

直接実行方式の提案

天野 英晴[†] 長谷川揚平[†] 阿部 昌平[†]

[†] 慶應義塾大学理工学部 〒223-8522 横浜市港北区日吉 3-14-1

drp@am.ics.keio.ac.jp

あらまし マルチコンテキスト型動的リコンフィギャラブルプロセッサ上でメディア処理を実行する場合、実行時間が大きく並列性が高いタスクは、PE(Processing Element)アレイ上で、それ以外のタスクは混載されている組み込みCPU上で実行するのが普通である。しかし、並列性の低い処理と高い処理の交代が頻繁になると、PEアレイと組み込みCPU間のデータ転送が頻繁になり、性能悪化と電力の浪費の原因となる。そこで、構成データメモリから構成データをPEアレイ上の一部のPEに送りつけ、そのまま実行する直接実行モードを提案する。効率の良い実行のため、構成情報テンプレートをを用いる方法、構成データバス自体をリコンフィギャラブルにする方法も提案する。予備評価の結果、構成データバス幅を2PE分にし、構成データのマルチキャストを実行した場合、MIPS 150MHzの組み込みCPUにデータを転送して実行するのと同程度程度の性能が確認できた。

キーワード リコンフィギャラブルプロセッサ, DRP, 動的再構成

The direct execution mode on dynamically reconfigurable processors

Hideharu AMANO[†], Yohei HASEGAWA[†], and Shohei ABE[†]

[†] Dept. of ICS, Keio University 3-14-1 Hiyoshi, Yokohama, 223-8522 Japan

drp@am.ics.keio.ac.jp

Abstract Most of media processing consists of a sequence of multiple tasks. When it is executed on a dynamically reconfigurable processor, a heavy weight task with high degree of parallelism is selected and executed on the Processing Element (PE) array. Other light weight tasks with little processor are executed on the embedded processor so as not to waste the context memory. However, frequent data transfer between PE array and an embedded processor may degrade the performance and increase the consuming power. In order to solve the problem, the direct execution method in which a part of PEs execute configuration data transferred from the central configuration memory directly is proposed. In order to enhance the execution performance, the configuration template and the reconfigurable configuration mechanism are also proposed. Preliminarily evaluation demonstrates that the execution performance is comparable with a 2-way superscalar MIPS with 150MHz clock when the configuration data bus is stretched so as to configure two PEs simultaneously.

Key words Reconfigurable Processor, DRP, Dynamic Reconfiguration

1. はじめに

動的リコンフィギャラブルプロセッサは、粗粒度のプロセッシングエレメント(PE)の機能と相互接続を動作中に変更することで、高い面積効率を実現可能であり、広い応用分野で利用可能な新しいアーキテクチャとして注目されている[15]。

なかでも NEC エレクトロニクス社の DRP [7] などのマルチコンテキスト型は、一つのタスクを複数のコンテキストを切り替えて時分割で実行することで面積効率を改善することができる。様々なアプリケーションの実装を通して、コンテキスト切り

替えを用いない場合に比べて4倍から14倍の面積効率を実現することが明らかになった[14]。

現在、DRP-1での実装は、単純だが計算量が多く、実行時間のうちの多くを占めるタスクを動的リコンフィギャラブルプロセッサであるDRPコアにより実行し、それ以外のタスクは同一チップ内に組み込まれるCPUが実行する。この方法は、ジョブの処理時間の大部分を、単純だが計算量の多い少数のタスクが占める場合は有効である。しかし、最近の動画処理など、タスクの組み合わせが複雑になると、並列性の少ない仕事を受け持つ組み込みCPUと、並列性の高い仕事を受け持つDRPコア

間のデータの転送が頻繁になり、性能的なボトルネックになる可能性がある。また組み込み CPU と DRP コア上の分散メモリとの間のデータの頻繁なやり取りは、消費電力の浪費にも繋がる。本来、DRP コア上の分散メモリに展開されたデータが、そのままの形で処理可能な場合は、いくら並列性が小さくても、DRP コア上で処理するのが効率的である。しかし、並列性の小さいコンテキストを多数実行するのは、コンテキストメモリの利用効率の点で不利であり、コンテキスト数の不足から、構成情報データをコンテキストメモリに転送する際のストールを招きかねない。

