社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＳＬＤＭ－128（18）
2007／1／18

マルチキャストコンフィギュレーションのスケジューリングアルゴリズム

堤 聡† Vasutan Tunbunheng† 長谷川 揚平† 松谷 宏紀† Adepu Parimala†
中村 拓郎† 西村 隆† 佐野 徹† 加東 勝† 斎藤 正太郎†
関 直臣† 平井 啓一郎† 毛 凱毅† 天野 英晴†

† 慶應義塾大学理工学部
〒223-8522 横浜市港北区日吉 3-14-1
E-mail: muccra@am.ics.keio.ac.jp

あらまし 動的リコンフィギャラブルプロセッサを用いた組み込みシステムにおいて，より多くのアプリケーションを搭載するためには，コンフィギュレーションの高速化が重要である．このために我々は，RoMultiC と呼ばれるマルチキャストによるコンフィギュレーションデータの配送手法を提案している．RoMultiC では，あらかじめコンフィギュレーションのスケジューリングを行う必要があり，この最適解を得るには膨大な計算量を必要とする．そこで，本研究では，このコンフィギュレーションスケジューリングアルゴリズムについて検討を行い，3 つの近似アルゴリズムを提案する．評価の結果，単純な総当たりの組み合わせ探索では時間がかかりすぎるスケジューリングを，これらのアルゴリズムを用いることにより，現実的な時間で行えることがわかった．また，RoMultiC の，後からコンフィギュレーションしたデータが有効になる特徴を活かしたスケジューリングを行うことで，8×8 アレイでは平均で最大 32%のコンフィギュレーションサイクルを削減できた．

キーワード 動的リコンフィギャラブルプロセッサ, コンフィギュレーション高速化, 動的再構成

Scheduling Algorithms for Multicast Configuration

Satoshi TSUTSUMI†, Vasutan TUNBUNHENG†, Yohei HASEGAWA†, Hiroki MATSUTANI†, Adepu PARIMALA†, Takuro NAKAMURA†, Takashi NISHIMURA†, Toru SANO†, Masaru KATO†, Shotaro SAITO†, Naomi SEKI†, Keiichiro HIRAI†, Mao KAIYI†, and Hideharu AMANO†

† Department of Information and Computer Science, Keio University
3-14-1 Hiyoshi, Yokohama, 223-8522 Japan
E-mail: muccra@am.ics.keio.ac.jp

Abstract Techniques for high speed configuration data delivery are essential to accommodate a variety of applications. RoMultiC is a novel reconfiguration mechanism well suited for dynamically reconfigurable processors. It needs configuration schedulings which require a great amount of time to solve. In this study, we propose three approximation algorithms for scheduling problems. They can complete configuration schedulings with realistic time which brute force search cannot compute, and they are possible to reduce configuration cycles maximum 32% with array size of 8 × 8 using overwriting rule of RoMultiC.

Key words Dynamically Reconfigurable Processor, High speed configuration, Dynamic Reconfiguration

1. はじめに

近年，動的リコンフィギャラブルプロセッサの開発が盛んに行われ，多くのアーキテクチャが発表されてきた [7]．これらのプロセッサは，メディア処理，通信処理，暗号処理など多くのアプリケーションを搭載する組み込みシステムをターゲットとしており，複数のタスクを高速に切り替える必要がある．

アーキテクチャ毎にタスクを格納するコンフィギュレーショ

ンメモリの配置，コンフィギュレーション方式は様々である．我々は特に，複数のコンフィギュレーションデータのセット (コンテキスト) を持つ，マルチコンテキスト型のアーキテクチャに注目をしている [2]．この方式では，Processing Element (PE) 毎にコンテキストメモリを持ち，コンテキストを 1 クロックで切り替えることができる．しかし，すべてのタスクを格納できるほどのコンテキストメモリを PE 毎に持たせるのは難しく，必要に応じて，より大きなバックアップ用の集中コンテキストメモリから PE のコンテキストメモリに新しいタスクを転送することが必要となる．その結果，マルチコンテキスト方式においても，再構成の時間はシステムの性能のボトルネックになることが多く，高速なコンフィギュレーション機構の開発は重要である．

