動的リコンフィギャラブルプロセッサにおける性能と消費電力の定量的解析

長谷川揚平† 西村 隆†† 阿部 昌平† 黒瀧 俊輔†

ヴ マン トウアン† 天野 英晴††

† 慶應義塾大学大学院理工学研究科
†† 慶應義塾大学理工学部情報工学科
〒 223–8522 神奈川県横浜市港北区日吉 3-14-1
E-mail: †drp@am.ics.keio.ac.jp

あらまし 本論文では，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による性能と消費電力の関係を議論する．対象となるアプリケーションの要求性能に対して，電力効率を最適化する構成を明らかにするため，時分割多重実行が性能および消費電力に与える影響を定量的に評価する．実際に，NEC エレクトロニクス社の Dynamically Reconfigurable Processor (DRP) とその開発環境を用いて，いくつかのアプリケーションを様々な構成を想定して実装を行なった．実装した各アプリケーションにおいて，性能および消費電力を評価し，時分割多重実行を行うことによる電力効率の変化を評価した．

キーワード 動的リコンフィギャラブルプロセッサ，DRP, マルチコンテキスト機能，時分割多重実行，消費電力

Application-Based Performance and Power Analysis on Dynamically Reconfigurable Processor

Yohei HASEGAWA†, Takashi NISHIMURA††, Shohei ABE†, Shunsuke KUROTAKI†,

Vu MANH TUAN†, and Hideharu AMANO††

† Graduate School of Science and Technology, Keio University
†† Faculty of Science and Technology, Keio University
3-14-1, Hiyoshi, Kohokuku, Yokohama, 223–8522, Japan
E-mail: †drp@am.ics.keio.ac.jp

Abstract Dynamically Reconfigurable Processor (DRP) developed by NEC Electronics is a coarse-grained reconfigurable processor that selects a datapath called a context from the on-chip repository of sixteen circuit configurations at run-time. The time-multiplexed execution based on the multicontext functionality is expected to improve area and power efficiency. To demonstrate the impact of the time-multiplexed execution, we have implemented several stream applications on DRP with various context sizes. Throughout the evaluation based on real application designs, we analyzed the impact of the time-multiplexed execution on performance and power dissipation quantitatively.

Key words Dynamically Reconfigurable Processor, DRP, Multicontext Functionality, Time-multiplexed Execution, Power Dissipation

1. ま え が き

近年，組み込み機器を対象とした新たなプログラマブルデバイスとして，動的リコンフィギャラブルプロセッサが注目を集めている [1]．近年の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) と分散メモリモジュールなどから構成され，これらの機能と接続を動作中に変更することが可能である．ハードウェアの高速性に加え柔軟性を備えたデバイスとして，特に組み込み機器への応用が期待されている．

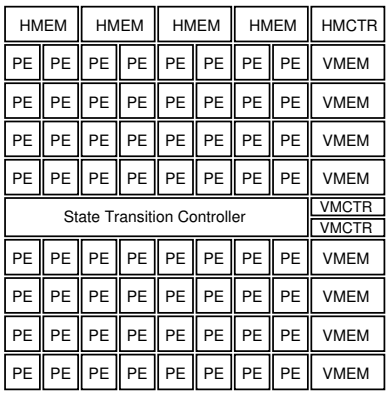


図 1　Tile の構造

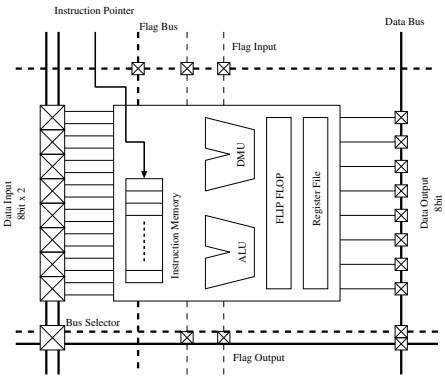


図 2　Processing Element (PE) の構造

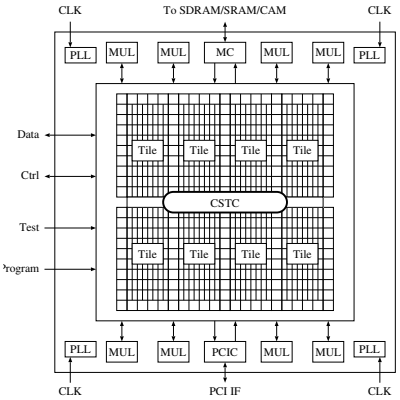


図 3　DRP-1 の構造

各 PE は ALU やレジスタから構成され，アレイ内外の分散メモリ中のデータを同時アクセスして並列処理することにより，特にマルチメディア処理や通信制御処理において高い処理能力を示す．さらに，LSI の高位合成技術を応用することで，開発期間の短縮が可能であり，設計変更にも容易に対応可能である．これらの利点から，近年では複数のメーカより商用のデバイスが登場しており [2]〜[5]，実際に家庭用ゲーム機などに応用されている [6].

