# 再構成回数削減による動的リコンフィギャラブルプロセッサの消費電力削減手法の提案

木村優之[†]　弘中和衛[†]　天野英晴[†]

本論文では動的リコンフィギャラブルデバイスのアーキテクチャを変更することなく電力を削減するためのマッピング手法を提案する．あらかじめ各再構成ユニットが同じ命令を実行するように配置を変更し，再構成回数を抑制することで，再構成ユニットで消費される電力オーバヘッドを削減する．提案アルゴリズムの効果を評価するために，提案手法を適用したコンパイラを開発し，本研究室で開発された動的リコンフィギャラブルプロセッサを用いて性能，構成情報サイズ，実行時消費電力をシミュレーションにより調査した．その結果，アプリケーションの実行時間を増加させることなく，消費電力を平均で 10%削減することができた．

# Power reduction for Dynamically Reconfigurable Processor Array with reducing the number of reconfiguration

Masayuki Kimura,[†] Kazuei Hironaka[†] and Hideharu Amano[†]

A power consumption centric assignment algorithm is proposed for dynamically reconfigurable processors. By assigning the same operations into the same PE (Processing Element) as far as possible, the number of changing configuration data for dynamic reconfiguration can be reduced, and so redundant power consumption for changing the configuration is also reduced as a result. The proposed algorithm is implemented in a compiler for a research dynamically reconfigurable processor MuCCRA-3, and its evaluation results showed that the power consumption was reduced by 10% in average without increasing the execution time.

## 1. はじめに

近年のモバイル機器の普及に伴い，組み込みデバイスには性能向上，低消費電力化に加え，開発期間の短縮，柔軟性がますます要求されるようになっている．これらの要求を満足するために，専用ハードウェアに代わるオフロードエンジンとして，動的リコンフィギャラブルプロセッサ (Dynamically Reconfigurable Processor Array, DRPA) が注目されている．すでに商用化もされており，その例として SONY の VME[1)]，NEC の STP エンジン[2)]，Panasonic の D-Fabrix[3)] などが挙げられる．

一般的な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は単純な演算器やレジスタファイルをひとまとめにした Processing Element(PE) をアレイ状に複数個並べた構成を取っている．各 PE の演算命令や PE 間結合網を決める構成情報 (コンフィギュレーションデータ) を複数セット保持し (コンテキストと呼ぶ)，これを実行時に高速に切り替えることでアプリケーションを実行する．

DRPA の演算単位は粗粒度であるため，専用ハードウェアや FPGA などの細粒度のデバイスに比べて C 言語などの高水準言語でのアプリケーション記述に適している．高水準言語で記述されたプログラムから構成情報を生成するためには，以下のような手順を踏む場合が多い．

(1) プログラムの構文解析，意味解析を行い，アセンブリ命令列を生成する．
(2) 中間命令表現のデータフローグラフ，制御フローグラフを生成する．その結果に基づきスケジューリングを行い，各コンテキストで実行すべき命令を決定する．
(3) 命令のマッピングを行う．PE 不足などでマッピングが出来ない場合は，(2) まで戻ってスケジューリングをやり直す．
(4) データフローグラフに基づきルーティングを行う．配線資源不足でルーティングが出来ない場合は，(2) まで戻ってスケジューリングをやり直す．
(5) DRPA で実行可能な構成情報を生成する．

これらの処理を人間の手で行うことは困難であるため，動的リコンフィギャラブルプロセッサのアプリケーションの開発にはコンパイラが必須である．このため，様々なコンパイラが提案され，性能を向上し，コンテキスト数や利用 PE 数を減らすためのマッピング手法などが検討されてきた[4)5)6)7)]．

動的リコンフィギャラブルプロセッサの利点の一つは，その消費電力が小さいことである．これは問題の解法アルゴリズムを直接 PE アレイ上で実行できるためで，この利点をさらに生かすため，様々な解析や構成上の工夫がなされている[8)]．この結果，動的リコンフィギャラブルプロセッサでは，コンテキストの切り替え時の構成情報を切り替えることによるデータフローの変化，すなわち演算の切り替えや，演算器に入力されるオペランドの変化によって多くの電力を消費していることが明らかにされた[9)]．このため，これらの消費電力を削減するアーキテクチャ上の工夫がいくつか提案されている[10)]．一方，演算の切り替え頻度やプログラムの性能はコンパイラの構成情報生成アルゴリズムにも依存しており，コンパイラによる構成情報の最適化は低消費電力化のために有効である．

† 慶應義塾大学大学院 理工学研究科
Graduate School of Science and Technology, Keio University

しかし，従来のコンパイラは，性能の向上や，コンテキスト数を抑えることを目標に設計されており，このような検討はほとんどなされていなかった．

そこで，本論文では，動的リコンフィギャラブルプロセッサの消費電力を抑えるための構成情報最適化手法として，Partially Fixed Configuration Mapping(PFCM) アルゴリズム提案する．この手法ではコンテキスト切り替え時に各ユニットの再構成回数を削減することにより，消費電力を削減する．

## 2. 関 連 研 究

動的リコンフィギャラブルプロセッサの電力効率をより向上させるために，様々な手法が提案されている．演算器に対するオペランドアイソレーションの適用[8]，細粒度部分再構成による消費電力削減技法[10]，二電源手法の利用[11] などが挙げられる．しかし，これらの提案手法の全ては，アーキテクチャ自体の改良手法であり，適用するためには，ハードウェアの実装し直しが必要となる．

これに対してコンパイラやマッピング手法による電力節約はハードウェアの変更なしに効果を得ることができ,FPGA 向けにはかなり以前から盛んに行われている[12][13][14]．一般的な FPGA に対しては, 配線による消費電力の最適化, モジュールの再利用, 高いレベルから電力を考慮して合成およびマッピングをする手法などが提案されている. また, 電源を二種類持つ FPGA[15] や, スレッショルドレベルを二種類持つ FPGA[16] など特殊な FPGA を用いて大きく消費消費を削減する方法も提案されている. しかし, これらはいずれも静的にマッピングされた回路の消費する電力を削減する研究であり, 本研究の狙いとは異なっている．