コンテキスト数の不足については、ダブルバッファを利用する仮想ハードウェア [11] [4]、コンテキストメモリの階層化 [13]、テンプレート再構成法 [16] などの改善手法が提案されており、コンテキスト情報の高速な転送法としては、コンテキスト情報の圧縮 [8]、差分再構成 [4] [12]、並列転送 [5] [10] などが提案されているが、いずれも利用される場合が制限されている。

そこで、本論文では、並列性の小さいタスクの実行時に、構成情報メモリから送られる構成情報データを特定の PE のみが、そのまま実行する直接実行法を提案する。直接実行法の利用により、本質的に分散メモリ上のデータ構造を変更する場合以外組み込み CPU との間の転送が不要となり、動的リコンフィギャラブルシステムの分散メモリ上にデータを載せっ放しにして、構成情報データのみを入れ替える「データレジデントコンピューティング」が可能となる。

2. 現在の手法と問題点

2.1 現在のマルチコンテキスト実現方式

マルチコンテキスト型リコンフィギャラブルプロセッサは、図 1 に示す論理構造を持つ。それぞれの PE の命令、PE 間、PE 内の接続を制御する情報を構成データと呼び、この構成データを格納するメモリを複数モジュール持つ。このメモリモジュールはマルチプレクサで PE のオペレーションやスイッチを制御する信号線に接続されており、これを 1 クロックで切り替えることで、全体の構成を変更することができる。それぞれの構成データにより実現される PE 構成のことをコンテキストと呼び、構成データの切り替えにより、これを変更する作業をコンテキストスイッチと呼ぶ。また、構成データを格納するメモリモジュールをコンテキストメモリと呼ぶ。

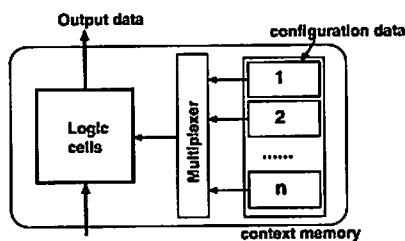


図 1 マルチコンテキスト方式の論理構造

しかし、この構成はあくまで論理的な構成であることに注意

されたい。実際には、マルチコンテキスト方式の実装は、図 2 に示す方法を取る場合が多い。すなわち、各 PE および周辺のスイッチ用の構成データは分散されてメモリに格納され、コンテキストスイッチを制御するコントローラからコンテキスト番号がブロードキャストされ、コンテキストメモリを同時に読み出し構成データを PE とスイッチに転送することで、コンテキストスイッチを行う。

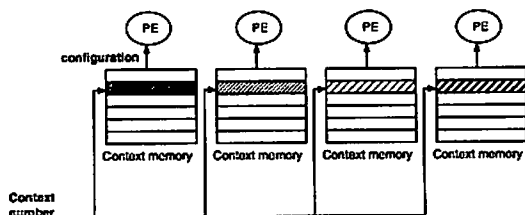


図 2 マルチコンテキスト方式の実装

この手法では、図 1 中のマルチプレクサはメモリの読み出しに置き換えられ、効率的な実装が可能となる。しかし、メモリはそれぞれの PE に分散されていることから、コンテキストメモリ量は一定の大きさに制限される。

2.2 コンテキストメモリへのデータ転送手法

それぞれの PE に分散されたコンテキストメモリに対しては、PE の位置をアドレスとして連続したアドレスを与え、バスを用いて順に構成データを転送する。転送元となる構成データメモリは、チップ内に置く場合とチップ外に置く場合があるが、高速な転送を可能とするために、チップ内に置く場合が増えている。チップ内の構成データメモリから PE アレイのコンテキストメモリに構成データを転送する手法は、PE アレイの構成に合わせて、様々な形で行われる。PACT Xpp [9] では、PE アレイの下部に構成情報メモリを置き、列単位でインターリーブして同時にデータを転送する。PipeRench [3] を商用化した Kilocore [1] も同様な方法を採用し、パイプラインの 1 ステージに当たる Stripe 全体を 1 クロックで再構成する。日立の DA-FE は、構成データメモリからの転送にバッファを設けて、データの重複を避ける工夫をしている。

