3 次元積層 SRAM/DRAM ハイブリッド・キャッシュ

橋口慎哉[†] 福本尚人[†]
井上弘士[††] 村上和彰[††]

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 次元積層された 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，提案方式は動的制御方式で平均 15%の性能向上を達成した．

3 Dimensional Integrated SRAM/DRAM Hybrid Cache

Shinya HASHIGUCHI,[†] Naoto FUKUMOTO,[†] Koji INOUE[††]
and Kazuaki MURAKAMI[††]

This paper proposes a novel cache architecture for 3D-implemented microprocessors. 3D-IC is one of the most interesting techniques to achieve high-performance, low-power VLSI systems. Stacking multiple dies makes it possible to implement microprocessor cores and large caches (or DRAM) into the same chip. Unfortunately, applying the 3D DRAM cache causes performance degradation for some programs, because increasing cache size makes access time longer. To tackle this issue, the proposed cache supports two operation modes: a fast but small SRAM cache mode and a slow but large DRAM cache mode. An appropriate operation mode is selected at run time based on the behavior of application programs. The evaluation results show that the proposed approach achieves 15% of memory performance improvement.

1. はじめに

半導体チップの新しい実現法として 3 次元積層が注目されている．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 次元積層 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プロセスを経て製造した複数のダイを積層する事も比較的容易になる．

† 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†† 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University

このような背景の中，3 次元積層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経て作成した大容量 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[1)2)3)]．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 1) 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 RAM 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すべく，新しいメモリ構成法として**SRAM/DRAM ハイブリッド・キャッシュ**を提案する．本方式は，高速かつ小容量な「SRAM キャッシュ・モード」，または，3 次元積層 DRAM を用いた低速かつ大容量な「DRAM キャッシュ・モード」での動作が可能である．アプリケーション特性に応じて適切な動作モードを選択することで，高速アクセスの実現とキャッシュミス率の改善を両立し，メモリ性能の向上を可能にする．

以下，第 2 節では文献 1) 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示す．第 4 節ではベンチマークプログラムを用いて定量的評価を行い，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 1) 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 次元積層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☆．このタグ RAM 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依存する．たとえば，上層 DRAM の容量が 64MB，

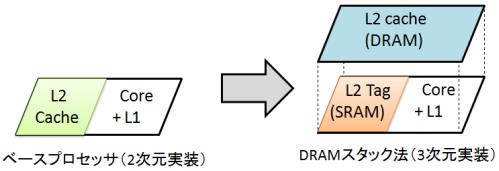

図 1 ベースプロセッサと DRAM スタック法の構成
Fig. 1 Architecture of BaseProcessor & DRAM-Stacking.

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（もしくはそれ以上）の実装面積を要する．そこで，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ため，DRAM スタック法では L2 キャッシュアクセス時間が増大し，引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．

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

- HT_{L1}/HT_{L2}:L1/L2 キャッシュヒット時間 [cycles]
- MR_{L1}/MR_{L2}：L1/L2 キャッシュミス率
- $MMAT$：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，メモリ階層における 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，$MMAT$ は同一となる．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．厳密には回路構成やデバイス特性に異存するが，一般的には以下の関係式が成り立つ．

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

$$MR_{L2\_BASE} \geq MR_{L2\_DRAM} \quad (3)$$

したがって，DRAM スタック法がベースプロセッサ

☆ 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．

の性能を上回るための条件は式 (4) となる．

$$HT_{L2\_BASE} + MR_{L2\_BASE} \times MMAT > HT_{L2\_DRAM} + MR_{L2\_DRAM} \times MMAT \quad (4)$$

この式を変形すると，式 (5) が導き出される．

$$MR_{L2\_BASE} - MR_{L2\_DRAM} > \frac{HT_{L2\_DRAM} - HT_{L2\_BASE}}{MMAT} \quad (5)$$