一方, 動的リコンフィギャラブルデバイス向けのコンパイラも数多く開発されている[5][6][7]．これらのコンパイラでは，FPGA 向けマッピング手法の適用[17] や，VLIW 向け並列化手法の適用[4] などを利用し，並列性を向上させ，性能を向上させることに成功している．しかし，コンパイラやマッピング手法により消費電力を削減する研究はこれまでに行われていない．

## 3. MuCCRA-3 アーキテクチャ

本研究では，評価対象アーキテクチャとして動的リコンフィギャラブルプロセッサ MuCCRA-3[9] を用いた．MuCCRA-3 は図 1 に示す小規模な PE とそれらを並べたアレイで構成される動的リコンフィギャラブルプロセッサである．

### 3.1 PE アレイ

MuCCRA-3 の PE は，演算を行う ALU，ALU のオペランドを選択する ALU_DATA_SEL，レジスタファイル (RF)，PE 間結合網を構成する Switching Element(SE) の 4 種類の再構成ユニットで構成される．PE の構造と，SE の構造をそれぞれ，図 2，3 に示す．

MuCCRA-3 は，PE による 4×4 の 2 次元アレイと，PE アレイの上端の下端に 4 つずつ，計 8 つの演算データ用メモリ (MEM) を持つ．このメモリは 128 ワードづつ 2 バンクあり，ダブルバッファリング方式によるデータ転送時間の隠蔽に利用する．

PE 間を接続する結合網は，SE を利用したアイランドスタイルに加えて，隣接している PE との直結網の 2 種類の結合網がある．図 1 の太線は直結網による接続を，細線は SE による接続網を示している．SE による接続網は A，B の 2 チャネルが利用でき，SE によりそれぞれのチャネルへの乗り換えが可能となっている．

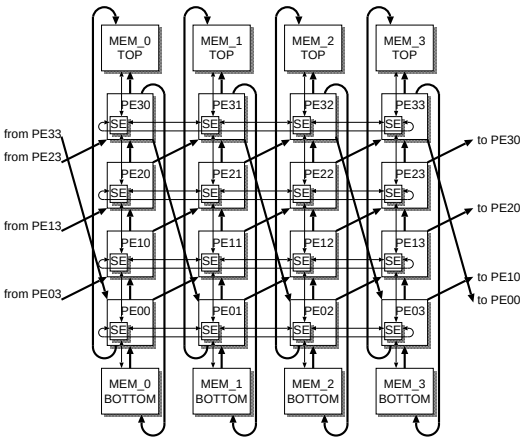


図 1 MuCCRA-3 の PE アレイ構成

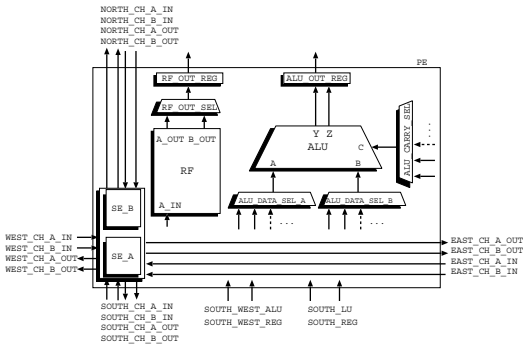


図 2 MuCCRA-3 の PE 構成

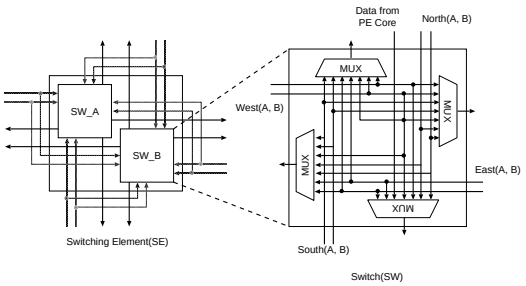


図 3 MuCCRA-3 の SE 構成

### 3.2 再構成制御機構

MuCCRA-3 はコンテキストと呼ばれる複数の構成情報のセットを切り替えて再構成を行うマルチコンテキスト方式を採用した DRPA である．MuCCRA-3 の再構成制御機構は，コンテキストの転送を制御する Task Configuration Controller(TCC) と，コンテキストの切り替えを制御する Conetxt Switch Controller(CSC) の 2 つから構成されている．

MuCCRA-3 では，1 つのアプリケーションを複数のタスクという単位に分割して制御を行う．1 つのタスクは，最大で 32 個のコンテキストの集合である．各タスクは Task Flow Table(TFT) と呼ばれるヘッダ情報が付加されており，TCC はその情報からコンテキストを外部のメモリより読み込み，各 PE などに転送を行う．タスク実行中のコンテキストの切

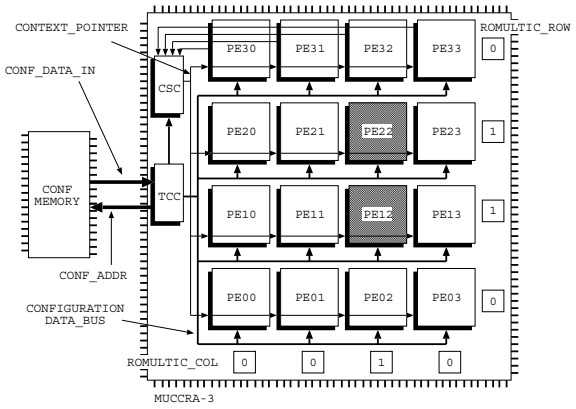


図 4　MuCCRA-3 の制御機構

り替えは，CSC が分岐などを判断して生成するポインタによって行う．

各 PE，MEM は，実行前に各ユニットに内蔵されたコンテキストメモリから構成情報を読み出して再構成を行う．そのため，これらのコンテキストメモリは実行前に構成情報を転送する必要があり，この転送は Task Configuration Controller(TCC) が行う．図 4 に MuCCRA-3 の制御機構を示す．TCC は全構成情報を保持している外部メモリに読み出しアドレス (CONF_ADDR) を与え，外部メモリから構成情報を読み出す．TCC と PE,MEM は共有バスである CONFIGURATION_DATA_BUS で接続され，TCC は構成情報と共に対象となるユニット種別情報も付加して送信する，この種別情報により，PE などのユニットは受け取るべきデータを判断し，それぞれのコンテキストメモリに書き込む．またこの際，MuCCRA-1,2 でも採用したマルチキャスト転送方式 RoMultiC[18] を用いる．RoMultiC は，PE アレイの各行と列に 1 ビットづつを割り当て，図 4 の例のように，行と列のビットが一致する複数の PE を対象 (図 4 の場合 PE12, PE22) に同じ構成情報をマルチキャストし，転送時間と回数を短縮する．

