動的リコンフィギャラブルプロセッサ MuCCRA-1 の実装と評価

長谷川 揚平[†] 堤 聡[†] 中村 拓郎[†]
西村 隆[†] 佐野 徹[†] 加東 勝[†]
斎藤 正太郎[†] 天野 英晴[†]

Multi-Core Configurable Reconfigurable Architecture (MuCCRA) は，アプリケーションに対して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．本論文では，MuCCRA のプロトタイプチップ MuCCRA-1 のアーキテクチャおよび面積，性能，消費電力の評価結果について述べる．MuCCRA-1 は，ローム社の 0.18μm 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．単一コアの小規模な構成であるが，コンフィギュレーションデータの高速転送と，仮想ハードウェア機構を備えている．PE アレイは典型的なアイランドスタイルの構造をもち，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．MuCCRA-1 上に実装したアプリケーション評価より，225MHz で動作する信号処理プロセッサと比較して最大で約 2 倍の実行速度を達成した．

Design and Implementation of the Dynamically Reconfigurable Processor MuCCRA-1

Yohei Hasegawa,[†] Satoshi Tsutsumi,[†] Takuro Nakamura,[†]
Takashi Nishimura,[†] Toru Sano,[†] Masaru Kato,[†]
Shotato Saito[†] and Hideharu Amano[†]

Multi-Core Configurable Reconfigurable Architecture (MuCCRA) aims to establish architectural techniques to develop low-power multi-core configurable dynamically reconfigurable processors. In this paper, MuCCRA-1, the first prototype of MuCCRA is introduced, and its area, speed, and power are evaluated. The MuCCRA-1 is implemented with Rohm's 0.18μm CMOS technology. On the 5mm-square die, 4 × 4 24-bit PE array, 4 multipliers, and 4 distributed shared memory modules are mounted. Although it is a small single core, it provides a high speed configuration mechanism and virtual hardware mechanism. PE array structure is a typical island-style architecture on which the trade-off between performance, power consumption, and cost can be analyzed. The evaluation results show that the DCT implemented on the MuCCRA-1 is 2 times faster than the TI's DSP operating at 225MHz.

1. はしがき

近年，発展の目覚ましい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と，ソフトウェアの柔軟性を併せもつデバイスとして注目されている[1)]．特に近年の動的リコンフィギャラブルプロセッサ[2)~5)] は，4〜32bit の粗粒度の Processing Element (PE) と分散メモリモジュールを二次元アレイ状に配置した構成をとり，各 PE の命令と PE 間の接続を動的に変更することが可能である．

動的リコンフィギャラブルプロセッサは多数の PE と分散メモリモジュールのアレイによる並列処理により，特にマルチメディア処理やネットワーク処理において高い処理性能を示す．また，マルチコンテキスト型の動的リコンフィギャラブルプロセッサでは，PE アレイ上で実現される回路構成情報をコンテキストと呼び，複数のコンテキストを内部のメモリに保持することが可能である．また，これを 1 クロック程度の短い時間で切り替えることができる．

対象のアプリケーションは複数のコンテキストに分割され，必要なコンテキストを必要なときのみ読み出して実行することができる．この動的再構成機能により，従来の Field Programmable Gate Array (FPGA) に代表される細粒度のリコンフィギャラブルデバイスの課題である面積効率，電力効率を改善することができると期待されている．

近年では，実際の商用のデバイスを用いたアプリケーション開発やアーキテクチャ検討が進んでいる．我々もこれまで NEC エレクトロニクス社が開発した Dynamically Reconfigurable Processor (DRP) を対象にして，そのハードウェア量，性能，消費電力に関して様々な解析を行ってきた[6)]．その結果，以下の点が明らかになってきた．

- PE の内部構造，乗算器の構成，分散メモリの量，PE 間接続はアプリケーションによって適不適がある．このため，アプリケーションの性質に適合した PE アレイを

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

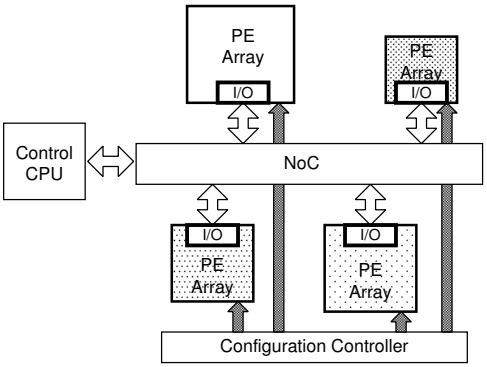

図 1 MuCCRA の概観
Fig. 1 MuCCRA Overview

利用するのが望ましい．すなわち，一般的なプロセッサ同様，動的リコンフィギャラブルプロセッサも System-on-a-Chip (SoC) に組み込む際にコンフィギャラブルであることが望ましい．このため，様々な PE アレイの構成について，そのトレードオフを解析する必要がある．

- PE アレイのサイズは，やや小さめにして，コンテキスト数を多くした方が，多くの問題に対して面積・電力効率が良い．高い性能が要求される場合は，複数の PE アレイをパイプライン的に用いるかスレッドレベルの並列性を利用する方法が優れている．すなわち，小規模の PE アレイを複数用いたマルチコア構成に関して，より詳細に検討する必要がある．
- 動的リコンフィギャラブルプロセッサの消費電力に関して，デバイス，アーキテクチャ，CAD を含めた解析と，それに基づく省電力技術の開発が必要である．これには，回路レベルに近い技術が必要になる．
- 多数のコンテキストから成る複数のタスクをなるべく小さなオーバーヘッドで制御する現実的な方法を検討する必要がある．