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ヘッド $HT_{L2\_OVERHEAD}$ 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を隠蔽できる 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．ここで，ベースプロセッサに関しては，2MB の 8 ウェイ・セットアソシアティブ方式 (ラインサイズは 64B) であり，そのアクセス時間は 6 クロックサイクルとする．一方，DRAM スタック法においては 32MB の 8 ウェイ・セットアソシアティブ方式 (ラインサイズは 64B) であり，アクセス時間は 28 クロックサイクルと仮定する．主記憶アクセス時間 $MMAT$ は 181 クロックサイクルであり，その他の実験環境の詳細については，第 4.1 節で示す内容と同じである．一般に，キャッシュヒット時間は，キャッシュ構成，ならびに，メモリ素子やデコーダ，センスアンプと言った各構成要素の動作速度によって決定されるため，$HT_{L2\_OVERHEAD}$ は実行プログラムによらない．これに対し，$MR_{L2\_REDUCTION}$ は実行対象となるプログラムの特性に大きく依存するため，DRAM スタック法では以下の問題点が生じる．

- **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 2MB から 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増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満たすとは限らない．$MR_{L2\_REDUCTION}$ 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．

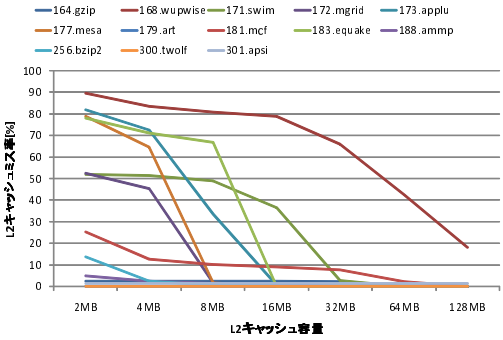

図 2 L2 キャッシュ容量と L2 キャッシュミス率：SPEC CPU 2000
Fig. 2 L2 Size-Missrate:SPEC CPU 2000.

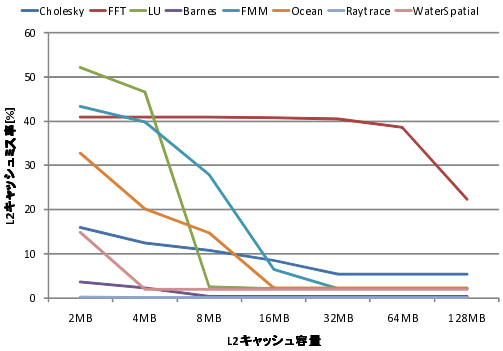

図 3 L2 キャッシュ容量と L2 キャッシュミス率：Splash2
Fig. 3 L2 Size-Missrate:Splash2.

$$Profit = \frac{MR_{L2\_REDUCTION} \times MMAT}{HT_{L2\_OVERHEAD}} \quad (6)$$

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

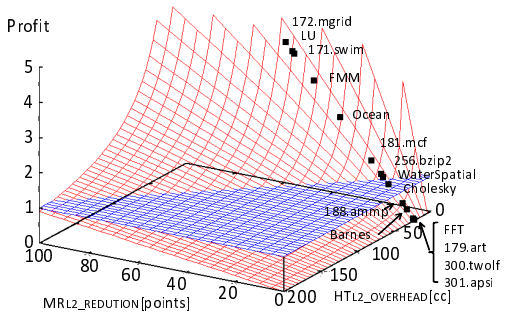

図 4 $MR_{L2\_REDUCTION}$ と $HT_{L2\_OVERHEAD}$ が $Profit$ に対して与える影響
Fig. 4 Effect of $MR_{L2\_REDUCTION}$ and $HT_{L2\_OVERHEAD}$.

を意味する．図 4 には，*Profit* が 1.0 以下のベンチマークプログラムが多くあるため，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単一プログラム実行中にも変動するため，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時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大きく変化していることが分かる．特に，キャッシュサイズを 2MB から 32MB へ増大することにより，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．

3. 3 次元積層プロセッサ向け SRAM/DRAM ハイブリッド・キャッシュ

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**図 7** に示す **SRAM/DRAM ハイブリッド・キャッシュ**を提案する．本方式では以下の 2 つの動作モードを有する．

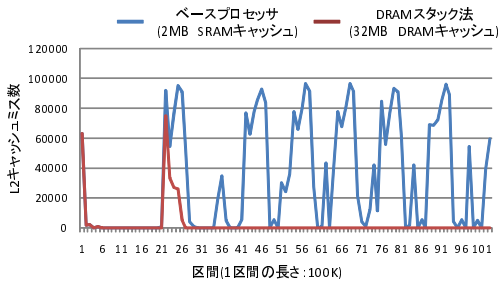

図 5 L2 キャッシュアクセス 10 万回あたりの L2 キャッシュミス数：Ocean
Fig. 5 L2 cache misses per 100000 accesses:Ocean.

