メニーコアプロセッサにおけるオンチップネットワークの可視化ツールの開発

森 洋介 †　植原 昂 †　吉瀬 謙二 †
東京工業大学大学院情報理工学研究科 †

1　はじめに

近年、1つのチップに複数のコアを搭載したマルチコアプロセッサが多く登場している．コア数は増加する傾向にあり、今後より多くのコアを搭載したメニーコアプロセッサが実現すると考えられる．メニーコアプロセッサではオンチップネットワークによって多数のコアが接続される．ネットワーク内のデータの流れをシミュレーションし、その挙動を可視化することは、ネットワークの解析や、実行プログラムのデバッグを支援し、メニーコアプロセッサ研究の速度向上につながる．

本稿ではオンチップネットワークにおけるデータの流れをシミュレーションし、可視化するツールを提案する．その初期段階として、メニーコアプロセッサアーキテクチャM-Core[1]を対象に、パケットの流れをシミュレーションし、視覚的に表現するツールの開発を行う．

2　M-Core アーキテクチャ

図1にM-Coreアーキテクチャ全体の構成を示す．M-Coreアーキテクチャでは、小規模で均一なコアをタイル状に多数並べる構成を採用している．M-Coreの各構成要素はノードと呼ばれる．各ノードは2次元メッシュネットワークで接続される．コア間のデータ転送には、パケット転送方式を採用する．

図2にノードの詳細を示す．ノードはルータ，演算処理ユニットであるコア、ノードメモリ、INCC(Inter-Node Communication Controller)で構成される．本研究ではパケットの流れを可視化するため、ルータに焦点を当てる．まずINCCはパケットを生成し、ルータに出力する．パケットは入力線を経由して入力バッファに格納され，図2のXBAR switchを通り，東西南北とINCCの5箇所のうち、しかるべき方向へと出力される．ルータは1サイクルで、パケットの1サイクルの送信単位である1フリットを処理する．

3　可視化ツール「flitplayer」

3.1　設計と実装

本章では、M-Coreのフリットの流れをシミュレートし、可視化する、**flitplayer**について述べる．flitplayerは、M-Coreアーキテクチャのcycle-accurateなシミュレータ「SimMc」上で実行した、アプリケーションプログラムの実行トレースを入力とする．

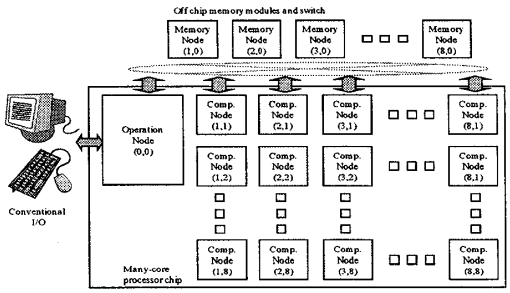

図1: M-Coreアーキテクチャ

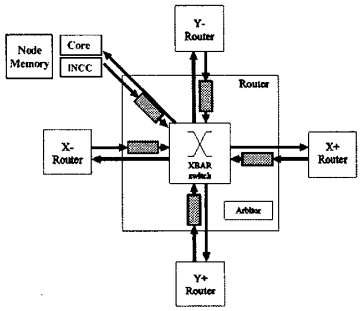

図2: ノードの詳細

flitplayerは実行トレースから、パケットの流れをシミュレートし、その挙動を表現する．シミュレーション機能を実現するために、flitplayerにはルータの機能を実装する．また、シミュレーションを詳細に行うために、フリット単位でシミュレーションを行う．

可視化ツールであるflitplayerがシミュレーション機能も有する理由は、今後、ネットワーク上の様々なデータの可視化を容易にするために、SimMcから出力するデータ容量を最小限に止めたいという方針からである．

flitplayerはJavaを用いて実装をしている．この理由として、様々な環境で実行できること、GUI周りの実装が比較的容易にできることが挙げられる．

flitplayerは、実行トレースを読み込むと、一定サイクルごとにネットワークの状態を書き出したダンプファイルを作成する．このダンプファイルは、高速に任意のサイクルの状態を表示するために使用される．ダンプファイルの作成が終わると、任意のサイクルから実行できる状態になり、シミュレーションの準備が完了する．

3.2　flitplayerの機能

図3にflitplayerの全体画面を示す．この図では5×6のネットワークを表示している．flitplayerの基本的な

Development of the Visualization Tool for On-Chip Network on Many-Core Processor
Yosuke MORI, Koh UEHARA and Kenji KISE
† Graduate School of Information Science and Engineering, Tokyo Institute of Technology