2.3 対象とする動的リコンフィギャラブルプロセッサ

本稿で提案する手法は、図 2 の手法を用いているマルチコンテキスト型動的リコンフィギャラブルプロセッサすべてに対して適用可能だが、説明および評価は、NEC エレクトロニクス社の DRP [7] を基としたモデルを対象に行う。なお、DRP の命令コードの詳細は非公開である。本稿では、提案手法の効果を見積もるため、数値を仮定して議論する場合があるが、これらの議論は全て一般的なものであり、DRP に直接対応するものではない点に注意されたい。

DRP は、粗粒度のリコンフィギャラブルデバイスであり、Tile と呼ばれるリコンフィギャラブルユニットを構成単位とする。各 Tile は図 3 に示す通り、8×8 の Processing Element(PE) アレイ、状態制御を行なうシーケンサである State Transition Controller(STC) から構成される。また、8bit×256 エントリの

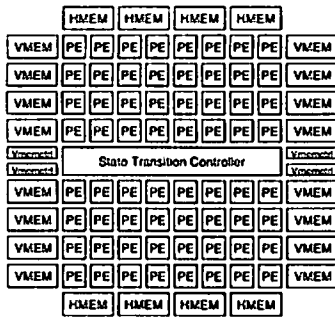


図3 Tileの構造

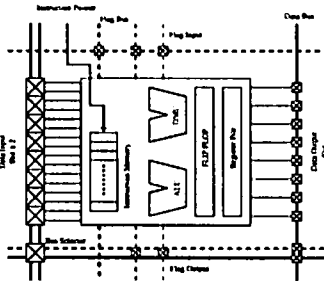


図4 PEの構造

メモリ (VMEM) 8セットとメモリコントローラ2セットをTileの左右にもち、8bit×8192 エントリのメモリ (HMEM)4 セットとコントローラをTileの上下にもつ。

各PEは図4に示すように、8bitのALU、シフトや簡単な論理演算を行なうData Management Unit(DMU)、8bitのFlip Flop、レジスタファイルから構成される。

PE間の配線は、PE内を縦横に通過するバスの交点に設けられたスイッチを介して行う。このスイッチは、当該PE内の素子とバスとを接続して、他のPEとのデータの授受に用いる他、単にデータを通過させたり、縦横のバスを相互接続することで、遡隔PE間の接続路を形成する役割も果たす。

PE内のスイッチの接続情報、ALU、DMUの機能、定数データ、レジスタファイル、フリップフロップの接続等は、構成データとしてまとめられ、図2に示す手法により16コンテキストの切り替えが実現される。ここで、DRPでは、コンテキストの制御に状態遷移を用いている。DRPはTile単位でSTCを持ち、内部に状態と、次に遷移する可能性のある状態(最大4)および条件を保持する状態表を持つ。遷移先の状態はPEからの情報によって決めることも可能であるため、多数の飛び先を持つ条件分岐に相当する状態遷移を行うことができる。ここで、DRPのコンテキスト制御の特徴は、状態とコンテキストの対応が必ずしも一対一ではない点にある。複数の状態を一つのコンテキストに割り当てることで、コンテキストを切り替えずに、コンテキスト内での逐次実行やループを含む処理の実行が可能である^(注1)。

現在のDRPのプロトタイプであるDRP-1は構成データメモリは、チップ外部に置かれ、構成データは、PCIバスインタ

フェース等を経由して、それぞれのPE内のコンテキストメモリに対して転送される。しかし、本稿では構成データメモリはチップ内を持つことを想定する。

3. 直接実行方式の提案

3.1 直接実行の必要性

多くのストリーム処理で、ジョブは連続したいくつかのタスクで構成される。動的リコンフィギャラブルシステムは、この中で計算時間を要するタスクをPEアレイ上での並列処理で高速実行することで、全体の実行時間を短縮する。しかし、タスクの中には、操作自体は複雑であるが、全体としてかかる時間は少なく、並列性もあまり高くはないものがある。このようなタスクは、動的リコンフィギャラブルシステムのPEアレイ上で処理するよりも、組み込みCPU上で実行する方が適している。そこで、PEアレイ上の分散共有メモリ上のデータをDMA等で組み込みCPU上のメモリに移動し、処理の後に再びPEアレイのメモリ上に移動する。この手法は、組み込みCPU上での処理が一定の大きさを持っていれば合理的であるが、そうでない場合は、データの移動に時間と電力を消費することになる。このような処理をPEアレイ上で簡単に行ってしまうことができれば、これらの無駄を省くことが可能となる。本来、PEアレイの処理能力は組み込みCPUよりも高いので、対象とする処理が演算が中心ならば、並列性が低くても処理速度の点でも有利になる可能性がある。しかし、このように並列性が低いタスクをPEアレイ上で実行する場合、コンテキスト数が大きくなる一方で、コンテキストメモリの無駄が大きい。このような無駄をなくする手法も提案[16]されているものの、その効果には限界がある。

