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

長谷川 揚平[†] 阿部 昌平[†] 黒瀧 俊輔[†]
ヴ マン トウアン[†] 天野 英晴[†]

動的リコンフィギャラブルプロセッサは, 粗粒度の Processing Element (PE) アレイによる並列処理と，動的再構成機能により，アプリケーションを時分割多重実行することで高い面積効率，電力効率を実現する．この特徴より System-on-a-Chip(SoC) における Intellectual Property コアとしての利用が期待されている．IP として実装する場合, 対象とするアプリケーションに対して高い面積効率と電力効率を実現することのできる PE アレイの構成をチップ設計の初期段階で明らかにしておくことが重要である．そこで, 本論文では，NEC エレクトロニクス社の Dynamically Reconfigurable Processor (DRP) を対象として，動的再構成機能による時分割多重実行の効果を定量的に評価し, 対象アプリケーションの要求性能に対して面積効率，電力効率を最適化する PE アレイサイズを予測する手法を提案する．この手法では, まず PE 数を無限大として時分割多重を行うモデルを基にコンテキスト分割と統合を行い, 与えられた PE サイズにおける性能およびコストを求める．実際にいくつかのアプリケーションを様々な PE アレイ構成に対して実装して評価を行い, 予測によって得られた値との比較を行った．

Evaluation of Time-multiplexed Execution on the Dynamically Reconfigurable Processor

Yohei Hasegawa,[†] Shohei Abe,[†] Shunsuke Kurotaki,[†]
Vu Manh Tuan[†] and Hideharu Amano[†]

Dynamically Reconfigurable Processors (DRPs) achieve a high degree of area/power efficiency by using both the parallel execution of coarse grain Processing Elements (PEs) and time-multiplexed execution with dynamic reconfiguration. They are expected to be a cost efficient Intellectual Property (IP) core in System-on-a-Chip (SoCs). By using IP cores, the structure of PE array which can achieve both a high degree of area efficiency and power efficiency to target applications in the early stage of chip design. Here, a method to prospect the efficient PE array size is proposed based on the quantitative evaluation of time-multiplexed execution. For a target device: NEC electronics' DRP-1, we propose a simple model for prospecting the performance and cost of possible PE array sizes. In this model, the infinite number of PEs are assumed, and then context division and integration are applied. Throughout the evaluation based on real application designs, we compared the prospected results with real ones, and discuss on the difference.

1. はしがき

近年，発展の目覚ましい動的リコンフィギャラブルプロセッサは，ハードウェアの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を集めている[1)]．近年の商用の動的リコンフィギャラブルプロセッサ[2),3)] は，8〜32bit 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上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，多くの 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，動的再構成機能により，ハードウェアの面積を削減し，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．さらに，高位合成技術との融合により，アプリケーションの開発期間を大幅に短縮することができる．

近年の動的リコンフィギャラブルプロセッサの最大の特徴は，マルチコンテキスト方式に基づく動的再構成機能である．物理的な PE アレイ上で実現されるデータパスをコンテキストと呼び，これを 1 クロック程度の短い時間で高速に切り替えることができる．対象となる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実行することができる．我々は，このマルチコンテキスト方式に基づく実行方式を時分割多重実行と呼ぶ．

時分割多重実行の概念は，細粒度の Field Programmable Gate Array (FPGA) を対象として，従来から活発に議論されてきた[4)]．近年，複数の企業より粗粒度の PE アレイによ

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

る高速な動的再構成能力をもつデバイスが登場し，FPGA に代表され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，電力効率を改善するデバイスとして期待されている．

一方，動的リコンフィギャラブルプロセッサは，System-on-a-Chip (SoC) における Intellectual Property (IP) コアとしての利用が想定されている．動的リコンフィギャラブルプロセッサ自身は，チップ製造後に用途に応じてユーザプログラム可能であるが，チップ製造段階では搭載する PE アレイのサイズやコンテキスト数を定義する必要がある．これらのパラメータは時分割多重実行において性能，面積，消費電力に直結する重要な要素である．このため，対象のアプリケーションに対して，高い面積効率および電力効率を得ることのできる PE アレイの構成をチップ設計の初期段階で明らかにしておくことが重要である．

