1. アーキテクチャ基盤技術

4

メモリ混載プロセッサ

中村 宏
東京大学 先端科学技術研究センター 情報物理システム
nakamura@hal.rcast.u-tokyo.ac.jp

アーキテクチャはメモリとの戦い

現在のコンピュータシステムは，プログラムをデータとして記憶装置に格納させるノイマン型を採用している．世界最初のノイマン型コンピュータは1949年に開発されたEDSACであるが，それ以降の50年以上にわたるコンピュータアーキテクチャの歴史は，メモリとの戦いの歴史でもある．

プログラムもデータもすべてメモリから読み込むノイマン型アーキテクチャは，本質的に大容量かつ高速なメモリを必要とする．しかし，メモリの大容量性と高速性は相反する特性であり，決して両立するものではない．このメモリとの戦いにおける，アーキテクチャ上の唯一の戦術は，**図-1**に示すメモリの階層化である．すなわち，高速小容量のメモリと低速大容量のメモリを階層的に組み合わせることで高速かつ大容量なメモリを擬似的に実現している．たとえば仮想記憶は，アドレス空間すべてをまかなう容量の主記憶を構成する代わりに，大容量かつより低速な補助記憶（ディスク）との階層構成を採用することで，主記憶と同等の高速性をアドレス空間全体に対して擬似的に実現する技術であるし，キャッシュメモリは，主記憶と小容量かつより高速なキャッシュメモリとの階層構成により，主記憶を擬似的に高速化する技術である．

では，何が最適なメモリ階層構成なのであろうか？この問いに対する普遍の答えは残念ながら存在しない．その答えは少なくとも，コンピュータシステムを実現する半導体技術と，対象とするアプリケーションの性質に依存するからである．そこが，アーキテクチャはメモリとの永遠の戦いだ，といわれる所以である．半導体技術は年々進歩しており集積度は今日でも指数的に向上している．また，コンピュータシステムの応用分野も今日では大型計算機から携帯端末にまで広がり，対象とするアプリケーションの性質も多岐にわたる．したがって，メモリとの戦いは厳しさを増しているとさえいえるであろう．

半導体技術の進展のおかげで，数年後には1つのチップ内に何十億というトランジスタを搭載可能になると予測されている[1]．そういう状況下で，プロセッサチップ内に実装すべきメモリ構成に関してさまざまな議論がなされている．また，これまでは性能と容量という2つの観点からメモリ階層は議論されていたが，最近になり消