Multi-Core Configurable Reconfigurable Architecture (MuCCRA) プロジェクトは，上記の課題解決に向け，詳細なアーキテクチャ解析と新たな手法の提案を目的とし，実際にチップを開発して解析および実証を行うものである．MuCCRA は，**図 1** に示すように，複数の PE アレイ (動的リコンフィギャラブルプロセッサコア) が Network-on-Chip (NoC) で接続されている構成をもつ．MuCCRA の PE アレイは，アプリケーションの特性に応じてコンフィギャラブルな構成を生成する．すなわち，コアの個数，PE アレイのサイズ，PE の内部構成等をアプリケーションの要求性能，電力，半導体面積に依存して，様々な構成を取ることを可能とする．一方で，それぞれのコアの I/O を介したデータ転送制御と，構成情報の転送制御，これに関連するコンテキスト制御は，すべての PE アレイ で共通化する．

我々は，MuCCRA プロジェクトの第一歩として，4 × 4 のサイズの PE アレイから成るシングルコアのプロトタイプ MuCCRA-1 を，ローム社の 0.18μm CMOS プロセスを用いて実装した．本論文では，MuCCRA-1 のアーキテクチャを紹介し，実装結果による面積評価と，MuCCRA-1 上に実装したアプリケーションに基づいた性能評価を示す．

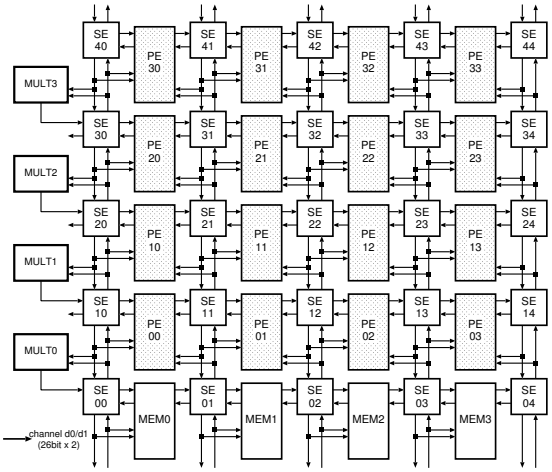

図 2 PE アレイ
Fig. 2 PE Array Architecture

本論文の構成は以下の通りである．次節では MuCCRA-1 のアーキテクチャについて述べる．続いて節 3 において，MuCCRA-1 の制御機構およびアプリケーションの実行モデルについて述べる．そして，節 4 で MuCCRA-1 の LSI 実装について述べ，節 5 でプリケーションによる性能評価について述べる．最後に，節 6 で結論を述べる．

2. MuCCRA-1 のアーキテクチャ

MuCCRA-1 は，プロジェクトの最初のプロトタイプチップとして，性能，面積，消費電力測定の基準とすることを目的に開発した．このため，PE アレイ自体の構成は，なるべく現在の動的リコンフィギャラブルプロセッサの多くで採用されている一般的な方式を採用した[7]．一方で，コンテキスト制御，構成情報の設定，入出力等は，MuCCRA のマルチコア全てで共通に用いることのできる新しい方式を提案し，これを実証することを目的としている．

2.1 PE アレイ

MuCCRA-1 の PE アレイは，ホモジーニアスな構造の比較的小規模な正方アレイで，乗算器，分散メモリをアレイの端にもつ構造を採用した. これは，DRP-1[2]，PACT-Xpp[4] などで用いられている構成である．**図 2** に示すように，4 × 4 の PE の左端および下端にそれぞれ乗算器 (MULT) 4 個，分散メモリ (MEM) 4 個を接続する．MEM は，24bit × 256 エントリで，後述するようにダブルバッファとして 2 セット実装されている．このため，片方のメモリで演算を行っている間に，もう一方のメモリでチップ外部との入出力を行うことができる．チップ外部との通信は，入力・出力それぞれ 64bit 幅である．MULT は 24bit 同士を演算し，結果は下位の 24bit のみを有効としている．

PE アレイにおける PE 間の結合網は，FPGA 同様のアイランドスタイルを採用したが，FPGA とは異なり各リンクは

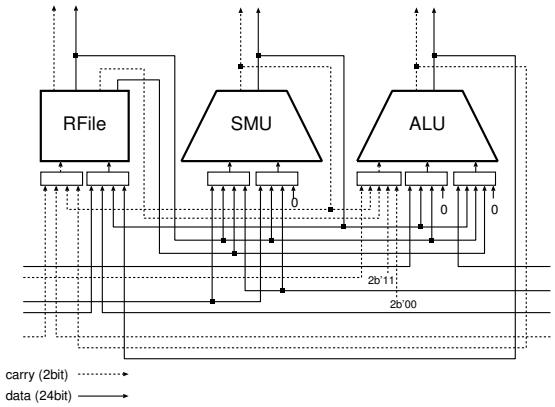

図 3 PE コアの構成
Fig. 3 PE Core Architecture

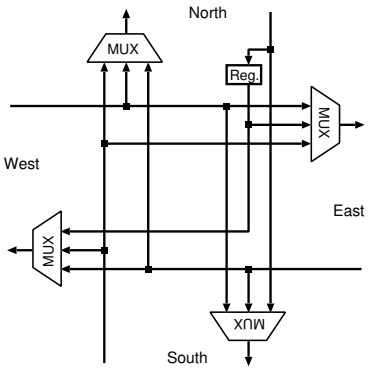

図 4 スイッチ (SW) の構成
Fig. 4 Switch (SW) Architecture

単方向とした．すなわち，配線領域間に Switching Element (SE) を設けて水平・垂直方向のネットワークを構成し，コネクションブロックによって，PE の入力と出力を接続する．SE 間インターコネクトは、FPGA 同様、構成情報によって静的に決まる．従って，それぞれのコンテキストの最大遅延時間は PE 間の接続経路に影響を受ける．MuCCRA-1 では 24bit の二系統のリンク (d0, d1) を上下・左右の方向にそれぞれ設けている．図中では省略されているが，MEM への書き込みデータは，最上位のスイッチからのフィードバックラインによって送られる．