### 3.3　電力の予備評価

動的リコンフィギャラブルプロセッサの構成情報の切り換えに要する消費電力を調査するため，図 5 に示すように PE アレイ上の演算を割り当てる PE の位置を変化させ，各 PE の構成情報を変化させた場合とさせなかった場合の消費電力を MuCCRA-3 のポストレイアウトシミュレーションによって解析を行った．

**2CONTEXT OP**　2 つのコンテキストを切り替えながら，どちらのコンテキストでもアレイの下半分だけを利用して同じ演算を行い，結果をレジスタへ代入する．各 PE の構成情報はコンテキスト間で変化させない．

**2CONTEXT+DP OP**　PE の演算は”2 CONTEXT OP”と同じだが，2 つのコンテキストの切り替え毎に演算する PE をアレイの上半分と切り替える．つまり，各 PE の構成情報をコンテキスト切り替え毎に変化させる．

加算，乗算，右シフト，左シフト，バレルシフトの各演算について上記の条件でそれぞれ実行し，消費電力の結果を図 6 に示す．すべての演算で 2CONTEXT+DP OP は 2CONTEXT OP よりも消費電力が大きく，特に乗算命令を実行した結果では 30%の電力オーバヘッドが生じた．

このオーバヘッドはコンテキストメモリから構成情報を読み出し，演算やデータを切り替える際，多くの配線がスイッチされるために生じるもので，シミュレーション結果より，再構成ユニットの構成情報はなるべく変化させず，ユニットの構成情報の切り替え回数を抑制することが消費電力を抑えるために有効であることが分かる．

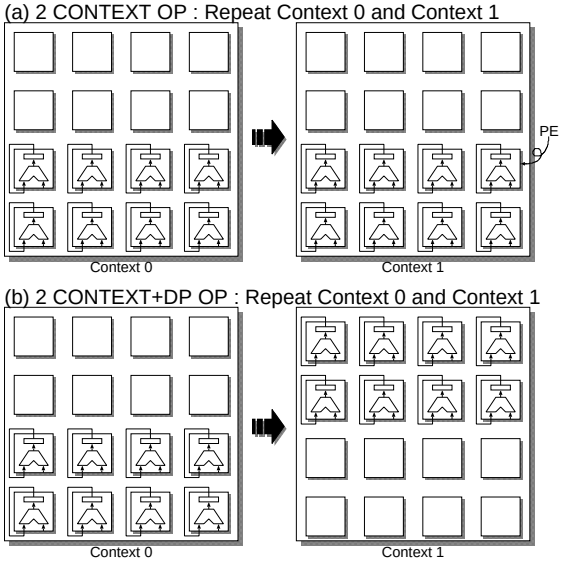


図 5　データパス切り替え時のオーバヘッドを調査するための構成情報

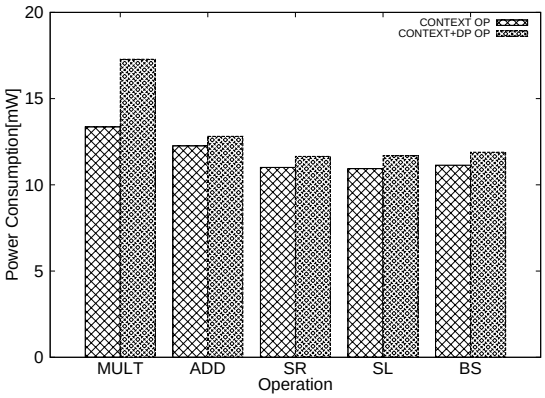


図 6　2 パターンのコンテキストでの消費電力

## 4.　提案手法:PFCM アルゴリズム

3.3 章の評価結果より，再構成ユニットが構成を切り替えることによる電力オーバヘッドを削減するために，なるべく構成情報を切り替えずに固定することが有効であることが分かった．

コンテキスト間の再構成ユニットの構成情報を維持するためには，コンパイラによりコンテキスト間の演算のスケジュールを調整し，なるべく 1 つの PE がコンテキスト間で同じ演算をし続けるように PE アレイに演算を割り当てる必要がある．

このスケジューリングを実現するためのアルゴリズムを Partial Fixed Configuration Mapping(PFCM) アルゴリズムと呼ぶ．PFCM はアプリケーション性能への影響を最小限に抑え，演算の配置の最適化と構成情報をコンテキスト間で一部共有 (伝搬) することでアプリケーションの消費電力を削減す

```
for each instruction inst
  numOfInst[ inst.kind ]++
sort( numOfInst ) in descend order
for each numOfInst
  kind = index of selected element of numOfInst
  inst_list = extractInstructionList(kind)
  for each inst_list inst
    s = scheduled context of inst
    (x0, y0) = geometry of inst
    (x, y) = findSameTypePE(x0, y0, kind)
    if failed to findSameTypePE &&
        status of allocTable_{x0,y0,s} is NONE
      call invalidate(kind, x0, y0)
      call validate(kind, x0, y0, s)
    else if success to findSameTypePE
      call validate(kind, x, y, s)
    else
      (x, y) = findEmptyPE(x0, y0, kind)
      if success to findEmptyPE
        call invalidate(kind, x, y)
        call validate(kind, x, y, s)
      else
        call validate(kind, x0, y0, s)
      endif
    endif
  endfor
endfor
function validate(kind, x, y, s)
  set allocTable_{x,y,s} as kind
  set status of allocTable_{x,y,s} as VALID
endfunction
function invalidate(kind, x, y)
  for each context c
    set allocTable_{x,y,c} as kind
    set status of allocTable_{x,y,c} as INVALID
endfunction
```