そこで我々は，これまでにマルチキャストコンフィギュレーション方式である RoMultiC [5] を提案している．RoMultiC は，並列性の高いアプリケーションには，同じコンフィギュレーションデータを持つ PE が複数存在することを利用して，これらをマルチキャストすることで，転送時間の短縮，および集中コンテキストメモリの削減を実現することができる．RoMultiC は PE アレイに対して 2 次元状のビットマップを用意して，長方形状の PE 領域に対してコンフィギュレーションデータをマルチキャストすることができる．

RoMultiC では，後からコンフィギュレーションしたデータが有効になるという点を利用してコンフィギュレーションのステップを削減することができる．しかし，コンフィギュレーションのスケジューリング方法は，膨大な組み合わせがあり，これを単純な総当たりの方法で求めることは困難である．

RoMultiC は開発中の動的リコンフィギャラブルプロセッサ Multi-Core Configurable Reconfigurable Array (MuCCRA) [6] で採用されている．

そこで，本研究では，比較的短時間でスケジューリングを行える 3 つのアルゴリズムについて検討を行い，評価によりその有用性について考察した．また，実際にこれらのアルゴリズムを搭載した MuCCRA のアプリケーション開発環境について紹介する．

2. コンフィギュレーション機構

2.1 関連研究

動的リコンフィギャラブルプロセッサの多くは，コンフィギュレーションメモリにシーケンシャルなアドレスを割り当て，そのアドレスに順にコンフィギュレーションデータを転送する方式を取っている．この場合，コンフィギュレーションはコンフィギュレーションデータバスのバンド幅と同じ速度である．関連研究として，コンフィギュレーション方法が異なる 3 つのアーキテクチャについて述べる．

NEC エレクトロニクス社が開発した Dynaically Reconfigurable Processor (DRP) [2] は，コンフィギュレーションデータのセットを複数持つマルチコンテキスト型デバイスであり，コンテキストメモリを PE 毎に分散して持つ．コンテキストメモリにはアドレスが割り当てられており，コンフィギュレーションデータはコンフィギュレーションバスを通して，指定されたアドレスに書き込まれる．

PACT XPP Technologies 社の XPP [3] は，2 次元構造の PE アレイを持つ粗粒度リコンフィギャラブルプロセッサである．コンフィギュレーションデータは，PE から独立したメモリモジュールから順に PE アレイに配送する．複数のメモリモジュールを持ち，並列にコンフィギュレーションデータを転送することができる．再構成トークンにより演算中にも利用していないプロセッサを再構成することができる．

MorphoSys [1] は，リコンフィギャラブルプロセッシングユニット (RC Array) と汎用の RISC プロセッサ (TinyRISC) を持つリコンフィギャラブルプロセッサである．MorphoSys はアプリケーションのデータ並列性を利用した SIMD モデルの RC Array を持つことに特徴がある．これにより，コンフィギュレーションを，行もしくは列すべての PE に対して同時に行うことができ，コンフィギュレーション時間を短縮することができる．

これら 3 つのデバイスは，転送方法は異なるものの，いずれも転送順序はコンフィギュレーションサイクルに影響を与えない．

2.2 RoMultiC

RoMultiC は，PE アレイ上の PE およびスイッチにシーケンシャルなアドレスを与え，これを指定してコンフィギュレーションデータを送るのではなく，図 1 に示すように，縦横のビットマップを用いる方法である．共通バス上のコンフィギュレーションデータは縦横のビットマップが共に'1' である PE あるいはスイッチのコンテキストメモリに送られる．

この手法はマルチキャストにより，コンフィギュレーションに要する時間を大幅に減らすことが可能である．転送用のビットマップの量は，PE およびスイッチにシーケンシャルなアドレスを振るよりも大きい．そのため，この分集中コンテキストメモリが大きくなるような印象を受けるが，実際には，マルチキャストするデータは共有されるため，集中コンテキストメモリに格納されるコンフィギュレーションデータの量は，全てのコンフィギュレーションデータを持たせる通常の方法に比べて激減する．このコンフィギュレーションデータの量の削減は，転送用ビットマップの分の増加をはるかに上回る効果がある．

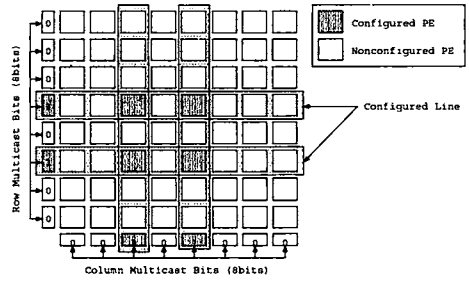

図 1 RoMultiC マルチキャスト方式

マルチキャストされる範囲は長方形状に限られるが，後で送ったデータが有効になるため，送る順番を工夫すれば，複雑なパターンにも対応可能である．次節では，この点を利用してコンフィギュレーションサイクルを削減するアルゴリズムについて述べる．

