社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

チップ内ネットワークを用いたマルチタスク向け リコンフィギャラブルアーキテクチャの検討

長谷川揚平† 松谷 宏紀† 鯉渕 道紘†† 天野 英晴†

† 慶應義塾大学大学院理工学研究科 〒 223–8522 神奈川県横浜市港北区日吉 3-14-1
†† 国立情報学研究所 〒 101–8430 東京都千代田区一ッ橋 2-1-2
E-mail: †drp@am.ics.keio.ac.jp, ††koibuchi@nii.ac.jp

あらまし 本論文では，リコンフィギャラブルコアを基本演算ユニットとし，複数のコアをチップ内ネットワークを用いて接続するリコンフィギャラブルアーキテクチャを提案する．本アーキテクチャでは，ストリーミングデータと構成情報の 2 つのコア間通信を 1 つのパケットネットワークに統合した点が特徴である．リコンフィギャラブルシステムにおけるアプリケーションの動作方式としては，演算データをコア間で移動させるパイプライン方式と，構成情報を移動させる方式が提唱されているが，両者は生じる通信パターンが大きく異なる．そのため，ネットワークの設計が，柔軟な特性をもつ本アーキテクチャの設計の鍵を握る．本稿ではコアのアーキテクチャとして NEC エレクトロニクス社の DRP-1 を想定し，JPEG エンコーダを例としてシミュレーションによる性能評価を行った．評価結果より，コア間で構成情報の転送が頻繁に発生する構成情報移動方式と比較すると，パイプライン動作の方がスループットの観点ではネットワークを効率良く使うことができることが分かった．一方で，構成情報用のキャッシュを設け，構成情報転送の経路を分散させることで，スループットを向上させることが可能であることがわかった．
キーワード リコンフィギャラブルアーキテクチャ，チップ内ネットワーク，マルチタスク，ストリーム処理

Reconfigurable Architectures with On-Chip Networks for Multitask Designs

Yohei HASEGAWA†, Hiroki MATSUTANI†, Michihiro KOIBUCHI††, and Hideharu AMANO†

† Department of Information and Computer Science, Keio University
3-14-1, Hiyoshi, Kohoku-ku, Yokohama, 223-8522 Japan
†† National Institute of Informatics 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan
E-mail: †drp@am.ics.keio.ac.jp, ††koibuchi@nii.ac.jp

Abstract A reconfigurable processor architecture which employs on-chip networks to connect multiple reconfigurable cores. Both the processed stream data and configuration data are integrated into a single packet and transferred in a unique network. Two application execution methods: task-level pipelining policy and configuration moving policy can be applied into the proposed architecture. Since these methods require different communication traffic, analysis of the network performance is indispensable. In this report, we designed each task of the JPEG encoder on the NEC electronics' DRP-1, and evaluated the performance of two execution methods using the flit-level network simulator. As a result, the pipelining policy outperformed the configuration moving policy from the viewpoint of the throughput. Even in the case of configuration moving policy, simulation result shows that the configuration cache can distribute the configuration data between each core to improve the throughput.
Key words Reconfigurable Architectures, On-Chip Networks, Multitasking, Stream Processing

1. はしがき

近年，組み込み機器においても，要求される性能や機能の多様化が進み，高速性という指標に加え，柔軟性や信頼性，消費電力などの指標が重要視されはじめている．このような背景のもと，チップ上に複数のプロセッサコアを搭載した組み込み向けのオンチップマルチプロセッサ [1], [2] や，用途に応じて構成要素を増やすことが可能なコンフィギャラブルプロセッサ [3], [4] などのマルチコアアーキテクチャが注目されている．マルチコアアーキテクチャは，タスク/プロセスレベルでの並列処理を実

現し，比較的低い周波数で動作させることで低消費電力化が可能である．

一方、柔軟性の導入という視点から，Field Programmable Gate Array (FPGA) に代表されるリコンフィギャラブルデバイスを利用したシステムも普及している．これらのリコンフィギャラブルデバイスは，ハードウェア実行による高速性に加え，ユーザプログラマブルであるという点でソフトウェアの柔軟性を兼ね備えたデバイスである．中でも, 面積効率や電力効率を追及した動的リコンフィギャラブルプロセッサ [5]〜[8] も普及している．