動的リコンフィギャラブルプロセッサの最大の特徴は，動的再構成機能による時分割多重実行である．特に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，複数のコンテキストと呼ばれる回路構成の命令情報（コンフィギュレーションデータ）を各 PE の内部の命令メモリに保持し，これらをクロックサイクル毎に切り替えることが可能である．対象のアプリケーションの一連の処理を複数のコンテキストに分割し，これらを動作時に切り替えながら実行する．このマルチコンテキスト機能に基づく時分割多重実行により，Field Programmable Gate Array (FPGA) を始めとする従来のプログラマブルデバイスの課題であった面積効率や電力効率を改善する．

また，動的リコンフィギャラブルプロセッサは，System-on-a-Chip (SoC) において Intellectual Property (IP) コアとして利用されることが想定されている．近年の組み込み機器では，バッテリー駆動デバイスの普及と，無線通信やポータブルアプリケーションの普及により，LSI における消費電力の削減が最重要課題となっている．このため，動的リコンフィギャラブルプロセッサは，組み込み機器を対象として，高い電力効率を実現するアーキテクチャとして期待されている．

しかしながら，動的リコンフィギャラブルプロセッサにおいて，時分割多重実行を行うことにより，電力効率にどのような影響があるのか明らかにした研究はこれまで報告されていない．また，SoC に搭載することを想定し，対象のアプリケーションに対して，どの程度の時分割多重化を行うことで電力効率を最適化できるのかを明らかにする必要がある．ここで，コンテキストサイズ（コンテキストあたりで利用可能なリソース量）とコンテキスト数は，時分割多重実行を行うにあたり重要な要素であり，対象とする問題の要求性能に応じて電力効率を最適化する構成を見出すことが重要である．

本論文では，実際の動的リコンフィギャラブルプロセッサである NEC エレクトロニクス社の Dynamically Reconfigurable Processor (DRP) を対象として議論する．DRP のプロトタイプチップ DRP-1 とその開発環境を用いて，いくつかのアプリケーションを実装し，性能および消費電力の評価を行った．また，各アプリケーションに対して，様々なコンテキストサイズのもとで設計し，時分割多重実行を導入することによる電力効率への影響を定量的に評価する．

本論文の構成は以下の通りである．節 2. では本論文で対象とする DRP のアーキテクチャについて述べる．続いて，節 3. において，DRP における時分割多重実行モデルを述べる．そして，節 4. で DRP-1 上に実装したアプリケーションに基づいた評価結果を述べ，節 5. で評価結果に関する考察と結論を述べる．

2. DRP アーキテクチャ

DRP [2] は，NEC エレクトロニクス社より 2002 年に発表された動的リコンフィギャラブルプロセッサである．DRP の基本ユニットである Tile の構造を図 1 に示す．DRP は，Tile をアレイ状に配置することで，所望の規模の DRP Core を実現する．各 Tile は，8bit の演算器とレジスタファイルなどから構成される Processing Element（PE）を 8 × 8 の 2 次元アレイ状に並べ，その周辺に 8 個の 2-port メモリである VMEM を垂直方向の端に配置し，4 個の 1-port メモリである HMEM を水平方向に配置した構成となっている．また，VMEM/HMEM をそれぞれ制御するコントローラである VMCTR が 2 つ，HMCTR が 1 つ搭載されている．さらに中央部に 1 つの State Transition Controller（STC）が配置されている．Tile ごとに STC と PE のアレイが組み合わされているため，各 Tile が独自のステートマシンの制御下で独立して動作することが可能である．

各 PE の構成を図 2 に示す．各 PE は 8bit の ALU，シフトやデータ制御，簡単な論理演算を行なう Data Manipulation Unit（DMU），8bit の Flip Flop，レジスタファイルから構成される．

DRP では，PE アレイで実現される複数の回路構成情報（コンフィギュレーションデータ）を，PE 内部の命令メモリに格

納しておく．本論文では，PE アレイで実現される回路構成をコンテキストと呼ぶ．そして，実行中に必要に応じて STC が発行した命令ポインタに切り替えることでコンテキストを切り替える．DRP では，クロックサイクル毎にコンテキストの切り替えが可能である．また，外部より命令メモリにコンフィギュレーションデータを書き込むことをコンフィギュレーションと呼ぶ．

DRP のプロトタイプチップである DRP-1 の構成を図 3 に示す．DRP-1 はコアの周辺部に 32bit の乗算器を 8 個，PCI バスインタフェース，SDRAM/SRAM/CAM インタフェースを搭載したシステム LSI である．また，PCI バスインタフェース，SDRAM/SRAM/CAM インタフェースにより単独で PCI バスへの接続や外部メモリの制御が可能である．

DRP-1 では 4 × 2 個の Tile が配置されており，全体では 512 個の PE をもつ．チップ中央には，DRP-1 全体の状態遷移を制御するためのシーケンサである Central State Transition Controller（CSTC）が配置されている．また，DRP-1 全体では VMEM を合計 80 個（DRP Core の両端には必ず VMEM を配置する必要があるため，1 列付加される），HMEM を合計 32 個もつ．

DRP-1 は内部の命令メモリに最大 16 コンテキスト分の情報を蓄えることが可能で，クロックサイクル毎にコンテキストの切り替えが可能なマルチコンテキストデバイスである．

DRP の開発環境として，DRP コンパイラおよび統合開発環境 Musketeer [7] が提供されている．対象のアプリケーションをソフトウェアライクに開発することができるため，短 TAT を実現することができる．DRP コンパイラは，C 言語をハードウェア記述用に拡張した Behavior Design Language（BDL）から，動作合成，マッピング，配置配線などの合成を経て，最終的に DRP 上で実行可能なコンフィギュレーションデータを生成する．