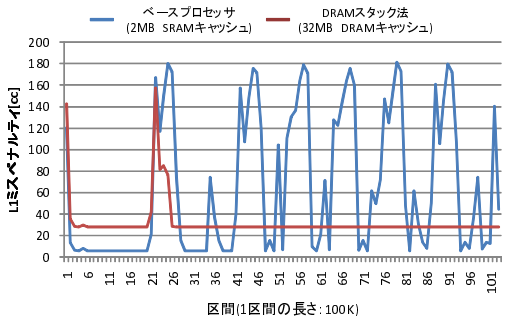

図 6 プログラム実行中の各区間における L1 ミスペナルティ：Ocean
Fig. 6 Run-time variation of L1 MissPenalty:Ocean.

- **SRAM キャッシュ・モード：**下層 SRAM が通常の L2 キャッシュとして動作する．このとき，上層 DRAM は使用せず，消費電力削減のために電源供給を遮断する．
- **DRAM キャッシュ・モード：**下層 SRAM が上層 DRAM 用のタグ RAM 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

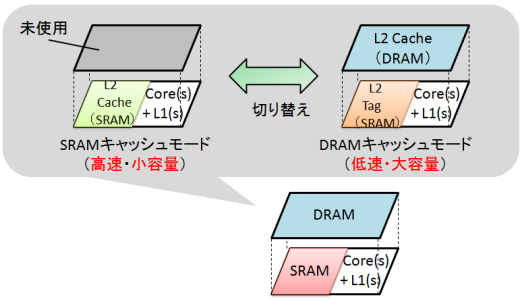

図 7 SRAM/DRAM ハイブリッド・キャッシュ
Fig. 7 SRAM/DRAM Hybrid Cache Architecture.

DRAM は L2 キャッシュのデータ（キャッシュライン）保存用メモリとして使用される．

SRAM/DRAM ハイブリッド・キャッシュでは，L2 キャッシュミス率削減効果が十分に得られる場合にのみ積層 DRAM を活用する．それ以外の場合には通常の L2 キャッシュメモリとして動作し，上層 DRAM へのアクセスは行わない．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．

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，下層 SRAM は通常のデータキャッシュならびにタグ RAM としての動作切り替えが可能でなければならない．DRAM キャッシュ・モード動作時，下層 SRAM 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，下層 SRAM の切替えを実現するためには以下の 2 点を考慮する必要がある．

- **タグ情報のマッピング：**一般に上層 DRAM は下層 SRAM 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．そこで，DRAM キャッシュ・モード時，下層 SRAM では 1 個のキャッシュラインに複数タグを格納する．連想度が 2 の場合の例を**図 8** に示す．上層 DRAM の各ウェイに対応するタグに関して，下層 SRAM へ垂直方向にマッピングする．なお，本マッピングは連想度が 2 以外の場合でも対応可能である．
- **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，下層 SRAM 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，下層 SRAM 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，下層 SRAM 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などが追加される．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．

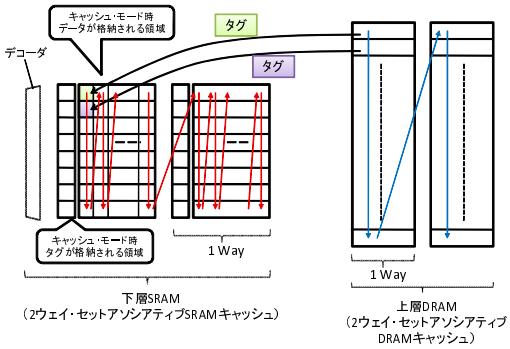

図 8 DRAM キャッシュモード時のタグのマッピング
Fig. 8 DRAM-Tag mapping for SRAM of bottom layler.

