コンテキストメモリの排除による 動的リコンフィギャラブルプロセッサの低電力、省面積化

木 村 優 之[†] 天 野 英 晴[†]

コンテキストメモリは, マルチコンテキスト型 CGDRP(Coarse-Grained Dynamically Reconfigurable Processor) の特徴的な構成要素であり, プロセッサおよびスイッチングエレメントに分散配置され, 複数のハードウェアコンテキストに対応する構成情報を保持する. コントローラからのポインタにより, コンテキストメモリから構成情報を読み出すことで, ハードウェアコンテキストスイッチを 1 クロックで行うことが可能である. しかし, 一方で, 面積と電力に占める割合いが大きく, 格納できるコンテキスト数の制限によりアプリケーションが制限される問題もある. そこで, 本論文では, マルチコンテキスト型 CGDRP からコンテキストメモリを排除する手法について検討した. コンテキストメモリの代りにダブルバッファを持たせ, 片方の構成情報で実行中に, 次の構成情報をバックグランドで転送する. 構成情報の転送にはマルチキャスト手法を用いて転送時間を短縮する. しかし, このような従来手法の組み合わせでは, 構成情報の転送が間に合わないため, 大きなストール時間が発生してしまう.

そこで, まず, スケジュール手法である Loop Separation を用いて, ループの実行時, 一つのコンテキストを, 一定以上のクロック数連続して実行するように変換する. さらに, 構成情報のマルチキャスト機構に差分再構成法,Spare register を組みあわることでバックグランド転送のクロック数を削減した. 評価の結果, ループ依存性がないプログラムでは, 最大 12-13%の性能低下で面積を 63%に, 消費電力を 40%にすることに成功した. ループ間の依存性があるプログラムでは差分構成法が有効であり, 今回評価した SQSUM では 18%程度まで性能低下を抑えることができた.

Reducing Power and Area of Dynamically Reconfigurable Processors by removing the context memory

Masayuki Kimura [†] and Hideharu Amano [†]

Although context memory or configuration cache is a key mechanism for quick dynamic reconfiguration of multi-context Coarse-Grained Dynamically Reconfigurable Processors (CGDRP), it requires a large amount of area and energy. In order to save them, methods to remove the context memory from multi-context DRPA are proposed. In order to keep a context without switching, a scheduling method; Loop Separation is introduced. By separating loops by the compiler the same context can be used without switching in a certain clock cycles with small additional hardware. The back-ground configuration data loading time can be reduced by multicasting configuration data with two dimensional bit-map. For further reduction, the differential loading and spare register are proposed. With combination of them, the increasing execution time is only up to 12-13% if the target application does not have loop-carried dependency. With the above overhead on the performance, the semiconductor area becomes 63%, and the energy consumption is reduced to 40%, thus, the performance per cost or energy is much improved.

1. はじめに

粗粒度動的リコンフィギャラブルプロセッサ (Coarse-Grained Dynamically Reconfigurable Processor: CGDRP)[1] は, 8bit から 32bit の粗粒度の PE(Processing Element) およびメモリモジュールを，アレイ状に多数配置した構成をとり，データ並列性の高いアプリケーションにおいて高い処理能力を発揮する．FPGA (Field Programmable Gate Array) などのプログラマブルデバイス同様，開発後に回路構成情報を追加，変更するだけで，様々な処理に対応することのできる拡張性，柔軟性を持ち，ハードウェアの開発期間やコストの削減につながると期待されている．

CGDRP の最大の特徴は高速な動的再構成機能であり, なかでもマルチコンテキスト型と呼ばれる方式では,1 クロックで PE アレイ上の演算用データパスを

† 慶應義塾大学理工学部 Faculty of science and Technology, Keio University

変更することができる．この方式では，PE の演算と PE 間の接続を決める構成情報の PE アレイ 1 セット分をハードウェアコンテキストと呼ぶ. そして複数のコンテキストに対応する構成情報をコンテキストメモリあるいはコンテキストキャッシュと呼ばれる小規模のメモリに搭載する．コンテキストメモリには，一定のアプリケーションを動かすのに必要なコンテキストに対応する構成情報を，実行前にあらかじめ外部より転送しておく．

このコンテキストメモリは，高速に切り替える必要上，PE や SE(Switching Element) に分散されて配置されており，コントローラから分配されるコンテキストポインタに従って, 一斉に内容を読み出すことでコンテキストの切り替え，すなわちコンテキストスイッチを行う．このハードウェアコンテキスト切り替え機構を用いてデータパスをクロック単位で変更することで, 多くのアプリケーションを高い PE の利用効率で実装することができる[2)]．