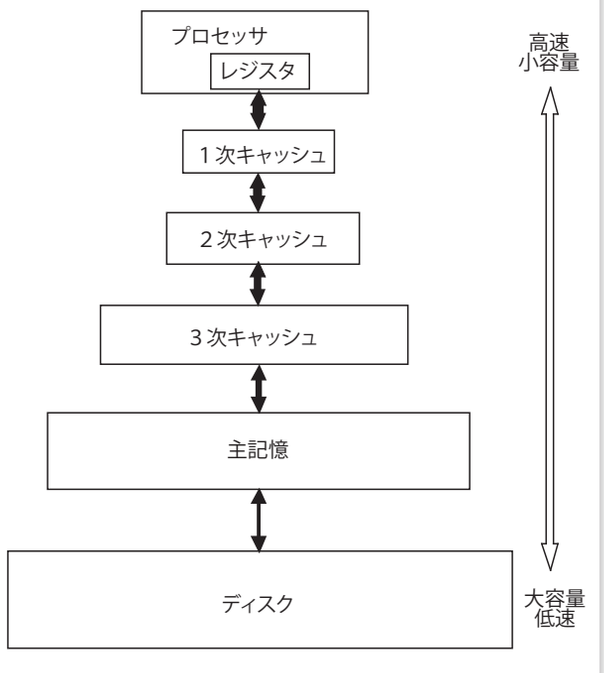

図-1 メモリ階層

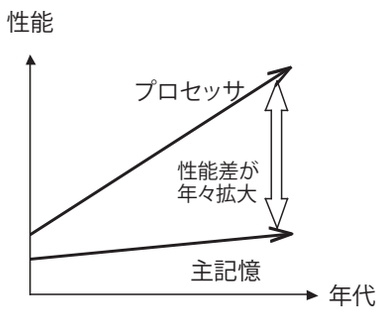

図-2　メモリウォール問題

費電力という第3の観点も出てきている．本稿では，これらの状況を踏まえ，現在どのような議論がなされているかをまず概観し，今後の展開についての予測を試みたいと思う．

メモリ混載アーキテクチャ

図-2 に示すように，プロセッサの速度向上に対しDRAMで構成される主記憶の速度向上は緩やかなため，その速度差が年々増大している．速度には，アクセス開始からアクセス終了までの遅延時間を表すレーテンシと，単位時間当たりに転送可能なデータ量を表すバンド幅の2つの意味があるが，いずれの意味でもプロセッサに対する主記憶の相対速度は年ごとに遅くなっている．この問題はメモリウォール(memory wall)とも呼ばれる．これはDRAM素子の問題もあるが，プロセッサと主記憶は異なるチップとして実装せざるを得ないことが主要因である．つまり，レーテンシに影響を与えるチップ間の信号伝播時間を改善することは難しく，また，チップのピン数の増加は物理的に難しくチップ間の周波数の改善も難しいからである．これに対し，同一チップ上のメモリはレーテンシ，バンド幅両面で優位であるため，プロセッサチップ上にメモリを実装することが有効となる．そこで，プロセッサチップ内に搭載すべきメモリの構成が，アーキテクチャ上の重要な検討事項となる．

■キャッシュメモリの発展

プロセッサに搭載するメモリとしてキャッシュメモリが広く採用されている．キャッシュとは，プロセッサと主記憶の間に置かれる，主記憶よりは小容量だが高速なメモリであり，参照頻度の高い主記憶内の情報の写しを保持することで低速な主記憶をあたかも高速に見せるものである．小容量のキャッシュが有効に機能する理由は，一般にプログラムには

- 時間的局所性：ある情報を参照すると，近い将来その情報を再び参照する．
- 空間的局所性：ある情報を参照するとその近隣の情報もあわせて参照する．

という2種類の局所性が存在するためである．時間的局所性があるため，小容量のキャッシュであっても参照される情報がキャッシュ内に存在する確率を高くすることが可能となる．また，主記憶は遅いため，主記憶とキャッシュの間で小さいデータを何度も転送するよりは複数のデータをまとめて転送したほうが転送に要する時間を短縮できる．この点も，空間的局所性を利用すればキャッシュと主記憶の間の転送をキャッシュのブロック単位でまとめて行うことが可能となる．この転送単位，および，リプレースメント（キャッシュに主記憶から新しいデータを持ってくるときにどのデータをキャッシュから破棄して置き換えるか）などのキャッシュの制御はすべてハードウェアとして実現されている．

半導体技術の進展に伴いプロセッサチップ上に今や数MBのキャッシュを搭載することが可能である．しかしキャッシュを大容量化すると，キャッシュのアクセス時間が長くなってしまうため，プロセッサの要求する性能（レーテンシとバンド幅）を実現できなくなる．そこで，プロセッサチップ上のキャッシュを階層化し，小容量の高速な1次キャッシュと比較的容量が大きく中程度の速度の2次キャッシュを搭載することが多い．さらに大きく低速な3次キャッシュを持つものもある．これらのキャッシュ階層化の階層数と各階層のキャッシュ容量といった最適なキャッシュ構成は，プロセッサチップ上でメモリとして利用可能なトランジスタ数やプロセッサからの要求性能に依存するが，プログラムの性質にも強く依存する．たとえば，局所性が高ければ小さいキャッシュで十分である．しかし，プログラムごとに局所性の性質は異なるため，最適な構成を1つ選択することは難しい．

また1つのプログラムの中でもデータによって局所性は異なる．たとえば1つのプログラムの処理中に再利用性の高いデータと，再利用性の低いデータが混在する場合，後者のデータをキャッシュに保持すると，キャッシュの容量が限られているため，せっかく保持している前者のデータを追い出してしまうことがあり，キャッシュの効率は著しく低下する．画像処理等におけるストリームデータは再利用性の低いデータの典型である．この問題を防ぐために，ストリームデータ用にキャッシュをバイパスして主記憶から直接レジスタへデータをロードする命令を持つプロセッサもある．

これは，ソフトウェア的手段によってキャッシュの画一的なハードウェア制御から逃れる手段であるが，その

	キャッシュメモリ	ソフトウェア可制御メモリ
リアルタイム性	×	○
ハードウェアコスト	大	小
ソフトウェアコスト	小	大
性能最適化の容易さ	アプリケーションによって異なる	