動的リコンフィギャラブルプロセッサでは，アイランドスタイルと共に PE 間を直結する結合方式も用いられるが，乗算器，分散メモリを PE アレイの端に置いた場合，アイランドスタイルが有利と考え，MuCCRA-1 ではこの方法を採用した．直結方式も今後，実装を行い比較する予定である．

2.2 PE の構成

MuCCRA-1 の各 PE は，実際に処理を行う PE コアと，コネクションブロック，コンテキストメモリから構成される．

MuCCRA-1 の PE コアの構成を**図 3** に示す．PE コアは，画像処理向けに RGB 画像データを扱い易くするため，24bit 構成の単純なプロセッサとした．PE は，多くの動的リコンフィギャラブルプロセッサと同様に，シフト，マスク，定数値供給を行う Shift and Mask Unit (SMU)，加減算，比較，論理演算を行う Arithmetic Logic Unit (ALU)，およびレジスタファイル (RFile) より構成される．ALU は，24bit を 12bit × 2 のデータとして扱い，2 つのデータを同時に計算する half-word 演算を行うことができる．RFile は 8 個のレジスタをもつ 2 ポート構成のレジスタファイルで，A ポートは読み書き，B ポートは読み出し専用である．SMU の出力は ALU の入力に接続することは可能だが，この逆は許さないことで，組み合わせ的なループ構造が生成されることを防いでいる．一方，RFile は ALU，SMU の入出力全てに接続することが可能である．

コンテキストメモリは，PE コアの命令と結合網との接続情報からなる構成情報を保持するメモリで，64bit × 64 エントリである．各 PE の 1 コンテキストあたりの構成情報は 64bit であり，各 PE のコンテキストメモリは最大で 64 コンテキスト分の構成情報を保持可能である．

2.3 SE の構成

MuCCRA-1 の SE は，各リンク (d0, d1) に対して，各方向からの入力をマルチプレクサで選択して出力するスイッチ (SW) と，マルチプレクサがどの入力を出力するかを決定する構成情報を保持するコンテキストメモリから構成される．マルチプレクサで構成するスイッチは，トランスファーゲートにより双方向の転送を許す FPGA のスイッチに比べて面積が大きいが，構成変更時に出力同士が短期間衝突する問題を避けることが可能なことから，動的リコンフィギャラブルプロセッサでアイランドスタイルを採用する場合は一般的な方法である．各 SE のコンテキストあたりの構成情報は 16bit であり，各 SE のコンテキストメモリは，PE と同様，最大で 64 コンテキスト分の構成情報を保持できる．

図 4 に SW の構成を示す．South，East，West からの入力はそのまま出力されるが，North からの入力で East，West に向かう場合には，一度内部レジスタに格納され，次のクロックで出力される．これは，結合網中で組合せ回路のループ構造が発生するのを防ぐためである．

2.4 PE 間結合網

図 5 に MuCCRA-1 の PE 間結合網を示す．各 PE は内部の入力用コネクションブロック (PICKIN) により，2 系統 (d0, d1) ある両側の結合網の垂直方向から信号を選択して取り入れる．結合網内の組み合わせループを避けるため，上から下へ向うデータ線から信号を取り入れる場合，必ず RFile にデータを格納する必要がある．一方，出力は出力用のコネクションブロック (PICKOUT) により水平方向のリンクに対して行う．ALU，SMU，RFile すべてのユニットの出力を左右どちらの方向にも取り出すことができる．

大規模な組み合わせ回路のデータパスを構成する場合，MuCCRA-1 では下端の分散メモリ (MEM) からデータを取り出し，これを上方向に流していき，中間結果を各 PE の RFile に格納したり，最終結果を最上位 SE からフィードバックラインを経由して MEM に格納する．

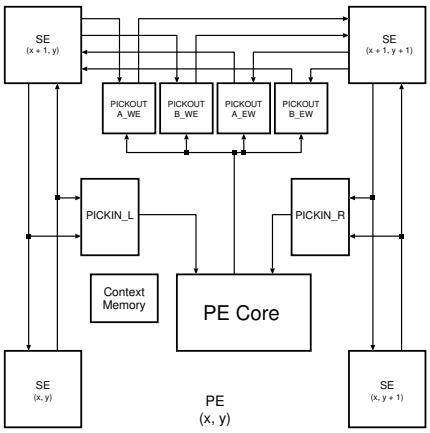

図 5 PE 間結合網
Fig. 5 Inter-PE Connections

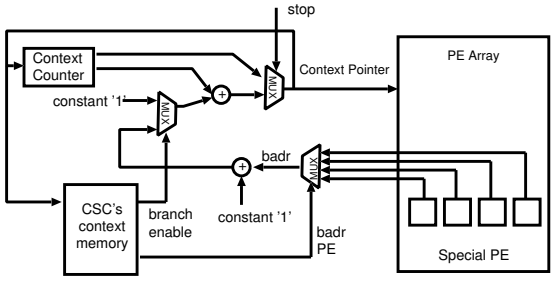

図 6 Context Switching Controller (CSC)
Fig. 6 Context Switching Controller (CSC)