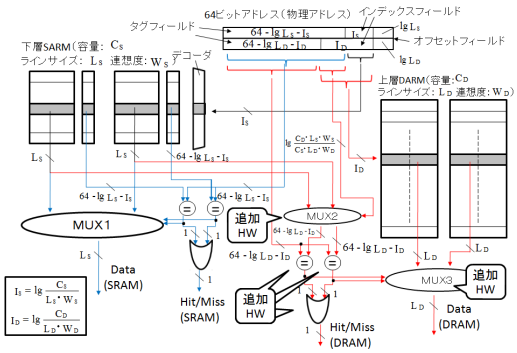

図 9 ハードウェア・アーキテクチャ
Fig. 9 HW-support.

DRAM キャッシュ・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 $I_D - I_S$ 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 ビットで下層 SRAM 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 $I_D - I_S$ 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，下層 SRAM 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，DRAM キャッシュ・モード動作時において，上層 DRAM のタグ読出しのため，下層 SRAM 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 の全てのタグを下層 SRAM キャッシュに格納できない．

3.3 動的動作モード決定法

3.3.1 動作モード決定方針

SRAM/DRAM ハイブリッド・キャッシュでは，如何に適切な動作モードを選択できるかが極めて重要になる．適切な動作モード (より高い性能を達成できる動作モード) は，プログラム実行におけるメモリ参照の振舞い，ならびに，SRAM と積層 DRAM のハードウェア特性に大きく依存する．ハイブリッド・キャッシュでの動作モード決定においては，動作モード決定時期ならびに動作モード設定時期に関して，プログラム実行前または実行中の選択肢が考えられる．本稿では，サーバやデスクトップ PC といった汎用システムでの応用を想定している．この場合，実行コードの互換性は極めて重要となる．また，組込みシステムとは異なり，多くの場合において，実行時の振舞いを事前に解析することは難しい．これらの理由により，本稿ではプログラム実行時にハードウェアレベルで動作モードを決定する選択肢を採用する．

3.3.2 動作モード切替えにおける性能オーバヘッド

ハイブリッド・キャッシュにおいて，プログラム実行中の動作モード切替えは以下の手順で行われる．

(1) L2 キャッシュへのアクセスを禁止する．プロセッサ側から L2 キャッシュへのアクセス要求があれば，動作モード切替えが終了するまでプロセッサはストールする．
(2) 現在使用している L2 キャッシュ（つまり，動作モード切替え前に使用している L2 キャッシュ）をフラッシュする．動作モードの切替えにより L2 キャッシュのデータ/タグ情報の記憶領域が変更されるために必要となる．
(3) データ/タグ情報記憶領域へのアクセスパスを変更する（詳細は第 3.2 節を参照）．これにより動作モードの切替えが完了する．
(4) L2 キャッシュへのアクセスを再開する．

動的に動作モードを切り替えることで大きな性能向上が期待できる一方で，上述したようにキャッシュをフラッシュする必要があるため，性能向上阻害要因として以下の 2 つの問題が生じる．

- **ライトバックに伴う性能オーバヘッド：**動作モード切替え前の L2 キャッシュ内のダーティ・ラインを全て主記憶へ書き戻す必要がある．そのため，メモリバンド幅を圧迫し性能低下が発生する可能性がある．
- **初期参照ミスの増加：**動作モード切替え直後，L2 キャッシュは空の状態である．そのため，見かけ上の初期参照ミス (動作モード切替えが原因で新たに発生する初期参照ミス) が増加する．