表-1　キャッシュメモリとソフトウェア可制御メモリの比較

ようなことをするのであれば，チップ上のメモリをソフトウェアから制御できるものにしてはどうか，という発想が当然生まれる．それが次節に述べるソフトウェア可制御メモリである．

■ソフトウェア可制御メモリ

本稿では命令が直接操作できる，プロセッサチップ上に搭載されたメモリをソフトウェア可制御メモリと呼ぶ．実はこの概念はキャッシュ以前から存在するもので，スクラッチパッドメモリとも呼ばれていた．しかし，その頃の導入理由は，集積度の不足から高速メモリはきわめて小容量しか実現できなかったため，それをソフトウェアから苦労して利用せざるを得なかったのである．その後，集積度の向上によりキャッシュメモリを導入できることとなり，ソフトウェア開発者を，スクラッチパッドメモリの利用最適化問題から解放した．ところが1990年代になると再びソフトウェア可制御メモリが，特に組み込みシステム用プロセッサで採用され始めた．その理由はこれらのシステムが重視する，リアルタイム性，ハードウェアコストに優れているためである．キャッシュメモリとソフトウェア可制御メモリの比較を**表-1**に示す．キャッシュの場合にはデータアクセスに要する時間がキャッシュの動的な状況に依存するため最悪実行時間を保証できず，リアルタイム性に劣る．ハードウェアコストの項は，同じ性能を達成するのに必要となるメモリ量を意味し，画一的な制御を行うキャッシュよりもデータアクセスの特徴に適した制御ができるために少ないメモリ量で同等の性能が原理的に達成できる．もっともそのためには，優れた制御をソフトウェア的に実現する必要があるため，ソフトウェア開発のコストが増える．性能に関しては，データアクセスパターンを静的に解析することが可能であればソフトウェア可制御メモリの方が有利であるが，それができない場合にはキャッシュメモリに任せるほうが有利である．また，ソフトウェア制御に必要となる命令追加による性能低下にも注意が必要である．組み込みシステムでは，対象とするアプリケーションがさほど広範囲ではないため，使いこなすソフトウェアの開発がそれほど難しくはないことも，採用しやすい一因である．

では，ソフトウェア可制御メモリの最適化はどのようにすればよいのであろうか？　1つには，よく使うデータを常駐させることが挙げられる．そのようなデータの総量がプロセッサ上のメモリ容量よりも小さい場合には，この最適化が最も効果的となる．対処アプリケーションが決まっている組み込みシステムにおいては，この戦略によりプロセッサ上のメモリ容量を決めることも可能となる．

一方で，最近では性能を追求するマイクロプロセッサでもソフトウェア可制御メモリが採用されつつある．この場合には，キャッシュにおいてハードウェアが制御する，主記憶との間のリプレースメント制御と，主記憶との間の転送単位の最適化が必要となる．いくつかの研究があるが，筆者のグループが研究しているSCIMA (Software Controlled Integrated Memory Architecture) を紹介する．SCIMAは，キャッシュの各ウェイをソフトウェア可制御メモリとして動的に再構成可能なアーキテクチャであり，主記憶とのデータ転送粒度も最適化の対象である．最適化においては，メモリアクセスパターンを**図-3**のように分類し，その特徴に応じて図中に示す最適化戦略を用いる．この戦略を簡単に説明する．まず，再利用性のないデータに関しては，ソフトウェア可制御メモリの一部をストリームバッファとして割り当てる．連続アクセスの場合は大粒度転送を行うことで下位メモリへのアクセス回数を削減できる．ストライドアクセスの場合もパッキングによりチップ上メモリの効率的な利用が期待できる．一方，再利用性のあるデータに関しては，ソフトウェア可制御メモリ上にできるだけ大きな領域を割り当て，再利用性を確実に活用する．ハードウェア制御のキャッシュでは避けられないライン競合が発生しないのでキャッシュより確実に再利用性を活用できる．ただし，アクセスが不規則な場合にはソフトウェア可制御メモリは利用できない．再利用性があり総データ量があらかじめ分かっている配列データの場合のみ，データ量分の領域をソフトウェア可制御メモリ上に割り