しかし，多くのマルチコンテキスト型 CGDRP では，PE および SE がそれぞれ個別にコンテキストメモリを持つため，全体の面積のうちコンテキストメモリの占める割合は大きい.32 コンテキストを持つ IMEC の ADRES では PE の面積の約半分を占め[3)], MuCCRA の各チップでも 40%程度を占める[4)]．また, 高速なコンテキストスイッチは, データパスの変更のための電力が大きいこともわかっている[5)]．さらに, コンテキストメモリの漏れ電流はバッテリー駆動の製品には無視できない大きさになる．

そこで，本論文では各 PE, SE には簡単なダブルバッファのみを持たせ，バックグランドで構成情報を転送することにより，コンテキストメモリを持たないでも，マルチコンテキスト型と同様なアプリケーションを実行可能とする手法について検討する．構成情報のマルチキャストをいかに工夫しても，コンテキストの切り替えが一定の頻度以上に行われると，構成情報の転送が間に合わない. このため，PE アレイ上のデータパスをなるべく変更しないためのスケジューリングと付加ハードウェア機構を設ける. これに，構成情報の転送をさらに高速化する手法を組み合わせる．提案手法により，コンテキストメモリを持つ動的リコンフィギャラブルプロセッサに比べ最大 18%の性能低下で面積を 63%に消費電力を 40%に抑えることができる．

本論文の構成は以下の通りである.2 章では典型的なマルチコンテキスト型 CGDRP を紹介し，コンテキストメモリのオーバーヘッドとこれを低減するための従来手法についてサーベイする.3 章では，提案手法を述べる.4 章では評価に用いた CGDRP の詳細を説明し，アプリケーション実行性能，面積，電力を評価する.5 章で結論を述べる．

2. マルチコンテキスト方式

2.1 基本的なマルチコンテキスト方式とその問題点

動的リコンフィギュレーションを実現する最も簡単な方法は, 構成情報を格納したオンチップメモリをチップ内に設け, この内容をバス等により順に PE や SE に対して転送することである (図 1). この方法はメモリモジュールが一個で済むため, 面積が小さくて済むが, 転送のために多数の転送クロック数を必要とするため, 時間がかかってしまう.PACT Xpp や D-Fabrix がこの方式に当たる. これらの動的リコンフィギャラブルプロセッサはコンテキストの切り替えに数十マイクロ秒が必要であり, その間処理は停止状態となる．

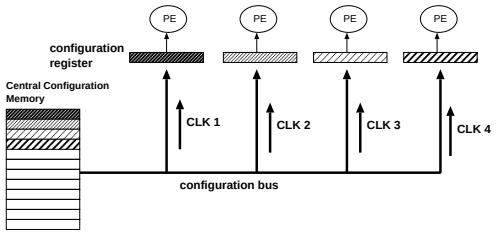

図 1 構成情報分配方式

より高速な再構成法が「マルチコンテキスト方式」である. この方式は古くから細粒度用に提案され[6)], 粗粒度構成でも一般的に用いられるようになっている. この方式は,PE または SE のそれぞれが自分の構成情報を 4-32 セット分格納する小規模なメモリを持つ. このメモリの内容は, 処理が始まる前にあらかじめチップ外部またはチップ内のコンフィギュレーションメモリから図 1 と同様の方法で転送しておく. この小規模なメモリをコンテキストメモリまたはコンテキストキャッシュと呼ぶ. 実行中のコンテキスト番号はコントローラからコンテキストポインタとして送られ, ポインタの値を切り替えることで, 対応する番地のコンテキストメモリがアクセスされ, そのコンテキスト用の構成情報が読み出されることでコンテキストスイッチが起きる (図 2). コンテキスト数は様々であり, 少ないものは 3(DAPDNA-II),4(FE-GA) から多いものは 32(STP engine,ADRES),64(MuCCRA-1) に達する．

コンテキストメモリに余裕があれば, 実行中に次のコンテキストに対応する構成情報をチップ外部や集中メモリからバックグランド転送することが可能である. 現在のタスクの実行中に, 余っているコンテキストメモリ領域に対して次のタスクに相当するコンテキスト

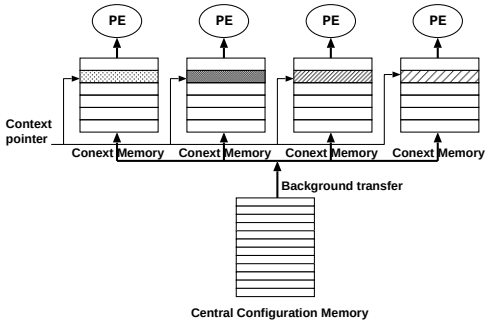

図 2 マルチコンテキスト方式

情報をすべて転送することができればストールなしで次のタスクへの切り替えが可能である. コンテキストの制御には様々な方法が提案されているが, 多くの実システムでは状態遷移を用いる方式やコンテキストカウンタ方式が用いられる.

