2007－ARC－175 (3)
2007／11／21

多機能メニーコアを実現するアーキテクチャ技術 Feature-Packingの構想

小林 良太郎† 吉瀬 謙二††

プロセッサ・アーキテクチャは，数百のコアを搭載するメニーコアの時代へと向かいつつある．その一方で，プロセッサが直面する課題はより多様に，より深刻になりつつある．今後，これらを解決する機能を数多く備えた多機能メニーコアの実現が重要になると考えられる．そこで我々は，コアが豊富に存在するというメニーコアの特徴に着目し，多機能メニーコアを実現するFeature-Packingと呼ぶアーキテクチャ技術の開発を行う．

Feature-Packing : Architectural-level Techniques for Multifunction Many-core

Ryotaro Kobayashi† and Kenji Kise††

Many core that facilitates some hundreds cores will be realiezd in the near future. On the other hand, the hurdles processors are faced is becomming more diverse and severe. A multifunction many core would be important to realize. We fucused on the abundant number of cores that is the benefit of many core and study architectural techniques called *Feature-Packing* for implementing multifunction many core.

1. はじめに

半導体技術の進歩は，チップ上に数個のコアを搭載するマルチコアの実現を可能とし[10)~12),17)]，現在では，マルチコアが重要なプロセッサ・アーキテクチャの1つになっている．消費電力や配線遅延の増加[4),13)]などにより大規模な単一コアの性能向上がますます困難になっていく一方で，集積度は着実に向上し続けており，今後，プロセッサ・アーキテクチャは，数百のコアを搭載するメニーコアの時代となる[5)]．メニーコアでは，利用可能なコアが豊富に存在し，ハードウェア資源に対する制約が大幅に緩和される．このため，アーキテクチャの可能性が一気に広がる．しかしその一方で，半導体技術の進歩によるハードウェア資源の爆発的な増加，情報機器の用途と求められる機能の拡大などにより，プロセッサが直面する課題はより多様に，そして，より深刻になりつつある．

直面する課題において特に重要となる要求は**高速化，省電力化，ディペンダビリティ向上，製造コスト低減**の4つである．高速化は，プロセッサ・アーキテクチャにおいてもっとも基本的で重要な課題である．これまで高速化を実現するための研究が数多く行われ，プロセッサの性能は飛躍的な向上を遂げたが，情報機器は急速に進歩し続けており，今後も，さらなる高速化が求められる．省電力化は，モバイル機器のバッテリー駆動時間の延長，冷却コストの削減などのため，今や必須の課題となっている．ディペンダビリティとは，ユーザーから見た，情報システムへの安心感，信頼感，便利さを統合した概念[8)]である．ディペンダビリティの向上は，情報システムが人々の社会生活を支える基盤の1つとなっている現代において欠かせない課題である．プロセッサ設計や検証，歩留まりを考慮した製造コストの低減も忘れてはならない．

このように，マイクロプロセッサは，解決すべき課題を数多く抱えている．そのため，メニーコアの時代を迎えるにあたり，これらを解決する数多くの機能を備えたメニーコア・プロセッサの実現が解決策となる．そこで，我々は，**コアが豊富に存在するというメニーコアの特徴に着目して，多機能なメニーコア・プロセッサを実現するFeature-Packingと呼ぶアーキテクチャ技術の枠組みを提案**し，直面する課題の解

† 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University
†† 東京工業大学大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

決を目指す.

以下, 2 章では Feature-Packing を定義する. その後, 3 章で高速化, 4 章で省電力化, 5 章でディペンダビリティ向上について議論する. 最後に, 6 章で本稿をまとめ, 今後の課題を明確にする.

2. Feature-Packing

Feature-Packing の目的は, アーキテクチャ技術を用いて, 多機能なメニーコア・プロセッサを実現することにある. しかし, やみくもに多機能化すれば良いというわけではない.

まず, メニーコアの最大の利点は, 消費電力と配線遅延を抑制しつつ, プロセッサ全体のスループットを向上させる点にあることを考慮する必要がある. 個々の機能を提供するために, それに特化した回路を各コアに設けてしまうと, コアの規模が拡大し, メニーコアの利点が失われてしまう.

