサイズの拡大によるミス率削減効果は実行対象プログラムの特性に依存するため，第 2.2 節で示した性能向上条件を必ずしも満足するとは限らない．$MR_{L2\_REDUCTION}$ と $HT_{L2\_OVERHEAD}$ が DRAM スタック法の利得（*Profit*）に対して与える影響を**図 4** に整理する．ここで，利得とは，大容量 DRAM キャッシュの導入に起因する L1 ミスペナルティ削減効果を表す指標であり，以下のように定義する．

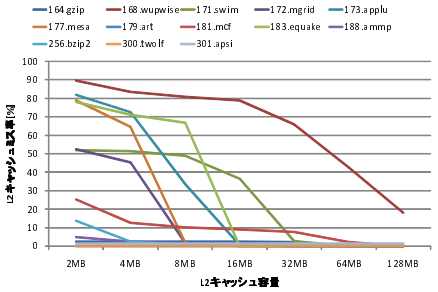

図 2 L2 キャッシュ容量と L2 キャッシュミス率：SPEC CPU 2000

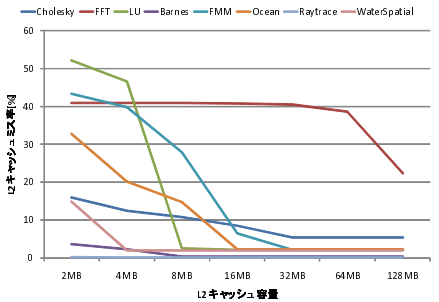

図 3 L2 キャッシュ容量と L2 キャッシュミス率：Splash2

$$Profit = \frac{MT_{L2\_REDUCTION}}{HT_{L2\_OVERHEAD}} \quad (5)$$

– $MT_{L2\_REDUCTION} = MR_{L2\_REDUCTION} \times MMAT$

つまり，*Profit* の値が 1.0 より大きい場合は DRAM スタック法の導入により性能改善が期待できることを意味する．なお，図 4 の $HT_{L2\_OVERHEAD}$ に関しては 200 クロックサイクルまでを表示している．図中には，各ベンチマークにおける利得をプロットしている．これらの結果から，多くのベンチマークにおいて DRAM スタック法の導入により性能低下が発生する可能性があることが分かる．

- **L2 キャッシュミス率改善効果はプログラム中に変動するため，DRAM スタック法の潜在能力を十分に引出すことができない．**一般に，プログラムの実行において，メモリ参照の振る舞いは多々刻々と変化するため，それに伴いキャッシュミス率も変動する．*Ocean* ベンチマークプログラムを実行した際の L2 キャッシュミス発生頻度の変動を**図 5** に示す．横軸は L2 キャッシュアクセス 10 万回を 1 区間とする時間経過を表しており，縦軸は各区間において発生した L2 キャッシュミス回数である．図 5 より，プログラム実行中に L2 キャッシュミス発生頻度が変化していることが分かる．特に，キャッシュサイズを 2MB