3. DRP における時分割多重実行

3.1 問題の並列性と逐次性

DRP の対象とするアプリケーションは，組み込み機器におけるマルチメディアデータを扱うストリーム処理である．JPEG や MPEG に代表されるストリーム処理は，まとまったデータを扱ういくつかのタスクから構成される．これらのアプリケーションを DRP 上で実行する場合，各タスクを複数のコンテキストに分割し，これらを切り替えながら実行する．コンテキストの切り替えは逐次的に行なわれるが，コンテキスト内の PE と分散メモリモジュールのアレイは並列に動作することが可能である．よって，面積/電力効率を高めるためには効率的にタスクを複数のコンテキストに分割する必要がある．

一般的に，計算機上で実行されるアルゴリズムには，潜在的に逐次性と並列性が存在するため，対象のアプリケーションを実行するためには一定数のサイクルを必要とする．特に，DRP 上でストリームアプリケーションを実行する場合，以下の典型的な処理フローが考えられる．

(1) データを分散メモリ（またはレジスタ）から読み出す

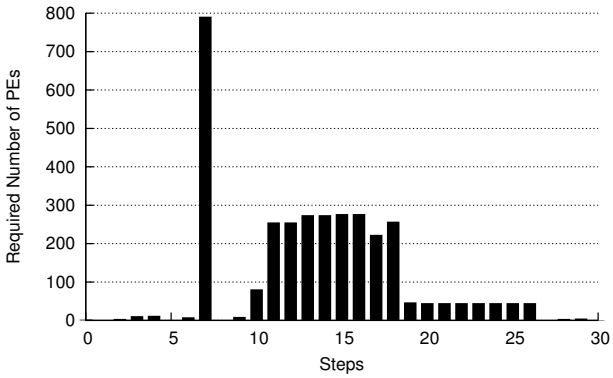


図 4 ステップ毎の必要 PE 数 (DCT)

(2) 要求された処理を多数の PE アレイにより並列に処理する

(3) 演算結果を分散メモリ（またはレジスタ）に書き戻す

ここで，これらの一連のプロセスは，対象のアルゴリズムにおける一つのステップと考えることができる．通常，各ステップは繰り返し実行され，演算結果を毎回分散メモリに書き戻すことから，各ステップは少なくとも 1 サイクルを必要とする．これらの全ステップが終了したとき，出力データストリームをチップ外部に出力する．

よって，対象とするアプリケーションは逐次実行の制約があると同時に，並列性は各ステップにおいて定義される．本稿では，各ステップにおいて要求される PE 数を，そのステップにおける並列性とみなす．必要 PE 数は並列性と同義ではないが，各ステップの並列性の目安として考えることができる．図 4 は，JPEG で用いられる離散コサイン変換 (Discrete Cosine Transform, DCT) を DRP-1 上で実行した場合の，各ステップにおける必要 PE 数を示している．この図では，無限個の PE と VMEM/HMEM が利用可能であると仮定している．ここで，各ステップはアルゴリズムに応じて反復して実行される．

DCT アルゴリズムでは，主に 8 × 8 の画像データに対して，行方向および列方向に積和演算を実行する．この実装例における各ステップの処理の内容は以下の通りである．

- 分散メモリ，レジスタの初期化を行い，入力データを分散メモリに格納する（ステップ 0〜6）
- 行方向への演算を行う（ステップ 7〜10 を 8 回反復する）
- 列方向への演算を行う（ステップ 11〜18 を 4 回反復し，続いてステップ 19〜26 を 1 回実行する）
- 出力データを外部に出力する (ステップ 27〜30)

この実装例では，8 × 8 の画像データを 8 つの独立した配列に格納し，行方向の演算において並列に処理が行えるようにしている．一旦データの準備が整ったら，行方向の演算を約 800 個の PE を使用して並列に処理している．また，この DCT の実装では，積和演算における乗数が定数であるため，乗算は DRP コンパイラによって Shift & Add に自動的に展開されている．しかし，列方向の演算は，配列の添字を変えながら順に 8 回アクセスする必要があることから，並列性が低下する．ここでは，2 つの分散メモリを同時にアクセスし，読み出して可

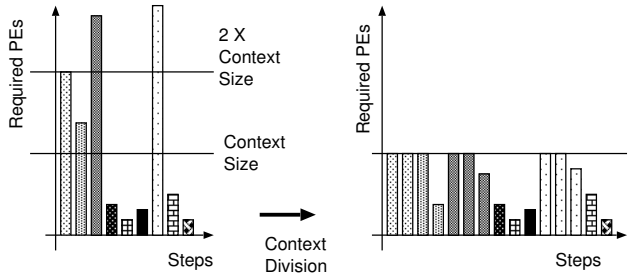


図 5　コンテキスト分割

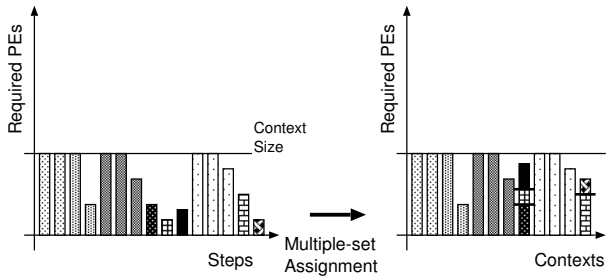


図 6　コンテキスト統合

能になった演算から順に実行する手法と，最終データの格納を演算とオーバラップさせる一種のソフトウェアパイプライニングを用いて，並列性を維持しつつ反復回数を削減している．