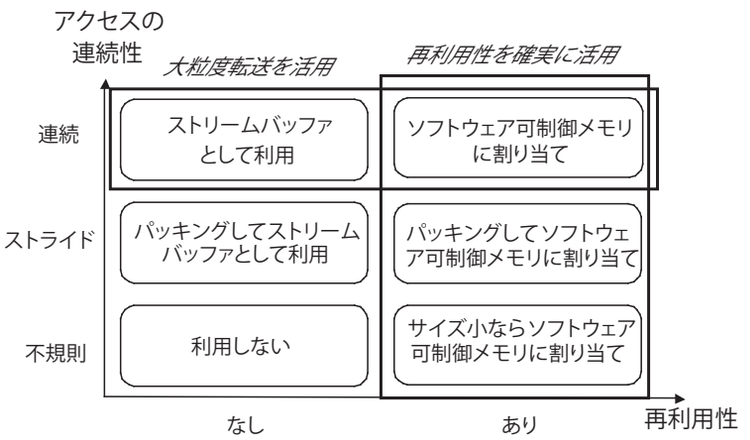

図-3　ソフトウェア可制御メモリの利用戦略

当てて再利用性を活用することができる．

この戦略に基づくコンパイラのプロトタイプ実装も行ったが，そこでは再利用性があるか否かだけはプログラム中でディレクティブにより与えることとし，データアクセスのパターンはコンパイラが自動的に解析を行った．図 -3 の戦略は単純なものであるが，この戦略を用いるだけでも，従来のキャッシュに比べ性能が向上するという評価結果も得ている．

ソフトウェア可制御メモリを採用する高性能商用プロセッサには以下のようなものがある．ルネサステクノロジ SH4 では 2 ウェイ（SH7751R では容量 32KB）キャッシュを持つが片方のウェイをアドレス指定可能なソフトウェア可制御メモリとして利用できる．Blue Gene/L のプロセッサチップは 4MB の 3 次キャッシュまで搭載するが，3 次キャッシュはあらかじめキャッシュとスクラッチパッドメモリに分割することができるようである [2]．また PlayStation3 に搭載予定の CELL Processor[3] は，1 つの汎用コアと 8 つの SPE と呼ばれる浮動小数点コアからなるが，各 SPE は 256KB のローカルストアと呼ばれるソフトウェア可制御メモリを搭載している．このように，容量が数 MB と大きくなると，スクラッチパッドメモリという名称はもはやふさわしくないであろう．

■ DRAM 混載プロセッサ

キャッシュかソフトウェア可制御メモリか，というのはアーキテクチャ上の論理的な構造の話であるが，実装上の話として SRAM を用いるか DRAM を用いるか，という選択肢もある．SRAM の方が集積度は低いが高速であり，トランジスタ構造が論理部と基本的に同じであるため製造上のハードルが低い．一方 DRAM の方が集積度は高いが製造上のハードルが高く，SRAM よりは低速である．製造上のハードルは，論理部のトランジスタは高速性を目指すためにソース・ドレインの抵抗を小さくするべきなのに対し，DRAM はデータの保持特性を良くするために，リーク電流を極力抑える必要があるため，そもそもデバイス素子に求められる性質が正反対であり製造プロセス上の親和性が低いことに起因する．

DRAM 混載プロセッサでは，RAM は主記憶あるいは前項のソフトウェア可制御メモリのように主記憶空間の一部を構成する．キャッシュは高速性が求められるため，DRAM を採用するのは難しい．1996 年にプロセッサチップに DRAM を混載させた M32R/D（2MB DRAM 搭載）が発表されたのを契機に DRAM 混載プロセッサも注目を集めており，前節で述べた Blue Gene/L のソフトウェア可制御メモリ（3 次キャッシュと併用）も DRAM で実現されている．

DRAM 混載プロセッサの商業的な価値は，DRAM の大容量性という利点が，製造上の困難さすなわちコスト増という欠点を補えるかどうかによるであろう．

今後のメモリ混載プロセッサ

本稿は「新世代マイクロプロセッサアーキテクチャ」特集なので，今後のメモリ混載プロセッサを展望したいと思う．

■性能バランス

半導体技術は今後も発展していくと予想されているが，プロセッサの処理能力と要求されるメモリバンド幅の比は基本的には半導体技術には依存せず，むしろアプリケ

ーションに依存する．そこで，プロセッサの処理能力が今後どのようなアーキテクチャでどのように伸びるのかを予測することが，今後のメモリ混載プロセッサを展望する手がかりになるであろう．「半導体チップに搭載されるトランジスタ数は 18 ～ 24 カ月で倍増する」というムーアの法則が今後も成立し近々 10 億個のトランジスタを搭載するプロセッサも実現すると予想されているので，まずは搭載可能なトランジスタ数という制約からは外れて自由に考えてみたい．