これらのリコンフィギャラブルデバイスをコアとしたマルチコアシステムの検討もはじまっている．我々はこれまで，NEC エレクトロニクス社の動的リコンフィギャラブルプロセッサである DRP を対象として，単一コアにおける面積効率，電力効率の評価を行ってきた [9]．組み込み機器を対象とするアプリケーション評価より，比較的小さなコアをシステム LSI に搭載することで多くの場合において効率が良いことがわかってきている．これらのリコンフィギャラブルコアを複数接続することで，柔軟性やスケーラビリティを高い面積効率や電力効率で実現することが可能であると考えられる．

本研究では，複数のリコンフィギャラブルコアをオンチップ上で接続したアーキテクチャを提案する．このアーキテクチャでは複数のコア同士の接続網として，データと構成情報の両方を転送可能なチップ内ネットワーク [10], [11] を用いる．チップ内ネットワークでは，従来の並列計算機で用いられてきたパケット転送方式をコア間のデータ通信に応用する．これにより，微細化による配線長の増大を抑え，高帯域のデータ通信を実現する．また，従来のオンチップバスよりも高バンド幅であるチップ内ネットワークを用いることで，ストリーミングデータとリコンフィギャラブルコアの構成情報を同一のインターコネクトで転送することを実現した．この点より，スケーラブルなマルチタスク処理システムを実現し，様々なアプリケーションを実行可能な仮想ハードウェアを提供する．

具体的には，NEC エレクトロニクス社の Dynamically Reconfigurable Processor (DRP) のマルチコア構成において，コア間通信にチップ内ネットワークを応用したアーキテクチャを検討する．また，アプリケーションの実行方式として，各コアで実行するタスクを固定しコア間でパイプライン処理を行う方式と，各コアの実行タスクを動的に変化させることでデータ通信を抑えた構成情報移動方式を，JPEG エンコーダアプリケーションにおけるネットワークの利用形態の点から比較，検討する．

2. 関 連 研 究

チップ内ネットワークを用いたリコンフィギャラブルアーキテクチャとして，QuickSilver 社の開発した Adaptive Computing Machinel (ACM) [12] があげられる．ACM では 各ノードを，Matrix Interconnect Network (MIN) と呼ばれる Tree トポロジのチップ内ネットワークにより接続する．MIN では，比較的単純な固定型ルーティングが用いられており，隣接ノード間は 1 サイクル，クラスタ間では 5 サイクルのレイテンシで通信することが可能である．ACM では，コア間のデータ通信だけではなく，コア内の基本構成要素間の通信にチップ内ネットワークを応用している点で本研究と立場が異なる．

また，FPGA の部分再構成機能を利用したマルチタスク処理を実現するシステムが提案されている．これらのシステムは，FPGA を一定の大きさのタイルに分割し，タイル単位で部分再構成を行う．しかしながら，タスク間の通信には，FPGA のプログラマブル配線による接続方法が用いられており，場合によってはタイルをまたいだ通信が発生し，一部分のみを再構成することが困難である．

このような問題を解決するため IMEC 社の Marescaux らは，タスク間の通信にチップ内ネットワークを応用することで，タイル単位での部分再構成を容易にし，マルチタスク処理を実現している [13]．Marescaux らのチップ内ネットワークでは，トポロジとして 2D Torus を採用している．また，タスク間通信を担う各ルータは 2 本の仮想チャネルをもち，スイッチング方式として wormhole 方式，ルーティングには固定型ルーティングである次元順ルーティングを採用している．Marescaux らは，これらのチップ内ネットワークを用いたリコンフィギャラブルシステムを，Xilinx 社の Virtex II FPGA (XC2V6000) 上に実装している．

IMEC 社の動的リコンフィギャラブルプロセッサ ADRES [8] は，乗算機能を含む 8×8 の機能ユニットとレジスタファイルをアレイ上に配置した構成をとる．これらの接続には従来のプログラマブル配線が用いられるが，実際に SoC で用いられる際には複数のコアをチップ内ネットワークで接続したマルチコア構成をとる．また，IO や RISC プロセッサ，SRAM なども同様にチップ内ネットワークで接続し，特にマルチメディア処理の高速化を目的としている．また，IPFlex 社の次世代アーキテクチャである DAPDNA-3A では，DAPDNA を 4 つ接続したマルチコア構成をとる．コア間の通信には AXION と呼ばれる細粒度のリコンフィギャラブルなバス結合網を採用している．