つぎに, アプリケーションの持つ性質が多様であることを考慮する, あるいは, プロセッサの性能, 消費電力, ディペンダビリティの間にはトレードオフが存在することを考慮する必要がある. あらかじめプロセッサの規模や機能を固定してしまうと, 付加した機能を活用できないといった問題や, 効率が悪化するといった問題が生じる.

これらを解決するため, シンプルな構造を維持しつつ多数のコアを柔軟に制御する **Feature-Packing アーキテクチャ**を新たに導入する. そして, このアーキテクチャをベースとして, 多機能なメニーコア・プロセッサを実現する.

我々は, ベースとなる Feature-Packing アーキテクチャを以下のように定める.

- **全体構成.**
  多数の均一な **Feature-Packing コア**と呼ぶコアをアレイ状に配置し, メッシュ等のシンプルな接続網で接続することでチップを構成する.
- **Feature-Packing コアの簡潔性.**
  Feature-Packing コアは, 大規模なキャッシュ, 複雑な分岐予測機構, 大規模な投機処理機構などを排除し, シンプルな構成を維持する. これにより, メニーコアの利点 (消費電力と配線遅延の抑制) をさらに高めることができ, さらに, タイル・アーキテクチャ[3] と同様, 設計と検証のコストを抑えることができる.
- **Feature-Packing コアの多様性.**
  動作させるプログラムを変更することにより Feature-Packing コアの役割を動的に変更できる仕組みを備える. これにより, アプリケーションの実行だけでなく, それを支援する役割 (例えばデータ供給の支援) も果たすことができるようになる.
- **Feature-Packing コアの融合性.**
  複数の Feature-Packing コアが協調動作 (融合) して, より規模の大きい 1 つの**ブロック**として動作する仕組みを備える. この仕組みは Core Fusion[7] を拡張したものとして捉えることができる.
- **Feature-Packing コアの独立性.**
  Feature-Packing コアは個別に (あるいは幾つかの Feature-Packing コアが融合したブロック毎に) 動作周波数と電源電圧を動的に変更できる仕組みを備える.
- **Feature-Packing コアの連続性.**
  従来の RISC 形式の命令セットがもたらす利点の大部分を維持しながら, Feature-Packing コアの間の通信を低いレイテンシでおこなう仕組みを取り入れる. これにより, 従来の最適化コンパイラ技術の大部分を流用できる.
- **モニタリングと制御.**
  Feature-Packing コアは, 自身の状況 (処理のスループットやハザードの発生頻度など) を計測し, これを報告する仕組みを備える. さらに, 他のコアから報告された状況などに基づいて, 他のコアに融合の指令や動作周波数変更の指令を与える (制御する) 仕組みも備える.

Feature-Packing アーキテクチャでは, 全体構成, および, Feature-Packing コアの簡潔性により, シンプルな構造を維持する. これにより, メニーコアの利点 (消費電力と配線遅延の抑制) をさらに高めることができ, さらに, タイル・アーキテクチャ[3] と同様, 設計と検証のコストを抑えることができる. また, Feature-Packing コアの多様性, 融合性, 独立性などを利用し, 多数のコアを柔軟に制御する. これによりアプリケーションの性質や, 様々なトレードオフを考慮して, プロセッサの規模や機能を動的に決定することができる.

図 1 に, Feature-Packing の概念図を示す. 図 1(a) は, チップ上に均一なコアが配置された通常のメニーコアを示す. 一方, 図 1(b) に, Feature-Packing プロセッサの例を示す.

図において, 番号が付加されていないコアは, 単一コアと同様にしてアプリケーションを実行する**計算コア**である.