マルチコンテキスト方式は, クロック単位で動的再構成が可能であり, 様々なアプリケーションを高い PE の利用率で搭載することが可能である. しかしこの方式には大きな問題点が二つある.

一つは, コンテキストメモリのオーバーヘッドである. 分散配置されたコンテキストメモリは,PE アレイのうちの大きな面積を占める.32 コンテキストを持つ ADRES では 50%の面積を占め[3],16 コンテキストで部分的に共有を行って面積を削減している MuCCRA-2 でも 40%に相当する[4]. さらに, コンテキストの切り替えによる電力消費, メモリのリーク電力も大きい. コンテキストメモリから構成情報を読み出してデータパスを切り替える電力は,CGDRP の動的電力の約 20%を占める[5].

二つめの問題点は, コンテキストメモリの容量の制限によるアプリケーションの制限である. コンテキストメモリに格納可能なコンテキスト数は多くても 64 程度であることから, アプリケーションによっては数が不足する. コンテキストメモリの一部を利用している間に, バックグランドで次の構成情報を転送する手法[7]は従来より研究されているが, 制約が大きく, バックグランド転送が間に合わない場合, ストールを生じる.

2.2 マルチコンテキスト方式の改良手法

マルチコンテキスト方式の問題点は,PE やスイッチングエレメントがそれぞれ異なった構成情報を持っている所から来ている. PE の演算および PE 間接続の再構成に制限を持たせれば, 構成情報を個別に持つ必要がなくなり, 問題点は解決する.Morphsys[8] は,PE の行, 列がすべて同じ構成情報で動作する部分 SIMD 方式を用いている. この場合, 行, 列単位に同一の構成情報を送れば良いため, 現実的な配線量で動的再構成が可能である. PipeRench[9] を代表とするパイプライン構造の CGDRP では,1 クロックで再構成されるのはパイプラインの 1 ステージに制限される. この場合も 1 ステージ分の構成情報のみを転送すれば良いため, 現実的な配線量で動的再構成が可能である.FLORA[10] では, コンテストメモリ間での構成情報の移動を可能としている. この方法では,PE の列単位で構成情報を移動させることで, 各列の機能を順に移動しながら演算を行って全体の再構成を行うことができる.

しかし, 上記の方法は, どれも動的再構成に制限が加わり, マルチコンテキスト型 CGDRP の高い柔軟性を犠牲にしている.

2.3 従来方式の組み合わせによる手法

コンテキストメモリを利用せず, 動的再構成に制限も加えない方法として, 最も可能性があるのは, 図 3 に示すダブルバッファに, 構成情報のマルチキャストを組み合わせる手法である. この手法では, マルチコンテキストメモリを用いた仮想ハードウェア手法[6] と同じく, 片方のバッファ中に現在実行中の構成情報を蓄えておいて, もう片方に対して, 次のコンテキストの構成情報を転送する. コンテキストスイッチより前に次のコンテキストの構成情報の転送が終わればストール無しに切り替えることができるが, そうでなければ性能上のオーバーヘッドが生じる.

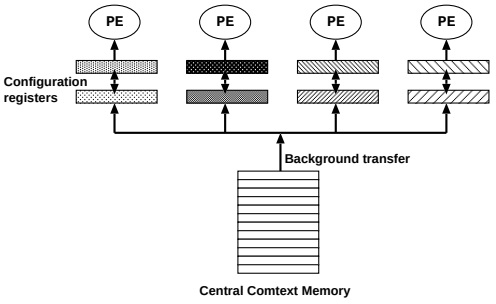

図 3 ダブルバッファの利用

1 クロックサイクルでコンテキストを切り替えることができるマルチコンテキスト型ではアプリケーションは頻繁にコンテキストを切り替えるのに対して, 構成情報の転送自体は, 図 1 に示すように, 順番に各 PE やスイッチングエレメントに配っていく必要がある. このため, 通常, 数百クロックから数千クロックを要する. このため, 単純にダブルバッファ方式を用いただけでは, 実行中のほとんどの時間帯で構成情報のためのストールが発生してしまう. このため, 構成情報転送の高速化が必要である.