ここで，無限個の PE と VMEM/HMEM が利用可能であるとし，あるアプリケーションを実行するために必要となるステップの延べ数の最小値を S_∞ とする．図 4 に示す DCT のケースでは，$S_\infty = 72$ である（行方向演算 4×8，列方向演算 $8 \times 4 + 8$）．なお，入出力に要するステップは，ダブルバッファ機構 [8] を採用することで隠蔽可能であることから，本論文では評価の対象としない．S_∞ は，アルゴリズムとデータ構造に依存するため，DCT のケースでは無限個の PE をもつとした場合でも 72 クロックを必要とする．よって，S_∞ は，対象のアプリケーションを DRP 上で実行する場合に必要となる実行クロック数の最小値とみなすことができる．

ここで，無限個の PE と VMEM/HMEM が利用可能であるとし，全てのステップが単一のコンテキスト上で実現される場合を考える．これは，時分割多重実行を行わないケースに相当する．この場合，対象のアプリケーションは最低でも S_∞ のサイクル数を必要とし，このときの実行時間 T_∞ は

$$T_\infty = C_\infty S_\infty$$

となる．ここで，C_∞ はこのときの最大遅延を示す．

時分割多重実行では，各ステップ内の処理は PE アレイで実現されるコンテキストに対応し，ステップの遷移は DRP のコンテキストの切り替えに相当すると考えるのが自然である．しかしながら，コンテキストあたりで利用可能な PE 数や，利用可能なコンテキスト数には限りがあるため，リソースの利用効率を高めるために以下のような操作が実際には必要となる．

3.2　コンテキストのスケジューリング方法

3.2.1　コンテキスト分割

時分割多重実行では，コンテキストあたりで利用可能な PE 数の制約の下で，各ステップはあるコンテキストに割り当てられる．ステップ毎に必要となる PE 数がこの制約を越える場合，そのステップは図 5 に示すように，さらに小さなステップに分割する必要がある．本稿ではこれをコンテキスト分割と呼ぶ．

形式的には，$PE_{\rm size}$ をコンテキストあたりで利用可能な PE 数とし，PE_i をステップ $i (i = 0, \cdots, S_\infty - 1)$ において必要となる PE 数とする．あるコンテキストサイズが与えられたとき，コンテキスト分割により，あるアプリケーションの実行に必要な最小ステップ数 $S_{\rm size}$ は，以下の式で示されるように増加する．

$$S_{\rm size} = \sum_{i=0}^{S_\infty - 1} \frac{PE_i}{PE_{\rm size}} \quad .$$

ここで，時分割多重実行により，あるコンテキストが半分のサイズとなった場合，最大利用 PE 数は減少するが，完全に半分になるわけではない．これは以下の理由に因る．

- 対象のアプリケーションを均等なサイズに分割することができない．例えば，定数乗算は，DRP コンパイラによりシフトと加算に展開されるが，これ以上細かく分割することが難しい．すなわち，割り当てる処理の中には，これ以上分割することのできないアトミックサイズが存在する．
- コンテキストサイズに適合しない分割が発生する．とりわけ，コンテキストサイズが大きい場合に起こり得る．
- コンテキスト間のデータ交信用のレジスタが増加する．

また，コンテキスト分割を行うことにより，最大遅延パスも分断され，$C_{\rm size}$ は削減される．コンテキスト分割を行なわないとしても，コンテキストサイズが小さくなれば，配線長が削減され動作周波数は向上する傾向にある．コンテキスト分割を行い，時分割多重実行を行う場合の実行時間 $T_{\rm size}$ は，

$$T_{\rm size} = C_{\rm size} S_{\rm size}$$

となる．

3.2.2　コンテキスト統合

図 4 に示した DCT の実装例では，メモリアクセスのみを行うステップが存在することから，必要 PE 数が少なく並列性の低いステップが数多く存在し，PE の利用効率が悪い．さらに，各ステップを各コンテキストに一対一で割り当てる場合，必要となるコンテキスト数が DRP-1 の利用可能コンテキスト数である 16 を越えてしまう．

この問題に対処するため，図 6 に示すように，一つのコンテキストに複数のステップを割り当て，各ステップを逐次的に実行することで，PE の利用効率を高めることができる．本論文では，これをコンテキスト統合と呼ぶ．DRP Core の STC は，状態遷移に従って実行するステップを駆動する．このようにして，コンテキスト統合により PE の利用効率を高め，必要コンテキスト数を削減する．

コンテキスト統合は，確かに PE の利用効率を改善し，必要コンテキスト数を削減するが，実行時間を改善することはないことに注意しなければならない．主な理由としては，コンテ

表 1　評価アプリケーションと実装結果

アプリケーション	略記	最大利用 PE 数	文献
JPEG エンコーダにおける離散コサイン変換	DCT	293	[9]
MP3 デコーダにおける修正逆離散コサイン変換	IMDCT	478	[9]
高速フーリエ変換	FFT	116	[10]
アクティブ方向通過型フィルタ	ADPF	100	[10]
ビタビデコーダ	Viterbi	336	[11]
Advanced Encryption Standard (ECB mode)	AES-ECB	512	[12]
Secure Hash Algorithm 1	SHA-1	65	[12]