今後どのようなプロセッサアーキテクチャが有望か，という点はまさに本特集で議論されている点であるが，プロセッサの性能向上を阻む主要因は，配線遅延に起因する周波数向上の限界と命令レベル並列度の限界である．これらの問題を解決するアーキテクチャとして，1 つのプロセッサにたくさんの機構を持たせ複雑化させて性能向上を目指すのではなく，さほど複雑ではないプロセッサを複数実装するチップマルチプロセッサあるいはタイルプロセッサは，メモリアーキテクチャの観点からは有利である．なぜならば，これらのアーキテクチャでは各々のプロセッサにおける処理において，メモリ階層を構成する上で重要となる局所性が維持されるからである．これに対し，たとえばマルチスレッドアーキテクチャでは，複数のスレッドが同時並行処理されるため局所性が乱され不利となる．チップマルチプロセッサにおいては，各プロセッサが必要とするメモリバンド幅を実現する独立のメモリ階層を持つことで，チップ全体でのプロセッサ能力とメモリバンド幅をバランスさせることが可能となる．この時の検討事項としては，キャッシュを含むチップ内のメモリ階層のうち，どの階層までをプロセッサごとに独立とし，どこから共有するのかという点が挙げられる．これも対象とするアプリケーションの性質に依存するであろうが，たとえば，前述の Cell Processor では，専用プロセッサである 8 つの SPE は独立のローカルストアを持ち，汎用プロセッサは自分だけが利用するキャッシュメモリを搭載している．

性能バランスという観点では，利用可能な膨大なトランジスタを，どのようにプロセッサ処理能力とメモリバンド幅のバランスを保ちながら有効に活用するのか，が本質的に重要であり，プロセッサアーキテクチャに適したメモリアーキテクチャを考えるだけでは不十分で，メモリとの親和性が良いプロセッサアーキテクチャ，という視点も今後は重要になるであろう．

■低消費電力

低消費電力化もコンピュータシステムに求められる重要な項目である．バッテリー駆動で動作する組み込み用途のプロセッサだけでなく，高性能プロセッサにおいても発熱の問題から性能向上が抑えられるようになっており，低消費電力化はきわめて重要となっている．

消費電力は，トランジスタのスイッチング動作に伴うダイナミック電力と，スイッチング動作とは無関係に常時流れるリーク電流に起因するスタティック電力の 2 種類がある．メモリシステムが消費する電力に焦点を当てると，局所性を活用しできるだけアクセスをプロセッサに近い上位階層のメモリに閉じさせることが，ダイナミック電力の低減につながる．遠くまで信号を伝えること自体が電力を消費するからである．この意味で，ダイナミック電力の低減は従来のメモリ階層による高性能化とほぼ同じアプローチで達成することが可能である．

一方，スタティック電力は，トランジスタあたりのリーク電力と総トランジスタ数に比例する．ここで問題となるのは，半導体技術の微細化の進展に伴い，第 1 項のトランジスタあたりのリーク電力が指数的に増加する点である．このため，数年のうちにスタティック電力がダイナミック電力を上回ると予想されている[4]．また，プロセッサチップに搭載されるトランジスタ数の大半をメモリが占めるため，プロセッサに混載されるメモリのスタティック電力の低減は重要である．

キャッシュメモリにおけるリーク電流削減手法として，cache decay[5] という手法がある．これは，各ラインのアクセス状況を把握するハードウェア機構を用意し，ある一定時間以上ラインがアクセスされない場合には，その後もアクセスされないと判断しそのラインの電源電圧をオフにするものである．電源をオフにすればリーク電流は流れないためスタティック電力を削減できる．このとき，電源をオフにするので該当ラインの情報は失われるが，必要な情報は主記憶に取りに行けばよいので，たとえ該当ラインの情報が必要になっても，主記憶アクセスが増えるために性能は低下するものの正しい動作は保証される．

これらの手法が消費電力低減に有効であるということはすなわち，キャッシュの容量は消費電力の観点からは不要に大きすぎるということを意味する．従来のメモリ階層の最適化は性能を重視しており，集積度の向上の恩恵のもと可能な限りできるだけ大きなキャッシュを搭載し，階層化を施してきた．しかしながら，電力の観点からはより小さいキャッシュの方がむしろ効率が良いのである．また，従来のメモリ階層において各階層のキャッシュ容量を小さくすれば良いわけではないことも分かる．なぜならば，cache decay において電源をオフにするべきラインの選択基準が，従来のキャッシュがリプレースメントに用いる LRU（Least Recently Used）アルゴリズムとは異なるからである．