3. スケジューリングアルゴリズム

3.1 コンフィギュレーションマスクパターン

コンフィギュレーションのスケジューリングについて説明するために，$M \times N$ アレイを k 種類のデータでコンフィギュレーションする場合を想定する．

RoMultiC では Row Multicast Bit と Column Multicast Bit が共に'1' の対象 (セル) がコンフィギュレーションされる．ここで言うセルは，PE やスイッチに対応する．このコンフィギュレーションするパターンをコンフィギュレーションマスクパターンと呼ぶことにする．コンフィギュレーションマスクパターンの総数 N_{mask} は，

$$N_{mask} = k \times (2^M - 1) \times (2^N - 1)$$

となる．すべてのコンフィギュレーションは，これらパターンのいずれかで行われ，その組み合わせによって最終的なコンフィギュレーションパターンを作りだすことになる．これは，適切なコンフィギュレーションマスクパターンを選択し，それらをコンフィギュレーションする順番に上に重ねていくことに対応する．その際，コンフィギュレーションマスクパターンにおいて，Row Multicast Bit もしくは Column Multicast Bit が'0' のセルは下のパターンが透過して見えることになる (以後このセルをウィンドウセルと呼ぶ)．

3.2 オーバライトコンフィギュレーション

RoMultiC では，後にコンフィギュレーションしたデータが有効になる．この特徴を利用することで，複数のコンフィギュレーションマスクパターンを一つのコンフィギュレーションマスクパターンで置き換えることができる．

オーバライトコンフィギュレーションは，コンフィギュレーションマスクパターンを重ねた際に，上のコンフィギュレーションマスクパターンによって隠された下のコンフィギュレーションマスクパターンの一部が上から見たときに見えなくなることに相当する．

3.3 最適解の探索空間

$M \times N$ アレイに対し，$k(\leq M \times N)$ 種類のデータによるコンフィギュレーションパターンが与えられたとする．ここでは，それをいくつのコンフィギュレーションマスクパターンの重ね方で作り出せるかを考える．コンフィギュレーションマスクパターンの数は，コンフィギュレーションのステップ数に相当するため，以後これをステップ数と呼ぶ．ステップ数の最小値は，1 種類のデータを 1 つのコンフィギュレーションマスクパターンのみでコンフィギュレーションする k ステップである．ステップ数の最大値は，すべてのセルを 1 つずつコンフィギュレーションする $M \times N$ ステップである．

単純にすべてのコンフィギュレーションマスクパターンを探索する場合は，求める最小ステップ数を S_{min} とすると，その組み合わせの総数 T は，

$$T = S_{min}! \times \prod_{i=1}^{k} {}_P C_{n_i},$$

$$(P = (2^M - 1) \times (2^N - 1), S_{min} = \sum_{i=1}^{k} n_i, n_i > 0)$$

となる．時間計算量は $O(k! \times 2^{k(M+N)})$ 以上となり，小さなアレイサイズでも現実的な時間で組み合わせを求めることは難しい．そのため本研究では評価を行っていない．

最適解を実用的なアレイサイズで求めることは困難であり，近似アルゴリズムが必要となる．そこで本研究では，次に述べる 3 つのスケジューリングアルゴリズムを提案する．

3.4 パターン分割スケジューリング

パターン分割スケジューリングでは，パターンを RoMultiC でコンフィギュレーション可能な，できる限り大きなパターンに分割する．これは，コンフィギュレーションパターンから，最大に一致するコンフィギュレーションマスクパターンを順に取り除いていくという，グリーディ集合カバーアルゴリズム [4] による近似解として求められる．この分割は，$O(M \times N \times 2^{(M+N)})$ 以下で完了する．

図 2(a) は 3×3 アレイで 2 種類のコンフィギュレーションデータが存在する状態である．これを分割して得られたコンフィギュレーションマスクパターンが，図 2(b)(c)(d)(e) となる．これらのコンフィギュレーションマスクパターンの間には，重なりはない．