3.4 動的動作モード決定アルゴリズム

プログラム実行中の動作モード決定では，1) 適切な動作モードを如何に選択できるか，ならびに，2) 動作モード切替えオーバヘッドを如何に低減するか，が重要となる．本アルゴリズムでは，プログラム実行を一定回数の L2 キャッシュアクセスで分割し (これを区間と呼ぶ)，各区間毎に適切な動作モードを決定する．具体的には，「**連続する 2 区間では適切な動作モードが同じになる確率が高い**」と想定して，**図 10** のようにある区間 N での適した動作モードを次区間 N+1 の動作モードとする．各区間の適した動作モードについては，式 (5) に基づき決定する．すなわち，区間 N での実行において，当該区間を SRAM キャッシュ・モードで動作したと仮定した場合の L2 ミス率 (式 (5) での MR_{BASE} に相当) および DRAM キャッシュ・モードで動作したと仮定した場合の L2 ミス率 (式 (5) での MR_{DRAM} に相当) を求める．そして，式 (5) が成り立つ場合には区間 N+1 を DRAM キャッシュ・モードで実行する．一方，成り立たない場合は区間 N+1 の実行は SRAM キャッシュ・モードとなる．図 10 に示すように，動作モードを切り替えた直後はウォームアップ区間として動作モード切替えを禁止する．両方の動作モードでの L2 ミス率は各区間の実行において同時に得る必要がある．現在実行中の動作モードでの L2 ミス率に関しては単純なハードウェア・カウン

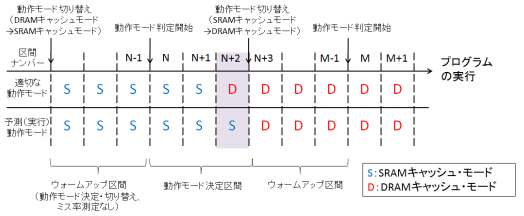

図 10 動作モード決定法
Fig. 10 Strategy of Run-time Operation Mode Selection.

タを搭載すれば良い．これに対し，現在の動作モードとは異なる動作モードの L2 ミス率を直接実測することは困難である．そこで，次節で説明するハードウェア・サポートにより L2 ミス率の推定を行う．なお，式 (5) における HT_{L2_BASE}，HT_{L2_DRAM}，ならびに，$MMAT$ はハードウェア特性にのみに依存しており，チップ製造後には基本的に既知と考えることができる．

3.5 キャッシュミス率推測法

キャッシュミス率はタグ情報のみから得ることができる．そこで，DRAM キャッシュ・モード時には，下層 SRAM のタグメモリが未使用になることを利用し，ここに SRAM キャッシュ・モードを想定してタグ情報を格納する．これにより，DRAM キャッシュ・モード時においても SRAM キャッシュ・モードでのヒット/ミス回数を求めることが可能となる．同様に，SRAM キャッシュ・モード時においても，DRAM キャッシュに格納可能なライン数分のタグ情報を保存するメモリを搭載すれば，DRAM キャッシュ・モードでの L2 ミス率を測定することができる．しかしながら，この場合には極めて大きな面積オーバヘッドが発生する．例えば，32MB の DRAM キャッシュにおいてラインサイズを 64B と仮定するとタグ領域は 2.5MB 程度必要となる．この問題を解決するため，文献[4] で提案された手法と同様，幾つかのセットに対応する少数のタグ情報のみを保存するアプローチを採用する．

図 11 に，少数のタグを保持し，DRAM キャッシュのミス率を推測するハードウェア・サポートを示す．本来必要な容量を持つタグ RAM とは別に，キャッシュセットをサンプリングし本来必要なタグ総数の $1/n$ のみ保持する小容量のタグ RAM を下層ダイに配置する．そして，L2 キャッシュアクセスが発生した際，タグを保持すべきと判断された場合にのみ実際にそのタグ RAM へのアクセスを行う．具体的には，インデックス下位 $lg(n)$ ビットを参照し，0 であるならば別途設けたタグ RAM へアクセスする．この時，タグ RAM に対するアクセス数，ヒット数をカウントすることでこのタグ RAM のキャッシュミス率を算出し，これを

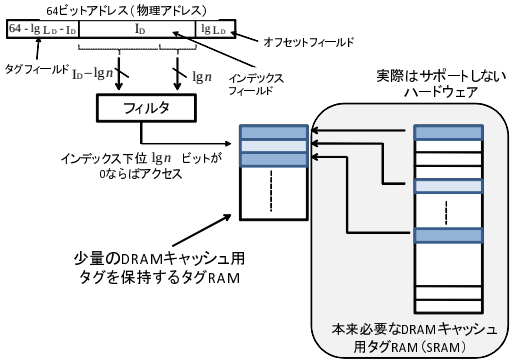

図 11 DRAM キャッシュミス率推測のためのハードウェア・サポート
Fig. 11 HW Support for DRAM Cache Miss Rate Estimation.