3. 対象アーキテクチャ

本節では，本研究で対象とするマルチコア構成のリコンフィギャラブルアーキテクチャについて述べる．

本研究で対象とする基本的なアーキテクチャを図 1 に示す．各コアには，FPGA や DRP などの単一種類のリコンフィギャラブルコアを規則的に配置するホモジーニアスな構成を想定するが，組み込み用途の RISC プロセッサや DSP, ASIC なども混載したヘテロジーニアスな構成であってもよい．このようなアーキテクチャでは，各コアは比較的小さな領域に限定するため，小さな面積に多くのアプリケーションを実行可能な動的リコンフィギャラブルプロセッサがよく適合すると考えられる．

3.1 コアアーキテクチャ

本論文では，NEC エレクトロニクス社の動的リコンフィギャラブルプロセッサである DRP-1 [6] を対象のコアアーキテクチャとする．DRP は Tile とよばれるリコンフィギャラブルユニットを単位として，これらを複数接続することで所望の規模の DRP コアを実現可能である．各 Tile は，8bit の演算器とレジスタファイルなどから構成される Processing Element (PE) を

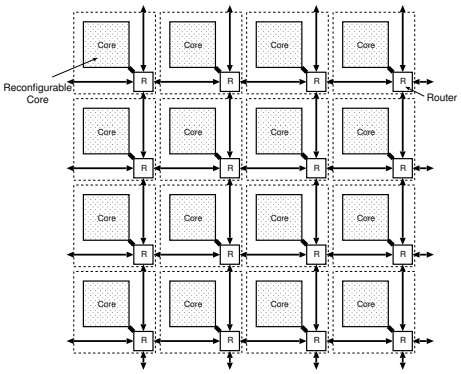


図 1　対象とするリコンフィギャラブルアーキテクチャ

8× 8 の 2 次元アレイ状に並べ，その周辺に VMEM，HMEM と呼ばれる分散メモリを縦横に拡散配置し，さらに中央部に StateTransition Controller (STC) を配置した構成となっている．

各 PE は 8-bit の ALU，シフトやデータ制御，簡単な論理演算を行なう Data Manipulation Unit (DMU)，8-bit の Flip Flop，レジスタファイルから構成される．また，PE アレイ上で実現されるデータパスをコンテキストと呼び，実行中に STC が発行した命令ポインタを各 PE の命令メモリが受け，利用する命令データをロードすることでコンテキストを切り替えることができる．

DRP のプロトタイプチップである DRP-1 は，8Tile の DRP コアを搭載し，全体として 512 個の PE と 80 個の VMEM, 32 個の HMEM をもつ．また，各 PE は内部の命令メモリに 16 コンテキスト分の構成情報を保持しておくことが可能で，これらをクロックサイクルごとに切り替えることが可能である．

3.2　チップ内ネットワークの基本構成

図 1 に示すように，本アーキテクチャでは各 DRP コアを基本的な演算ユニットとして搭載し，各コアごとに 1 つのネットワークルータで接続した構成をとる．

チップ内ネットワークでは，従来の並列計算機で用いられてきたパケット転送技術が一般的に用いられている．トポロジでは比較的単純な構造となる 2D Mesh/Torus や Tree 構造が用いられることが多い．また，リコンフィギャラブルシステム向けのトポロジとして Fat H-Tree なども提案されている．ルーティング方式では，経路の混雑状況に従って動的に経路を設定する適応型ルーティングと，経路を静的に決定する固定型ルーティングが用いられる．適応型ルーティングは制御が複雑となり，ハードウェアコストも大きいため，コスト制限の厳しいチップ内ネットワークではあまり用いられていない．

本アーキテクチャにおいても，各コアは比較的小規模な構成で実装されることを想定しており，ルータ等の構成も小規模であることが望ましい．このため，本アーキテクチャでは，各コアを 2D Mesh 状に接続し，双方向に通信することが可能なネットワークルータを想定する．また，ルーティング方式では代表的な固定型ルーティングである次元順ルーティングを採用する．