1 番と 2 番は, 計算コアを高速化するために, Feature-Packing コアの融合性を利用して規模を増加

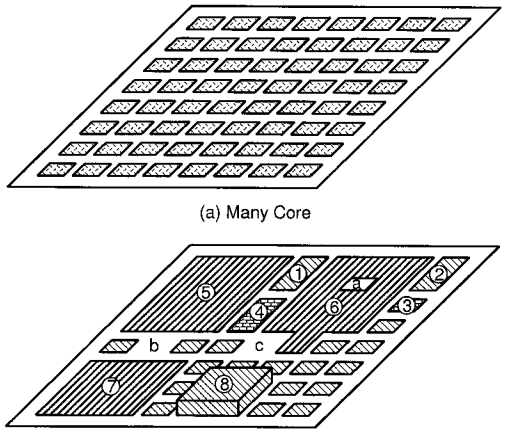

図 1 Feature-Packing many core processor

させた**計算ブロック**を示す.

3 番は，高精度な分岐予測によって他の計算コア/ブロックの命令供給を支援するために，Feature-Packing コアの多様性を利用して役割を変更した**命令供給支援コア**を示す．4 番は，Feature-Packing コアの多様性と融合性を利用し，3 番よりも高い予測精度を達成できるようにした**命令供給支援ブロック**を示す.

5 番〜7 番は，他の計算コア/ブロックに高性能なキャッシュを提供するために，Feature-Packing コアの多様性と融合性を利用して，役割の変更と規模の増加を行った**データ供給支援ブロック**を示す.

8 番は，Feature-Packing コアの融合性と独立性を利用して，規模と動作周波数を増加させた計算ブロックを示す.

図中の a〜c は，経年変化や製造不良といった致命的な永久故障が発生したコアや，素子ばらつきによってタイミング違反を頻繁に起こすコアを示す．こうしたコアが存在する場合，Feature-Packing コアの独立性を利用して，正常なコアだけを使う．これにより，**歩留まりを大幅に改善**する.

以下では，Feature-Packing プロセッサの特性を利用して，マイクロプロセッサの各課題を解決する方法を述べる.

3. 高 速 化

メニーコアは，多数の小規模なコアを搭載することによって，消費電力と配線遅延を抑制しつつ，プロセッサ全体のスループットを向上させることに成功している．しかし，このアプローチによって，各コアのスループットは低下するため，個々のアプリケーションの処理時間は増加してしまう.

この問題に対し，まず，シングルスレッドのアプリケーションを高速化する方策を考える．これまでに，複数のコアを 1 つのより高速なコアとして動作させる研究がいくつも行われてきた[1),2),7)]．例えば Core Fusion[7)] では，発行幅の狭い複数のコアが融合し，より発行幅の広い 1 つのコアとして，シングルスレッドのプログラムを実行する．融合するコアの数は動的に決定することができ，任意の発行幅のコアを任意の数用意することができる.

我々は，このアプローチを発展させ，複数のコアを用いて，柔軟な融合をおこなうことをこころみる．これは，**Feature-Packing コアの多様性と融合性**をうまく利用することで実現する.

基本的には，まず，コアの多様性を利用して，計算コアと，それを支援する，命令供給支援コア，データ供給支援コアなどを用意する．命令供給支援コアでは**ソフトウェア分岐予測**[9)] の導入を，データ供給支援コアでは**ソフトウェアキャッシュ**[15)] の導入を検討している．機能をソフトウェアで実現するため，アプリケーションに応じて，アルゴリズム (分岐予測アルゴリズム，キャッシュの連想度や容量などを決定するアルゴリズムなど) を容易に変更できる.

つぎに，コア融合性を利用して，幾つかの Feature-Packing コアを融合することによって，特定の機能をもつ高性能な 1 つのモジュールを実現する．こうすることにより，メニーコア上にはシンプルなハードウェアしか搭載しないにも関わらず，高度な機構を利用することができる.

さらに，**モニタリングと制御**を利用することで，各 Feature-Packing コアは必要に応じて，アプリケーションに最適な規模やアルゴリズムを選択することができる.