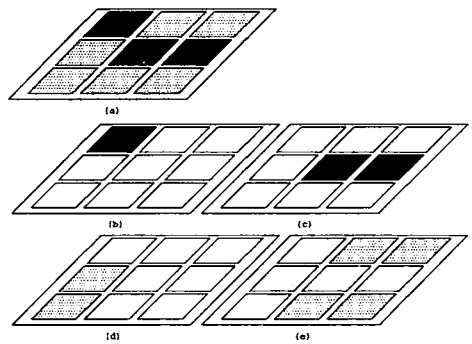

図 2 コンフィギュレーションパターンの分割

3.5 パターン結合スケジューリング

パターン結合スケジューリングでは，コンフィギュレーションデータの種類毎に，基本となるコンフィギュレーションマスクパターンの組み合わせを求め，オーバライトコンフィギュレーションを利用して，コンフィギュレーションのステップ数を削減する．このパターン結合スケジューリングには，基本パターンの選択方法の違いにより 2 種類ある．

3.5.1 基本パターンの選択

まず，結合するコンフィギュレーションマスクパターンの組み合わせを求める．ここでは，先程パターン分割スケジューリングで述べた最大に一致するコンフィギュレーションマスクパターンを除いて行くことで得られる組み合わせ **(分割サイズ大)** と，セルが 1 つのみの最小のコンフィギュレーションマスクパターンからなる組み合わせ **(分割サイズ最小)** を考える．

3.5.2 コンフィギュレーションマスクパターンの結合

図 3(a) は図 2(b)(c)(d)(e) のコンフィギュレーションマスクパターンを重ね合わせた様子を示している．

コンフィギュレーションマスクパターンの結合は，結合する

コンフィギュレーションマスクパターンが無くなるまで次の手順を繰り返す.

(1) すべてのコンフィギュレーションマスクパターンを重ね合わせ，上から見たパターンを投影する.

(2) 投影されたパターンにおいて，最も多くのコンフィギュレーションマスクパターンを完全に含み，かつ，ウィンドウセルに重ならないコンフィギュレーションマスクパターン X を選択する.

(3) X に完全に含まれるコンフィギュレーションマスクパターンを除く.

(4) X を一番下に挿入する.

この結合は，分割した結果のコンフィギュレーションマスクパターン数 n に対して，$O(k \times n \times 2^{(M+N)})$ 以下で完了する.

図 3(b) は，図 3(a) に対してこの手順を一度行い，一番下のコンフィギュレーションマスクパターンとその 1 つ上のコンフィギュレーションマスクパターンが結合した状態である.

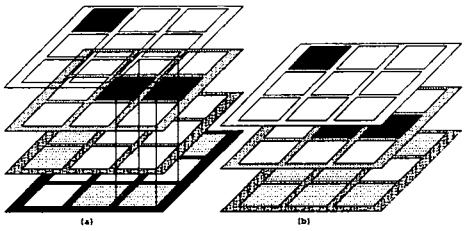

図 3 コンフィギュレーションマスクパターンの結合

4. MuCCRA

4.1 MuCCRA の概要

MuCCRA プロジェクトは，コンフィギャラブルな低電力マルチコア動的リコンフィギャラブルプロセッサに関するアーキテクチャ技術をチップレベルから提案，開発，解析することを目的としている．MuCCRA のコンフィギュレーション機構には，RoMultiC を採用しており，前述のコンフィギュレーションスケジューリングアルゴリズムを，MuCCRA のアプリケーション開発環境 MuCCRA-editor に搭載している.

MuCCRA-1 は MuCCRA アーキテクチャのプロトタイプであり，ローム社の 0.18um プロセスを用いて 5mm 角のダイ上に 4 × 4 の 24bit PE アレイ，乗算器 4，分散メモリ 4 を実装している．PE アレイは典型的なアイランドスタイルの構造を持ち，性能とコストのトレードオフ，電力モデルの構築等に用いることができる.

4.2 PE アレイ

図 4 に示すように 4 × 4 の PE(ここでは EXPE) の左端および下端にそれぞれ乗算器 4 個，分散メモリを 4 個接続した構造を持っている．それぞれの PE は 24bit の ALU，DMU，レジスタファイルから構成される．分散メモリは，$24bit \times 256$ エントリで，ダブルバッファとして 2 セット実装されている．コンテキスト数は 64 とし，集中コンテキストメモリ上には 512 コンフィギュレーションセット分のコンフィギュレーションデータが格納され，RoMultiC によって PE，スイッチなどのコンテキストメモリにマルチキャストされる．PE アレイ間の結合網は，FPGA 同様のアイランドスタイルを採用している．すなわち，配線領域間にスイッチ (EXSW) を設けて縦横のネットワークを構成し，インターコネクトモジュールによって，PE の入力と出力を接続する.