一方，現在のリコンフィギャラブルシステムでは，構成情報の設定は，演算データ通信用のシステムバスとは分離した専用のバスを利用することが多い．しかし，あまり構成情報を移動しない場合には，構成情報バスの利用効率が極めて悪い．一方で，一般的に構成情報量は演算データ量よりも大きいため，システムバスと共有するとバス全体の混雑を招く．このため，対象アーキテクチャでは，構成情報も演算データと同様に同一ネットワークを介して転送することを検討する．これにより，コア間で構成情報を共有した状況では，ネットワークを介して並列に構成情報の設定が可能となるため，効率的にコンフィギュレーションを隠蔽することが可能であると考えられる．

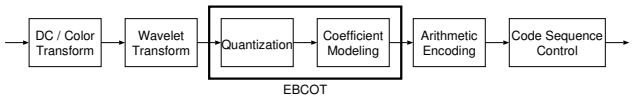


図 2　JPEG2000 のタスクフローグラフ

4.　アプリケーションの実行方式

本節では，対象アーキテクチャにおける典型的なアプリケーションの実行方式を説明する．

4.1　ストリーム処理とパイプライン動作

JPEG や MPEG に代表されるストリーム処理は，次々に到着するある一定の大きさのデータに対して，一連の処理が実行されるものである．本論文では，このようなストリーム処理において，一つの処理単位をタスクと呼ぶ．図 2 に JPEG2000 エンコーダのタスクの流れ (タスクフロー) を示す．

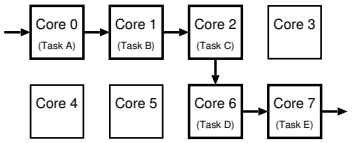
通常のチップ内ネットワークを用いた SoC では，各タスクを各コアにそれぞれ割り当て，パイプライン的に動作させる方式が一般的である．図 3 にパイプライン動作を行う場合の一般的なタスクの割り当てと，タスク間のデータの流れを示す．図 3 では，タスク間の通信は隣接間に限られている．JPEG2000 の場合には EBCOT と呼ばれる符号化処理の計算負荷が高く，これらをそれぞれ一つのコアで実行すると，EBCOT の処理がボトルネックとなる．このような場合には，EBCOT をさらに複数のサブタスクに分割して，ストリーム処理の負荷を分散することができる．

図 3(b) に示すように，パイプライン動作の場合には，各コアは割り当てられたある特定のタスクを実行し続けることとなる．そして，各コア間で演算データの通信を行い，タスクレベルでのパイプライン的に処理を行い，逐次実行に比べてスループットを向上させる．一方で，この場合には，アプリケーションの実行中にタスクの変更は発生しないため，コア間で動作中に構成情報の通信が行われることはない．

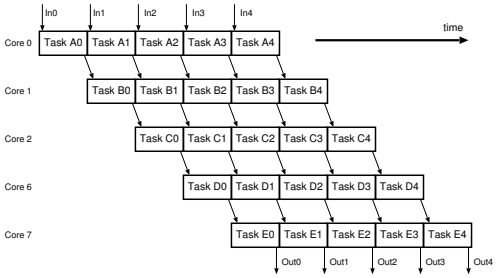
4.2　構成情報移動方式

前述したストリーム処理におけるパイプライン動作は，次々に入力される入力ストリームデータに対して，タスクレベルで並列処理を行うことが可能で，スループットを向上させる手段として有効である．しかし，より多くのアプリケーションに対応させる場合や，組み込み機器において OS と連携したマルチタスク処理を行う場合や，プリエンプションを実現する場合に，タスクのスケジューリングという観点では柔軟性に欠ける．

例えば，図 2 の JPEG2000 エンコーダの例のように，パイプライン動作によるストリーム処理では，ストリームデータをい

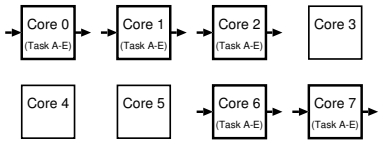


(a) Task Mapping (Pipeline)

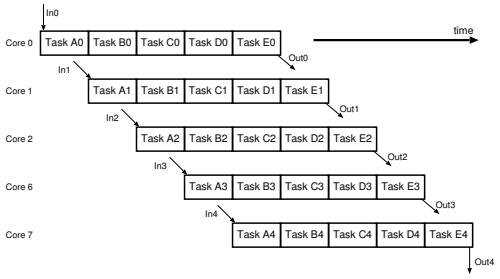


(b) Data Flow (Pipeline)