3.2 直接実行のための手法

そこで、考えられるのは、構成データメモリから転送する構成データをそのままPE上の命令として受け付けて直接実行する手法である。図5にその概念を示す。構成データメモリから読み出された構成データ(命令)は、処理を必要とするPEのみに位置指定で転送され、Instruction Bufferに格納されて、コンテキストメモリ上の構成データに代って、即時に実行される。しかし、この手法は以下の問題点がある。

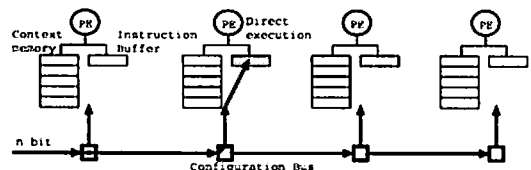


図5 直接実行の概念

- 構成データ転送用のバス幅は多くの場合、PE一個に対する命令を転送する程度である。ところが、例えばDRPにおいては8bitのPEを用いているが、32bitのデータ処理を行う場合、8bitのPE一個では単独に動作することはできず、構成データ転送用バス幅の転送能力が不足する。

(注1)：この手法をDRPでは面内多状態と呼んでいる

• 多くの PE アレイの処理では分散メモリからのデータを読み出し、いくつかの処理を行った後に別の分散メモリや別の PE のレジスタにデータを書き込む場合が多い。したがって位置的に離れた場所にある複数の PE への構成データの同時転送が必要となる。

• 構成データ転送用のバス幅は、直接実行に効果があるだけでなく、コンテキストメモリへの高速転送も可能とするため、ある程度増やすことは必要であろう。しかし、PE 全域に渡するため、コストの点で無暗に大きくすることはできない。

これらの問題点を解決するため、以下の手法を導入する。

a) テンプレートコンテキスト:

利用する PE のレジスタおよび分散メモリが決まっていれば、基本的な接続と構成をあらかじめテンプレートコンテキストとして持たせておく。そして、実行時に必要な部分の PE のみの構成データを転送する。また、PE はいくつかの構成要素から構成されるため、複数の PE にまたがって異なった部分を再構成の対象とすることも考えられる。この概念を図 6 に示す。この場合、一番下に示すコンテキストがテンプレートとなり、転送したデータと Instruction buffer 上で組み合わせる。再構成データが転送されない PE に対してはテンプレートが命令としてそのまま与えられる。この例では異なった要素、例えば ALU と DMU を別々の PE に対して再構成した場合を示す。このように再構成を柔軟にするためには要素単位でバスを分離する必要があり、アドレスバスの分配線コストが必要となる。構成データを 1 クロックで実行できない場合は、このテンプレートコンテキストをベースとして Instruction Buffer 上で複数クロックを用いてコンテキストを構成し、形成後に実行することも可能である。

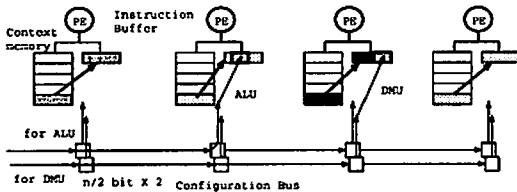


図 6 テンプレートコンテキストの概念

b) リンコンフィギャラブルなコンフィギュレーション機構:

対象とする問題が決まれば、コンフィギュレーションのパターンはある程度決まってくる。一例として、32 ビットデータを扱う場合、4つの PE を同じ演算を行って一定の桁上げ接続パターンを行う構成となる。したがって、これらの PE に対して同じ構成データをマルチキャストする手法が考えられる。組にして用いる PE はあらかじめ決まっているため、このパターンで構成データをマルチキャストする機構を備えることが可能である。これは、「コンフィギュレーション機構自体をリコンフィギャラブルにする」という概念であり、あらかじめ、コンフィギュレーション機構に構成情報を転送しておく必要がある。この概念を図 7 に示す。この例では右側の 3 つの PE が組になっており、同じデータがマルチキャストされる。このため、