DRAM キャッシュのミス率とする．1/32 程度のタグのみ保持する場合でも高い精度でキャッシュミス率を推測することが可能である．

4. 評価実験

4.1 評価環境

本評価では，ミシガン大学で開発された M5 シミュレータ[5] を用いてトレースを取得し，メモリ性能値を算出した．評価指標は式 (1) で表す AMAT(平均メモリアクセス時間) である．提案方式においては，動作モード切替え時に発生するライトバック時間も含んでいる．具体的には，8GB/s のメモリバンド幅を想定し，全ダーティ・ラインのデータ転送に要する時間を用いて近似した．インオーダ命令発行のシングルコア・プロセッサを想定し，L1 命令/データキャッシュは，それぞれ，容量 32KB，連想度 2，ラインサイズ 64B，アクセス時間は 2 クロックサイクルとした．L2 キャッシュの容量，アクセス時間などを表 1 に示す．また，主記憶アクセス時間は 181 クロックサイクルとした．これらの値に関しては，CACTI ならびに関連研究を参考に決定した[1)6)7)]．ベンチマークプログラムは，SPEC-CPU2000[8]（入力は train）ならびに SPLASH-2[9]（入力は *Cholesky*：*tk29.0*，*FFT*：*4M data points*，*LU*：*1024* × *1024 matrix*，*Barnes*：*32K particles*，*FMM*：*64K particles*，*Ocean*：*258* × *258 ocean*，*WaterSpatial*：*4096 molecules* ）から 14 個選択した．評価対象モデルは以下の通りである．

- **DRAM-STACK**：大容量 DRAM をスタックする従来の DRAM スタック法 (図 1).
- **HYBRID-S-IDEAL**：各プログラムにおいて全ての実行を同一の動作モードで実行するハイブ

表 1 L2 キャッシュに関する設定
Table 1 Configuration of L2 Cache.

	SRAM $モード	DRAM-STACK DRAM $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	6 clock cycles	28 clock cycles

リッド・キャッシュ．ここで，各プログラムにおける適切な動作モードは既知であると仮定する．

- **HYBRID-D-IDEAL:**理想ハイブリッド・キャッシュ．HYBRID-D において，各区間において常に適切な動作モードを選択でき，かつ，動作モード切替えに伴う性能オーバヘッドは発生しないと仮定する．なお，1 区間は L2 キャッシュアクセス 10 万回である．
- **HYBRID-D-OPT：**第 3.4 節で示したアルゴリズムに基づき動的に動作モードを決定するハイブリッド・キャッシュ．実行開始時の動作モードは SRAM キャッシュ・モードであり，SRAM キャッシュ・モード時の DRAM キャッシュミス率推定用タグは本来必要な量の 1/32 を搭載している．各プログラムにおいて，1 区間の長さは最適な値を選択できると仮定する．
- **HYBRID-D：**1 区間の長さを 5M 回の L2 アクセスで固定とするハイブリッド・キャッシュ．区間の長さに関しては基礎実験に基づき平均的に良い結果となる値を選択した．これ以外に関しては HYBRID-D-OPT と同じである．

4.2 評価結果

性能評価結果を**図 12** に示す．横軸はベンチマークプログラム，縦軸は従来の 3 次元積層法である DRAM-STACK に対する性能向上比である．

まず，ハイブリッド・キャッシュの潜在的な性能向上について考察する．静的動作モード選択方式である HYBRID-S-IDEAL は，上層 DRAM の活用により性能が低下する場合は SRAM キャッシュ・モードとして動作するため，DRAM-STACK に対し性能が向上する．これにより，平均約 20%の性能向上を達成している．理想的な動的動作モード選択方式 HYBRID-D-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．これは，実行中のメモリ参照の振る舞いの変化に追従でき，性能オーバヘッドが発生しないためである．*ammp* では約 33%性能が向上し，ハイブリッド・キャッシュの潜在能力が高いことがわかる．