構成情報の転送の高速化手法として, 最も有効なの

はコンフィギュレーションデータのマルチキャスト手法であり,RoMultiC[11] はこの手法の一つである．RoMultiC は図 4 に示すような 2 次元のビットマップを用い,x 方向と y 方向が共に 1 になる PE のグループに対して同一の構成情報を一度に送ることによって転送時間を減らす. 転送対象は矩形領域に限定されるが, 転送の順番を工夫することで様々な形状を対象とすることが可能である．また, 構成情報のフィールドの切り方を工夫することで, さらに高速化を図ることができる[12]．

RoMultiC は, 特に同一の処理をする PE 数の多いマルチメディア処理などで特に効果的だが, それでもまだ多くのアプリケーションでは数十クロックを要するため, 依然としてストールを数多く発生する．本論文はこのストールを可能な限り少なくし, ダブルバッファとマルチキャストの組み合わせを現実的にする手法を提案する．

3. ストール削減手法の提案

3.1 Loop Separation

マルチコンテキスト型 CGDRP は往々にして 1 クロックに一回コンテキストスイッチを行う. この頻度でコンテキストを切り替えられては，ダブルバッファに対するバックグランド転送が間に合うはずがない. しかし，コンテキストの切り替えに制約を課してはマルチコンテキスト型 CGDRP の利点を損ねることになる．そこで，マルチコンテキスト型 CGDRP が実行するプログラムを，コンテキストスイッチの回数を減らすように，スケジューリングしなおす手法を提案する．

CGDRP のアプリケーションはマルティメディア処理など, 一定の大きさのデータに繰り返し処理を施すものが多い．これらの処理は, ループ内の処理は複雑でも, ループ構造自体は単純なものが多く, 多数のデータに対して同一処理が繰り返して施される. 図 5 は, このタイプの処理を非常に小さい 2×2 の PE アレイで実行する例を示す. マルチコンテキスト型の DRPA では, 通常,PE アレイ上のデータパスを毎回切り替えて図 5a に示すようにこの処理を実行する．

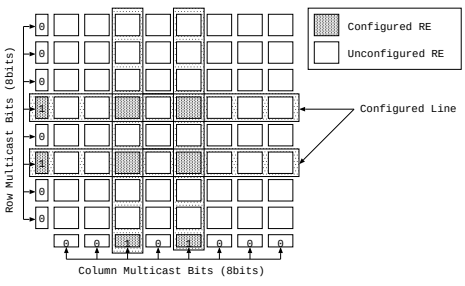

図 4 マルチキャスト手法：RoMultiC

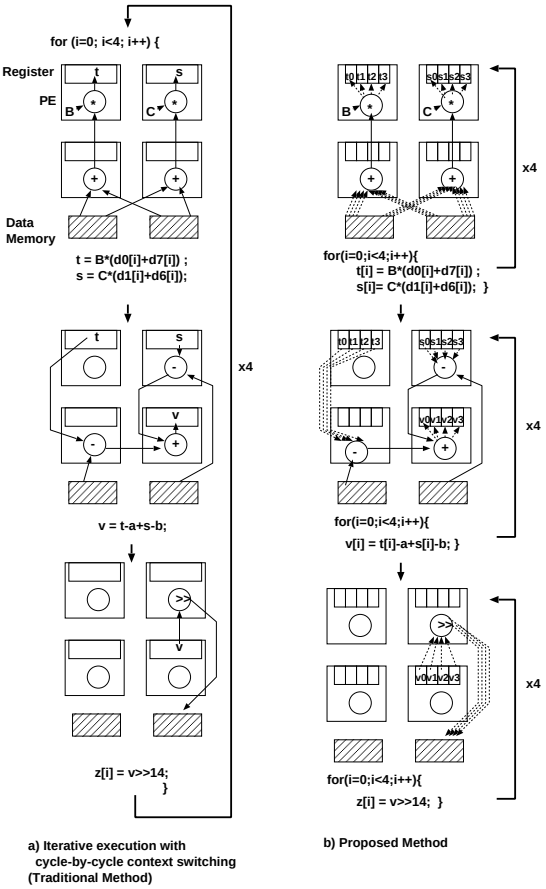

図 5 Loop Separation

しかし, この処理はループキャリ依存性がないので, 図 5b に示すようにそれぞれのステップにおけるループに置き換えることが可能である. ここで, 以下に示すハードウェアの助けがあればコンテキストを切り替えずに，演算処理を実行できる．

- ループを数える専用のループカウンタ
- ループカウンタによって切り替わる中間結果格納用レジスタアレイ
- ループカウンタに依存して適切なアドレスを発生してデータメモリを読み出すアドレスカウンタ. 処理の幅を広げるには単純なカウントアップだけではなく, ストライドアクセスなどが必要である.

ループの回数, すなわち一つのコンテキストを続けて実行できるクロック数は, 中間結果を格納するレジスタアレイのサイズによって決まる. コンテキストメモリの代わりにこのアレイが巨大化してしまうと元も

子もないので, ここでは 8 程度のサイズを想定する. これならば, 実用的な CGDRP で用いられているサイズであり, ハードウェア量も小さい. レジスタアレイのサイズを 8 とした場合には, 扱う対象のループを 8 づつに区切って分離する. これは, Loop Unrolling が Unroll するループの回数と類似しており, 割り切れない場合は後処理が必要となる.