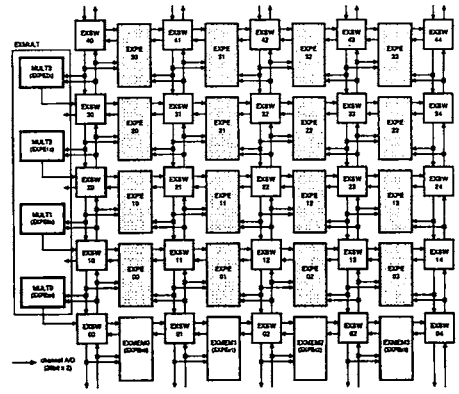

図 4 MuCCRA-1 の PE アレイ構成

4.3 コンフィギュレーション機構

他の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同様，MuCCRA は，各 PE，スイッチに小規模のコンテキストメモリを分散して持つ．チップ上には，それぞれのコンテキストメモリのバックアップ用の集中コンテキストメモリを持ち，立ち上げ時に，チップ外部よりこの集中コンテキストメモリに対して，全ての PE およびスイッチで利用するコンフィギュレーションデータを転送する．実行開始後は必要に応じてこの集中コンテキストメモリからそれぞれの PE，スイッチのコンテキストメモリに対してコンフィギュレーションデータを転送する．チップ面積を節約する必要がある場合は，この集中コンテキストメモリはチップ外に出すことも可能である．集中コンテキストメモリから，それぞれの PE，スイッチのコンテキストメモリへのデータ転送基本的にはバスを用いて順番に行うが，ここで，RoMultiC を利用することでマルチキャストによる高速化を実現する．スイッチには 3 セットの Multicast Bit およびコンフィギュレーションデータバスを設けてあり，3 並列でコンフィギュレーションを行うことができる.

4.4 アプリケーション開発環境

図 5 は，MuCCRA のアプリケーション開発環境 MuCCRA-editor の GUI である．アプリケーションの開発は，ツール上でマウス操作により，PE の機能，スイッチ間のルーティングを選択することで進める．すべての選択が完了した後，MuCCRA-1 の Verilog-HDL シミュレーションモデルで読み込むコンフィギュレーションデータファイルを出力することができる．この際に，前章で述べたパターン結合スケジューリングアルゴリズムにしたがって，コンフィギュレーションのスケジューリングが行われる.

5. 評　　価

評価では，3 章で提案した次の 3 つのアルゴリズムについて，ステップ数，スケジューリング計算時間を比較する.

(a) パターン分割スケジューリング

(b) パターン結合スケジューリング (分割サイズ大)

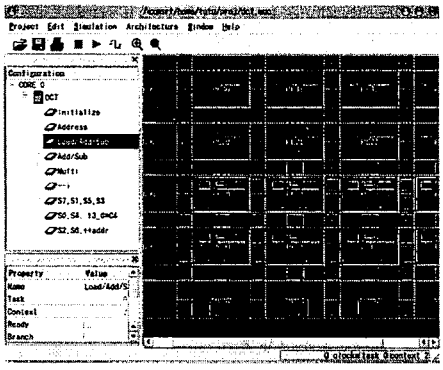

図 5　アプリケーション開発環境 MuCCRA-editor

(c)　パターン結合スケジューリング (分割サイズ最小)

ステップ数，スケジューリング計算時間は，それぞれ 4×4，6×6，8×8 のアレイを想定して，ランダムなコンフィギュレーションパターンをスケジューリングすることで求めた．コンフィギュレーションパターンは，アレイサイズ N に対して，$2 \sim N-1$ 種類のコンフィギュレーションデータを用いて生成している．それぞれのアレイサイズ，コンフィギュレーションデータ数に対して，4×4，6×6 のアレイには 1000 通り，8×8 のアレイには 100 通りの配置パターンをランダムに生成している．