図 3　パイプライン動作 (演算データ移動方式)



(a) Task Mapping (Configuration Moving)



(b) Task and Data Flow (Configuration Moving)

図 4　構成情報移動方式

かに効率的にコア間で通信するかが重要である．このため，エンコード中にコア単位での構成情報の入れ替えは難しく，入れ替える際にはアプリケーション全体を入れ替えるような方法が現実的となる．一方で，FPGA や DRP をはじめとするリコンフィギャラブルデバイスでは，コア内部に保持する構成情報メモリは全体の面積うちの大きな割合いを占める．そのため，小さなアプリケーションであればすべてのタスクの構成情報をコア内部のメモリに格納することが可能である．この特性により，従来のパイプライン動作と異なる動作モデルが可能である．

すなわち，全く考え方を変え，各コアはストリームデータに対してあるタスクを実行し，終了したら構成情報を入れ替えて同一のストリームデータに対して次のタスクを実行する方式が考えられる．この実行中に，次のストリームが到着する場合には，別のコアにおいて同様の処理が割り当てられ，次々に構成情報を入れ替えながら実行する．この方法は，あるストリームデータは一旦あるコアに到達したら，一連の処理が終了するまでそのコアに留まることになる．このため，入出力は最初と最後の一回のみとなる．本論文ではこの方式を構成情報移動方式と呼ぶ．構成情報移動方式におけるタスク割り当てとデータフローを図 4 に示す．

各リコンフィギャラブルコアにおいて，構成情報メモリが十分であれば，各コアがそれぞれ全タスクを格納しておけば，構成情報の移動も不要となり，ネットワーク上を流れるデータ量を最小限に抑えることが可能となる．また，ストリームデータを並列に処理することが可能であれば，コア数が多いほどストリームレベルでの並列処理が可能となり，スループットが向上する．しかし，実際には一つの構成情報に対して一つのタスクを実装する場合が多く，これによりタスクの設計も単純化される．このため，タスクの切り替えには少なからず構成情報の移動が必要となり，一般的に構成情報の量は大きなものであるため，構成情報の通信がスループット低下の要因となり得る．しかしながら，パイプライン動作と比較して，より柔軟にタスクの入れ替えが可能であり，マルチタスク処理の実現には不可欠な機能である．

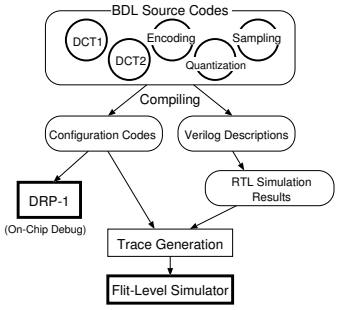


図 5　シミュレーション環境

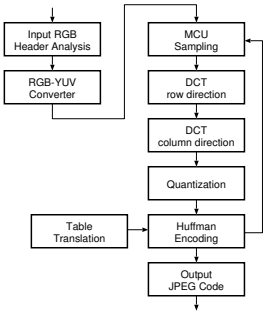


図 6　JPEG のタスク分割

本論文では，実際にアプリケーションによる性能評価を行い，パイプライン動作と構成情報移動方式の性能評価や，そのネットワーク上の特性などを評価する．その上で，対象とするアプリケーション対して，どのような構成のリコンフィギャラブル・マルチコアアーキテクチャが効率的であるかを検討する．

5. 評　　価

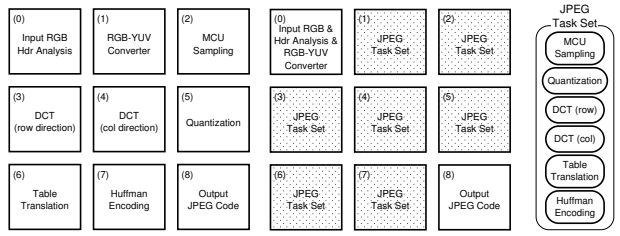
本節では，典型的なストリーム処理のアプリケーションである JPEG エンコーダの通信パターンを用いて，対象アーキテクチャにおけるパイプライン方式と構成情報移動方式における通信特性について評価を行う．