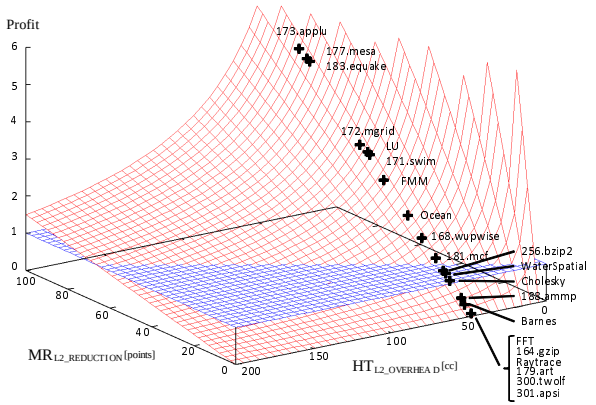

図 4 $MR_{L2\_REDUCTION}$ と $HT_{L2\_OVERHEAD}$ が $Profit$ に対して与える影響

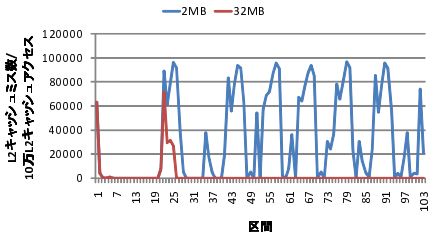

図 5 L2 キャッシュアクセス 10 万回あたりの L2 キャッシュミス数：Ocean

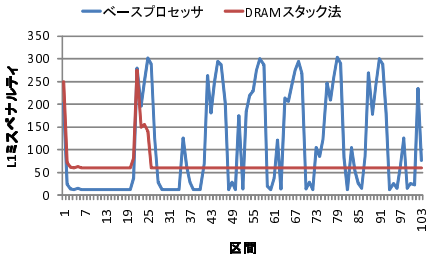

図 6 プログラム実行中の各区間における L1 ミスペナルティ：Ocean

から 32MB へと拡大することにより，ミス発生回数削減効果が大きい区間（たとえば，区間 41 から 47）と小さい区間（たとえば，区間 1 から 20）が存在することが分かる．また，L2 ヒット時間も考慮した DRAM スタック法によるメモリ性能改善効果の変動を**図 6** に示す．縦軸は各区間における L1 ミスペナルティ（$HT_{L2} + MR_{L2} \times MMAT$）であり，横軸は図 5 と同様に区間を表す．これらの実験結果から，プログラムの実行において，DRAM スタック法の適用により性能低下が生じる区間が存在することが分かる．

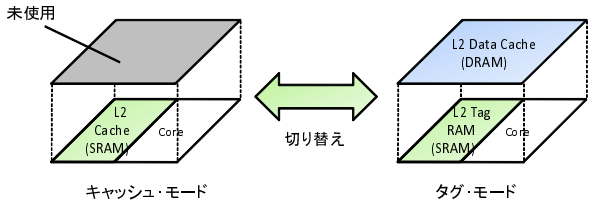

図 7 ハイブリッド・キャッシュを搭載した 3 次元 DRAM 積層プロセッサ

3. 3 次元積層プロセッサ向けハイブリッド・キャッシュ・アーキテクチャ

3.1 基 本 概 念

第 2.3 節で述べたように，従来の DRAM スタック法では，十分な L2 キャッシュミス削減率を達成できなければ性能が低下する．この問題を解決する単純な方法として，ベースプロセッサと同様，SRAM で構成される L2 キャッシュを下層に実装し，上層 DRAM を L3 キャッシュとして活用することが考えられる．しかしながら，この場合，第 2.1 節で説明したように，下層に実装されるタグメモリはベースプロセッサの L2 キャッシュと同程度の容量が必要となるため，下層ダイの面積オーバヘッドが大きくなる．

そこで本研究では，大幅な面積の増加を伴うことなく，DRAM スタック法の問題を解決し，さらなる高性能化を実現する新しいメモリ構成法として**ハイブリッド・キャッシュ・アーキテクチャ**を提案する．**図 7** に示すように，ハイブリット・キャッシュは下層に実装される SRAM メモリ領域であり，以下 2 つの動作モードを有する．

- **キャッシュ・モード：**通常の L2 キャッシュメモリとして動作する．このとき，上層 DRAM は使用されず，オンチップ・メモリアーキテクチャはベースプロセッサと同様になる．
- **タグ・モード：**上層 DRAM 用のタグメモリとして動作する．このとき，従来の DRAM スタック法と同様に上層 DRAM は L2 キャッシュとして使用される．

つまり，実行対象プログラムのワーキングセット・サイズが小さい場合には，ハイブリッド・キャッシュは通常の L2 キャッシュメモリとして動作し，上層 DRAM へのアクセスを禁止する．一方，より大きな L2 キャッシュが必要だと判断した場合には，上層 DRAM を L2 キャッシュとして使用するためにタグメモリとして動作する．このように，選択的に大容量 DRAM を活用することで，高速アクセスとキャッシュミス率改善の両立を可能にし，3 次元積層 DRAM の潜在能力を最大限に引き出す．