次に，実行時動作モード決定アルゴリズムに基づくハイブリッド・キャッシュについて考察する．*twolf* や *LU* のように HYBRID-S-IDEAL と HYBRID-D-IDEAL の性能がほぼ同じプログラムでは HYBRID-D-OPT でも変化がない．**図 13** は twolf を実行における区間ごとの L1 ミスペナルティ変化 (縦軸) を示している．twolf はプログラム実行を通してほぼ全ての区間にて適切なキャッシュモードが SRAM キャッシュ・モードであるため，HYBRID-D-OPT の変化が少ない．HYBRID-S-IDEAL に対し HYBRID-D-IDEAL の性能が高いプログラムでは，多くの場合 HYBRID-D-OPT の性能が低下している．これは，切替えオーバヘッドが大きく影響しているためである．**図 14** に *mcf* の実行中の L2 キャッシュミス率の変化を示す．HYBRID-D-OPT(*mcf* では 1 区間の長さは L2 キャッシュアクセス 5M 回) は適した動作モードを選択できているが，動作モード切り替え後の初期参照ミスが大きい区間があることが分かる．しかしながら，より高性能となる区間の長さで実行することにより HYBRID-S-IDEAL と同等の性能を達成しており，平均約 15%性能が向上する．一方で，全てのベンチマークプログラムで 1 区間の長さを L2 キャッシュアクセス 5M 回とした HYBRID-D は，DRAM-STACK 以下の性能となっているプログラムが存在する．これは，前述した切替えオーバヘッドの影響，ならびに，メモリ参照の追従性の低下が原因として挙げられる．**図 15** に *Ocean* の実行における平均 L1 ミスペナルティの変化を示す．図上は HYBRID-D-IDEAL(1 区間の L2 キャッシュアクセス回数は 100K)，図下は HYBRID-D の結果である．ここで，図 15 から，理想的には大きく性能が向上するが，1 区間が長くなるとメモリ参照の振る舞いの変化に対する追従性が大きく低下することが分かる．平均すると，DRAM-STACK に対し HYBRID-D では約 6%性能が向上する．

5. おわりに

本稿では，3 次元積層 DRAM を活用する新しいメモリ構成法として SRAM/DRAM ハイブリッド・キャッシュを提案した．評価実験の結果，SRAM/DRAM ハイブリッド・キャッシュは，従来手法と比較し平均約 15%の性能が向上し，有効であることを確認した．今後は，プログラム毎に動作モードを決定および設定する静的方式の検討を行う．また，消費エネルギーについての評価も行う予定である．

謝辞
日頃から御討論頂いております九州大学安浦・村上・

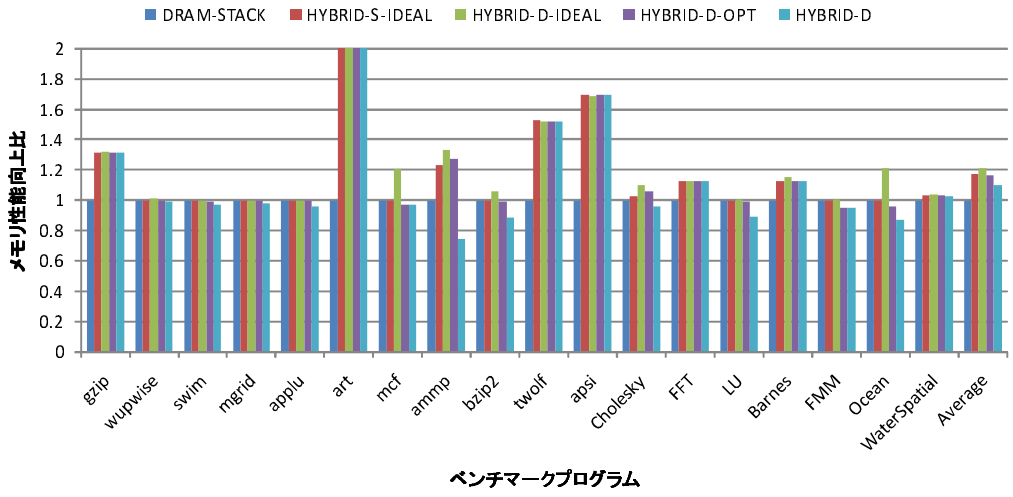

図 12 性能評価結果
Fig. 12 Performance Result.