構成データ転送用バスのスイッチに、マルチキャストを決定するビット列 (Reconf bit) を設ける必要がある。この Reconf bit を複数持たせて転送中に選択するために、Reconf address bus が必要となる。

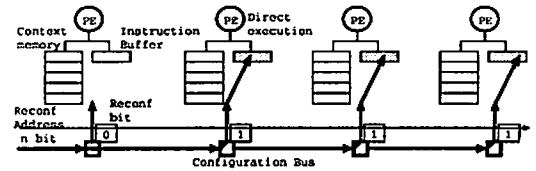


図 7 リンコンフィギャラブルなコンフィギュレーション機構

3.3 状態遷移の制御

今回、対象とした DRP ではコンテキストの切り替えに状態遷移による制御を利用している。しかし、直接実行モードは、この手法で制御すると状態数が膨大になるため、プログラムカウンタ (コンテキストカウンタ) が必要となる。状態遷移とプログラムカウンタ制御との切り替えは以下のように行う。状態の一つにプログラムカウンタ利用モード状態を設け、この状態に遷移すると、プログラムカウンタが状態遷移表中に格納されている初期値に設定される。状態遷移表は、次に遷移する状態番号を複数持っているが、ここで利用するのは、直接実行モード終了時に遷移すべき状態のみで、複数持つ必要はないため、その分のフィールドを利用可能である。図 8 に実装例を示す。この例は DRP-1 の制御と類似しており、状態遷移表は、遷移先状態を 4 つ (ns1-ns4) と、対応するコンテキスト番号 (ctx) から構成される。通常の状態遷移では、ctx をポインタとして各 PE のコンテキストメモリに送り、PE アレイからの制御信号に従って次の状態を選んで遷移することで、コンテキストスイッチを制御する。

ここで、現在の状態を保持するレジスタ state が、直接実行モードの状態に遷移すると、ctx により、実行モードであることが認識される。実行モードで送り出す状態データメモリの前頭番地は ns2-ns4 に格納されており、これが Program Counter に設定されて、実行がスタートする。

構成データメモリ中の命令は、通常の PE アレイの構成データと同一であるが、分岐命令と終了命令という二種類の特権命令を持つ。分岐命令は、飛び先アドレスが構成データ中に格納されており、分岐するかどうかの判断は、PE アレイからの制御信号によって行われる。分岐するかどうかの判断は、状態遷移の遷移先を選択する方法と同様である。すなわち、制御信号は、PE アレイ上のレジスタに格納されており、少数の信号線を用いて制御機構に送られ、判断に用いられる。直接実行モードにおける制御は簡単な条件分岐のみとし、サブルーチンコール等は持たない。

終了命令を実行すると、レジスタ state に ns1 を設定し、状態遷移によるコンテキストスイッチを再開する。

4. 評価

直接実行モードは、PE アレイの並列性を利用できないため、

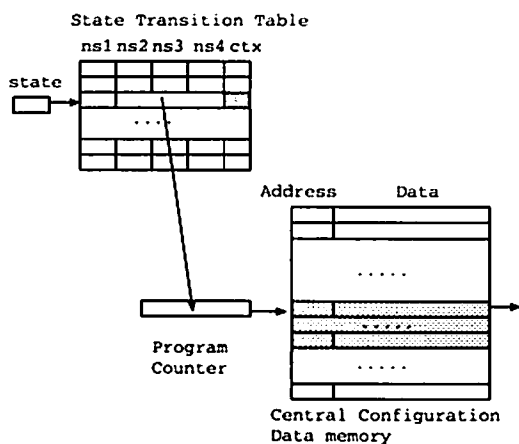


図 8 直接実行モードの制御機構

通常の実行モードより大きく性能が落ちるが、混載されている通常の組み込み CPU と同程度の性能が得られれば、無駄なデータ転送を省くという目的は達成される。

あまり並列性が高くないアプリケーションとして、JPEG における `rgb_ycc_convert` の処理を DRP-1 上に直接実行モードを想定して実装し、その性能を測定した。この処理は、テーブル引きを用いてデータを変換するもので、あまり並列性が高くなく、処理時間も大きくないことから、組み込み CPU で実行されることが多い。

