2008－ARC－180 (12)
2008／10／21

メニーコアプロセッサのディペンダビリティ向上と高性能化を目指すSmartCoreシステム

吉瀬謙二† 植原昂† 佐藤真平†

SmartCore システム (Smart many-core system with redundant cores and multifunction routers) とは，多数のコアの利用と高機能ルータによってメニーコアプロセッサの性能向上を目指す仕組みである．冗長実行の構成を動的に変化させることにより (1) ディペンダビリティの向上，(2) バンド幅の向上，(3) レイテンシの削減，という重要項目の改善を目指す．本稿では，ディペンダビリティおよびバンド幅の向上に絞って，その方式を議論する．

SmartCore systems to realize high-performance and dependable many-core processors

Kenji Kise ,† Koh Uehara † and Shimpei Sato †

The SmartCore system is the mechanism of pursuing high performance and dependable many-core processors with redundant cores and multifunction routers. By changing the organization of redundant execution dynamically, SmartCore system makes it possible to realize the dependable, high-bandwidth, and low latency many-core processors. In this paper, we discuss the method to achieve dependable and high-bandwidth many-core processors.

1. はじめに

プロセッサの性能向上と消費電力の削減を目的に，広い分野にわたり，複数個のコアを搭載するチップマルチプロセッサが主流となりつつある．今後は，半導体の集積度向上により，さらに多くのコアを集積するメニーコアプロセッサへと向かうと考えられる．この様なトレンドを受け，我々は，1 チップに搭載可能なコアが**数十から数千コアというオーダまで増え続ける**ことを想定し，そのような豊富なコアを活用するプロセッサの開発を目指して研究を進めている．

図 1 に，我々が想定しているメニーコアアーキテクチャのモデルを示す．2 次元のメッシュ状に配置された 64 個のユニットをノードと呼び，それ以外をモジュールと呼ぶことにする．ここでは 64 ノードの構成を示しているが，数千ノードのメニーコアプロセッサのためのアーキテクチャ開発を目指している．ノード間のデータ転送にはチップ内のネットワークを使用する．ノード及びモジュールは X 座標と Y 座標との組合せで一意に指定される識別子 (ID) を持つ．この ID を用いて，特定のノード及びモジュール間での DMA 転送を実現する．本稿では，(X,Y) という ID を持つノードのことを Node(X,Y) と表記する．左上に位置するモジュール (0,0) は，従来の汎用プロセッサと同等の機能を有するコアである．図の上部に配置されるモジュールはオフチップのメモリコントローラを含むメインメモリである．

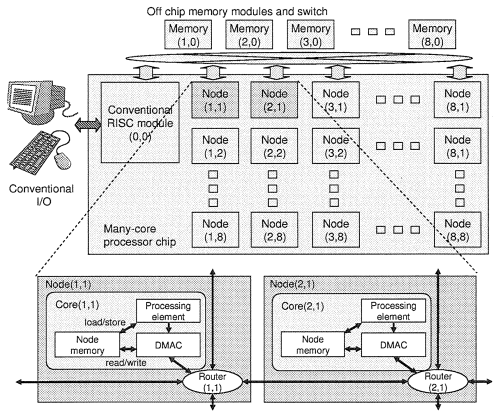

図 1 想定するメニーコアアーキテクチャのモデル．個々のノード (Node) はプロセッサコア (Core) とルータ (Router) を持つ．

ノードの内部構成を図 1 の下部に示す．ノードはルータとコアによって構成される．コアは，アプリケーショ

† 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

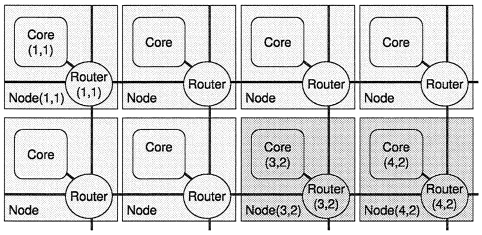

図 2 SmartCore システムによるディペンダビリティの向上．Core(3,2) のディペンダビリティを向上させるために，Core(4,2) を利用して多重実行をおこなう．

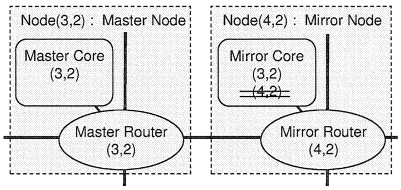

図 3 マスターノード Node(3,2) とミラーノード Node(4,2) との多重実行によるディペンダビリティの向上．

ンプログラムなどを実行するプロセッシングエレメント (PE)，ノードメモリ (ノードが持つ小規模のメモリ)，DMAC(Direct Memory Access Controller) で構成される．PE はロード/ストア命令により自ノードのノードメモリにアクセスする．また，DMAC に対して DMA 転送を発行する．DMA 転送は DMA_PUT と DMA_GET に大別される．前者は，自ノードのノードメモリのデータを他ノードのノードメモリに転送する．後者は，他ノードのノードメモリのデータを自ノードのノードメモリに転送する．