キスト統合を行っても，必要となるステップ数の最小値 S_{size} が変化しないことがあげられる．さらに，単一コンテキストのデータパス全体が空間的に拡大するので，通常は配線遅延が増大する．さらに，単一コンテキスト内の複数のステップ間で分散メモリやレジスタの入出力を共有するため，マルチプレクサを追加する必要がある．これにより，コンテキストあたりの利用 PE 数は増加し，最大遅延も増大する．結果的に，コンテキスト統合により，性能を悪化させる可能性がある．しかし，最大遅延パスを含むステップは，通常他のステップに比べそれ自身で多くの PE を利用するため，コンテキスト統合により他のステップと統合されることはない．よって，多くの場合において，コンテキスト統合が性能を悪化させることはないと考えられる．

なお，コンテキスト統合により，全ステップが単一のコンテキストに割り当てられた場合，これは時分割多重化を行わないケースに相当する．ただし，DRP では，単一のコンテキストにマッピング可能なステップは最大 4 つまでである，

3.3　時分割多重実行の効果とトレードオフ

本節では，時分割多重実行や，これまで述べたコンテキストスケジューリングによって，性能と消費電力にどのような効果があるのかを定性的に議論する．

マルチコンテキスト機能に基づく時分割多重実行は，組み込み向けの SoC において，高い面積効率，電力効率を提供することが期待されている．これは，必要なコンテキストを必要な時に実行することが可能であるという，動的リコンフィギャラブルプロセッサの性質に基づいている．よって，対象となるアプリケーションは，最小のハードウェアコストと消費電力で，要求性能を達成することが可能である．これまで述べたように，コンテキストあたりで利用可能な計算資源に対応するコンテキストサイズは，時分割多重実行において重要な要素の一つである．面積効率および電力効率を最適化するためには，対象のアプリケーションにおいて，最適なコンテキストサイズを選択する必要がある．

コンテキストサイズが小さい場合，物理的なハードウェア面積も小さいが，コンテキスト分割により必要ステップ数，必要コンテキスト数が増加する．一方で，コンテキスト分割によって，最大遅延パスも分断されるため，最大動作周波数は向上する．また，コンテキストサイズが小さい場合，PE アレイが消費する電力は減少するが，一方で動作周波数が向上することにより，クロック網が消費する電力が増加する．また，コンテキストの切り替え頻度が高まることで，コンテキストの切り替えに要する電力も増加する．

一方，コンテキストサイズが大きい場合，より多くの PE を用いて並列処理を行うことが可能となる．これにより，高スループットを達成し，必要ステップ数および必要コンテキスト数を削減する．また，動作周波数とコンテキストの切り替え頻度の低下によりこれらの消費電力は減少するが，命令の割り当てられていない PE も含め，駆動する PE の数が増加するため，PE アレイの消費する電力は増加する．必要以上にコンテキストサイズが大きい場合，不適切なコンテキスト統合により，マルチプレクサの増加などにより，最大遅延に重大な損失が生じる恐れがある．

対象となるアプリケーションに対して，電力効率を最大化するコンテキストサイズを選択する際，上記のトレードオフを考慮しなければならない．以下の節では，実際に複数のストリームアプリケーションを DRP-1 上に実装し，電力効率について定量的な評価を行う．

4. 評　　価

4.1　評価アプリケーション

本論文では，前節で示した時分割多重実行による効果を明らかにするため，実際に複数のストリームアプリケーションを DRP-1 上に実装し，評価を行った．各アプリケーションを，C レベルの動作記述言語である BDL を用いて記述し，DRP コンパイラおよび統合開発環境 Musketeer を用いてコンパイルした．実装したアプリケーションと，それぞれの DRP-1 上における最大利用 PE 数を表 1 に示す．

最大利用 PE 数は，使用したコンテキストの中で最も多く使用した PE 数であり，空間的に消費している PE の数を意味する．前節で述べたように，対象アプリケーションの各ステップにおける並列性は必要となる PE 数で定義されるため，最大利用 PE 数は対象アプリケーションの最大並列性と考えることができる．表 1 より，DCT, IMDCT, Viterbi, AES-ECB は，FFT, ADPF, SHA-1 に比べ，相対的に並列性が高いアプリケーションであるといえる．

4.2　電力効率の評価

4.2.1　評 価 方 法

我々は，時分割多重実行を導入することによる性能と消費電力の変化を評価するために，それぞれのアプリケーションに対して，コンテキストサイズを変化させて合成した．コンテキストサイズは，一つのコンテキストで利用可能な PE 数である．DRP の場合には Tile と呼ばれるクラスタ構造をもち，各 Tile は 64 個の PE と，分散メモリである VMEM/HMEM をもつ．本論文では，コンテキストサイズの変更は Tile 単位で行うも

のとし，実際に利用可能な Tile 数を変化させて合成した．

コンテキスト分割とコンテキスト統合は，手動で行うか，DRP コンパイラのオプションとして自動的に行うことが可能である．今回の実装では，コンテキスト分割は手動で最適化を行い，コンテキスト統合は基本的にはコンパイラに委ね，最適化の必要がある場合のみ手動で行った．

まず，時分割多重実行を行う場合と行わない場合の実行時間の比率 R_{size} を評価する．すなわち，

$$R_{\text{size}} = \frac{T_{\text{size}}}{T_\infty} = \frac{C_{\text{size}} S_{\text{size}}}{C_\infty S_\infty}$$