3. MuCCRA-1 の制御機構

3.1 MuCCRA のコンテキスト制御

MuCCRA-1 のコンテキストの切り替え制御は，単純なカウンタベースの Context Switching Controller (CSC) によって行なわれる．MuCCRA-1 における PE, SE, MULT, MEM といったモジュールは，リコンフィギャラブルモジュール (Reconfigurable Module) であり，それぞれがコンテキストメモリと呼ばれる構成情報を保持するメモリをもつ．各モジュールは CSC が生成するコンテキストポインタに従って，コンテキストメモリから自身の構成情報を読み出して動作する．CSC 自身もリコンフィギャラブルモジュールであり，PE や SE と同様，コンテキストメモリから構成情報を読み出し，コンテキストを切り替える．

コンテキストポインタは，**図 6** に示すように，現在実行中のコンテキストポインタを保持するコンテキストカウンタを元に決定される．コンテキスト分岐をしない場合には，単純にインクリメントしていく．一方，コンテキスト分岐をする場合，特定の PE から相対分岐アドレス (badr) をコンテキストカウンタの値に加算する．MuCCRA-1 では，PE アレイ上の右端の 4 つの PE を Special PE として，RFile の出力を相対分岐アドレスとして出力することができるように拡張している．4 つの Special PE のうち，どれが相対分岐アドレスを出力するかは，CSC の構成情報で決定する．

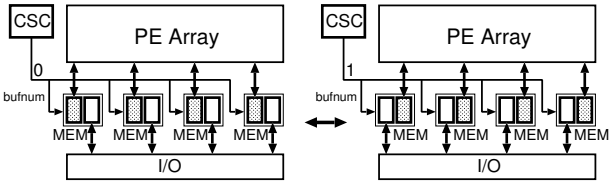

図 7 MuCCRA-1 の入出力インタフェース
Fig. 7 IO Interface of MuCCRA-1

3.2 入出力インタフェース

MuCCRA-1 の入出力インタフェースは，すでに我々が提案した独立 I/O コントローラとダブルバッファを用いる方法[8] に基づく．この手法では，分散メモリモジュールを 2 種類に分けてダブルバッファとして利用し，PE アレイとは独立に動作する I/O コントローラにより制御する．

MuCCRA-1 では，コンテキストスイッチと連動してダブルバッファを切り替えることとし，CSC がダブルバッファの制御を行う．MuCCRA-1 における入出力インタフェースの概略を**図 7** に示す．PE アレイはまず，ダブルバッファの一方を用いて演算を行い，この間，もう一方に対して次のデータストリームの入出力を行う．演算が終了したら，分散メモリの接続を入れ替え，再び入出力と演算を並行して行う．

上記の構成により，PE アレイ本体の演算とデータ入出力は，完全に同時動作するため，入出力に要する時間が演算時間よりも大きくなければ，これを完全に隠蔽可能である．

3.3 MuCCRA-1 のタスク制御

MuCCRA-1 は，各モジュールが 64 コンテキストを保持可能なコンテキストメモリをもっている．これは他の動的リコンフィギャラブルプロセッサに比べかなり大きい (DRP-1[2] は 16, ADRES[5] は 32, DAPDNA-2[3] は 4) が，MPEG や JPEG などのプリケーション全体を 64 コンテキスト以内で実装することは難しい．そこで，アプリケーション全体を複数のタスクに分割し，タスク単位で PE アレイに構成情報をロードし，実行する必要がある．

MuCCRA-1 のタスク管理は，Task Configuration Controller (TCC) が行う．TCC は内部に 512 エントリの構成情報メモリをもつ．MuCCRA-1 では，全タスクの構成情報を TCC の構成情報メモリに格納しておき，TCC は実行するタスクの必要な構成情報のみ各モジュールのコンテキストメモリに転送する．

TCC は，Task Flow Table (TFT) と呼ばれるタスク情報を参照して構成情報の転送を行う．TFT のフォーマットと，タスクフローの例を**図 8** に示す．TFT の各タスクに対するエントリの詳細は以下の通りである．

- start: 対象タスクに対応する構成情報が保持されている構成情報メモリの先頭番地
- size: 対象タスクの構成情報のエントリ数
- context size: 対象タスクの利用コンテキスト数
- branch task: タスク分岐時の分岐先タスク番号
- default task: タスク分岐しない場合のタスク番号
- jobend: 対象タスクで当該ジョブの終了を行うかどうか

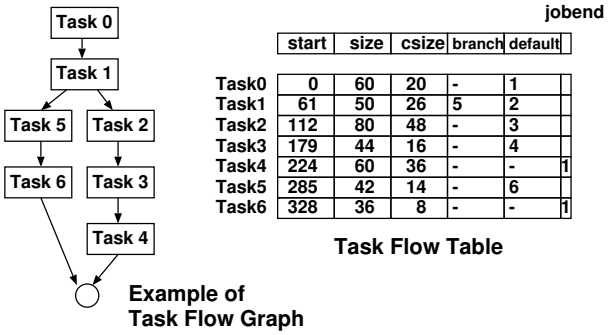

図 8 Task Flow Table (TFT) の例
Fig. 8 An Example of Task Flow Table (TFT)