5.1 シミュレーション環境

今回用いたシミュレーション環境の構成を図 5 に示す．

チップ内ネットワークのシミュレーションに C++ 言語で記述されたフリットレベル・シミュレータを使用した．シミュレーションには，C 言語などで記述された対象アプリケーションのプログラムから得られるタスク間の通信パターンを使用する．このようなトレースデータの時間軸を変化させることで，様々な注入データレートに対するスループットとレイテンシを計測することが可能である．

今回の評価に用いたチップ内ネットワークの基本構成は以下の通りである．ネットワークトポロジとして 3 × 3 のサイズをもつ 2D Mesh を想定し，ルーティングアルゴリズムとして次



(a) Pipeline (b) Configuration Moving

図 7　JPEG エンコーダのマッピング例

元順ルーティングを用いた．各ルータは 5 つのポートをもち，一つは DRP コアとの接続に，残りは隣接ルータとの接続に使用する．また，各ルータのスイッチング機構として， IO バッファ，クロスバ，クロスバコントローラを単純化したモデルを採用し，ヘッダフリットが隣接ルータや DRP コアに転送されるのに 3 サイクルかかるものとする．ルータのチャネル幅は 64bit で，各コアの入出力のポートでのデータ幅も同様に 64bit である．パケット転送方式として wormhole 方式を採用し，パケット長はヘッダの 1 フリット分を含め最大 16 フリットとした．

また，本論文では，コアとして NEC エレクトロニクス社の DRP-1 を想定しており，個々のタスクが DRP-1 上で実行可能となるように，JPEG コーデック全体を図 6 に示すように分割した．そして，個々のタスクを DRP 向けの C レベルの動作記述言語 Behavioral Design Language (BDL) を用いて記述し，DRP コンパイラ [14] を用いて合成を行った．これらのタスクは，個別に構成情報を生成し，実際に DRP-1 上で実機動作を確認した．

そして，マルチコア構成での性能評価にあたり，各タスクを図 7 に示すようにマッピングを行った．JPEG コーデックは，タスク間で効率的にパイプライン化が可能なアプリケーションである．このため，パイプライン動作を想定した場合，ほぼ理想的なマッピングが可能で，ほぼパケットの衝突の起こらないデータ通信を行うとができる (図 7(a))．一方，構成情報移動方式を対象としたマッピング (図 7(b)) では，RGB 入力と RGB-YUV 変換タスクを統合したタスクと，JPEG Code の出力用タスクを用意し，この 2 つはそれぞれコア 0 とコア 8 に固定した．それ以外の 7 つのコアが構成情報を入れ替えながらストリームレベルでの並列処理を行う．なお，本論文では，コア 0 が全タスクの構成情報を保持可能であると仮定し，コア 0 から各コアに構成情報を転送するものとする．

ここで，構成情報移動方式において，各コアが全タスクを内部の構成情報メモリに格納可能であるとすると，ネットワーク上を流れるデータは，コア 0 から各演算コアに分配される RGB データと，各コアからコア 8 に収集される符号化データのみとなる．しかし，入出力はそれぞれ 1 つのコアに限定されているため，7 つのコアに対して独立にデータを分配・収集を行うと，コア 0，8 に対してトラフィックが集中することとなり，極度のスループットの低下を招くと予想される．このため，各コアの入出力ストリームは，パイプライン的に各コアに対して分配・収集が行うように実装した．

フリットレベルシミュレータで用いるトレースデータは，各

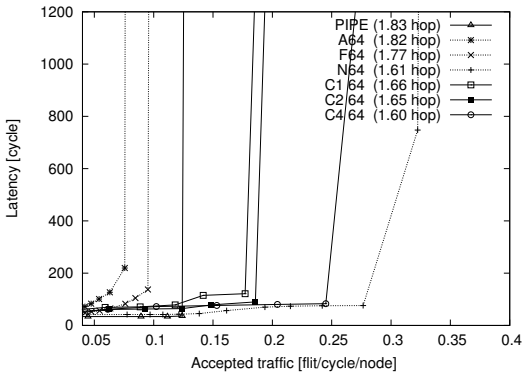


図 8　シミュレーション結果

タスクの動作合成後の RTL シミュレーションの結果を用いて生成した．個々のタスクは DRP-1 上に実装されており，各タスクの実行サイクル数やデータ転送量を得ることができる．また，生成した構成情報のサイズも取得し，単純なタスク切り替えの制御機構をモデル化し，構成情報移動方式における構成情報の転送パターンも生成し，トレースデータに加えた．