9×9 以上のアレイについては，現実的な時間で評価が終了しないため，評価を行っていないが，10×10 までのサイズであれば，条件によっては数十秒から数分でスケジューリングが可能である．また，10×10 より大きなアレイは，Multicast Bit 幅が増大し，集中コンテキストメモリのオーバヘッドが大きくなるため，現段階では，最大 10×10 のアレイがコンフィギュレーションできれば十分であると考えている．

最後に，実際のアプリケーションにおけるステップ数の削減効果について，JPEG デコーダの DCT と 2×2 の行列積を求めるアプリケーションを用いて評価した．

スケジューリングを行うプログラムは C++を用いて，gcc-3.4.6 (-O3) によりコンパイルした．評価に使用した環境は表 1 の通りである．

表 1　評 価 環 境

CPU	Athlon 64 FX-57 (2.8GHz)
Memory	2GB
OS	Linux kernel-2.6.9 (64bit)

5.1　オーバライトスケジューリングによるステップ数削減

図 6 は，アルゴリズム (a) の総ステップ数を 100%として，アルゴリズム (b)，アルゴリズム (c) の平均ステップ数削減率を示している．

この図より，オーバライトコンフィギュレーションを利用したアルゴリズム (c) を用いることで，コンフィギュレーションステップ数を最大 68%まで削減できることがわかる．

図 7 は，アルゴリズム (b)，アルゴリズム (c) によるコンフィギュレーションデータの数 (k) と平均ステップ削減率の関係を示している．この図より，すべてのアレイサイズにおいて，コンフィギュレーションデータの数が増えるにつれて減少していることがわかる．また，すべてのアレイサイズにおいてアルゴリズム (c) がもっとも削減率が高いことがわかる．

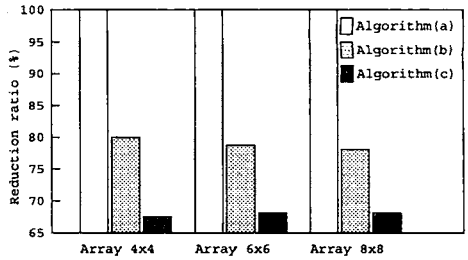

図 6　オーバライトスケジューリングによる平均ステップ数の減少率

これらは，ランダムなパターンにおける平均ステップ削減率を示しているが，実際のアプリケーションでは，RoMultiC が有効に働くマッピングを行うことで，さらにステップ数を削減できる可能性がある．

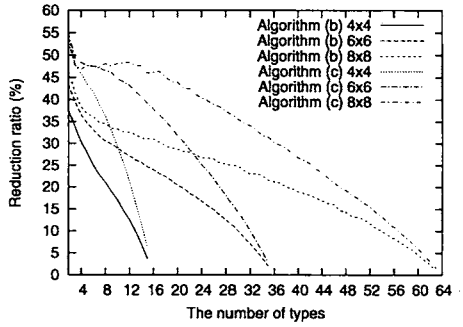

図 7　データ数とステップ数の削減率

5.2　コンフィギュレーションスケジューリングに要する時間

図 8 は，アレイサイズとコンフィギュレーションスケジューリングに要する平均時間との関係を示している．

図 9〜図 11 は，各アレイサイズにおけるコンフィギュレーションデータの数 (k) と平均スケジューリング時間の関係を示している．図 9 でアルゴリズム (a) のグラフが見えないのは，測定精度が ms 以下のためである．

すべてのアレイサイズにおいて，アルゴリズム (a) はデータ数の増加とともに，スケジューリング時間が減少している．こ

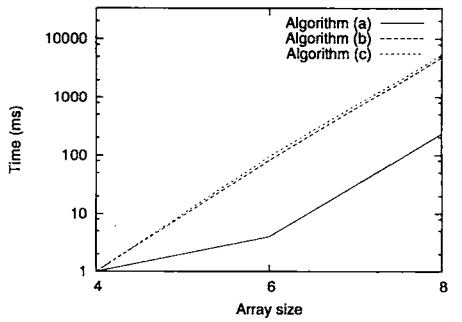

図 8　スケジューリング時間

れは，データ数が増えるにつれて，コンフィギュレーションパターンから除いていく，最大に一致するコンフィギュレーションマスクパターンの選択肢が減っていくためである．

また，図9〜図11において，アルゴリズム (b)，アルゴリズム (c) は，ある特定のデータ数まで時間が増加した後に減少に転じている．

5.3 ケーススタディ

表2は，DCT，2×2の行列積計算をMuCCRA-1のアプリケーションとして実装し，スケジューリングしたステップ数の結果である．今回の実装では，DCTを行方向演算，行列転置，列方向演算の3つのタスクに分割した．これらのアプリケーションのマッピングはMuCCRA-editorにより手動で行っている．

