# メニーコアプロセッサ向けプロトタイピングシステムの高速化

渡邉 伸平 † 高前田 伸也 † 姜 軒 † 三好 健文 †‡ 吉瀬 謙二 †

東京工業大学 大学院情報理工学研究科 † 科学技術振興機構 ‡

## 1 はじめに

プロセッサアーキテクチャは，1 チップに数十から数百のコアを搭載するメニーコアへと向かっている．今後，メニーコアにおけるソフトウェア・ハードウェアの研究がますます重要になると考えられる．

メニーコアプロセッサに関する研究・開発を効率的に行う為には，様々なアイディアを迅速に検証できる環境が必要となる．構成の変更が柔軟であること，専用のハードウェアが必要でないことから，現在はソフトウェアのシミュレータが検証に用いられることが多い．しかし，メニーコアプロセッサのシミュレーションの場合，シミュレーション対象のコア数が増加すると，シミュレーション時間が大幅に長くなってしまうという欠点がある．そこで我々は，ハードウェアによる高速プロトタイピングシステム ScalableCore を開発している．シミュレーション対象の一構成要素を実装した “ScalableCore Unit” を，共通の接続インターフェースである “ScalableCore Board” を用いて複数接続することにより，スケーラビリティを満たしつつ，柔軟かつ高速なシミュレーション環境の実現を目指す．

## 2 M-Core アーキテクチャ

本稿では M-Core アーキテクチャ[1] をプロトタイピングの対象とする．図 1 にその構成を示す．M-Core アーキテクチャは，多数のノードをタイル上に配置した構成を取る．ノードには，計算を行う計算ノード，メインメモリとのアクセスを行うメモリノード，パケットの転送のみを行うパスノードがある．ノード間のデータ送受信には各ノードに割り当てたチップ内で固有のノード ID を用い，DMA 転送を利用したパケット転送方式で行う．M-Core における計算ノードは，コア，ノードメモリ (小規模

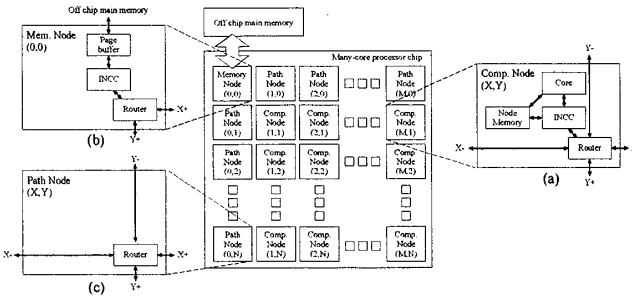


図 1: M-Core アーキテクチャ．

Speedup of a Prototyping System for Many-Core Processors
Shimpei WATANABE†, Shinya TAKAMAEDA†, Ken KYOU†, Takefumi MIYOSHI†‡, Kenji KISE†
†Graduate School of Information Science and Engineering
‡Japan Science and Technology Agency(JST)

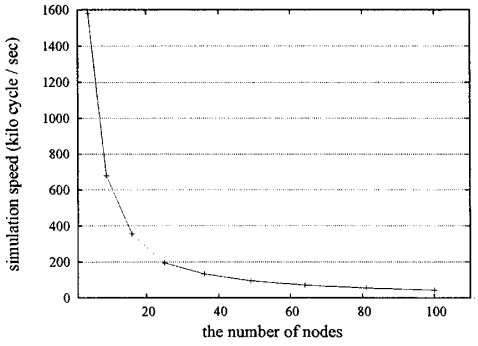


図 2: SimMc によるシミュレーション速度．

なローカルメモリ)，INCC(Inter Node Communication Controller)，ルータを搭載する．INCC は DMA 要求を受け，メモリを読み書きし，パケットへの整形を行い，ルータへと渡す．パケットはルータ経由でオンチップネットワーク上を転送される．

M-Core アーキテクチャの cycle-accurate なシミュレータとして SimMc がある．SimMc 上で，並列化した N-queen を実行した際のコア数とシミュレーション速度の関係を図 2 に示す．実行環境は CPU:Intel Core 2 Duo CPU E8400(3.00GHz)，Memory:4GB である．SimMc は各ノードのシミュレーションを直列に行うことから，対象とするのノード数が増えるのに従い，速度が低下する．従って特にシミュレーション対象のノード数が大きい場合，高速化を行う必要が出てくる．

## 3 ScalableCore

### 3.1 概要

我々はソフトウェアシミュレーションの速度という問題点を解決し，メニーコアプロセッサに関する研究を加速するためのシステムとして，ハードウェアによるプロトタイピングシステム ScalableCore[2] を提案している．ScalableCore の主な構成要素は，シミュレーションノード “ScalableCore Unit” と，接続インターフェース “ScalableCore Board” である．各 Unit にシミュレーション対象の一構成要素と Unit 間の通信機構を実装し，それらを Board を用いて接続することにより，メニーコアプロセッサの評価環境を実現する．4 枚の Board が 1 つの Unit を取り囲むように配置し，各 Unit の I/O ポートを隣接する 4 つの Unit の I/O ポートと接続する．ポート数の制限により，Unit 間の通信はシリアル転送とし，シリアル通信モジュールを上下左右の 4 方向分実装する．この接続方法により，シミュレーション速度を保ちつつ，ScalableCore Unit の増減が可能となる．その他の構成要素として小型の液晶ディスプレイがある．これを用い