図 7　演算配置の最適化処理アルゴリズム

ることができる．

PFCM アルゴリズムは，大きく 2 つの段階に分けられる．

- 命令配置の最適化
- 構成情報の伝搬

命令配置の最適化では，命令の配置を初期配置から次段階の構成情報の伝搬処理のために，各 PE が同じ演算をコンテキスト間で実行し続けられる位置へ再配置し配置を最適化する．

次に，構成情報の伝搬処理を行い，再構成ユニットの構成情報切り替え回数を削減する．

各段階について，以下で詳細なアルゴリズムを説明する．

### 4.1　命令配置の最適化

図 7 に PFCM の演算配置の最適化処理アルゴリズムを示す．

ここでは，再構成ユニットに以下の 3 種類の状態を定義する．

- 何も命令が構成されていない (NONE)

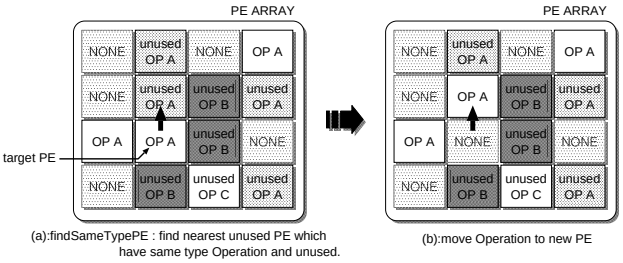


図 8　命令の再配置処理 ($findSameTypePE()$)

- 現在のコンテキストで利用する命令を構成済み (VALID)
- 現在のコンテキストで利用しない命令を構成済み (INVALID)

再配置を行う前に，最適化前のアプリケーション全体で利用する命令の種類，数，演算の配置を調査する．

アプリケーション中で利用される回数の多い命令種が消費電力に与える影響が大きいと考えられるため，最も利用される回数の多い命令種順に再配置処理を行う．図 7 では，numOfInst という配列に命令種の出現回数を格納し，この結果に基づき，もっとも回数の多い命令種順に再配置を実行する．

図 7 の $kind$ は，再構成ユニットで実行される命令種を表す．例えば, 加算命令 add，addi はどちらも再構成ユニット上では同じ加算であり，これらは同じ $kind$ を割り当てる．

$extractInstructionList()$ を用いて $kind$ に分類されている命令を取りだし，これらの命令列を $inst\_list$ に格納する．

命令列の各命令を $inst$ とし，各命令毎に再配置処理を行う．また，命令 $inst$ の実行されるコンテキスト番号を $s$ とし，この $inst$ の現在の配置を $(x0, y0)$ とする．

$findSameTypePE$ 関数は，$inst$ の命令種 $kind$ が現在のコンテキストで構成済みで，$inst$ の初期配置 $(x0, y0)$ から最も近い位置に位置する現在のコンテキストでは演算に利用しない (状態が INVALID の) 再構成ユニットを探索する．

図 8 に，$findSameTypePE$ 関数の動作を示す．命令種が等しく，現在のコンテキストでは利用しない命令がアレイ上に配置されている場合，その再構成ユニットに命令 $inst$ の配置を移動する (命令の再配置)．

一方，命令種が等しい命令が割り当てられた PE が見つからず，再配置に失敗した場合，利用されていない再構成ユニットを探す必要がある．この探索を行う関数が $findEmptyPE()$ である．$findEmptyPE()$ によりすべてのコンテキストで利用されていない再構成ユニットが見つかった場合，その再構成ユニットはすべてのコンテキストで同一の命令種の命令を配置する．図 9 に，命令の再配置動作の概要を示す．CONTEXT$i$ で命令が配置されると，CONTEXT$i+1$, CONTEXT$i+2$ にも同一の命令が配置され，これらはそれぞれのコンテキストでの $findSameTypePE()$ で利用する．図 8 に，$findEmptyPE()$ の動作を示す．

$findEmptyPE()$ により，すべてのコンテキストで利用されていない再構成ユニットが見つからなかった場合は，その命令が初期配置されている再構成ユニットに割り当てる．(初期配置から変更しない) この場合，他に利用できる再構成ユニットがないため，配置の最適化ができず，その再構成ユニットでは構成情報の切り替えが生じる結果になる．

この演算配置の最適化処理では，コンテキストの命令の配置だけを最適化し，命令を他のコンテキストに移動することは行わない．もし，実行される演算命令のコンテキストを変

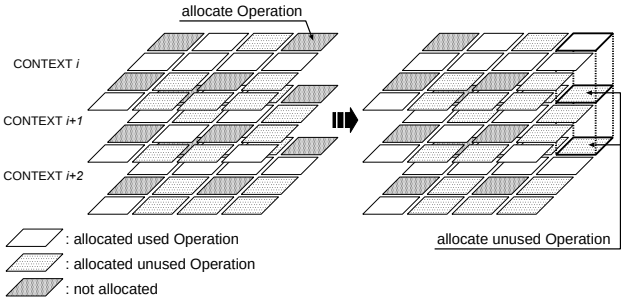


図 9　命令の再配置

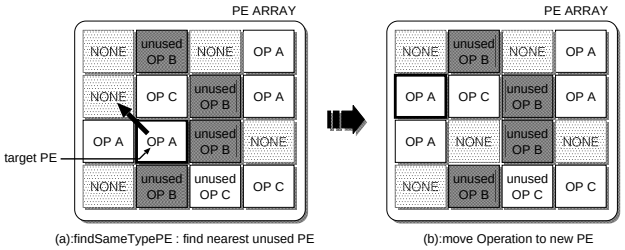


図 10　利用されていない再構成ユニットへの命令の再配置 ($findEmptyPE()$))

```
pred_s = 0;
s = 1;
while all region of x and y
  while all region of s
    if status of allocTable_{x,y,s} is not VALID &&
          inst of PE[x, y, s] != inst of PE[x, y, pred_s]
      propagate_conf( inst of PE[x, y, s],
                      inst of PE[x, y, pred_s]);
    endif
    pred_s = s;
    s = next context of s
  endwhile
endwhile
```