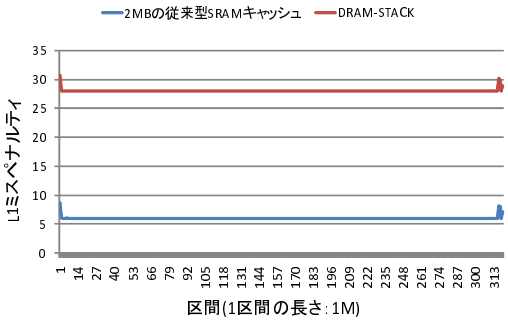

図 13 プログラム実行中の L1 ミスペナルティの変化：twolf
Fig. 13 Run-time variation of L1 MissPenalty:twolf.

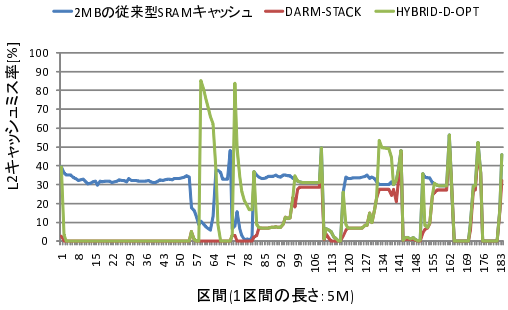

図 14 プログラム実行中のキャッシュミス率の変化：mcf
Fig. 14 Run-time variation of L2 MissRate:mcf.

松永・井上・石原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) Black, B., Annavaram, M., Brekelbau, N., DeVale, J., Jiang, L., Loh, G. H., McCauley, 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).
2) Loh, G. H.: 3D-Stacked Memory Architectures for Multi-core Processors, *SIGARCH Comput. Archit. News*, Vol. 36, No. 3, pp. 453–464 (2008).
3) 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).
4) Qureshi, M. K. and Patt, Y. N.: Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches, *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, Washington, DC, USA, IEEE Computer Society, pp. 423–432 (2006).

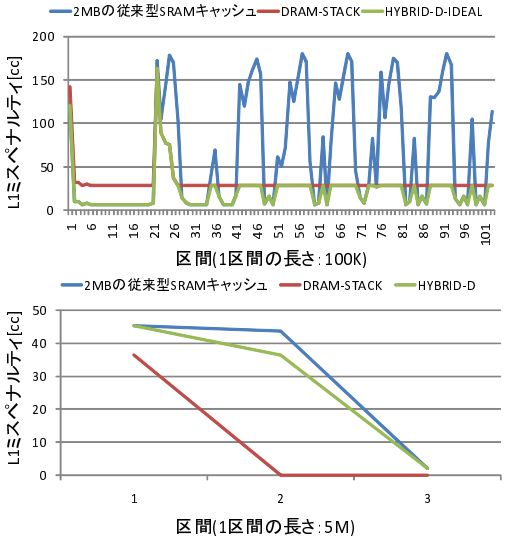

図 15 実行中の平均 L1 ミスペナルティの変化：Ocean
Fig. 15 Run-time variation of L1 MissPenalty:Ocean.

5) 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).
6) Loi, G. L., Agrawal, B., Srivastava, N., Lin, S.-C., Sherwood, T. and Banerjee, K.: A thermally-aware performance analysis of vertically integrated (3-D) processor-memory hierarchy, *DAC '06: Proceedings of the 43rd annual Design Automation Conference*, New York, NY, USA, ACM, pp. 991–996 (2006).
7) Thoziyoor, S., Muralimanohar, N., Ahn, J. H. and Jouppi, N. P.: CACTI5.1, Technical report, HP Lab (2008).
8) Henning, J. L.: SPEC CPU2000: Measuring CPU Performance in the New Millennium, *Computer*, Vol. 33, pp. 28–35 (2000).
9) 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).