文献 5）および後続の関連研究によると，cache decay

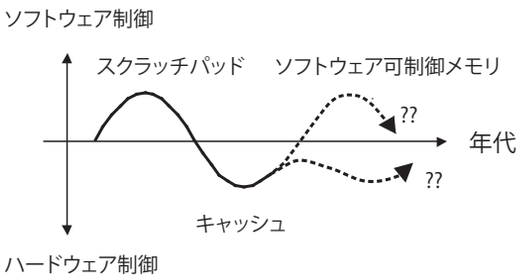

図-4　メモリアーキテクチャの波

において電源オフの判定根拠に用いるべき「一定時間」はアプリケーションの性質に大きく依存することが分かっている．これらのことは，ハードウェアによる画一的な制御がなされるキャッシュよりも，アプリケーションの特徴に応じて効率よく利用できるソフトウェア可制御メモリの方が適切な電源制御によるスタティック電力削減が実現できることを示唆している．そこで我々のグループは，ソフトウェア可制御メモリにおいてコンパイラがデータアクセスの特徴を把握することで不要と思われる個所の電源をオフにする手法を提案し，cache decay との比較評価を行った[6]．その結果，提案手法の方が cache decay より多くのスタティック電力を削減できることが分かったが，その理由は，電源をオフにしてよい領域を cache decay アルゴリズムよりも的確に把握できるからである．

スタティック電力を考えた場合には，できるだけ小さい容量のメモリだけを利用するべきであり，そのような場合にはハードウェアによる自動制御よりもソフトウェアによる手動制御の方が優れている，と現時点では言えるであろう．しかしながら，すべてのキャッシュがソフトウェア可制御メモリにとってかわられるわけではないだろう．昔主記憶がきわめて小容量だったとき，補助記憶と主記憶の間の情報の入れ替えを手動でやるオーバーレイという技術があり，その後それなりに主記憶の容量が増えてきて，仮想記憶という自動制御が出現した．スクラッチパッドメモリではなくキャッシュが主に用いられるようになったのも同様な経緯である．そして今再び，キャッシュからソフトウェア可制御メモリへ回帰する流れは出てきているが，これは 2 つの両極のコンセプトの間で，その時々の利用可能な実装技術と実装上の制約のもとで，選択すべきアーキテクチャが波のように行ったり来たりしているに過ぎないと思われるのである（**図 -4**）．

メモリとの戦いは続く

メモリ混載プロセッサに関して，現状と今後の展開について述べた．はじめに述べたように，コンピュータアーキテクチャはメモリとの戦いである．しかし，メモリに勝つための戦いではなく，プロセッサとメモリの間の良い妥協点を探ることで，優れたコンピュータシステムを実現するための戦いである．今後のメモリ混載プロセッサを展望するところで述べた，メモリシステムとの親和性の良いプロセッサアーキテクチャ，というのもまさにこの視点である．性能だけではなく低消費電力という新しい要件が加わることで，アーキテクチャが回帰する可能性にも言及したが，最近では信頼性という新しい要件も加わってきており，プロセッサに混載されるメモリに関して新しい展開が生じるかもしれない．プロセッサとメモリの終わることのない戦いを通してコンピュータシステムが今後さらに発展することは間違いないであろう．

参考文献

1）Burger, D. et al.: Billion Transistor Architectures: There and Back Again, IEEE Computer, Vol.37, No.3, pp.22-28 (2004).
2）Ohmacht, M. et al. : Blue Gene/L Compute Chip: Memory and Ethernet Subsystem, IBM Journal of R&D, Vol.49, No.2, 3, pp.255-264 (2005).
3）Flachs, B. et al. : A Streaming Processing Unit for a CELL Processor, Proc. of ISSCC 2005, pp.134-135 (2005).
4）Austin, T. et al. : Leakage Current: Moore's Law Meets Static Power, IEEE Computer, Vol. 36, No.12, pp.68-75 (2003).
5）Kaxiras, S., et al. : Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power, Proc. of 28th ISCA, pp.240-251 (2001).
6）藤田元信 他：ソフトウェア制御オンチップメモリにおけるスタティック消費電力削減手法，情報処理学会論文誌，Vol.45, No.SIG11(ACS7), pp.219-228 (2004).

（平成 17 年 8 月 11 日受付）