上記のハードウェアの助けを前提に行うループの再構成をここでは Loop Separation と呼ぶ. Loop Separation は, ループを一定回数で区切る点以外にも, 中間結果を格納するレジスタを必要とする点でもループアンローリングと似ている. しかし, 並列性を高めるためにループを展開するのではなく, データパスを連続利用するためにループ構造を分離する点で異なる. キャッシュのヒット率を上げるために類似の方法が利用されるが[13], 目的が全く異なっており,Loop Separation はサポートハードウェアを必要とする. また, タスクの割り当てとは関係ないため, モジュロスケジューリング等の手法[14] とは共存可能である.

3.2 構成情報転送のさらなる高速化

Loop Separation を利用すれば, 一つのコンテキストに, 最低でもレジスタアレイ中のレジスタ数である 8 クロック分滞在することになる. 次に, この 8 クロック以内に構成情報の転送を終えるために, 転送時間をさらに高速化する手法を二つ提案する.

3.2.1 差分構成法

現在実行中のコンテキストと次のコンテキスト間に類似性があれば, 違っている部分だけを転送すれば転送時間を削減することができる. そこで図 6 に示す通り, コンテキストレジスタを結合網, コンスタント, 入力選択,ALU などに相当するフィールドに分割し, 各々のフィールドに対してフラグを持たせ, これが 1 の部分のみ転送を行う. この方式では, コンフィギュレーションレジスタは, ダブルバッファとしてまるごと入れ替えるのではなく, バッファとして使われているレジスタの 1 が立っているフィールドのみデータ転送を行うことでコンテキストスイッチを行う. このようなフィールド分けにより RoMultiC のマルチキャスト効率も向上することが知られている[12].

この差分構成法はハードウェア量のオーバーヘッドはフラグのみで小さい. しかし, 効果があるのあ連続する二つのコンテキスト間に類似性がある場合だけである.

3.2.2 Spare register

構成情報の量は, コンテキストによって差が大きく, 場合によってはバックグランド転送が, コンテキストの実行よりも早く終わる可能性もある. Spare regiter

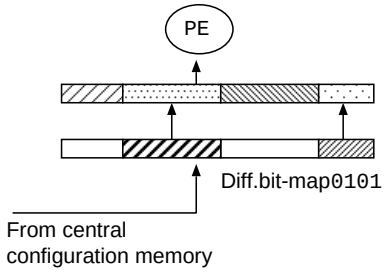

図 6 差分構成法

と呼ぶ第三のレジスタを設け, 空いた時間を利用して, マルチキャストが最もうまく働かず, 最も長い転送時間を要するコンテキストの構成情報を転送しておく.

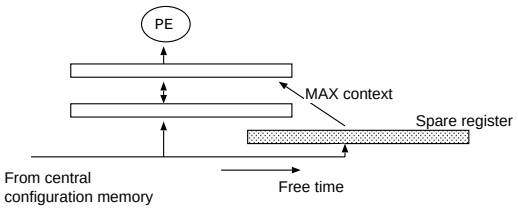

図 7 Spare register

図 7 に示すように, いくつかのコンテキストにおいて, 転送時間に余裕があれば, その分, 数多くの PE やスイッチングエレメントの Spare register に転送が終わっていることになり, 対象となるコンテキストへの構成情報の転送時間がその分節約できる. そのコンテキストへのスイッチ時は, 通常のバッファのスイッチではなく Spare register からの構成情報の転送が行われる.

この方式は一つのコンテキストに対する構成情報の転送時間を節約できるだけである. しかし, 多くのアプリケーションでは, ストールを引き起こすコンテキストの個数は多くないことから常に一定の効果を得ることができる. 一面, レジスタが一個増え, その分のハードウェア量が増える.

4. 提案方式の評価

4.1 対象モデル MuCCRA-3NC

4.1.1 PE アレイアーキテクチャ

まず, 評価の対象とする モデルアーキテクチャである MuCCRA-3NC(No-Context memory) について簡単に紹介する. MuCCRA-3NC は MuCCRA-3[15][16] を基に設計されており, アレイサイズ, データ幅等は MuCCRA-3 と同じである. すなわち, 4x4 の PE によるアレイ構造を持ち, データ幅は 16bit とした. 演算データを保持する MEM は, アレイ下部とアレイ上

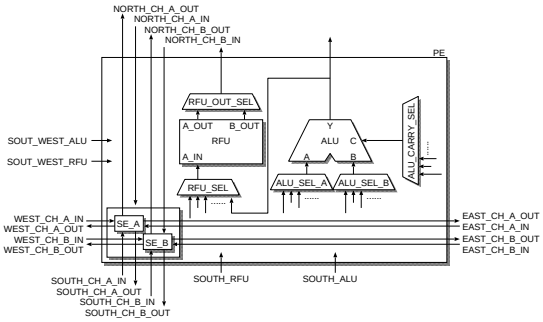

図 8 MuCCRA-3NC の PE

部両方に配置されており合計 8 つである.