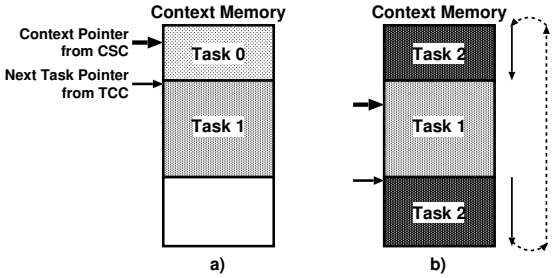

図 9 タスクのコンフィギュレーションの実行例
Fig. 9 An Example of Task Configuration

を示すフラグ

MuCCRA-1 では，あるタスクの実行中に，TFT の default task で示された番号のタスクが PE アレイ内の各コンテキストメモリに転送される (バックグランド転送)．図 8 に示した例では，まず，Task 0 の実行中に Task 1 の構成情報を転送する．Task 1 の転送が終了した時点の，各モジュールのコンテキストメモリの状態を**図 9**(a) に示す．

コンテキストメモリのエントリ数 64 に対して，この時点で転送されたコンテキスト数は Task 0 と Task 1 の和で 46 なので，コンテキストメモリには空きがあるが，さらに次のコンテキストを転送することはしない．すなわち，あるタスクの実行中には，TFT の default task で示された番号のタスクのみをロードし，ロードが終了すると TCC はサスペンドする．

Task 0 の実行が終了すると，PE アレイから TCC にタスク終了信号が送られる．このとき，Task 1 のロードが終わっていれば即座に Task 1 の実行を開始すると共に，Task 1 の default task である Task 2 の先行ロードを開始する．ロードする番地は，Task 1 の最後のコンテキストの直後からとなる．すなわち，64 コンテキストは 図 9(b) に示すように，サイクリックバッファ状に使われることになる．ここで，Task 1 (26) と Task 2 (48) の必要コンテキスト数を合わせると 64 を越えてしまうため，Task 2 のコンテキストを全てロードすることができない．この場合は，Task 2 を 38 コンテキスト分だけロードし，TCC はサスペンドする．

また，タスク分岐をする場合には，コンテキスト分岐の場合と同様に，PE アレイ上の右端の 4 つの Special PE のうちのいずれかが，RFile の出力をタスク分岐信号として出力することができる．どの Special PE が出力するかは，CSC の構成情報で指定することが可能である．

Task 1 の実行終了時に，PE アレイからタスク分岐信号が送られてこなかった場合には，Task 2 を実行するため，先行ロードできなかった残りの 10 コンテキストをロードする．この間，PE アレイはサスペンドする．Task 1 の実行終了時にタスク分岐信号が送られてきた場合には，次は TFT の branch task に示された Task 5 が実行されなければならない．この場合，PE アレイはサスペンドし，TCC は Task 5 のコンテキストを必要無くなった Task 2 のコンテキストに上書きする形でロードする．

この手法では PE アレイ上の各モジュールがもつコンテキストメモリのエントリ数を超えるアプリケーションの実装が可能であり (仮想ハードウェア)，さらにコンテキスト配送のレイテンシを大幅に隠蔽できる．ただし，以下の 3 つのケースでストールが発生する．

(1) 現在のタスクと先行ロードするタスクのコンテキスト数の和が，コンテキストメモリのエントリ数 64 を超えたとき
(2) タスク分岐が発生したとき
(3) 先行ロードが終了する前に，現在のタスクの実行が終了したとき

また，個々のタスクはコンテキストメモリのエントリ数 64 に収まる必要がある．したがって，大きなタスクは 64 コンテキストに収まるように分割する必要がある．

3.4 RoMultiC によるコンテキスト配送

MuCCRA-1 は，各モジュールに小規模のコンテキストメモリを分散してもつ．さらにチップ上には，構成情報メモリをもち，立ち上げ時にチップ外部より構成情報メモリに対して，全モジュールで利用する構成情報を転送する．実行開始後は必要に応じてこの構成情報メモリから各モジュールのコンテキストメモリに構成情報を転送する．構成情報メモリから，各コンテキストメモリへのデータ転送は，基本的にはバスを用いて逐次的に行うが，ここで，すでに我々が提案している RoMultiC[9] 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．

RoMultiC は，PE アレイ上のモジュールに逐次的な番号を与え，これを指定して構成情報を送るのではなく，**図 10** に示すように，縦横のビットマップを用いる方法である．共通バス上の構成情報は，縦横のビットマップが共に 1 であるモジュール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，構成情報の転送に要する時間を削減することが可能である．また，マルチキャストするデータは共有されるため，TCC 内の構成情報メモリに格納されるデータ量は，全ての構成情報を保持する通常の方法に比べて少なくなる．