表 1 `convert` 処理の実行クロック数

条件	実行クロック数	実行時間 (us)
1PE 直接実行	5542800	797
マルチキャスト	1752578	251
2PE+マルチキャスト	1180235	168
MIPS(150MHz)	2203637	162

表 2 各方式の必要資源

条件	必要バス幅	必要メモリ量
1PE 直接実行	PE_c	$65 \cdot (PE_c + 6)$
マルチキャスト	PE_c	$65 \cdot (PE_c + 6 + 64)$
2PE+マルチキャスト	$2 \cdot PE_c$	$2 \cdot 65 \cdot (PE_c + 6 + 64)$

処理を行うのに必要なクロック数を表 1 に示す。比較のため、2 命令同時実行の MIPS(NEC エレクトロニクス VR5500) における測定結果を示す。入出力に関しては評価に入っていない。

各処理方式が必要とする資源を表 2 に示す。今回のアプリケーションでは 65 命令で処理が実行可能なため、この分の命令が必要となる。ここで、1PE 直接実行は、1 PE の構成データに必要なビット幅 (PE_c) + アドレス (ここでは 1Tile 分で 64 を想定して 6bit) を一組として構成データバスを想定する。 PE_c は、32bit から 128bit 程度までで、PE アーキテクチャによっ

てかなり異なった値を取る^(注2)。DRP の PE は 8bit 構成であるのに対し、`convert` 処理は基本的に 32bit データを扱うため、単純な直接実行では、全く処理を行うことができない。そこで、テンプレート方式を想定し、複数クロックをかけて構成データを作って実行する方式を採用した。このため実行クロック数が相当大きくなっている。

これに対して「マルチキャスト」は、構成データバスのスイッチにテーブルを持たせて再構成可能とし、同一の構成データの PE を組として、構成データを同時転送した場合を示す。32bit データを扱うため、レジスタや ALU などは組みとなって用いられる場合が多く、この方法はかなり実行クロック数を節減することができる。しかし、今回のアプリケーションは、PE の組み合わせについての再利用がほとんどできず、スイッチのテーブルは必要な直接実行命令数分必要とする。

最後に 2PE+マルチキャストは、構成データバスを 2PE 分に拡張すると共に、マルチキャストも可能とした構成である。前者二つに対して、最後の構成は構成データバス幅が 2 倍となる点でコストの増加が大きいが、その分実行クロック数を削減できている。

今回実装した設計は 68MHz で動作可能である。MIPS との実行結果と比較すると、2PE+マルチキャストの実行時間が、MIPS が 150MHz で動作して、50MHz で入出力データの転送したと仮定した場合の実行時間とほぼ等しくなることがわかる。

ただし、今回の測定データは、直接実行モードのために意識して命令を構成していないことからあくまで予備的な評価である点に注意されたい。

5. 関連研究

直接実行に近い手法をサポートしているアーキテクチャとして IMEC による ADRES [2] がある。ADRES は、PE アレイのうちもっともレジスタファイルに近い一行が `VIEW` モードとして、直接命令を実行することが可能である。ADRES と本稿の方式との違いは、PE アレイ内部でも直接実行が可能な点である。また、Stretch [6] を代表とする拡張命令タイプの動的リコンフィギュラブルプロセッサは、基本的な実行モードが本稿における直接実行に近い考え方であると考えられる。しかし、これらのプロセッサでは、PE アレイ自体が小さく、PE アレイ全体として動作する場合の並列性が制限されている。

構成データをマルチキャストすることにより高速化を行う手法は一部本稿で提案の手法と類似している。MorphoSys [5] は、同一の構成データを PE の行あるいは列単位にブロードキャストする。しかし、この手法は SIMD に限られる。RoMultiC [10] は、より柔軟な構成データのマルチキャストを可能とするが、基本的にこれらの方法は、行、列単位のデータマルチキャストのみ可能である。

本稿で提案するテンプレート方式は、構成データの一部のみを変更する点で差分変更法 [4] [12] およびテンプレート再構成法 [16] と似ている。しかし、差分変更法は、構成データ自体の

(注2) : DRP-1 の命令セットおよび PE 当りの必要ビット数は非公開である

変更が目的であるのに対して、本稿の手法はテンプレートと組み合わせてその場で実行する命令を生成する点で異なる。また、テンプレート再構成法は、テンプレートの利用が PE 単位であるのに対して本稿の手法は PE アレイ上でグローバルにテンプレートを構成する点で異なる。