PE 間を接続する結合網は，Switching Element(SE) によるアイランドスタイルに加え，近接の PE と接続する直結接続網との両方により構成される．アプリケーションのマッピング時に，近接の PE へデータを移動する場合には SE を再構成する必要のない直接接続網を利用し，遠い PE へデータを渡す場合には，SE を利用してアイランドスタイルの接続網を利用する．アイランドスタイルの接続網は，A,B の 2 チャネルが利用可能で，SE にて A から B，B から A へ乗り換えを可能とした．SE は配置配線のし易さから，PE 内のユニットとして持つようにした.

4.1.2 PE の構成

MuCCRA-3NC は，2 項の四則演算，乗算，ビットシフト，比較演算を行う ALU，PE 内外からのデータ選択または定数生成を行い，1 項のシフト演算，ビット反転をする 2 つの ALU_SEL，8 エントリの 2 リードポート，1 ライトポートの RFU によって PE は構成される．RF は，深さ 8 の FIFO として動作する FIFO モードを持つ．図 8 に MuCCRA-3NC の PE の内部構造を示す.

ALU_SEL_A,ALU_SEL_B, および RF の A_IN には，RF の出力，直接接続網，SE による接続網が接続されており，構成情報により選択が可能である．ALU の機能は単純な 13 命令のみとし，ビット反転などは ALU_SEL が行う．これにより各ユニットの機能分離が明確になり，アプリケーションマッピングを容易にする.

RFU は 8 エントリの 2r/1w のレジスタファイルで構成され, Loop Separation による中間結果の格納に用いる. しかし,Loop Separation を用いない通常のモードでは独立したレジスタとして中間結果の格納に用いることも可能である.

4.2 評価結果

4.2.1 評価環境

MuCCRA-3NC は MuCCRA-3 と同じく，富士通の 65nm CMOS セルライブラリを用いて設計した.Verilog HDL で記述し,Synopsys 社 Design Compiler 2007.12-SP3 を用いて合成を行った. 電力評価は, Prime-Time 2007.12-SP3. を用い, 遅延付き合成後シミュレーションを NC-Verilog 8.1 で行い, 動作するかどうかを調べた.

アプリケーションプログラムとしては, ループにまたがる依存性を持たないアプリケーションとして, 離散コサイン変換 (DCT: 64 データ), アルファブレンダ (ALPHA: 64 ピクセル), セピアフィルタ (SEPIA: 32 ピクセル), エッジフィルタ (EDGE：64 ピクセル) の 4 つを用いた. また, ループにまたがる依存性を持つプログラムとして,64 個のデータの自乗和を計算する SQSUM を用いた.

アプリケーションの実装にはリターゲッタブルコンパイラ Black Diamond[17] を用いた. このコンパイラには Loop Separation は実装されていないため, ループの変換は人手で行った.

4.2.2 実行時間

評価は,MuCCRA-3NC の基本構成 (Fund.), 差分設定機構を持つもの (Diff.) および spare register (Spare) の三種類で行った. いずれも 50MHz での動作を仮定した.

図 9 は,32 コンテキストを持つ MuCCRA-3 との実行時間の比較を示す．MuCCRA-3NC はコンテキストメモリを持たないため, コンテキストレジスタに対する構成情報の設定が間に合わないと, その時間がオーバーヘッドとなり, 実行時間が増加する．ループ依存性のないものの中ではコンテキストの構成が複雑な DCT が最もオーバヘッドが大きく,23%である. しかし差分構成法を用いることで 12%にまで縮小できる.Spare register も効果があり,13%にまで縮小できている．ALPHA と Sepia は構成情報が単純であるため, 基本方式でもさほどオーバヘッドは生ぜず,Spare register を用いることでほとんどなくすることができる. ただしコンテキスト間の類似性は少ないため, 差分構成法は効果がない．EDGE はメモリアクセスがボトルネックになるプログラムで, 並列処理できる PE 数が少ないため, 基本方式でもオーバーヘッドは生じない.

SQSUM はループ間の依存性がないため,Loop Separation を効果的に用いることができず, オーバーヘッドは最も大きく 29%になっている. この方式では差分構成法が有効で, オーバーヘッドをおよそ半分に減らしている.