4. MuCCRA-1 の実装

MuCCRA-1 は，ローム社の 0.18μm CMOS プロセスを

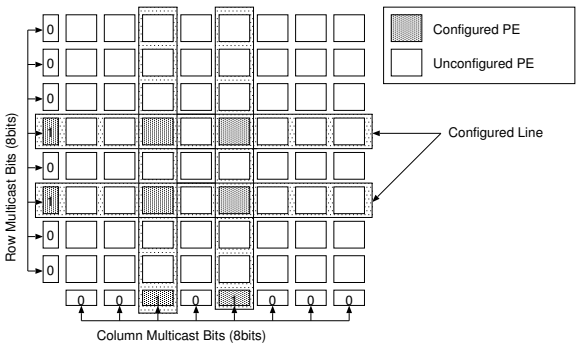

図 10 RoMultiC マルチキャスト法
Fig. 10 RoMultiC: A Context Multicasting Scheme

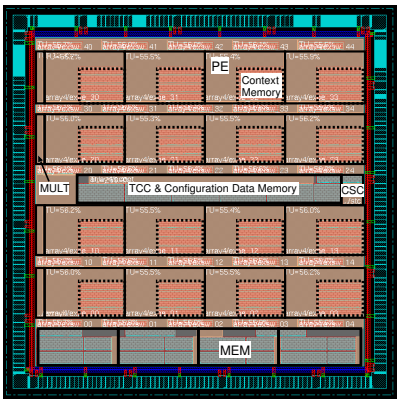

図 11 MuCCRA-1 のフロアプラン
Fig. 11 Layout of MuCCRA-1 Chip

表 1 MuCCRA-1 の使用セル数および面積
Table 1 Cell Usage and Area of MuCCRA-1

使用セル数	121305
ゲート数	1125231
PE アレイ面積	10.89 mm^2
メモリ総面積	7.33 mm^2

用いて実装した．設計には Verilog-HDL を用い，VDEC でサポートされるシノプシス社 Design Compiler 2006.06-SP2 を用いて論理合成を行った．また，レイアウト，フロアプラン，配置配線には，ケイデンス社の SoC Encounter 5.2 を用いた．**図 11** にコア中心部のフロアプラン図を示す．図 2 のアレイ構成をほぼそのままの形で配置し，中心部に構成情報メモリを配置している．

表 1 に，配置配線後のセル数および面積を示す．PE アレイの総面積に対して，メモリの総面積 (データメモリ，コンテキストメモリ，構成情報メモリを含む) は約 67% である．なお，メモリ総面積のうち約 71% (コア面積の約 46%) は，各モジュールがもつコンテキストメモリの面積である．本実装では各モジュールが 64 コンテキストを保持可能であるが，これはローム社のメモリライブラリがサポートする最も小さいエントリ数が 64 であったためで，最適なコンテキスト数は，トレードオフを評価した上で検討する必要がある．

図 12 に，PE アレイ全体の面積における各モジュールが

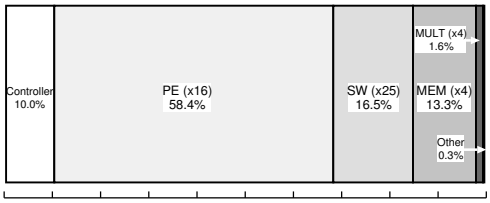

図 12 各モジュールの面積の割合
Fig. 12 Area Breakdown of each Reconfigurable Modules

占める面積の内訳を示す．各面積は各モジュールがもつコンテキストメモリの面積を含む値である．図中の Controller は，CSC, TCC および構成情報メモリを含む．MEM はダブルバッファ方式のデータメモリ，MULT は乗算器である．

図 12 より，PE の面積がアレイの半分以上を占めていることがわかる．また，PE 1 つあたりの面積のうち，コンテキストメモリの占める割合は約 55.7% となった．これより，64 コンテキスト分のコンテキストメモリの面積が，1 PE のロジックのみの面積とほぼ等しいということがわかる．また，1 PE あたりのロジックの面積に対する 1 コンテキスト分のコンテキストメモリの面積の割合は 0.019 となった．この数値は，マルチコンテキスト構成を用いることで実現可能なハードウェアの面積と，このための構成情報を格納するためのメモリの面積の比であり，この数値が大きいと，動的再構成による面積効率の改善効果が少なくなる．MuCCRA-1 の値は，IMEC 社の ADRES の値である 0.031[5] よりも小さく，高い面積効率が期待できる．

ただし，今回の実装では PE の面積について更に最適化の余地があることから，実際にはコンテキストメモリの割合は更に大きくなると考えられる．また，Controller の面積のうち，構成情報メモリの占める割合は約 86.8% (PE アレイ全体の 8.1%) である．これより，CSC，TCC の占めるロジック面積はごくわずかで，少ない面積オーバヘッドで実現可能であることがわかった．

5. アプリケーションによる評価

本節では，実アプリケーションの MuCCRA-1 上への実装と評価結果を述べる．

5.1 評価アプリケーションと開発環境

実装した評価アプリケーションは，JPEG エンコーダで用いられる離散コサイン変換 (DCT)，二画像を半透明合成するアルファブレンダフィルタ (α-Blender)，認証や電子署名で用いられる一方向ハッシュ関数 (SHA-1) の 3 つである．

アプリケーションの実装には，MuCCRA 向けに独自に開発した MuCCRA-editor を使用した．MuCCRA-editor の GUI を **図 13** に示す．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，SE 間の接続関係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．MuCCRA-editor では，マッピングや配線は手動で行う必

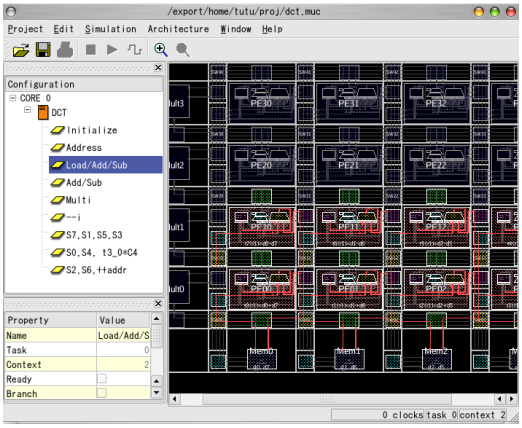

図 13 アプリケーション開発環境 MuCCRA-editor
Fig. 13 MuCCRA-editor: An Application Design Environment