である．ここで，時分割多重実行を行わない場合の必要 PE 数は，しばしば DRP-1 の利用可能 PE 数 (512) を越えるため，実際には DRP-1 上にマッピングすることができないことが多い．この場合，時分割多重実行を行わない場合の最大遅延 C_∞ は，実際には計測することができない．このような場合には，最もコンテキストサイズの大きい，Tile 数が 8 のケースの最大遅延 C_8 を代りに用いるものとする．一般的に，$C_\infty \geqq C_8$ であることから，実際の C_∞ を用いるよりも，R_{size} の値は小さくなる可能性がある．

次に，実装した各アプリケーションに対して予測される消費電力を計測し，アプリケーションの実行に要する必要エネルギーを算出する．消費電力は，統合開発環境 Musketeer における電力プロファイラの算出した値である．電力プロファイラは，DRP-1 を対象デバイスとして，シミュレーション結果より電力値を算出しており，実測値とは異なる可能性がある．また，対象となる電力は DRP-1 のコア部のみで，I/O などの 3.3V 系の電力は含まれていない．さらに，使用した電力プロファイラは DRP-1 を対象としており，算出される電力値は 8 Tile のケースでの消費電力である．このため，電力プロファイラの推定値を補正して各コンテキストサイズに対する消費電力を見積った．

必要エネルギーは，対象のアプリケーションを実行するために必要な電力量である．すなわち，消費電力を P_{size} とすると，必要エネルギー E_{size} は以下で表すことができる．

$$E_{\text{size}} = P_{\text{size}} T_{\text{size}} = P_{\text{size}} C_{\text{size}} S_{\text{size}}$$

T_{size} と同様，P_{size} と E_{size} は，時分割多重実行を行なわない場合の値 P_∞ と E_∞ に対して，それぞれ相対的に評価される．しかし，P_∞, E_∞ は C_{size} と同様，実際に計測することができないため，それぞれ P_8, E_8 を代わりに使用する．

4.2.2 評価結果

本節では，コンテキストサイズを変えてアプリケーションを実装することで，時分割多重化の程度により性能および消費電力に与える影響を定量的に評価する．図 7 は，各アプリケーションにおいて，コンテキストサイズに対する実行時間，消費電力，必要エネルギーの関係を示す．これらの値は，8 Tile で実現した場合の値に正規化したものである．

性能に関する評価結果より，各アプリケーションにおいて，コンテキストサイズを増やすことで，実行時間は減少し一定の性能向上を達成可能であることがわかった．特に DCT のように，並列性の高いアプリケーションでは，さらに Tile 数を増やすことで更なる性能向上を期待することが可能である．一方で，並列性が低く必要 PE 数の少ない FFT や ADPF, SHA-1 では，コンテキストサイズを増やしても更に並列実行が可能な部分がないため，必要な実行クロック数を削減することができない．例えば，SHA-1 のケースでは，2 Tile のコンテキストサイズで性能向上の限界が存在する．また，6 Tile での IMDCT のケースのように，コンテキストサイズが大きすぎることにより，コンテキスト統合によるマルチプレクサが増加し，最大遅延時間の増大を招き，返って性能を損ねている．

コンテキストサイズと消費電力の関係において，コンテキストサイズが大きくなると，通常消費電力は増加する．これは，より大きなサイズのコンテキストでは，命令の割り当てられていない PE も含め稼働する PE の数が増加することにより，消費電力も増加するためである．また，DRP-1 の場合，消費電力のうちコンテキストスイッチに要する電力の割合は，経験則として平均 5% 程度であることがわかった．また，PE アレイやクロック網などの基本的な部分の消費電力が支配的であるため，コンテキストスイッチの頻度は消費電力において大きく影響していない．よって，コンテキストサイズが増加すると，消費電力の面では不利であることがわかった．

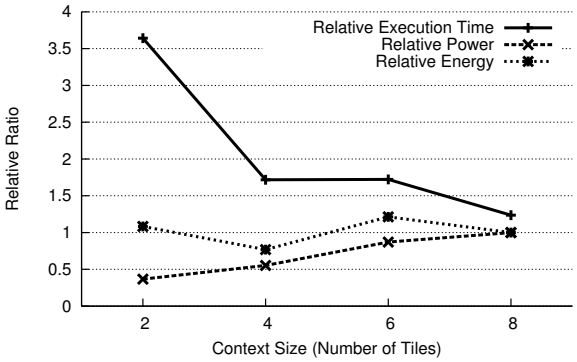
コンテキストサイズの増加により，消費電力も増加するが，実行時間は削減されるため，必要となるエネルギーは減少する．4 Tile の DCT のケースでは，2 Tile のケースと比較して実行時間が大幅に減少するため，必要エネルギーが最も少ない．同様に，Viterbi や AES-ECB においては，コンテキストサイズの増加により実行時間は線形に減少し，それに従って必要エネルギーも減少する．よって，比較的並列性の高いアプリケーションでは，対象の問題を解くための必要エネルギーの観点では，コンテキストサイズを増やした方が有利であるといえる．

対照的に，FFT や ADPF, SHA-1 などの並列性の低いアプリケーションでは，1,2 Tile 以上にコンテキストサイズを増やしても，それ以上の並列性を引き出すことができず必要クロック数は変化しない．このため，コンテキストサイズの増加に伴ない消費電力，必要エネルギーともに増加する．また，コンテキストサイズを増やすと，コンテキスト統合により最大遅延が増大し，必要エネルギーとしては不利になる傾向がある．よって，このようなアプリケーションでは，コンテキストサイズは 1，2 Tile 程度の小さい方が，必要エネルギーの観点では有利である．