これらの高速化の仕組みは，シングルスレッドのアプリケーションの高速化に貢献するだけではなく，**マルチスレッド化された並列アプリケーションの高速化**にも大いに役立つ．全てのスレッドが均一な性質を持つとは限らないため，並列に実行するスレッドをモニタし，ボトルネックになる幾つかのスレッドの性能が向上するように Feature-Packing コアの規模やアルゴリズムを制御すれば，全体として並列アプリケーションの効率化を達成できる．すなわち，Heterogeneous Multi-Core Architectures[6)] と同様の利点をもたらす．これらに加えて，機能と規模を動的に変更する能力は，逐次プログラムから並列プログラムまで，また，**インクリメンタルな並列プログラムの開発過程**のそれぞれ

の状況に応じて適切な構成を提供できる利点を持つ．

4. 省電力化

Feature-Packing アーキテクチャは，各コアが非常にシンプルな構成を維持することができるため，メニーコアが持つ最大の利点の1つである省電力効果を最大限引き出すことができる．しかし，Feature-Packing アーキテクチャは，上記だけでなく，多数の均一なコアを柔軟に制御できるという特徴も持っている．この特徴を利用すれば，さまざまなアプローチを用いて消費電力を削減することができる．

消費電力を削減する有効なアプローチの1つとして，動作周波数と電源電圧を動的に制御する DVFS (Dynamic Voltage and Frequency Scaling) と呼ばれる手法が用いられている．Feature-Packing アーキテクチャは，**Feature-Packing コアの独立性**を利用することによって，DVFS を計算コア/ブロックに適用する．DVFS を適用した計算コア/ブロックは，与えられた負荷に応じて，動作周波数と電源電圧を低下させることで，消費電力を削減することができる．

このアプローチを発展させ，さらに効率良く消費電力を削減することをこころみる．これは，**Feature-Packing コアの多様性，融合性，独立性**，および，**モニタリングと制御**をうまく利用することで実現する．

まず，コアの多様性と融合性を用いて，計算ブロックとそれを支援するブロック (データ供給支援ブロック，命令供給支援ブロックなど) を用意し，これらを融合することで，高性能な1つのモジュールを実現する．また，コアの独立性を用いて，各ブロックに DVFS を適用し，ブロック毎に動作周波数と電源電圧を変更できるようにする．そして，モジュールの実行中に，各ブロックは自身の状況をモニタし，それぞれの役割に応じて，動作周波数と電源電圧を低下させることによって，消費電力を削減することができる．モジュール内にクロックが異なるブロックが複数存在していることから，GALS プロセッサのアプローチ[16] を用いて消費電力を削減しているとみなすこともできる．

Feature-Packing アーキテクチャの持つ柔軟さを利用すれば，コアの独立性を用いなくても，省電力化を達成できる．コアの多様性と融合性を用いて実現したモジュールにおいて，与えられた負荷に応じて，各ブロックの規模を変更することで，消費電力を削減することができる．

5. ディペンダビリティ向上

多数の小規模なコアを柔軟に制御する Feature-Packing アーキテクチャの構造は，チップのディペンダビリティの面にも様々な恩恵をもたらす．

経年劣化や製造不良といった致命的な永続故障に対し，メニーコア構成は柔軟な代替機能を有している．これは，8つの SPE のうち1つまでの不良を許容して歩留まりを向上させる Cell/B.E. や，発生した故障に応じて動的にチップ規模を縮退させるアーキテクチャ研究などでも示されている．

また，放射線や熱など動作環境によって引き起こされる過渡故障対策の基本は冗長実行であり，メニーコア構成がもたらすスループット向上は，実行信頼性への大きな寄与となる．特に，ランダムばらつきや熱によって引き起こされるタイミングエラー検出には，マージンの異なる冗長化が必須である．Feature-Packing アーキテクチャでは，各コア毎に動作周波数と電源電圧を変えることができるため，実行用の高周波数コアと保証用の低周波数コアといった構成を簡単に作り出すことができる．また，柔軟な制御を利用して，温度の上がりすぎたコアから他コアへのマイグレーションを行うなど，様々な対策が可能である．

ディペンダビリティ確保は電力や計算力とのトレードオフである．制御がソフト的に行われる Feature-Packing アーキテクチャでは，実行プロセスのクリティカリティに応じた，柔軟なリソース投入が可能となる．