本稿では，このようなメニーコアアーキテクチャのモデルを前提として，メニーコアプロセッサのディペンダビリティ向上と高性能化を目指す **SmartCore システム**の構想を述べる．

2. SmartCore システムの構想

SmartCore システム (Smart many-core system with redundant cores and multifunction routers) とは，多数コアの利用と高機能ルータによってメニーコアプロセッサの性能向上を目指す仕組みである．冗長実行の構成を動的に変化させることにより，(1) ディペンダビリティの向上，(2) バンド幅の向上，(3) レイテンシの削減，という 3 つの重要項目の改善を目指す．本章では，(1) と (2) のディペンダビリティおよびバンド幅の向上について議論する．

2.1 ディペンダビリティの向上

SmartCore システムの機能の 1 つとして，高機能ルータを中心にして送受信パケットのレベルで冗長実行を実現するディペンダビリティ支援高機能ルータアーキテクチャを提案する．

図 2 を用いて，ディペンダビリティ支援アーキテクチャの概要について述べる．ここでは，Node(3,2) のディペンダビリティ向上のため，同等の機能を持つ Node(4,2) を冗長実行のために割り当てるとする．また，Core(4,2) をソースとする通信が特定の高機能ルータ，ここでは Router(3,2)，を経由するように経路を固定する．Core(3,2) が受け取るべきパケットは，高機能ルータ Router(3,2) が複製し，パケットの系列を同じ順番で Core(3,2) および Core(4,2) の 2 つのコアに送信する．2 つのコアが送信したパケットは，経路が固定されているため，必ず Router(3,2) を経由する．このルータにおいて，それらのパケットの内容が一致することを検査してエラーが発生していないことを確かめた後に，1 つのパケットとして適切な宛先へと送信する．2 つのコアが送信したパケットが一致していない場合にはエラーが検出され，適切な処置が施される．このように，本方式では，高機能ルータを中心にしてコアの単位で冗長実行をおこない，かつ送受信パケットのレベルでエラーを検出する．このため，コアの実装の詳細に依存することなく，様々なハードウェア構成に対してディペンダビリティを向上させることができるという利点がある．

図 3 を用いて，ディペンダビリティ支援アーキテクチャの詳細を説明する．ここでも，Node(3,2) のディペンダビリティ向上のために，Node(4,2) を利用する例を考える．本稿では，初期検討として，2 つのコアが同じタイミングで同じパケットを受信し，かつ，2 つのコアが同じタイミングで同じパケットを送信する，という厳しい条件を満たすものとする．

ディペンダビリティを要求するコアをマスターコア，冗長実行する Core(4,2) をミラーコアと定義する．まず，マスターコアとミラーコアでは同じタイミングで同じプログラムの実行を開始する．ただし，コア識別子が異なると，アプリケーションの挙動も変わってくるため，ミラーコアがマスターコアの識別子を利用するように工夫する必要がある．例えば，コア識別子はコアが持つ DMAC のレジスタに保存されているとすると，Router(4,2) が Node(4,2) のコア識別子を更新することで，マスターコアと同じ識別子を持つ Core(3,2) の様に見せかける．

2 つのコアが同じタイミングでパケットを受信する仕組みについて述べる．Router(3,2) が Core(3,2) をデスティネーションとするパケットを受け取ると，ルータがパケットを複製し，同じパケットをミラーコアにも送信する．ただし，Router(3,2) が複製する 2 つのパケットを同じタイミングで送信すると，2 つのコアにおいてパケットを受信するタイミングが一致しなくなる．このことを回避するため，Router(3,2) は，ミ

ラーコアが受信するまでのサイクル数だけ遅らせて☆，マスターコアにパケットを送信することで，2 つのコアが同じタイミングでパケットを受信することを保証する．

この様に，マスターコアとミラーコアが同じ状態からアプリケーションの実行を開始し，かつ，同じパケットを同じタイミングで受信しているため，2 つのコアの状態は一致する．これにより，2 つのコアが同じタイミングで同じパケットが送信されることになる．

次に，エラー検出の仕組みを述べる．ミラーコアのパケットの送信先にかかわらず，ミラールータはマスタールータ Router(3,2) にパケットを送信する．これは，ミラールータ Router(4,2) におけるミラーコアからのパケットの宛先を置換するハードウェアにより実現する．マスタールータでは，マスターコアとミラーコアからのパケットが到着するのを待ち，これらのパケットを比較することでエラーを検出する．2 つのパケットが一致する場合には，これらを 1 つのパケットとして適切な宛先へと送信する．