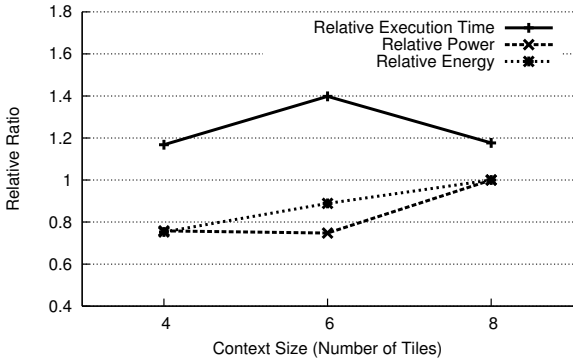
これまでの評価結果より，コスト対性能比が最適なコンテキストサイズを選ぶことで，多くのアプリケーションにおいて必要エネルギーも最小となることがわかった．

5. むすび

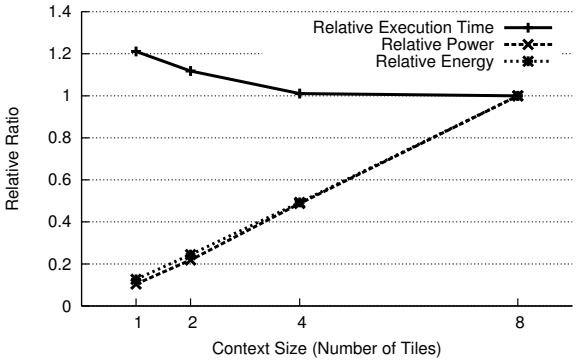
本論文では，動的リコンフィギャラブルプロセッサにおいて，時分割多重実行による性能と消費電力の関係を議論した．実際にいくつかのストリームアプリケーションを NEC エレクトロニクス社の DRP-1 上に実装し，評価を行った．各アプリケーションは異なるコンテキストサイズを想定して設計され，それ



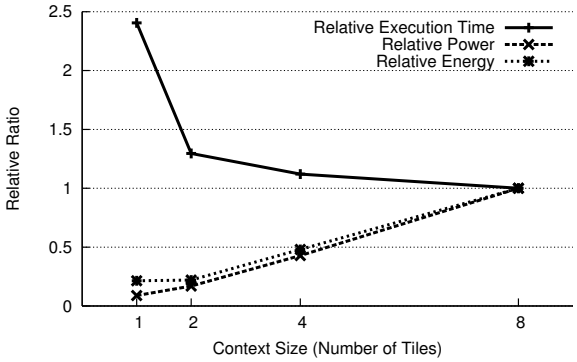
(a) DCT



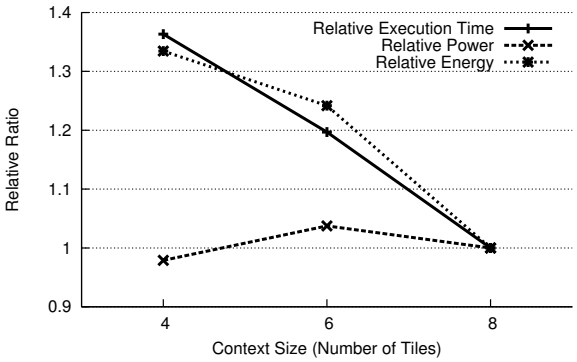
(b) IMDCT



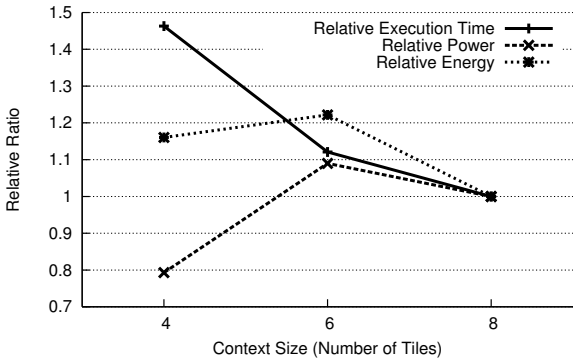
(c) FFT



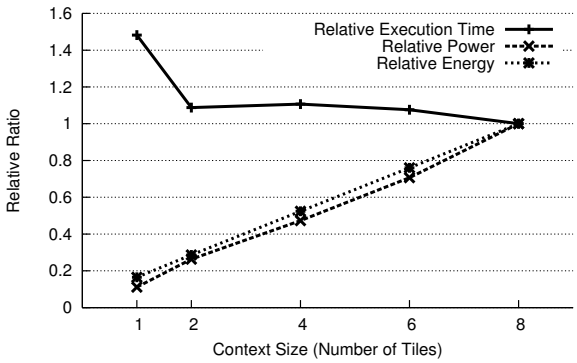
(d) ADPF



(e) Viterbi



(f) AES-ECB



(g) SHA-1

図 7　コンテキストサイズと性能および消費電力の関係

ぞれのコンテキストサイズに対して評価を行った．

まず，コンテキストサイズに対するアプリケーションの実行時間を評価した．DCT や IMDCT のように，並列性の高いアプリケーションでは，コンテキストサイズを増やすことで一定の性能向上を達成できることがわかった．一方で，並列性の低いアプリケーションでは，コンテキストサイズを増やしても更に並列実行が可能な部分がないため，必要な実行クロック数を削減することができないことがわかった．さらに，過剰なコンテキストサイズを供給することで，コンテキスト統合により最大遅延が増大し，結果的に実行時間を増大させることがあることがわかった．