このように，メニーコア構成と柔軟な制御はディペンダビリティ向上に大きく寄与するが，その一方で，コア間ネットワークやコアマネージメント機構の信頼性が重要である．これらの課題には，大型計算機システムにおけるノウハウが有効と考えられる．

6. まとめ

プロセッサ・アーキテクチャは，数百のコアを搭載するメニーコアの時代へと向かいつつある一方で，プロセッサへの要求はより多様に，より深刻になりつつある．

そこで本稿では，コアが豊富に存在するというメニーコアの特徴に着目し，多機能なメニーコア・プロセッサを実現する Feature-Packing の枠組みの提案と議論を行った．

しかし，Feature-Packing の開発のために必要な課題は山積している．主なものとして以下の課題を挙げることができる．

(1) FP コアの詳細な構成の決定．
(2) さまざまな機能を提供するために FP コアが果たすべき役割 (データ供給支援など) の検討．

(3) FP コアを効率よく実現する命令セットと統一的なマイクロアーキテクチャの決定.
(4) FP コアを効率的に融合, 分離する仕組みの実現.
(5) モニタすべきコアの状況, および, コア全体の制御方法の検討.
(6) ソフトウェアシミュレータによる詳細評価.
(7) チップ試作による消費電力や実現可能性の検証.

シミュレータ等に関する準備[14] も着実に進んでおり, 間もなく本格的にこれらの課題に取り組む予定である.

謝　辞

適切なアドバイスをしていただいた科学技術振興機構 (JST) の入江英嗣氏に感謝いたします.

参 考 文 献

[1] K. Sankaralingam, et al., "Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture," In *Proc. ISCA-30*, 2003.
[2] K. Mai, et al., "Smart Memories: a modular reconfigurable architecture," In *Proc. ISCA-27*, 2000.
[3] M. B. Taylor, et al., "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams," In *Proc. ISCA-31*, 2004.
[4] K. Bernstein, "Caution Flag Out: Microarchitecture's Race for Power Performance," Keynote Presentation to MICRO-36, 2003.
[5] S. Borkar, et al., "Platform 2015: Intel Processor and Platform Evolution for the Next Decade," Technology@Intel Magazine, 2005.
[6] K. I. Farkas, et al., "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction Rakesh Kumar," In *Proc. MICRO-36*, 2003.
[7] E. Ipek, et al., "Core Fusion: Accommodating Software Diversity in Chip Multiprocessors," In *Proc. ISCA-34*, 2007.
[8] 入江 英嗣ほか, "超ディペンダブル・プロセッサアーキテクチャの構想," 電子情報通信学会技術研究報告 CPSY2006, Vol.106, No.3, 2006.
[9] 吉瀬 謙二ほか, "Cell プロセッサの分岐ペナルティを軽減するソフトウェア分岐予測の可能性検討," 情報処理学会研究報告 2007-ARC-172, 2007.
[10] K. Krewell, "UltraSPARC IV Mirrors Predecessor," Microprocessor Report 2003-11-10, 2003.
[11] S. Naffziger, et al., "The Implementation of a 2-core Multi-Threaded Itanium Family Processor," In *Proc. ISSCC*, 2005.
[12] D. Pham, et al., "The Design and Implementation of a First-Generation CELL Processor," In *Proc. ISSCC*, 2005.
[13] F. Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies," Keynote Presentation to MICRO-32, 1999.
[14] 佐藤 真平ほか, "実用的かつコードのシンプルさを追求した Cell BE の機能レベルシミュレータ SimCell の設計と実装," 第 19 回コンピュータシステム・シンポジウム ComSys, 2007.
[15] 佐藤 芳紀ほか, "Cell Broadband Engine への SPE ソフトウェアデータキャッシュの実装," 情報処理学会研究報告 2007-HPC-110, 2007.
[16] G. Semeraro, et al., "Hiding Synchronization Delays in a GALS Processor Microarchitecture," In *Proc. ASYNC-10*, 2004.
[17] J. M. Tendler, et al., "POWER4 System Microarchitecture," IBM Journal of Research and Development, 46(1), 2002.