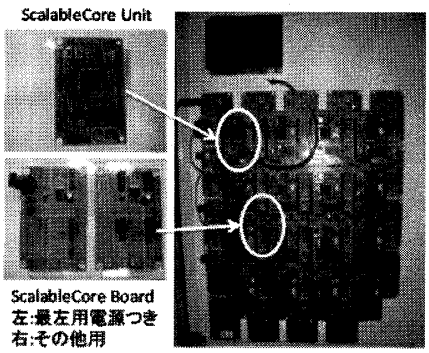


図 3: 実装した基板.

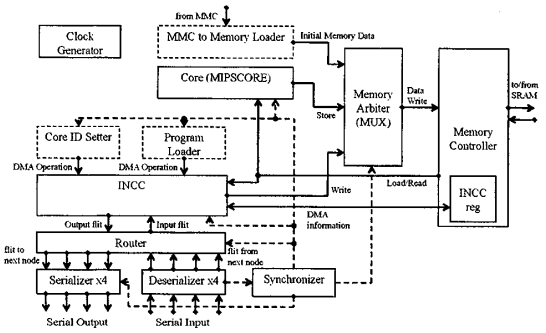


図 4: HDL 実装のブロック図.

て，シミュレーションの状況などを表示する.

### 3.2 実装と評価

M-Core アーキテクチャの計算ノード群部分を，ScalableCore システム上に実装する．その上でベンチマークプログラムを実行し，動作速度について評価する.

図 3 に今回制作した基板を示す．左上の図が，制作した ScalableCore Unit である．Xilinx Spartan-3E XC3S500E(50 万ゲートの FPGA)，512KB の SRAM を搭載する．左下の図が，制作した ScalableCore Board である．左方向への通信ができない代わりに，電源および MMC(MultiMedia Card) が接続できる最左用のものと，上下左右 4 方向との通信が可能なその他用のものがある．右側の図は，実際に Unit と Board を接続した状態である．図中左上に見えるのはコマンドインタプリタ型の液晶ディスプレイで，シリアル通信によりコマンドを送ることでシミュレーションの状況などが表示できる.

Verilog HDL を用いて，各 ScalableCore Unit 上に M-Core アーキテクチャの 1 計算ノードと，ノード間通信機構を実装した．図 4 に HDL 実装のブロック図を示す.

システムに電源が投入されると，ID のセット，MMC から各ノードへのロード等の初期化フェイズを経て，アプリケーションプログラムの実行が開始される．ここでは，基とするシミュレータである SimMc の 1 シミュレーションサイクルを 1 仮想サイクルとし，1 仮想サイクルあたり数十サイクルかけてシミュレーションを行う．各仮想サイクル中ではコア・INCC・ルータ・データ送受信の各処理をそれぞれ直列に行う．これにより，SRAM

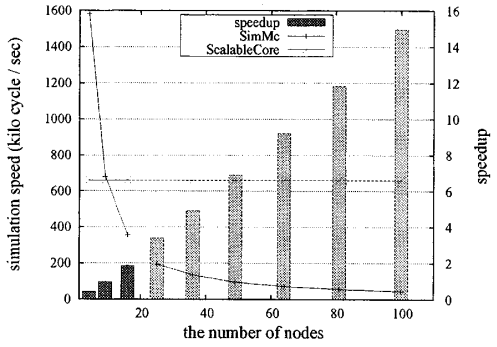


図 5: ScalableCore システムを用いた場合のシミュレーション速度及び速度向上.

へのアクセス競合を排除し，モジュール間における依存関係の保証を容易化できる．また，データの送受信が終了したところで次の仮想サイクルに進むことにより，隣接するノードとの同期をとりつつアプリケーションを実行することができる.

ScalableCore システムを用いた場合のシミュレーション速度及び速度向上を図 5 に示す．ベンチマークとして，並列化した N-queen を用いる．現行バージョンの ScalableCore システムでは，動作検証は 16 ノードまで行っている．そのため今回の評価において，破線及び色の薄い棒グラフで示される部分は，ScalableCore が 16 ノード以下でのシミュレーション同様に 661Kcycle/sec で動作すると仮定した場合の値となっている．16 ノードでのシミュレーションでは 2 倍程度の速度向上が実現できた．また，100 ノードでのシミュレーションでは 15 倍程度の速度向上が見込まれる.

## 4 まとめ

メニーコアプロセッサの為のプロトタイピングシステム ScalableCore を提案・実装し，16 ノードのシミュレーションで 2 倍程度の高速化を実現した．今後の課題として，ノード数を増加した場合の動作検証，さらなる高速化，メモリノード・メインメモリを含めた M-Core アーキテクチャの全機能の実装などが挙げられる.

**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.

**参考文献**

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

[2] 高前田伸也，他，メニーコアアーキテクチャ研究のためのスケーラブルな HW 評価環境 ScalableCore システム，情処研報 2009-ARC-185.