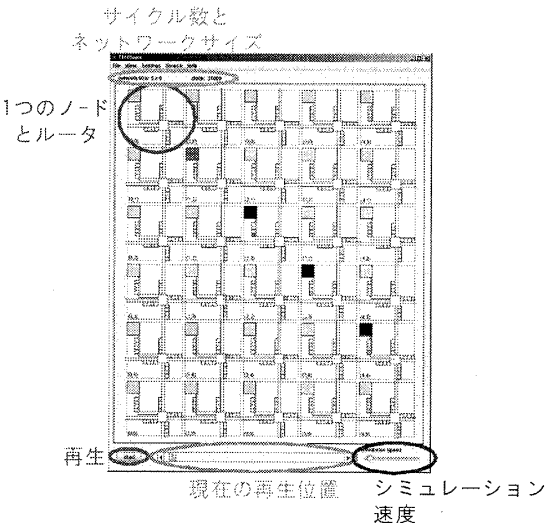

図 3: flitplayer の全体画面

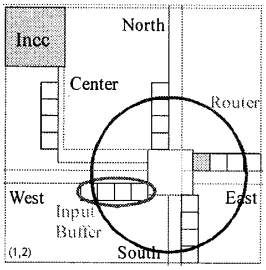

図 4: 1 つのノードの表現

機能として、左下の再生ボタンを押すと、ネットワークの状態がサイクル単位で順次再生され、フリットの流れが視覚的に表現される.

図 4 に flitplayer で表現される 1 つのノードを示す. ここにはルータの入力バッファの状態や、フリットの軌跡が表示される. 現在、以下の機能を実装している.

フリットの軌跡を表示

各ルータに、フリットが通過したラインを表示する. フリットが詰まっている場合はラインの色が変わる. 例えば、図 4 では、East から North と South から North にフリットが流れている. South から North のフリットの流れは、East から South のフリットの流れに妨げられて止まっている.

フリットの情報を表示

表示されているフリットをダブルクリックすると、対象のフリットの情報をダイアログに出力する. 現在取得できる情報は、フリットのタイプ、送信元ノード、送信先ノード等である.

ジャンプ機能

スクロールバーとサイクル指定の 2 種類の方法によって、任意のサイクルにおけるネットワークの状態を確認できる.

入力バッファの状態を表示

ルータの各入力バッファの状態として、貯まっているフリットを表示する. 停滞しているフリットに対しては、色を変えて強調する.

その他に、表示する部分の拡大・縮小・移動、表示色の変更、キーボードによる操作など、ユーザーインターフェースの面での機能も搭載している.

4 今後の課題

本稿で述べた内容は、本研究の初期段階である. 今後、メニーコアプロセッサの研究開発の加速を支援するツールとして有用な機能を持たせるためには、多くの課題が存在する.

柔軟性のある機能の充実

例えば、M-Core アーキテクチャは各ノードがメッシュ接続であるため、現在 flitplayer ではメッシュ型のネットワークのシミュレーションのみ対応している. 将来的には、リング型、ツリー型など様々なトポロジに対応する予定である.

同じく、ルータの仕様もシミュレーションする対象に応じて柔軟に対応する. このように、様々なメニーコアプロセッサに対応した柔軟性のあるツールを目指す.

可視化機能の充実

ネットワーク全体の情報の取得も課題である. 現在では個々のフリットの情報が取得できるという段階にとどまっている. 例として、ネットワークの混雑度を色の濃淡で表現し、混雑しているノードを視覚的に分かりやすく示したり、ネットワーク内のフリット数がサイクルごとにどのように変化するかを、グラフを用いて表現する機能などを検討している.

デバッグツールとしての機能の充実

デバッグツールとして、一番充実させるべき機能は検索機能である. 細かい条件を指定して検索できる事はデバッグには必須の要件である.

5 まとめ

本稿では、メニーコアプロセッサの研究開発の加速を目指し、オンチップネットワークの可視化ツールを提案した. その初期段階として M-Core アーキテクチャを対象とした、パケットの流れをシミュレートし、可視化するツールを開発した. 今後このツールを元に、様々な機能の充実を行っていく.

参考文献

[1] Koh Uehara, et al. A Study of an Infrastructure for Research and Development of Many-Core Processors, UPDAS, 2009. p. 414-419.