図 11　構成情報の伝搬アルゴリズム

えてしまった場合，演算命令の再スケジュールの必要が生じてしまい，場合によってはコンテキスト数が増えてしまう為である．

### 4.2 構成情報の伝搬

構成情報の伝搬処理は，コンテキスト $i$ で各再構成ユニットが利用されているか調査する．ユニットが利用されていない場合，コンテキスト $i$ の構成情報を一つ前のコンテキスト $i-1$ の構成情報で置き換える．

図 11 に構成情報の伝搬アルゴリズムを示す．

プログラムの実行順に構成情報の伝搬処理を行っていく．再構成ユニットの状態が INVALID もしくは NONE である場合，一つ前のコンテキストの構成情報をそのまま引き継ぐ (アルゴリズム中の $propagate\_conf()$) 操作を，すべての構成情報に対して適用する．

このとき，$propagate\_conf()$ は，一律に構成情報を次のコンテキストに複写するのでは無く，再構成ユニットごとに構成情報の最適な置き換え方針を決める．表 1 に再構成ユニットごとの置き換えの方法についてまとめる．

表 1　再構成ユニットにおける PFCM の適用方針

| 再構成ユニット | PFCM の適用方針 |
|---|---|
| ALU | オペコード等，すべての構成情報を 1 つ前のコンテキストの構成情報で置き換える |
| ALU_DATA_SEL | オペランド等，すべての構成情報を一つ前のコンテキストの構成情報で置き換える |
| RF | 読み込み・書き込みアドレスを一つ前のコンテキストの構成情報で置き換える．書き込み信号は無効にする |
| SW_A, SW_B | 構成情報を引き継がずに，0 を出力する |
| MEM | 読み込み・書き込みアドレスを一つ前のコンテキストの構成情報で置き換える．書き込み信号は無効にする |
| CONTROL | 構成情報を引き継がない |

ALU では，命令の種類やキャリーインの選択など，すべての構成情報を引き継ぎ，ALU_DATA_SEL ユニットでも，同じくすべての構成情報を引き継ぐ．

RF ユニットでも大部分の構成情報を引き継ぐが，RF への書き込み信号は伝搬させずに無効化する．これは，一つ前のコンテキストでレジスタ書き込みが行われていた場合，その結果をそのまま伝搬させると後続のすべてのコンテキストで意図しない書き込みが生じることを防ぐためである．

SW_A, SW_B ユニットに対しては，構成情報の伝搬は行わず，代わりにスイッチの出力を停止する構成情報に置き換える．これは，スイッチングエレメントはデータ転送用のユニットであるため，構成情報を前のコンテキストから引き継ぐと無意味なデータが結合網上を流れることを防ぐ為である．

同様に MEM ユニットについても，DMEM への書き込み信号伝搬せず無効とする．

CONTROL ユニットは，デバイスの制御を司るユニットであり，デバイス内に 1 つしか存在しないことから，構成情報は引き継がない．

このようにして，PFCM は同一コンテキスト内でのみ演算の配置を最適化するため，アプリケーションのコンテキスト数が変化することは無い．従って，実行サイクル数は変化せず，性能に影響を与えることなく再構成回数を最小化することができる．

## 5. 評　　価

### 5.1 評 価 環 境

提案手法の評価のため，MuCCRA-3 を富士通 65nm プロセスライブラリを用いて論理合成を行い，ゲートレベルシミュレーションの結果を用いて電力評価を行った．全ての手法で共通の動作周波数制約として 40MHz を与え，これを満たすことを確認した．

評価対象のアプリケーションには $\alpha$ ブレンダ，グレイスケールフィルタ，セピアフィルタ，SSD(差分 2 乗和)，2 次元離散コサイン変換 (2D-DCT)，2 次元逆離散コサイン変換 (2D-IDCT) を利用した．

提案手法を評価するために，MuCCRA-3 用のコンパイラを実装した．本コンパイラは，C 言語で記述されたプログラムを入力すると，フロントエンド処理，スケジューリング，配置，提案アルゴリズムである演算ノードの再配置処理を自動的に行う．本コンパイラでは配線処理は行わず，MuCCRA-3 用アセンブラを用いて配線を行い，その後，提案アルゴリズ