5.2　シミュレーション結果

図 8 にシミュレーション結果を示す．シミュレーションでは，パイプライン動作を行うもの (PIPE) と構成情報移動方式に関して以下のパターンを比較した．

- A64: タスクを切り替える度に構成情報をコア 0 から転送するケース
- F64: 後述のように最も構成情報量の大きい DCT (row/col) を各コアがそれぞれ常時保持し，それ以外のタスクの切り替え時には構成情報をコア 0 から転送するケース
- N64: 全タスクの構成情報を各コアに保持可能であるとし，構成情報の転送が一切ない理想的なケース
- C1, C2, C4: 各コアが構成情報用のキャッシュをもち，同時に保持可能なタスクの数をそれぞれ 1, 2, 4 とし，演算コア間で構成情報の共有を行うケース

図 8 では，それぞれのスループット (Accepted Traffic) とレイテンシを示している．評価結果より，構成情報の移動が一切なく，ストリームデータの転送が最小限に抑えられた N64 が他のパターンと比較して高いスループットを実現可能であることがわかる．また，パイプライン動作 (PIPE) は，A64 や F64 と比較して，高いスループットを実現している．これは，JPEG エンコーダが元来パイプライン化をしやすいアルゴリズムであるという点と，コア間通信においてほとんどパケットの衝突が起こらないためである．これに対し，A64 や F64 は構成情報転送時に ノード 0 からのトラフィックが集中することで特定の経路が混雑し，スループットが低下している．

また F64 は A64 と比較して転送する構成情報量を削減することが可能であるが，大幅なスループット向上には結びついていない．これは依然として，コア 0 からネットワーク上を流れる構成情報量が大きいためにトラフィックの衝突が頻発すること，またタスク切り替えに要する時間がボトルネックとなるためであると考えられる．一方，構成情報用のキャッシュを設け

表 1　JPEG エンコーダの各タスクの利用コンテキスト数と構成情報量

Task Name	# of Contexts	Configuration Size [byte]
Header Analysis	2	2224
RGB-YUV Converter	5	21308
MCU Sampling	4	6292
DCT (row)	7	33504
DCT (column)	7	34544
Quantization	7	16604
Table Translation	4	8060
Huffman Encoding	10	18596

た C1, C2, C4 では，F64 のケースと異なり，各コアに構成情報を分散保持させコア間で共有することが可能となるため，構成情報の転送経路を分散することが可能となる．このため，C1, C2, C4 では，F64 や PIPE に比べ高いスループットを達成しており，C4 までキャッシュを増やせば N64 に近いスループットを達成可能であることがわかる．

ここで，各タスクの利用コンテキスト数と構成情報量を表 1 に示す．これより，比較的小さなタスクの場合でも構成情報量は非常に大きいことがわかる．また，パイプライン動作の場合に発生するストリームデータの転送量に比べても，A64 における構成情報の転送量は大きいものである．一方，利用コンテキスト数をみると，DRP-1 は 16 コンテキストを保持可能であるので，比較的余裕があるといえる．また，将来的には 32 コンテキスト程度の構成情報を保持可能となることが予想され，今回の評価における C2, C4 程度の構成は現実的なものであると考えられる．

今回の評価では，8Tile の DRP コアを想定したが，我々の評価では 2〜4 Tile 程度の比較的小さいサイズのコアがコスト対性能比がよいことがわかっている．より小さなサイズを想定すると，各タスクの実現により多くのコンテキスト数が必要となるため，コア内に保持可能なタスクの数に制約を受ける．ここで，構成情報を高速に転送するより強力なネットワークを構築することも考えられるが，今回の評価結果より，各コアに保持可能なコンテキスト数をより拡張する方がスループットの点では有利であると考えられる．

6. ま と め

本論文では，複数のリコンフィギャラブルコアをチップ内ネットワークで接続したマルチコア構成のリコンフィギャラブルアーキテクチャを提案した．また，このアーキテクチャにおいて，ストリームデータをパイプライン状に転送しながら処理をする方式と，ストリームデータはコア間で動かさず構成情報を入れ替えながら実行する構成情報移動方式を検討した．