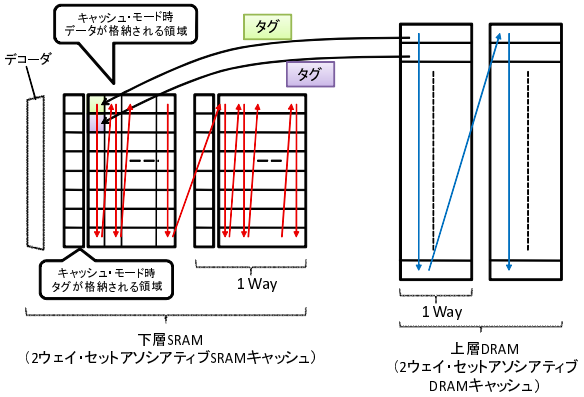

図 8　タグモード時のタグのマッピング

3.2　マイクロ・アーキテクチャ

ハイブリッド・キャッシュでは，通常のデータキャッシュならびにタグメモリとしての動作切り替えが可能でなければならない．そこで，タグモード動作時，ハイブリッド・キャッシュは自身のデータアレイに上層 DRAM キャッシュ用タグを格納する．ここで，ハイブリッド・キャッシュを実装するためには以下の 2 点を考慮する必要がある．

- **下層ハイブリッド・キャッシュに対するタグ情報のマッピング：**一般に上層 DRAM はハイブリッド・キャッシュより大きな容量を有する．したがって，ラインサイズが同じ場合には，上層 DRAM と比較して，下層 SRAM キャッシュの総キャッシュライン数は少なくなる．そのため，ハイブリッド・キャッシュでは 1 個のキャッシュラインに複数タグを格納しなければならない．この様子を**図 8** に示す．上層 DRAM の各ウェイに対応するタグに関して，ハイブリッド・キャッシュへ垂直方向にマッピングする．なお，この例では，上層 DRAM キャッシュと下層ハイブリッド・キャッシュの連想度は共に 2 と仮定しているが，実際には異なる連想度の場合でも対応可能である（厳密には条件があるが，詳細は後ほど述べる）．
- **データアレイに格納されたタグ読出しのためのハードウェア・サポート：**ハイブリッド・キャッシュのマイクロアーキテクチャを**図 9** に示す．ここで，ハイブリッド・キャッシュならびに上層 DRAM に関して，それぞれの容量を C_S と C_D，ラインサイズを L_S と L_D，連想度を W_S と W_D で記す．アドレス長は 64 ビットと仮定する．連想度とラインサイズが同じ場合，ハイブリッド・キャッシュのセット数は上層 DRAM のそれと比較して少ない．そのため，前述したように，ハイブリッド・キャッシュは上層 DRAM のタグを垂直方向に折り畳んで格納する．したがって，データアレイより読出したデータの中から該当するタグを選択するための専用マルチプレクサが必要となる（図 9 の MUX2 ならびに MUX3）．また，タグ比較を行うための比較回路等が追加される．これらの追加構成要素は小規模かつ容易に実装でき，アクセス時間や実装面積に与える影響は無視できるほど小さいと考えられる．