これより，コンフィギュレーションステップ数は，DCT全体，2×2の行列積でそれぞれ最大11.2%，22.8%ずつ削減されていることがわかる．

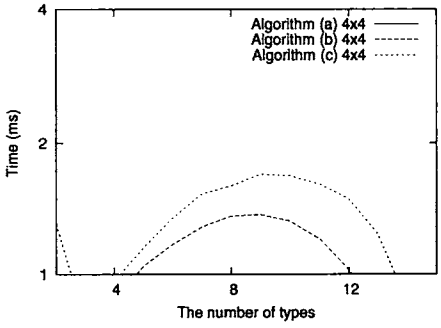

図9 データ数とスケジューリング時間 (4 × 4)

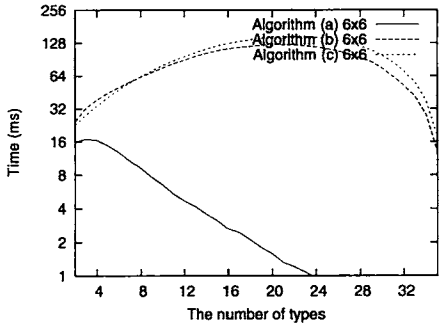

図10 データ数とスケジューリング時間 (6 × 6)

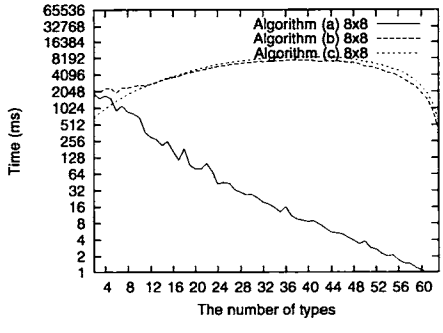

図11 データ数とスケジューリング時間 (8 × 8)

表2 アプリケーションのスケジューリング結果

アプリケーション	(a)	(b)	(c)
DCT 行方向	158	136	135
DCT 転置	193	182	182
DCT 列方向	158	136	135
2 × 2 行列積	57	44	44

6. おわりに

本研究では，マルチキャストコンフィギュレーション手法であるRoMultiCのスケジューリングアルゴリズムについて検討を行い，3つの近似アルゴリズムを提案した．

最小ステップを単純に求めるアルゴリズムでは，小さいアレイサイズにおいても現実的な時間で完了しないスケジューリングを，これらのアルゴリズムを用いることにより，現実的な時間で行えることがわかった．8×8アレイでは，スケジューリングを平均5.4秒で行え，また，オーバライトコンフィギュレーションを利用することで，平均で最大32%コンフィギュレーションサイクルを削減することができた．

今後の課題として，

- スケジューリングアルゴリズムの近似率の考察
- オーバライトする際の電力オーバヘッド
- RoMultiCが効率的に機能する配置配線アルゴリズム

の検討が挙げられる．

謝　辞

本研究は，科学技術振興機構「JST」の戦略的創造研究推進事業「CREST」における研究領域「情報システムの超低消費電力化を目指した技術革新と統合化技術」の研究課題「革新的電源制御による次世代超低電力高性能システムLSIの研究」による．

本研究は東京大学大規模集積システム設計教育研究センターを通し，ローム (株)・凸版印刷 (株)・シノプシス株式会社・日本ケイデンス株式会社・メンター株式会社の協力で行なわれたものである．

文　献

[1] 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. pages 465–480, Vol.49, No. 5, 2000.

[2] M. Motomura. A dynamically reconfigurable processor architecture. Microprocessor Forum, Oct. 2002.

[3] 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.

[4] Vijay V. Vazirani(原著), 浅野 孝夫 (翻訳), editor. **近似アルゴリズム**. Springer Japan, 2002.

[5] V.Tunbunheng, 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, 2005.

[6] 天野，長谷川，中村，西村，Vasutan. 動的リコンフィギャラブルプロセッサMuCCRAのコンフィギュレーション機構. **電子情報通信学会技術研究報告** *RECONF2006-30*, 106(247):19–24, Sep. 2006.

[7] 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.