また，コンテキストサイズと消費電力，必要エネルギーの評価結果より，DCT や IMDCT などの並列性の高いアプリケーションでは，コンテキストサイズの増加に従い消費電力は増加するが，並列性が高まることにより実行クロック数が減少し必要エネルギー量は減少する．一方，FFT や ADPF, SHA-1 などでは，必要以上にコンテキストサイズを増やしても，性能向上を期待することができない．従って，コンテキストサイズの増加に伴ない消費電力は増加し，必要エネルギーも増加する．よって，このようなアプリケーションでは，必要エネルギーの観点では，1,2 Tile 程度の小さなコンテキストサイズを選択することが望ましいことがわかった．

全体的に，要求性能に対して，面積効率および電力効率を最適化するためには，事前に対象のアプリケーションの並列性と逐次性を調査しておく必要がある．その上で，IP コアとして登載する DRP コアの規模を決定することが有効であると考えられる．DRP の場合，これらの操作は一般的な C プログラミングの技術をもっていれば，統合開発環境を用いることで比較的容易に実現することが可能である．

より高い電力効率を実現するアーキテクチャに向けて，動的リコンフィギャラブルプロセッサの消費電力をより詳細にわたって調査する必要がある．特に，消費電力とコンテキストスイッチの関係を明らかにすることは今後の課題の一つである．また，電力効率に関して FPGA におけるアーキテクチャ評価は多数報告されており，高電力効率，低消費電力を実現するアーキテクチャがすでに検討されている [13]〜[15]．今後は，Xilinx 社の Spartan FPGA [16] など，低消費電力を意識した他のプログラマブルデバイスとの比較を行う必要があると考えられる．その上で，動的リコンフィギャラブルプロセッサにおける低消費電力のアプリケーション設計技法や，高電力効率のアーキテクチャを検討していく予定である．

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] 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] M. Motomura: “A Dynamically Reconfigurable Processor Architecture”, Microprocessor Forum (2002).

[3] T.Sugawara, K.Ide and T. Sato: “Dynamically Reconfigurable Processor Implemented with IPFlex’s DAPDNA Technology”, IEICE Transaction on Information & System, **E87-D**, 8, pp. 1997–2003 (2004).

[4] Elixent. http://www.elixent.com/.

[5] PACT. http://www.pactcorp.com/.

[6] M. Okabe: “A 90nm embedded DRAM single chip LSI with a 3D graphics, H.264 codec engine, and a reconfigurable processor”, Proceedings of the International Symposium on High Performance Chips (Hot Chips 16) (2004).

[7] 粟島, 戸井, 中村, 紙, 加藤, 若林, 宮澤, 李：“動的再構成可能チップ DRP の C コンパイラ”, 電子情報通信学会技術研究報告 VLD2003-118, **103**, 578, pp. 23–28 (2004).

[8] H. Amano, S. Abe, K. Deguchi and Y. Hasegawa: “An I/O mechanism on a Dynamically Reconfigurable Processor -Which should be moved: Data or Configuration?-”, Proceedings of International Conference on Field Programmable Logic and Application (FPL2005), pp. 347–352 (2005).

[9] M. Suzuki, Y. Hasegawa, Y. Yamada, N. Kaneko, K. Deguchi, H. Amano, K. Anjo, M. Motomura, K. Wakabayashi, T. Toi and T. Awashima: “Stream Applications on the Dynamically Reconfigurable Processor”, Proceedings of International Conference on Field Programmable Technology (FPT2004), pp. 137–144 (2004).

[10] S. Kurotaki, N. Suzuki, K. Nakadai, H. G. Okuno and H. Amano: “Implementation of Active Direction-Pass Filter on Dynamically Reconfigurable Processor”, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2005), pp. 215–220 (2005).

[11] 阿部, 長谷川, 戸井, 犬尾, 天野：“動的リコンフィギャラブルデバイス DRP-1 におけるアダプティブコンピューティング”, 電子情報通信学会技術研究報告 RECONF (デザインガイア) (2005). to apear.

[12] Y. Hasegawa, S. Abe, H. Matsutani, K. Anjo, T. Awashima and H. Amano: “An Adaptive Cryptographic Accelerator for IPsec on Dynamically Reconfigurable Processor”, Proceedings of IEEE International Conference on Field Programmable Technology (FPT2005) (2005). to apear.

[13] L. Shang, A. Kaviani and K. Bathala: “Dynamic Power Consumption in Virtex-II FPGA Family”, Proceedings of ACM International Symposium on Field-Programmable Gate Arrays (FPGA2002), pp. 157–164 (2002).

[14] K. Poon, A. Yan and S. Wilton: “A Flexible Power Model for FPGAs”, Proceedings of the International Conference on Field-Programmable Logic and Applications (FPL2002), pp. 312–321 (2002).

[15] F. Li, D. Chen, L. He and J. Cong: “Architecture Evaluation for Power-Efficient FPGAs”, Proceedings of ACM International Symposium on Field-Programmable Gate Arrays (FPGA2003), pp. 175–184 (2003).

[16] Xilinx. http://www.xilinx.com/.