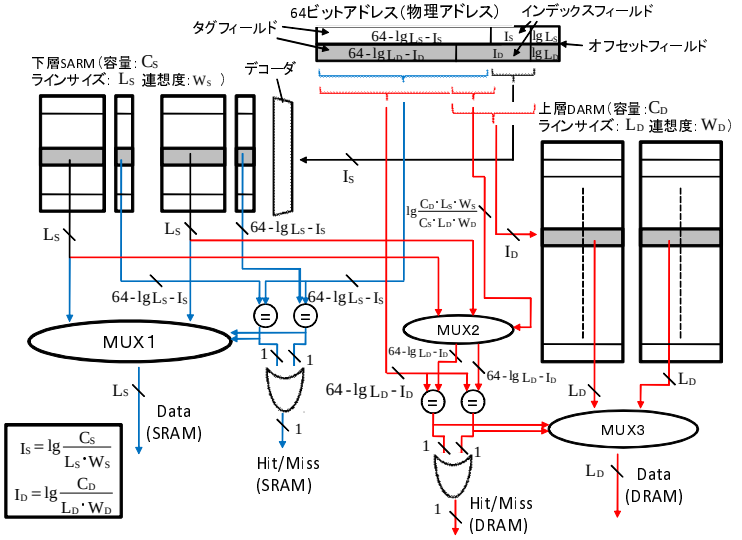

図 9　ハードウェア・アーキテクチャ

タグモード時のデータ読出しは以下の流れで行われる．インデックスは上位 $\frac{C_D L_S W_S}{C_S L_D W_D}$ ビットと下位 I_S ビットの 2 つに分割される．まず，下位 I_S ビットでハイブリッド・キャッシュの 1 セットに格納されている全タグを読出す．次に，上位 $\frac{C_D L_S W_S}{C_S L_D W_D}$ ビットで 1 つのセットに格納されているすべてのタグから W_D 個のタグを選択する．以降は通常のキャッシュと同様，メモリ参照アドレスより得たタグと比較し，一致するものが存在すればヒット，存在しなければミスとなる．ヒットの場合には，インデックスにより上層 DRAM から読出されたデータから当該タグに対応するデータを選択する．ただし，図 8 で示したタグ情報のマッピ

ング，ならびに，図 9 のマイクロアーキテクチャを前提とした場合，ハイブリッド・キャッシュが正しく動作するためには以下の条件を満足する必要がある．

- 下層 SRAM と上層 DRAM の連想度が 2 のべき乗である：この条件を満たさない場合，インデックスの上位ビットではビット数が不足し，1 つのキャッシュラインに格納されているタグから選択できないためである．
- $I_S \leq I_D$（$\frac{C_S}{L_S W_S} \leq \frac{C_D}{L_D W_D}$）：この条件を満たさない場合には，タグモード動作時において，上層 DRAM のタグ読出しにのためにハイブリッド・キャッシュの複数セットにアクセスする必要が生じる．
- $\frac{C_D W_S}{C_S W_D}(64 - lgL_D - lgI_D) \leq L_S$：この条件を満たさない場合は，上層 DRAM の全てのタグをハイブリッド・キャッシュに格納できない．

3.3 動作モード切り替え方針

プログラムが必要とするキャッシュ容量は，プログラム間ならびにプログラム内にて変化する．これに対し，提案するハイブリッド・キャッシュでは，動作モードの変更により上層 DRAM を選択的に活用し，プログラム実行に応じた適切な L2 キャッシュサイズを実現する．したがって，提案方式の有効性は,「如何に適切に動作モードを切り替え可能か?」に大きく依存する．ハイブリッド・キャッシュの動作モード選択に関しては，少なくとも以下 2 つの選択肢が考えられる．

- **静的切り替え：**プログラムの実行前にいずれの動作モードがより高性能となるかを予測し，それに基づきプログラム実行開始時に動作モードを設定する．また，プログラム実行中は動作モードを変更しない．動作モード切り替えに要するオーバヘッドは殆ど発生しないという利点がある．しかしながら，同一プログラムにおいても入力データによって 2 つの動作モードの優劣が大きく異なる場合には適切な動作モードの予測が難しい．また，プログラム実行におけるメモリ参照の振舞いの変化には対応できない．
- **動的切り替え：**プログラム実行中に動作モードを切り替える．プログラム実行におけるメモリ参照の振舞いの変化にも追従できるため，静的切り替えよりも大きな性能向上を期待できる．しかしながら，動作モード切り替えにおけるオーバヘッドが発生する場合には，ベースプロセッサや DRAM スタック法より性能が低下する可能性がある．また，適切な動作モードを決定するための専用回路が必要となり，場合によっては面積の増大を引起こすといった欠点がある．

3.4 利点と欠点

提案するハイブリッド・キャッシュは，実行対象プログラムが有するワーキングセット・サイズに応じて上層 DRAM を選択的に活用することで，小容量かつ高速 L2 キャッシュと大容量低速 L2 キャッシュの使い分けを可能にする．これにより，従来の DRAM スタック法と比較して，より高いメモリ性能を実現することができる．これに加え，ハイブリッド・キャッシュがキャッシュモードとして動作する際には上層 DRAM は使用されない．そのため，消費エネルギー削減効果も期待できる．