表 2 利用リソース数の評価結果
Table 2 Evaluation Results of Resource Usage

	CTXT	ALU	SMU	RFile	MULT	MEM
DCT	41	76 11.6%	217 33.1%	180 27.4%	56 34.2%	92 56.1%
α-Blender	8	14 10.9%	26 20.3%	41 32.0%	6 19.0%	12 38.0%
SHA-1	12	70 36.4%	128 66.7%	134 69.8%	0 0.0%	23 47.9%

要があるが，RoMultiC による構成情報のマルチキャストが効率良く行われるように，転送の順番と相手先のスケジューリングは自動で行なわれる[10]．スケジューリングを行った後，RoMulTiC のビットマップを含む構成情報を生成する．

5.2 実装結果

5.2.1 利用リソース数

実装した評価アプリケーションの利用コンテキスト数 (CTXT)，利用コンテキスト中の各モジュールの総利用数を **表 2** に示す．下段に示す値は，各モジュールの利用可能な総モジュール数に対する利用率を示す．

評価結果より，DCT は他のアプリケーションに比べて，MEM と MULT の利用率が高く，ALU などの PE 内モジュールの利用率が低いことがわかる．これは，DCT が，8 × 8 画素の RGB データを分散メモリ (MEM) から並列に読み出し，乗算器 (MUL) を用いて積和・積差演算を行うことによる．α-Blender も，DCT と同様に，複数の RGB データに対して乗算を行うが，アプリケーションの規模が比較的小さいため，全体の利用率は低くなっている．

一方，SHA-1 は，乗算は行わず，シフトや論理演算などの比率が高いため，ALU および SMU の利用率が高い．また，RFile に格納した定数との演算やテーブル引きが多いため，RFile と MEM の利用率が高い．SHA-1 の演算粒度は 32bit であり，32bit の演算を 24bit の PE アレイ上にマッピングすると，PE を数多く必要とすることから，全体的にリソースの利用率は高めである．

5.2.2 実行速度

実装したアプリケーションのコンフィギュレーションデータと，配置配線後の MuCCRA-1 のネットリストを用いてシミュレーションを行った．そして，実行サイクル数および最大遅延時間を求め，MuCCRA-1 の性能を評価した．

表 3 実行速度の評価結果
Table 3 Evaluation Results of Execution Time

	BlockSize[bit]	ExecClocks	Delay[ns]	ExecTime[μs]
DCT	1024	195	40	7.8
α-Blender	8192	644	24	15.5
SHA-1	512	418	50	20.9

表 4 コンテキスト配送クロック数の評価結果
Table 4 Evaluation Results of Context Deliver Clock Cycles

	with RoMultiC	w/o RoMultiC	Reduction%
DCT	492	1025	52.0%
α-Blender	67	200	66.5%
SHA-1	220	300	26.7%

各アプリケーションの実行速度に関する評価結果を **表 3** に示す．実行クロック数 (Exec.Clock) は，アプリケーションの演算に要する実行サイクル数であり，コンテキスト配送，入出力に要するサイクル数は含めていない．これらのアプリケーションはいずれも定期的に処理すべきデータが到着するストリーム処理である．また，必要コンテキスト数は 64 以内であることから，MuCCRA-1 のコンテキスト配送制御と入出力ダブルバッファ機構により隠蔽が可能であり，表中の演算時間のみでの実行が可能である．

評価結果より，各アプリケーションは 20MHz から 42MHz 程度で動作可能であることがわかる．画素レベルで並列処理を行うことが可能で，リソースの利用率の比較的小さい α-Blender は高い動作周波数を実現可能である．しかし，SHA-1 は 32-bit 加算などでデータパスのパス段数が増加し，最大遅延も増加する結果となった．

また，TI 社の 225MHz で動作する信号処理プロセッサ (DSP) TMS320C6713 と比較すると，DCT では実行時間で約 2 倍高速であり，比較的古いプロセスを用いているにも関らず，組込み向けのアクセラレータとして利用可能な性能を達成している．

5.3 コンテキスト配送時間

各アプリケーションのコンテキスト配送時間に関する評価結果を **表 4** に示す．RoMultiC を用いてコンテキストを各モジュールにマルチキャストした場合 (with RoMulTiC) と RoMulTiC を用いずに各モジュールに順番にコンテキストを転送する場合 (w/o RoMulTiC) の必要サイクル数と，その削減率を示す．なお，RoMulTiC を用いない場合，1 コンテキストあたりの転送に要する時間は 25 サイクル (16 (PE) + 8 (SE) + 1 (CSC, MEM, MULT)) である．

アプリケーションの実行時間を比較すると，コンテキスト数の多い DCT は，実行クロック数よりもコンテキスト配送に要するクロック数の方が大きい．すなわち，他のタスクと組み合わせて実行する場合，現在のコンテキスト配送制御では，DCT はコンテキスト配送時間を隠蔽できないことになる．一方，α-Blender と SHA-1 は，コンテキスト数が比較的少ないため，コンテキスト配送に要する時間も小さく隠