4.2.3 ハードウェア要求量

MuCCRA-3, MuCCRA-3NC の基本方式, 差分構

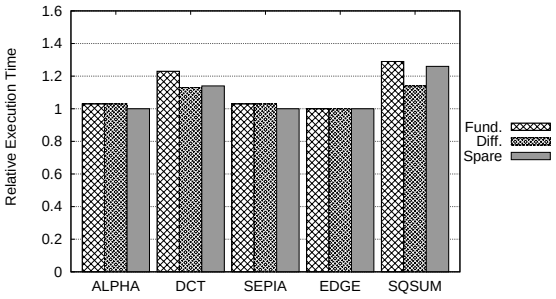

図 9 MuCCRA-3NC の相対実行時間

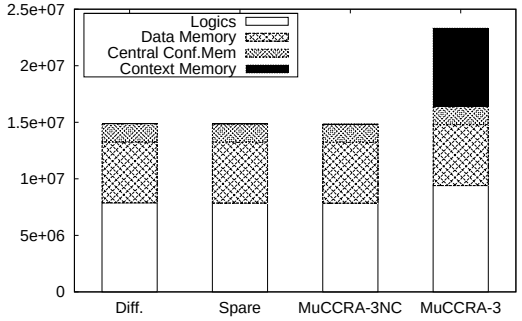

図 10 Hardware requirement (μm^2)

成法,Spare register のハードウェア要求量を比較して図 10 に示す．MuCCRA-3NC は MuCCRA-3 に比べてコンテキストメモリを持たず, その代わりにコンテキストレジスタのダブルバッファが用いられている．ここでは, 半導体の面積に換算してあり,Logic はダブルバッファとして使われているレジスタを含むメモリ以外の面積を示す.MuCCRA-3NC は MuCCRA-3 に比べて Logic 量も減っている．これは, タスク制御が必要な MuCCRA-3 に比べて, 全てが構成情報レジスタのダブルバッファで実行可能な MuCCRA-3NC の方がコンテキスト制御が単純になるためである. コンテキストメモリが存在しないことと合わせて,MuCCRA-3 の 63%の面積ですんでいる. 差分構成法と Spare register は基本方式に比べて若干ハードウェア量は増加するが,1%前後である．

4.2.4 消費エネルギー

図 11 に MuCCRA-3NC の消費エネルギーを MuCCRA-3 を 1 として相対値で示す．差分構成法および Space Register の利用は，消費エネルギーにほとんど影響を与えなかったため, 図ではその差は示していない．

各部の動作率が同じならば, 消費エネルギーの相対値は, 面積の相対値と実行時間の相対値の積になることが見積られる．しかし, 今回,SQSUM を除いては,

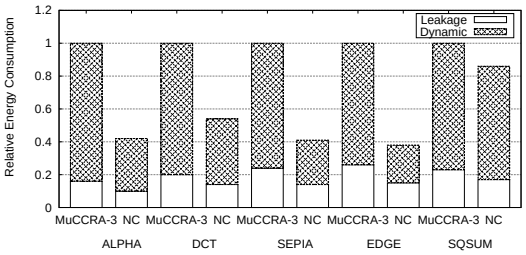

図 11 相対消費エネルギー

この値以上にエネルギーの相対値が小さくなっている．これは,Loop Separation によってデータパスの変更頻度が小さくなったため, これに要するエネルギーが節約されたためである．SQSUM はループにまたがる依存性を持つため,Loop Separation を適用できる部分が少ないためこの効果が少なくなっている．

5. ま と め

本報告では, マルチコンテキスト型 CGDRP から, コンテキストメモリを削減するための手法について検討した. コンテキストメモリの代りにダブルバッファを持たせ, 片方の構成情報で実行中に, 次の構成情報をバックグランドで転送する．コンパイラと付加ハードウェアを用いた Loop Separation により, 一つのコンテキストを必ず一定以上のクロック数連続して実行するように変換する. さらに, 構成情報のマルチキャスト機構に差分再構成法,Spare register を組みあわせてバックグランド転送のクロック数を削減した．

評価の結果, ループ依存性がないプログラムでは, 最大 12-13%の性能低下で面積を 63%に, 消費電力を 40%にすることに成功した. ループ間の依存性があるプログラムでは, 差分構成法が有効で今回評価した SQSUM では 18%程度まで性能低下を抑えることができる. 現在，Black Diamond コンパイラは Loop Separation を自動的に行う機能がなく，今回の評価は人手によりループ構造を分離している．この自動化が今後の課題である. また，Loop Separtion の通用が難しい複雑なループを持つアプリケーション実行時のストール削減手法をさらに開発する必要がある．