一方，ハイブリッド・キャッシュの実装においては下層ダイの面積が増加するといった問題がある．原因は次の 2 つである．まず，マルチプレクサ等の追加により回路規模が増大する．しかしながら，通常，キャッシュ面積の殆どはメモリアレイが占める．したがって，この面積拡大による影響は無視できる程度に小さいと考えられる．次に，第 3.2 節で述べた実装条件の 3 番目を満足するために発生する面積オーバヘッドである．上層 DRAM キャッシュに関する全てのタグ情報を格納可能とするためには，必要に応じてハイブリッド・キャッシュのデータアレイを拡大する必要がある．たとえば，ハイブリッド・キャッシュの容量を 4MB，ラインサイズを 64B，連想度を 16 とする．また，上層 DRAM キャッシュの容量を 64MB，ラインサイズを 64B，連想度を 16 とする．このような状況において，上層 DRAM キャッシュの全てのタグ情報を格納するためには 4.875MB のメモリ容量が必要となり，ハイブリッド・キャッシュのデータアレイ面積が約 20%拡大する．

4. 性 能 評 価

4.1 評 価 環 境

第 3.3 節で述べたように，ハイブリッド・キャッシュの実現法としては静的動作モード切り替えと動的動作モード切り替えが考えられる．これら 2 つの実現方式において，その性能は動作モード決定アルゴリズムに大きく依存する．しかしながら，動作モード決定アルゴリズムと実装法に関する詳細は現在検討中である．そこで本稿では，ハイブリッド・キャッシュによる性能向上限界の評価を目的とする．評価対象モデルは以下の通りである．

- **2D-BASE：**第 2.1 節の図 1 で示したベースプロセッサ．L1 キャッシュ・ミスが発生した際には，常に SRAM 実装された L2 キャッシュにアクセスする．
- **3D-CONV：**第 2.1 節の図 1 で示した従来の DRAM スタック法であり，上層 DRAM は L2 キャッシュとして動作する．L1 キャッシュ・ミスが発生した際には，SRAM で実装された下層タグメモリにアクセスしヒット/ミス判定を行うと同時に，上層 DRAM より当該データを読出す（読出しアクセスの場合）．
- **3D-HYBRID-STATIC：**静的モード切り替えに基づくハイブリッド・キャッシュ．図 9

表 1 L2 キャッシュ以外のシミュレータの設定

コア数		1
命令発行方式		インオーダ
L1 命令/データキャッシュ	サイズ	32KB
	連想度	8
	ラインサイズ	64B
	アクセス時間	1clock cycle
主記憶アクセス時間		300 clock cycles
L1-L2 間バス幅		64B
L2-主記憶間バス幅		16B

表 2 L2 キャッシュに関するシミュレータの設定

	2D-BASE キャッシュ・モード	3D-CONV タグ・モード
サイズ	2MB	32MB
連想度	8	8
ラインサイズ	64B	64B
アクセス時間	12 clock cycles	60 clock cycles

表 3 Splash2 ベンチマークプログラムの入力

Cholesky	tk29.0
FFT	4M data points
LU	1024 × 1024 matrix
Barnes	32K particles
FMM	64K particles
Ocean	258 × 258 ocean
Raytrace	balls4
WaterSpatial	4096 molecules

で示すように，実行対象プログラムのワーキングセット・サイズに応じてプログラム実行開始時にタグモードもしくはキャッシュモードのいずれかを設定する．プログラム実行中には動作モードの変更は行わない．ここで，より高い性能を達成できる動作モードは既知であると仮定する．具体的には，各ベンチマーク・プログラムにおいて評価時と同一入力を用いた事前解析を行い，ハイブリッド・キャッシュの動作モードを決定した．

- **3D-HYBRID-DYNAMIC**：動的モード切り替えに基づくハイブリッド・キャッシュ．図 9 で示すように，実行対象プログラムのワーキングセット・サイズに応じてプログラム実行中にタグモードもしくはキャッシュモードのいずれかを選択する．また，動作モード切り替えに要する性能オーバヘッドは発生しない．ここで，プログラム実行での各区間（1 区間は 10 万 L2 キャッシュ・アクセス）それぞれにおいて，より高い性能を達成できる動作モードは既知であると仮定する．具体的には，タグモードならびにキャッシュモードそれぞれに関して評価時と同一入力を使用した事前実行を行い，より高い性能を達成可能な動作モードを解析した．