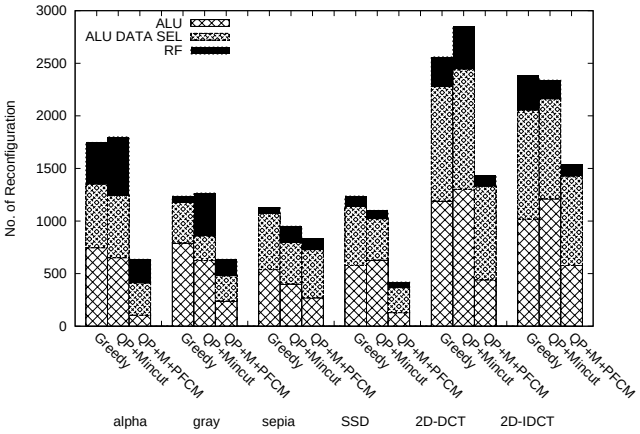


図 12　再構成ユニット毎の構成情報の変更回数

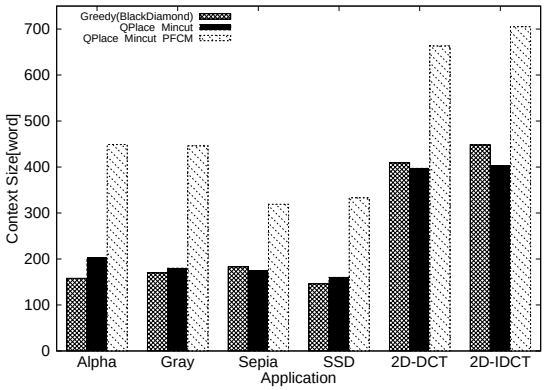


図 14　構成情報のサイズ

ムである構成情報の伝搬処理を行い，構成情報を出力する．

また，本研究では，構成情報出力アルゴリズムとして以下の 3 種類を用意し，それぞれ比較を行った．

1. **Greedy アルゴリズム**　演算命令を PE アレイの下側から順に配置する．配線が不可能になるとコンテキストを切り替える．本研究室で開発された Black-Diamond コンパイラ[6] が本アルゴリズムを利用している．
2. **Quadratic & Mincut Placement(QPlace & Mincut)**　VLSI の配置アルゴリズムである Quadratic Placement[19] と Mincut Placement[20] を交互に適用する．本研究で開発したコンパイラが利用する配置アルゴリズムである
3. **QPlace & Mincut & PFCM**　上記の配置結果に対して本研究の提案手法を適用する

1，2 は，提案手法を適用せず，スケジュール・配置配線のみを実行して構成情報を生成するものである．そして 2. と 3. の評価結果を比較し，提案手法の効果を検証する．

### 5.2　再構成頻度

提案手法の適用により再構成ユニットの再構成回数の変化について調査するため，各再構成ユニットの再構成回数のシミュレーションを行った．

図 12 に，ALU, ALU_DATA_SEL, RF ユニットの PFCM 適用時と非適用時の再構成回数を示す．なお，今回構成情報の伝搬処理を行うのは上記の 3 つのユニットのみであるため，SW_A, SW_B については再構成回数のシミュレーションを行っていない．

すべてのアプリケーションにおいて，提案手法を適用した場合にすべての再構成ユニットで再構成回数が削減された．特に ALU ユニットの再構成回数の削減率が高く，$\alpha$-Blender では Greedy アルゴリズムでの再構成回数に比べて 86%程度再構成回数が削減できた．

一方で，sepia フィルタではすべてのユニットで再構成回数の削減率が小さく，Greedy アルゴリズムに比べて 26%程度の削減率に留まっている．これは，sepia フィルタは各コンテキストの再構成ユニットの利用率が低く，半分以上の PE を利用せずにアプリケーションが実行されているため，利用されていない再構成ユニットでは構成情報の変更が行われず，大部分のユニットが再構成を行われていないためである．

図 13(a) に，2D-DCT における PFCM を適用しない際の演算ユニットの構成情報出力結果，図 13(b) に，PFCM を適用した際の演算ユニットの構成情報出力結果を示す．

2D-DCT は演算数が多いため，構成情報の切替え回数も多く，PFCM を適用しない場合，すべてのコンテキストで構成情報の切り替えが生じている．一方，PFCM を適用した場合では，非適用時に比べて再構成回数が減少していることが分かる．同アプリケーションは，加算 26 命令，乗算 16 命令，シフト 10 命令を利用しており，一度も再構成を行わないように PE アレイ上に配置することは不可能である．しかし同一種類の命令を同じ PE 上で実行するように演算を移動することにより，PFCM 非適用時に比べて再構成回数を削減することができている．

結論として，提案手法はある程度規模の大きなアプリケーションで，PE アレイの利用率が高い場合に効果があると言うことができる．

### 5.3　構成情報データサイズ

構成情報のデータサイズはアプリケーション実行前の構成情報転送時間に大きく影響する．本提案手法を適用したことによる，構成情報データサイズへの影響について検証を行った．

図 14 に，各アプリケーションにおいて Greedy アルゴリズムを用いた場合，QPlace & Mincut のみを用いた場合，QPlace & Mincut & PFCM を適用した場合について構成情報データサイズを示す．

QPlace& Mincut のみを適用した場合，Greedy アルゴリズムを適用した場合に比べてデータサイズはわずかな増加に止まっており，2D-DCT においては減少する場合も見られた．これは MuCCRA-3 に採用されているコンフィギュレーションデータ転送アルゴリズム RoMultiC の利用効率に強く依存しており，構成情報の PE アレイの配置方法にも強く依存する．

一方，PFCM を適用した場合，データサイズは 2 倍近く増加した．この原因としては提案手法の構成情報の伝搬処理が挙げられる．PFCM を適用しない場合，あるコンテキストでの構成情報は，RoMultiC のアルゴリズムにより PE の利用してない部分を活用して広く囲むことができ，その結果構成情報を小さくすることができていた．ところが，PFCM を適用することで，各コンテキストにおいて多くの PE を細かく制御する必要があり，その結果として RoMultiC による構成情報の圧縮効果が薄れ，全体として構成情報データサイズの増加を招いたと考えられる．