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．

また, 本チップ設計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の協力で行われたものである．

参 考 文 献

1) 末吉敏則, 天野英晴: リコンフィギャラブルシステム, オーム社 (2005). (編著).
2) Suzuki, M., Hasegawa, Y., Yamada, Y., Kaneko, N., Deguchi, K., Amano, H., Anjo, K., Motomura, M., Wakabayashi, K., Toi, T. and Awashima, T.: Stream Applications on the Dynamically Reconfigurable Processor, *Proc. of the 3rd IEEE Int'l Conf. on Field Programmable Technology (ICFPT2004)*, pp. 137–144 (2004).
3) F.J.Veradas, M.Scheppler, W.Moffat, B.Mei: Custom Implementation of the Coarse-Grained Reconfigurable ADRES architecture for multimedia Purposes, Proc. of International Conference on Field Programmable Logic and Applications (FPL05), Springer, Berlin, pp. 106–111 (2005).
4) H.Amano and Y.Hasegawa and S.Tsutsumi and T.Nakamura and T.Nisimura and V.Tanbunheng and A.Parimala and T.Sano and M.Kato: MuCCRA Chips: Configurable Dynamically-Reconfigurable Processors, *Proc. of ASSCC 2007*, pp. 384–387 (2007).
5) Y.Saito and T.Sano and M.Kato and and V.Tunbunheng and Y.Yasuda and H.Amano: A Real Chip Evaluation of MuCCRA-3: A Low Power Dynamically Reconfigurable Processor Array, *Proc. of Int'l Conf. on Engineering of Reconfigurable Systems and Algorithms (ERSA)* (2009).
6) X.-P. Ling, H. Amano: WASMII: A Data Driven Computer on a Virtual Hardware, Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines(FCCM'93), Springer, Berlin, pp. 33–42 (1993).
7) Amano, H., Inuo, T., Kami, H., Fujii, T. and Suzuki, M.: Techniques for Virtual Hardware on a Dynamic Reconfigurable Processor -An approach to tough cases-, *Proc. of the 14th Int'l Conf. on Field Programmable Logic and Applications (FPL2004)*, pp. 464–473 (2004).
8) 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, IEEE Trans. on Computers, Springer, Berlin, pp. 465–480 (Vol.49, no.5, 2000).
9) H. Shimit, D. Whelihan, A.Tsai, M.Moe, B.Levine, R.Taylor: PipeRench: A virtualized programmable datapath in 0.18 micron technology, Proc. of CICC, pp. 63–66 (2002).
10) Yoojin Kim, Rabi N.Mahapatra, Ilhyn Park, and Kiyoung Choi: Low Power Reconfiguration Technique for Coarse-Grained Reconfigurable Architecture, *IEEE Trans on VLSI Systems VOL.17, NO.5, MAY 2009*, pp.593–603 (2009).
11) Tunbunheng, V., Suzuki, M. and Amano, H.: RoMultiC: Fast and Simple Configuration Data Multicasting Scheme for Coarse Grain Reconfigurable Devices, *Proc. of the 4th IEEE Int'l Conf. on Field Programmable Technology (ICFPT2005)*, pp. 129–136 (2005).
12) Tsutsumi, S., Tunbunheng, V., Hasegawa, Y., Parimala, A., Nakamura, T., Nishimura, T. and Amano, H.: Overwrite Configuration Technique in Multicast Configurati on Scheme for Dynamically Reconfigurable Processor Arrays, *Proceedings of International Conference on Field Programmab le Technology (ICFPT2007)*, pp. 273–276 (2007).
13) R.Allen and K. Kennedy: Optimizing Compilers for Modern Architectures, Elsevier (2001).
14) B. Mei and S. Vernalde and D.Verkest and H.DeMan and R.Lauwereins: Exploiting Loop-Level Parallelism on Coarse-Grained Reconfigurable Architectures Using Modulo Scheduling, *Proc. of Design, Automation and Test in Europe COnference and Exhibition (DATA'03)* (2003).
15) Y.Saito, T.Sano, M.Kato, V.Tunbunheng, Y. Yasuda, H.Amano: A Real Chip Evaluation of MuCCRA-3: A Low Power Dynamically Reconfigurable Processor Array, *Proc. of ERSA 2009*, pp. 283–286 (2009).
16) M.Kimura, Y.Saito, T.Sano, M.Kato, V.Tunbunhen, Y.Yasuda, H.Amano: Low Power Image Processing using MuCCRA-3: A Dynamically Reconfigurable Processor Array, *Proc. of IEEE Int'l Conf. on Field Programmable Technology (FPT)*, p. To appear (2009).
17) V. Tunbunheng and H. Amano: BlackDiamond: a Retargetable Compiler Using Graph with Configuration Bits for Dynamically Reconfigura ble Architectures, *Proc. of The 14th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, pp. 412–419 (2007).