評価にはミシガン大学で開発された M5 シミュレータ[1] を用いた．また，評価指標は式 (1) で表される平均メモリアクセス時間 AMAT とする．各種コンフィギュレーション設定値を**表 1** ならびに**表 2** にまとめる．これらの設定値は，主に文献 2) を参考に決定した．ベンチマークプログラムは，SPEC-CPU2000 ベンチマークセット[3] から 13 個（入力は train），Splash2 ベンチマークセット[6] から 8 個（入力は**表 3**）を選択した．

4.2 評価結果

評価結果を**図 10** ならびに**図 11** に示す．それぞれの図において，横軸はベンチマークプログラム，縦軸はベースプロセッサに対するメモリ性能向上比である．各ベンチマークプログラムにおいて，左から 2D-BASE，3D-CONV, 3D-HYBRID-STATIC，ならびに，3D-HYBRID-DYNAMIC の性能向上比を表す．

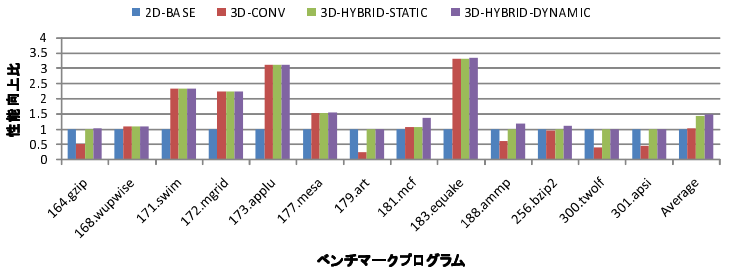

図 10 性能向上比：SPEC CPU 2000

まず，従来の DRAM スタック法である 3D-CONV に関して議論する．幾つかのプログラムにおいて性能が低下している．たとえば，*164.gzip*，*Cholesky*，*FFT* などが相当する．これは，第 2.2 節ならびに第 2.3 節で示したように，L2 キャッシュアクセス時間オーバヘッドに対し，それを隠蔽できるだけの L2 キャッシュミス率削減効果を得ることができなかったためである．実際，性能向上条件式 (4) に対して本実験で得た値を代入すると，

$$MR_{BASE} - MR_{DRAM} > 0.16 \quad (6)$$

となる．つまり，3D-CONV において性能向上を達成するためには，上層 DRAM の活用により 16%ポイント以上の L2 キャッシュミス率削減を実現しなければならない．しかしながら，図 2 ならびに図 3 より，これらのプログラムに関しては L2 キャッシュサイズを 2MB から 32MB に増加した場合でも十分な L2 ミス率削減を達成できていないことが分かる．

次に，提案するハイブリッド・キャッシュの性能向上効果について考察する．静的動作モード選択方式である 3D-HYBRID-STATIC では，上層 DRAM の活用により性能が低下するプログラムを実行する際にはキャッシュモードとして動作する．そのため，3D-CONV に

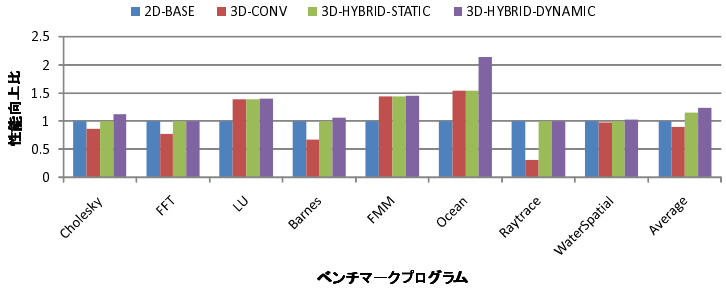

図 11 性能向上比：Splash2

表 4 Splash2 ベンチマークプログラムの $Rate_{tag}$

	Cholesky	FFT	LU	Barnes	FMM	Ocean	Raytrace	WaterSpatial
$Rate_{tag}$[%]	12.7	0	96.8	4.11	57.0	48.5	0.0289	25.9