しかし，動的リコンフィギャラブルデバイスは，一度構成情報を転送してしまうと，アプリケーションを切り替えるま

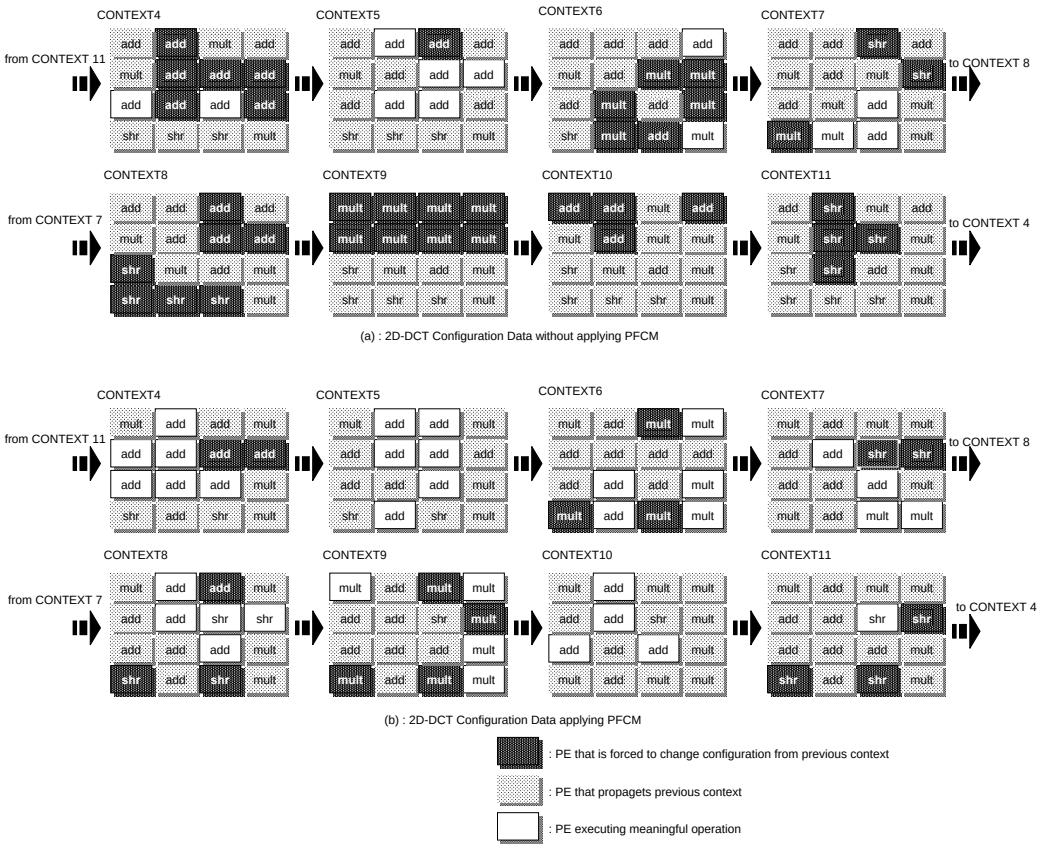


図 **13** 2D-DCT の PFCM 非適用時 (a) および適用時 (b) の構成情報の遷移

でコンフィギュレーションデータを転送する必要が無く，したがって，アプリケーションにつき処理すべきデータが多いほど，構成情報データの転送時間は隠蔽されることとなる．動的リコンフィギャラブルデバイスはメディアストリーム処理向けアクセラレータであるため，大量のデータを処理するアプリケーションを実行することが多い．したがって，小さなデータサイズのアプリケーションを頻繁に切り替えて実行しない限り，性能への影響は少ないと考えられる．

### 5.4 アプリケーションの実行サイクル数

提案手法を適用したことによるアプリケーションの実行速度を解析するために，評価アプリケーションの実行サイクルのシミュレーションを行った．

表 2 に，各アプリケーションの実行サイクルを示す．

表より，QPlace& Mincut に対して PFCM を適用しても，実行サイクルに変化は生じず，PFCM を適用してもアプリケーションの実行速度への影響は少ない．

Greedy アルゴリズムと比べて $\alpha$-Blender のみ実行サイクル数がわずかに変化しているが，これは繰り返しループより外側の初期化の行い方が異なっているためである．

アプリケーションの実行速度が低下しない理由として，PFCM は構成情報からコンテキスト毎の命令配置を最適化するだけであり，この範囲は同一コンテキスト内のみを対象とし，コンテキストをまたいだ構成情報の再配置を行わないため，コンテキストの数や順序に影響を与えないためである．

### 5.5 実行時消費電力

提案手法を適用した際のアプリケーション実行時の消費電力について評価を行った．図 15 に，各アプリケーションにおいて Greedy アルゴリズムを用いた場合，QPlace & Mincut のみを用いた場合，QPlace & Mincut & PFCM を適用した場合について実行時の消費電力を示す．図 15 の Internal Power は，トランジスタ ON/OFF 時のセル内で消費される貫通電力である．また，Switching Power は，トランジスタに接続されている配線がトグルすることによる消費電力である．Leak Power はトランジスタのリーク電力である．どの評価でも，ハードウェア自体は同じであり，したがってリーク電力は 3 つの手法で同一である．

表 **2** アプリケーションの実行サイクル

| | alpha | gray | sepia | ssd | DCT | IDCT |
|---|---|---|---|---|---|---|
| Greedy | 83 | 82 | 67 | 67 | 127 | 113 |
| QPlace & Mincut | 84 | 82 | 67 | 67 | 127 | 113 |
| QPlace & Mincut & PFCM | 84 | 82 | 67 | 67 | 127 | 113 |

Greedy アルゴリズムでは，演算に利用する PE をアレイの下側に集中させる傾向があるため，各 PE の再構成回数が多くなっており，消費電力の増加につながっている．

QPlace & Mincut を適用した場合，つまり配置配線アルゴリズムを変更した場合でも，消費電力は平均で 5%程度削減されている．これは 3.3 節の MuCCRA-3 の予備評価でも示したとおり，演算命令の配置の結果が消費電力に強く影響しているとを示している．

しかし，2D-DCT や 2D-IDCT のような比較的複雑なアプリケーションでは，配置配線アルゴリズムを変更しても消費電力は大幅に削減されていない．これは PE アレイの利用率

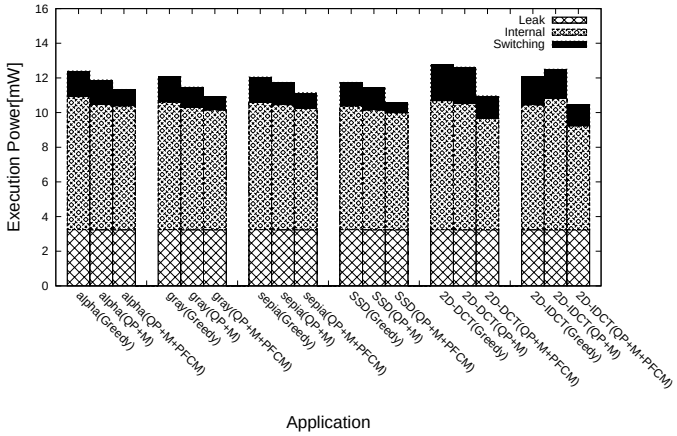


図 **15** 実行時消費電力

が高い場合，配置アルゴリズム毎の差が小さくなるためと考える．

一方 PFCM を適用した場合，平均で 10%程度の消費電力が削減できた．QPlace & Mincut での消費電力の削減効果が薄いアプリケーションにおいても，PFCM を適用した場合，大幅な消費電力の削減を行うことができている．