表 5 消費電力の評価結果
Table 5 Evaluation Results of Power Consumption

	PE	SE	MULT	MEM	Ctrl	Other	Total[mW]
DCT (25 MHz)	69.1	12.5	3.6	3.2	1.8	9.8	85.1
α-Blender(42MHz)	65.0	11.8	4.1	4.0	2.3	12.8	103.3
SHA-1 (20MHz)	62.1	14.8	3.2	3.6	2.6	13.7	50.6

蔽可能である．

また，RoMulTiC の効果を比較すると，最大で 66.5% のコンテキスト配送時間を削減することが可能であることがわかる．多くのアプリケーションでは，データ依存性等により並列性が低下し，モジュールの利用効率の低いコンテキストが存在する．このような場合，利用していないモジュールの構成情報をマルチキャストすることで，コンテキスト配送時間を削減できるため，RoMulTiC は効果的であるといえる．一方で，全般的にコンテキストの利用効率の高い SHA-1 では，RoMulTiC の効果は 26.7% に留まった．

5.4 消 費 電 力

各アプリケーションを配置配線後のネットリストでシミュレーションを行い，スイッチング確率を抽出し，シノプシス社の Power Compiler を用いて電力の評価を行った．**表 5** に，消費電力の見積りと，各モジュールの消費する割合 (%) を示す．なお，評価結果は，各アプリケーションの最大動作周波数で動作させた場合のものである．

評価結果より，最大で 103.3mW という低い消費電力で動作可能であることがわかった．また，各モジュールの消費電力の内訳から，PE の消費割合が 60% 以上を占めることがわかる．CSC, TCC など制御部 (Ctrl) は，面積の場合と同様に，少ない消費電力で動作可能である．

一方，その他 (Other) の占める割合が，13% 程度と比較的大きいことがわかる．この中には，外部から入力されるクロックネットワークや，PE/SE 間の配線に挿入されるバッファなどの消費電力が含まれる．この結果より，結合網の消費消費電力が無視できないことがわかる．また，利用していないモジュールも定常的にスイッチングをしてしまい，電力を消費していることがわかった．

6. む す び

本論文では，動的リコンフィギャラブルプロセッサ MuCCRA のプロトタイプチップ MuCCRA-1 について述べた．MuCCRA-1 は，ローム社の 0.18μm CMOS プロセスを用いて実装し，5mm^2 のダイ上に 4 × 4 の 24bit PE アレイと，乗算器，ダブルバッファ方式の分散メモリをもつ．この他に，RoMultiC によるコンテキストのマルチキャスト配送，タスク制御機構が実装されている．実装結果より，制御機構は少ない面積・消費電力で実装可能であることがわかった．

また，アプリケーションによる評価により，JPEG で用いられる DCT では，TI 社の 225MHz で動作する DSP の 2 倍の速度を達成した．また，RoMulTiC によるコンテキストのマルチキャスト配送により，最大で 66.5% のコンテキスト配送時間を削減可能であることがわかった．

今後は，MuCCRA-1 をベースとして，PE アレイの実チップレベルでの基本的なトレードの解析，PE アレイのマルチコア構成の検討などを行う予定である．また，90nm CMOS プロセスを用いて，省電力技術を導入したチップを開発すると同時に，動的リコンフィギャラブルプロセッサの電力モデルも構築する予定である．

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．

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．

参 考 文 献

1) 末吉, 天野 (編): リコンフィギャラブルシステム, オーム社 (2005).
2) Motomura, M.: A Dynamically Reconfigurable Processor Architecture, *Microprocessor Forum* (2002).
3) Sugawara, T., Ide, K. and Sato, T.: Dynamically Reconfigurable Processor Implemented with IPFlex's DAPDNA Technology, *IEICE Trans. on Information & System*, Vol. E87-D, No. 8, pp. 1997–2003 (2004).
4) Petrov, M., Murgan, T., May, F., Vorbach, M., Zipf, P. and Glesner, M.: The XPP Architecture and Its Co-simulation within the Simulink Environment, *Proc. of Int'l Conf. on Field Programmable Logic and Application (FPL)*, pp. 761–770 (2004).
5) Veredas, F., Scheppler, M., Moffat, W. and Mei, B.: Custom Implementation of the Coarse-Grained Reconfigurable ADRES Architecture for Multimedia Purposes, *Proc. of Int'l Conf. on Field Programmable Logic and Application (FPL)*, pp. 106–111 (2005).
6) 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, 情報処理学会論文誌コンピューティングシステム, Vol. 47, No. SIG12(ACS15), pp. 171–181 (2006).
7) Amano, H.: A Survey on Dynamically Reconfigurable Processors, *IEICE Transactions on Communications*, Vol. E89-B, No. 12, pp. 3179–3187 (2006).
8) Amano, H., Abe, S., Deguchi, K. and Hasegawa, Y.: An I/O mechanism on a Dynamically Reconfigurable Processor - Which should be moved: Data or Configuration, *Proc. of Int'l Conf. on Field Programmable Logic and Applications (FPL)*, pp. 347–352 (2005).
9) Tunbunheng, V., Suzuki, M. and Amano, H.: RoMultiC: Fast and Simple Configuration Data Multicasting Scheme for Coarse Grain Reconfigurable Devices, *Proc. of IEEE Int'l Conf. on Field Programmable Technology (FPT)*, pp. 129–136 (2005).
10) 堤他: 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, 電子情報通信学会技術研究報告 VLD2006-102, pp. 49–54 (2007).