本方式には，コアの実装に依存することなく様々なハードウェア構成に対して，必要に応じて，ディペンダビリティを向上できるという利点がある．一方で，幾つかの欠点を持つ．まずは，ディペンダビリティを向上させるために 2 倍以上のハードウェアを要することである．この欠点については有効な解決策は見つかっていない．次に，コアにおいて間違った挙動が発生した時刻とエラーを検出する時刻との間にある程度の遅延が生じることである．例えば，まれにしかパケットを送信しないコアでは，エラー検出がとても遅れる可能性がある．このことは，エラーを検出した場合の回復を困難にする．エラーが検出された場合には，オペレーティングシステムにシグナルを通知し，アプリケーションを再実行するといった方式が現実的である．

本方式を実現するためには，ルータの高機能化が必要となる．パケットの複製，パケットの同一性の検出などを追加した高機能ルータの実現可能性およびハードウェア規模の見積もりは今後の課題である．

2.2 バンド幅の向上による高性能化

SmartCore の 2 つ目の機能として，バンド幅を向上させる方式を提案する．並列アプリケーションでは，多数のコアの DMA が特定のコアに集中することで性能が低下するケースが頻繁に発生する．このような場合に，集中しているコアへのリード/ライトのバンド幅を向上させることで，プロセッサの高性能化を目指す．

図 4 を用いて，バンド幅向上の方式について述べる．ここでは，Core(1,1)，Core(2,1)，Core(3,1)，Core(4,1) の 4 つのコアが Core(2,2) に頻繁に DMA をおこない，そこがボトルネックとなっている場合を考える．この時，Core(2,2) をマスターコア，Core(3,2) をミラーコアとして多重実行をおこない，マスターコアへの要求を分散させることで，バンド幅の向上を狙う．

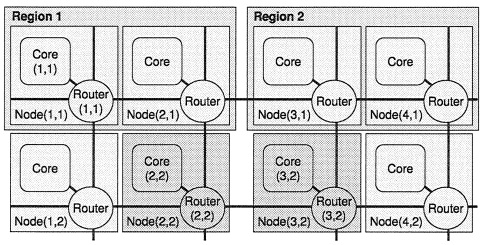

図 4 SmartCore システムによるバンド幅の向上．Core(2,2) がマスターコア，Core(3,2) がミラーコアとなり，それぞれが Region 1 と Region 2 の要求に応えることでバンド幅の向上を目指す．

方式の詳細を述べる．前節で述べたように，マスターコアとミラーコアでは同じタイミングで同じプログラムの実行を開始する．

マスターコアあるいはミラーコアからデータを読み出す場合を考える．これは，例えば，Core(1,1) から Core(2,2) への DMA_GET に相当する．DMA が分散されるように，リージョンと呼ばれる単位を導入する．図 4 では，Node(1,1) と Node(2,1) を Region 1，Node(3,1) と Node(4,1) を Region 2 と定義している．Region 1 の DMA_GET はマスターコア Core(2,2) に送信され，必要としているデータを得る．一方，Region 2 の DMA_GET はミラーコア Core(3,2) に送信され，Core(3,2) から必要としているデータを得る．このように，多重実行しているそれぞれのコアが並列に DMA を処理することでバンド幅を向上させる．これを実現するためには，ミラーコアに要求すべき Region 2 のルータを高機能化し，Core(2,2) への DMA_GET のパケットを Core(3,2) に送信するように宛先を書き換えればよい．

次に，マスターコアあるいはミラーコアがパケットを送信する場合を考える．基本的にマスターコアとミラーコアは同じ状態を保ちながら同じプログラムを実行しているので，同様のパケットを送信することになる．一方，これらのパケットを 2 つのコアがそのまま送信してしまうと，受信側のコアは同一のパケットを 2 回受信することになり，アプリケーションが正しく実行されなくなる．この問題を回避するためには，マスターコアあるいはミラーコアから送信されたパケットの宛先が担当しているリージョンではない場合に，そのパケットを破棄すればよい．これは，マスタールータおよびミラールータに，同一ノードのコアが担当するリージョンを判定して，異なるリージョンへのパケッ

☆ このためには，マスタールータはミラーコアが受信するまでのサイクル数を知る必要がある．例えば，メッシュ接続と次元順ルーティングの特徴をうまく利用して，2 つのルータ間を独占的に利用できるように構成すればよい．