2D-DCT のアプリケーションに注目すると，図 12 では，ALU の再構成回数が半分以上に削減されており，図 15 での 2D-DCT の消費電力が大きく削減されていることが分かる．

これらの評価により，本提案アルゴリズムは再構成ユニットの再構成回数を抑制し，消費電力の削減に有効であることが分かる．

## 6. 結　　論

本論文は，動的リコンフィギャラブルデバイスにおける消費電力を削減するマッピング・スケジューリング・アルゴリズム Partially Fixed Configuration Mapping を提案し，その効果について検証を行った．提案アルゴリズムを適用するための開発環境を整備し，同提案手法の効果を検証した結果，アプリケーションの性能をに影響を与えることなく，消費電力を平均で 10%程度削減することができることを示した．

謝　　辞

本研究の一部は，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．

本研究におけるチップ開発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，株式会社半導体理工学研究センター・(株) イー・シャトルおよび富士通株式会社・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．

## 参 考 文 献

1) Kurose, Y., Okabe, M., Seno, K., Ozawa, H., Wada, T., Taniguchi, K., Hokazono, H., Hirano, T., Kumata, I., Hanaki, H., Hasegawa, K., Horiike, S., Arima, S., Ono, K., Hiroi, T. and Takashima, S.: A 90nm embedded DRAM single chip LSI with a 3D graphics, H.264 codec engine, and a reconfigurable processor, *Hot Chips 16* (2004).
2) Motomura, M.: C-based Programmable-HW Core “STP Engine”Current Status and Future, *IECE Technical Report, RECONF2008-48(invited talk)* (2008).
3) Panasonic: D-Fabrix. www.panasoniceurope.com.
4) Mei. B., et al.: DRESC: a retargetable compiler for coarse-grained reconfigurable architectures, *International Conference on Field Programmable Technology(FPT2002)*, pp. 155–173.
5) T. Toi., et al.: High-Level Synthesis Challenges and Solutions for a Dynamically Reconfigurable Processor, *The International Conference on Computer-Aided Design(ICCAD'06)*, pp. 702–708 (2006).
6) Tunbunheng, V. and Amano, H.: Black-Diamond: a Retargetable Compiler Using Graph with Configuration Bits for Dynamically Reconfigurable Architectures, *The 14th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, pp. 412–419 (2007).
7) Yoshikawa, T.: A dynamically reconfigurable processor for streaming processing, *Tutorial of the ICFPT 2007* (2007).
8) T.Nishimura, K.Hirai, Y.Saito, T.Nakamura, Y.Hasegawa, S.Tsutsusmi, V.Tunbunheng and H.Amano: Power Reduction Techniques for Dynamically Reconfigurable Processor Arrays, *International Conference Field Programmable Logic and Application (FPL2008)* (2008).
9) Saito, Y., Sano, T., Kato, M., Tanbunheng, V., Yasuda, Y. and Amano, H.: A Real Chip Evaluation of MuCCRA-3: A Low Power Dynamically Reconfigurable Processor Array, *The 2009 World Congress in Computer Science, Computer Engineering and Applied Computing*, pp. 283–286 (2009).
10) Sano, T., Saito, Y., Kato, M. and Amano, H.: Fine Grain Partial Reconfiguration for energy saving in Dynamically Reconfigurable Procesors, *19th International Conference on Field Programmable Logic and Applications(FPL2009)*, pp. 530–533 (2009).
11) Yamamoto, T., Hironaka, K., Hayakawa, Y., Kimura, M., Amano, H. and Usami, K.: Dynamic VDD Switching Technique and Mapping Optimization in Dynamically Reconfigurable Processor for Efficient Energy Reduction, *7th International Symposium on Applied Reconfigurable Computing (ARC2011)* (2011).
12) D.Chen, J.Cong and Y.Fan: Low-Power High-Level Synthesis for FPGA Architecture, *Proc. of International Symposium on Low Power Electronics and Design*, pp.134–139 (2003).
13) R.Pandey and S.Chattopadhyay: Low-Power Technology Mapping for LUT based FPGA: A Genetic Algorithm Approach, *Proc. of 16th International Conference on VLSI Design*, p. 79 (2003).
14) J.Kim, H.Yang, K.Ryu and H.Kim: FPGA Low-Power Technology Mapping for Reuse Module Design under the Time Constraint, *Proc. of Future Generation Communication and Networking*, pp. 57–61 (2008).
15) Chen, D., Cong, J., Dong, C., Li, F. and Peng, C.: Technology Mapping and Clustering for FPGA Architectures with Dual Supply Voltages, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 29, No. 11, pp. 1709–1722 (2010).
16) H.Hassan, M.Anis and M.Elmary: A leakage-aware CAD flow for MTCMOS FPGA architecture, *Proc. of the 2005 ACM/SIGDA 13th international symposium on Field programmable gate arrays*, p. 267 (2005).
17) Friedman, S., Carroll, A., Essen, B. V., Ylvisaker, B., Ebeling, C. and Hauck, S.: SPR : an architecture-adaptive CGRA mapping tool, *In Proc. of the ACM/SIGDA international symposium on Field Programmable Gate Arrays(FPGA)*, pp. 191–200 (2009).
18) Tunbunheng, V., Suzuki, M. and Amano, H.: RoMultiC: Fast and Simple Configuration Data Multicasting Scheme for Coarse Grain Reconfigurable Devices, *International Conference on Field Programmable Technology(FPT2005)*, pp. 129–136 (2005).
19) M.Kleinhans, J., Sigl, G., M.Johannes, F. and J.Antreich, K.: GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization, *In Proc. of IEEE Trans. on CAD*, pp. 356–365 (1991).
20) Breuer, M. A.: Min-Cut Placement, *In Journal of Design Automation and Fault Trelance*, pp. 343–362 (1977).