コアのアーキテクチャを NEC エレクトロニクス社の DRP-1 とし，単純なネットワーク構造を想定したシミュレーションを行った．JPEG エンコーダを評価アプリケーションとし，各タスクを DRP-1 上に実装し，パイプライン動作と構成情報移動方式の性能評価を行った．

評価結果より，構成情報移動方式では，構成情報量が大きいために，頻繁にタスクの入れ替えが発生する場合にはパイプライン動作に比べ大幅にスループットが低下することがわかった．一方で，各コアに構成情報用のキャッシュを設け，タスクをコア間で共有することで，転送経路が分散されスループットを向上さることが可能であることがわかった．また，コンテキスト数を拡張し，各コアにすべてのタスクを保持することができれば，大幅にスループットを改善することが可能である．このとき，ストリームレベルでの並列処理が可能であれば，台数効果は極めて高く，コア数が増えた場合にもチップ内ネットワークの高帯域のデータ通信により，高いスループットを維持することが可能であると考えられる．

今回の評価で用いた JPEG エンコーダは，もともとパイプライン動作に適しており，今後はより多様な通信パターンをもつアプリケーションで評価を行う必要がある．

謝辞　本研究を進めるにあたり DRP-1 とその開発環境を提供していただいた NEC エレクトロニクス社および NEC システムデバイス研究所の皆様に感謝致します．本研究にはメンターグラフィックス社のユニバーシティプログラムを利用しており，そのシミュレーションにはメンターグラフィックス社の ModelSim を使用しています．

文　献

[1] S. Kaneko et. al. A 600 MHz Single-Chip Multiprocessor with 4.8 GB/s Internal Shared Pipelined Bus and 512 kB Internal Memory. *IEEE Journal of Solid-State Circuits*, Vol. 39, No. 1, pp. 184–193, January 2004.

[2] J. Sakai et. al. Multi-Tasking Parallel Method on MP211 Multicore Application Processor. In *Proceedings of the IEEE Symposium on COOLChips VIII*, pp. 198–211, April 2005.

[3] J. Arnold. S5: The Architecture and Development Flow of a Software Configurable Processor. In *Proceedings of the IEEE International Conference on Field Programmable Technology (FPT2005)*, pp. 120–128, December 2005.

[4] 宮森. コンフィギャラブルプロセッサ MeP とその SoC 応用開発事例. 信学技報 CPSY2004-81, pp. 31–36, January 2005.

[5] 末吉, 天野（編）. リコンフィギャラブルシステム. オーム社, 2005.

[6] M. Motomura. A Dynamically Reconfigurable Processor Architecture. *Microprocessor Forum*, October 2002.

[7] T. Sugawara, K. Ide, and T. Sato. Dynamically Reconfigurable Processor Implemented with IPFlex's DAPDNA Technology. *IEICE Transaction on Information & System*, Vol. E87-D, No. 8, pp. 1997–2003, May 2004.

[8] F. Veredas et. al. Custom Implementation of the Coarse-Grained Reconfigurable ADRES Architecture for Multimedia Purposes. In *Proceedings of International Conference on Field Programmable Logic and Application (FPL2005)*, pp. 106–111, August 2005.

[9] 長谷川, 阿部, 黒瀧, ヴ, 天野. 動的リコンフィギャラブルプロセッサにおける時分割多重実行の評価. 先進的計算基盤システムシンポジウム (SACSIS2006) 論文集, May 2006. to apear.

[10] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proceedings of the Design Automation Conference (DAC2001)*, pp. 684–689, June 2001.

[11] A. Andriahantenaina et. al. SPIN: a Scalable, Packet Switched, On-chip Micro-network. In *Proceedings of the Design Automation and Test in Europe Conference (DATE2003)*, pp. 70–73, March 2003.

[12] QuichSilver Technology, Inc. `http://www.qstech.com/`.

[13] T. Marescaux et. al. Interconnection Networks Enable Fine-Grain Dynamic Multitasking on FPGAs. In *Proceedings of the Field-Programmable Logic and Applications (FPL2002)*, pp. 795–805, September 2002.

[14] 粟島他. 動的再構成可能チップ DRP の C コンパイラ. 信学技報 VLD2003-118, pp. 23–28, January 2004.