トを破棄する仕組みを実装することで実現できる．

次に，マスターコアあるいはミラーコアにデータを格納する場合を考える．これは，例えば，Core(1,1)からマスターコア Core(2,2) への DMA_PUT に相当する．先の読み出しの場合と同様に，リージョンに応じて，宛先をミラーコアに書き換える．ただし，マスターコアあるいはミラーコアへの書き込みが他方にも同様に行われる必要があることに注意しなければならない．これを実現するために，マスタールータあるいはミラールータは，パケットが DMA_PUT であることを識別する．もし，パケットが DMA_PUT であれば，パケットを複製して，他方のコアにもパケットを送信する．

前節で議論したディペンダビリティ向上の方式では，2 つのコアが同じタイミングで同じパケットを受信し，かつ，2 つのコアが同じタイミングで同じパケットを送信するという厳しい条件を課していた．一方，本節の方式では，アプリケーションプログラムが割り当てられるコアの場所 (識別子) が変化して，DMA のレイテンシが変化しても正しく動作するように記述されていなければならないという条件から，2 つのコアの送受信のタイミングを一致させていない点に注意する必要がある．

3. 関連研究

提案する SmartCore システムは，複数のハードウェアを用いて同一の処理をおこなう空間冗長化により，ディペンダビリティ向上と高性能化を目指す方式である．高性能プロセッサのディペンダビリティの向上を目指す空間冗長化による方式として DIVA(Dynamic Implementation Verification Architecture)[1] が提案されている．これは，メインパイプラインと検証用パイプラインという異なるハードウェアにおいて同一の命令を実行し，それらの結果を比較することにより故障を検出する方式である．このような命令のレベルの多重実行としては，単一ハードウェアにおいて異なる時刻に同一の命令を複数回実行するという時間的冗長性を用いる IRI(Instruction Re-Issue)[2] が提案されている．これらの方式は，プロセッサアーキテクチャの仕組みを巧みに利用して，少ない性能およびハードウェアのオーバヘッドで命令レベルの多重実行を実現している．一方，提案する SmartCore システムは，マルチコアプロセッサを対象として，プロセッサ構成を変更することなく，プロセッサの入出力として観測されるパケットのレベルで故障を検出するところに特徴がある．

マルチコアプロセッサの特徴を利用する関連研究として，Slipstream プロセッサ[3] がある．こちらは，通常の命令列を 1 つのコアで動作させ，同時に，同様の挙動を示す短い命令列を異なるコアで動作させることで，高性能化とディペンダビリティの向上を達成する．この方式は，プロセッサ構成の大幅な変更が必要となる．また，低レイテンシのコア間通信の仕組みを必要とするため，我々が想定している大規模なメニーコアアーキテクチャにおける利用は困難である．

複数のチップに同一の動作をさせることで故障を検出する技術として，PowerPC 750GX に採用されている LockStep がある．プロセッサのエラーは最終的にプロセッサの入出力で観測されるのでそこで比較すればよいという方向は SmartCore システムに近い．SmartCore システムでは，高機能ルータがパケットのレベルで比較する，大規模なメニーコアアーキテクチャを想定しているという点で大きく異なっている．

4. おわりに

我々は，1 チップに搭載可能なコアが数十から数千コアというオーダまで増え続けることを想定し，そのような豊富なコアを活用するメニーコアアーキテクチャの開発を目指して研究を進めている．

本稿では，メニーコアプロセッサのディペンダビリティ向上と高性能化を目指す SmartCore システムの構想を述べた．SmartCore システムとは，多数コアの利用と高機能ルータによってメニーコアプロセッサの性能向上を目指す仕組みである．冗長実行の構成を動的に変化させることにより，ディペンダビリティおよびバンド幅を向上させる方式を議論した．

今後，ソフトウェアシミュレータを用いて，SmartCore システムの性能オーバヘッドおよびバンド幅向上の効果を明らかにしていく．

謝　辞

本研究の一部は，科学技術振興機構・戦略的創造研究推進事業 (CREST) 「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による．ルータの基礎から実装に至るまで丁寧に指導していただきました．電気通信大学の吉永努先生，慶応義塾大学の松谷宏紀博士に感謝いたします．

参考文献

1) T. M. Austin: DIVA: a reliable substrate for deep submicron microarchitecture design, *MICRO-32*, pp. 196–207 (1999).
2) T. Sato: A Transparent Transient Faults Tolerance Mechanism for Superscalar Processors, *IEICE transactions on information and systems*, Vol. 86, No. 12, pp. 2508–2516 (2003).
3) K. Sundaramoorthy, Z. Purser, and E. Rotenburg: Slipstream processors: improving both performance and fault tolerance, *ASPLOS-IX*, ACM, pp. 257–268 (2000).