6. おわりに

直接実行モードを提案し、簡単なアプリケーションでその効果を予備評価した。その結果、構成データバス幅を 2PE 分にし、構成データのマルチキャストを実行した場合、MIPS 150MHz の組み込み CPU にデータを転送して実行するのと同程度の性能が確認できた。しかし、今回の予備評価は、単一のアプリケーションであり、より広範かつ正確な評価が必要である。また、直接実行モードは 8bit の PE よりも、粒度の大きい PE に適していると考えられる。対象とするアーキテクチャについても様々なものを想定して評価する必要がある。

また、構成データのマルチキャスト法など、より効率の良い手法について検討して行く予定である。

謝 辞

本研究を進めるにあたり DRP-1 とその開発環境は 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に提供して頂きました。デバイスおよび開発環境について、日頃から有益な御助言、御指導を頂きました事に著者一同深く感謝致します。

文 献

- [1] B.Levine. Kilocore: Scalable, High Performance and Power Efficient Coarse Grained Reconfigurable Fabrics. In Proc. of International Symposium on Advanced Reconfigurable Systems, pages 129-158, 2005.
- [2] F.J.Veradas, M.Scheppler, W.Moffat, B.Mei. Custom Implementation of the Coarse-Grained Reconfigurable ADRES architecture for multimedia Purposes. In Proc. of International Conference on Field Programmable Logic and Applications (FPL05)/, pages 106-111.
- [3] H. Shimit, D. Whelihan, A.Tsai, M.Moe, B.Levine, R.Taylor. PipeRench: A virtualized programmable datapath in 0.18 micron technology. In Proc. of CICC, pages 63-66, 2002.
- [4] H.Amano, T.Inuo, H.Kami, T.Fujii, M.Suzuki. Techniques for Virtual Hardware on a Dynamically Reconfigurable Processor - An approach to Tough Cases. In Proc. of the FPL 2004, pages 464-473.
- [5] H.Singh, M-H.Lee, G.Lu, F.J.Kurdahi, N.Bagherzadeh, Em.Chaves. MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications. In IEEE Trans. on Computers, pages 465-480.
- [6] J.M.Arnold. S5: The Architecture and Development Flow of a Software Configurable Processor. In Proc. of IEEE FPT, pages 121-128, 2005.
- [7] M. Motomura. A Dynamically Reconfigurable Processor Architecture. Microprocessor Forum, Oct. 2002.
- [8] T.Kitaoka, H.Amano, K.Anjo. Reducing the configuration loading time of a coarse grain multicontext reconfigurable device.
- [9] V. Baumgarte, F. May, A. Nückel, M. Vorbach, and M. Weinhardt. PACT XPP - A Self-Reconfigurable Data Processing Architecture. In Proc. of International Conference on Engineering of Reconfigurable Systems and Algorithms(ERSA'01), 2001.
- [10] V.Tanunheng, M.Suzuki, H.Amano. RoMultiC: Fast and Simple Configuration Data Multicasting Scheme for Coarse Grain Reconfigurable Devices. In Proc. of IEEE FPT, pages 129-136.
- [11] X.-P. Ling, H. Amano. WASMII: A Data Driven Computer on a Virtual Hardware. In Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines(FCCM'93), pages 33-42.
- [12] 川崎, 谷川, 弘中, 小島. “再構成情報圧縮機構を備えた粗粒度再構成アーキテクチャ”. pages 95-102. 第 1 回リコンフィギャラブルシステム研究会, 2003 年 9 月.
- [13] 津野田, 高田, 秋田, 田中, 佐藤, 伊藤. デジタルメディア向け再構成可能プロセッサ FE-GA の概要. In 信学報 RECONF2005-65 (デザインガイア 2005)/, pages 37-42.
- [14] 天野, 阿部, 出口. “動的リコンフィギャラブルプロセッサの基本的トレードオフの解析”. 第 4 回リコンフィギャラブルシステム研究会, 2004 年 9 月.
- [15] 末吉敏則, 天野英晴 編. リコンフィギャラブルシステム. オーム社, 2005.
- [16] 鈴木正康, 天野英晴. テンプレート再構成法の提案. In 信学技報 RECONF2005-24, pages 55-60, 2005.