おいて性能低下が発生するプログラムに関しては，2D-BASE と同じ性能になる．これにより，全ベンチマークの平均では，2D-BASE に対して約 31%，3D-CONV に対して約 35%の性能向上を実現している．一方，動的動作モード選択方式である 3D-HYBRID-DYNAMIC では，全てのベンチマーク・プログラムにおいて性能向上を達成している．また，*181.mcf*，*188.ammp*，ならびに，*Ocean* に関しては，3D-HYBRID-STATIC よりも更なる高性能化を実現する．平均すると，2D-BASE に対して約 38%，3D-CONV に対して約 43%の性能向上となる．しかしながら，*181.mcf* 等を除く多くのベンチマークにおいて，3D-HYBRID-STATIC からの性能改善は平均で約 5%と低かった．これは，*181.mcf* 等を除く多くのベンチマークプログラムがプログラム実行中殆どの区間においてキャッシュ・モードが高性能となる，もしくはタグ・モードが高性能となるためである．**表 4** に Splash2 ベンチマークプログラムの 3D-HYBRID-DYNAMIC においてタグ・モードが高性能となる区間の割合 $Rate_{tag}$ を示す．この表から分かるように，*Ocean* や *FMM* 以外のプログラムはタグ・モードが高性能となる区間が 0%，もしくは 100% に近い．*FMM* は 50%に近いが，キャッシュ・モードが高性能となる区間では，タグ・モードの性能に対し性能向上が低いために，3D-HYBRID-DYNAMIC の性能向上が低い．

5. おわりに

本稿では，3 次元積層 DRAM の利用を前提としたキャッシュ・アーキテクチャとして，ハイブリッド・キャッシュ・アーキテクチャを提案した．評価実験の結果，提案方式の静的制御方式で平均 35%，動的制御方式で平均 43%の性能向上を達成した．

今後は，動作モード決定アルゴリズムの考案とハイブリッド・キャッシュの実装，およびその性能評価を行う．さらに，各評価対象モデルにおいて L2 キャッシュのヒット時間を変化させた場合の性能の評価を行う．また，ハイブリッド・キャッシュでは消費エネルギーの削減も期待できる．したがって，消費エネルギーの評価も今後行う予定である．

謝辞

日頃から御討論頂いております九州大学安浦・村上・松永・井上研究室ならびにシステム LSI 研究センターの諸氏に感謝します．本研究は主に九州大学情報基盤研究開発センターの研究用計算機システムを利用しました．なお，本研究は，独立行政法人新エネルギー・産業技術総合開発機構 (NEDO) 若手グラントの支援による．

参考文献

1) Binkert, N.L., Dreslinski, R.G., Hsu, L.R., Lim, K.T., Saidi, A.G. and Reinhardt, S.K.: The M5 Simulator: Modeling Networked Systems, *IEEE Micro*, Vol.26, No.4, pp.52–60 (2006).
2) Black, B., Annavaram, M., Brekelbaum, N., DeVale, J., Jiang, L., Loh, G.H., McCaule, D., Morrow, P., Nelson, D.W., Pantuso, D., Reed, P., Rupley, J., Shankar, S., Shen, J. and Webb, C.: Die Stacking (3D) Microarchitecture, *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE Computer Society, pp.469–479 (2006).
3) Jin, Z. and Cheng, A.C.: Evolutionary Benchmark Subsetting, *IEEE Micro*, Vol.28, No.6, pp.20–36 (2008).
4) Loh, G.H.: 3D-Stacked Memory Architectures for Multi-core Processors, *SIGARCH Comput. Archit. News*, Vol.36, No.3, pp.453–464 (2008).
5) Puttaswamy, K. and Loh, G.H.: Implementing Caches in a 3D Technology for High Performance Processors, *ICCD '05: Proceedings of the 2005 International Conference on Computer Design*, IEEE Computer Society, pp.525–532 (2005).
6) Woo, S.C., Ohara, M., Torrie, E., Singh, J.P. and Gupta, A.: The SPLASH-2 programs: characterization and methodological considerations, *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, ACM, pp.24–36 (1995).