一般的なプロセッサに関しては，設計時に構成をカスタマイズする技術が進んでおり，目的に応じて専用の命令セットを定義したり，専用ハードウェア回路を拡張するコンフィギャラブルプロセッサ[5)~7)] が既に広く利用されている．しかし，動的リコンフィギャラブルプロセッサの場合，様々なサイズの PE アレイで時分割多重実行をした際の，面積効率および電力効率に関する定量的な評価はあまり行われておらず，最適な PE アレイ構成を見出す方法論は確立されていない．

そこで, 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，時分割多重実行の性能，面積，消費電力の PE アレイサイズに対してのトレードオフを議論し, 対象アプリケーションの要求性能に対して面積効率，電力効率を最適化する PE アレイサイズを予測する手法を提案する．この方法では, まず，無限の PE アレイサイズを想定してアプリケーションの並列性を評価することで，それぞれの PE アレイサイズで実行時間とコストがどのように変化するかをモデル化し，それぞれのサイズに対してコンテキスト分割と統合を行うことで, 性能とコストを予測する．実際に 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，様々な PE アレイサイズのもとで設計し，理論値との比較を行う．さらに，面積効率，電力効率に関する評価も行い，対象アプリケーションに対する最適な PE アレイ構成を見い出す．本研究の最終目標は，対象のアプリケーションに対して必要性能を最も効率的に実現する PE アレイの構成を見出だすための方法論を確立することであり，PE アレイのサイズに対するモデルはそのための第一歩である．

本論文の構成は以下の通りである．節 2 では，対象デバイスである 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と，その予測手法について述べる．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，理論値との間の相違に関して検討する．

2. DRP アーキテクチャ

DRP 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[2)]．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，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．

DRP のプロトタイプチップ 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[8)] 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，テクノロジマッピング，配

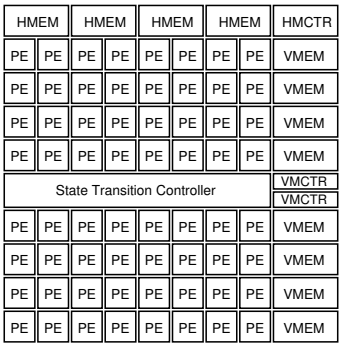

図 1 Tile の構造
Fig. 1 DRP Tile Architecture

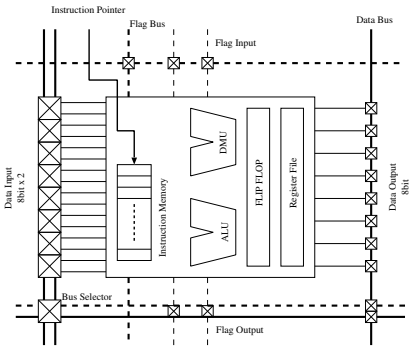

図 2 Processing Element (PE) の構造
Fig. 2 Processing Element (PE)

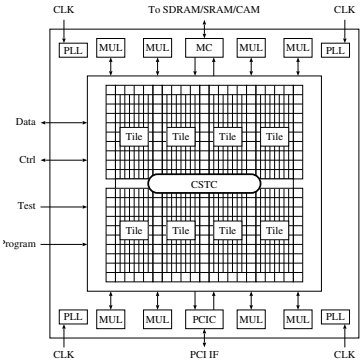

図 3 DRP-1 の構造
Fig. 3 Prototype Chip: DRP-1

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 時分割多重実行の基本モデル

DRP の対象とするアプリケーションは，組み込みシステム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．動的リコンフィギャラブルプロセッサにおいて，アプリケーションは複数のコンテキストに分割され，これらを動作中に切り替えながら実行する．このとき，コンテキストの切り替えは逐次的に行われるが，コンテキスト内ではアレイ状に配置された多数の PE と分散メモリによって並列処理が行なわれる．よって，アプリケーションの実行性能はコンテキストのスケジューリング方法に大きく依存する．

通常，計算機上で実行されるアルゴリズムには潜在的に並列性と逐次性が存在するため，アプリケーションを実行するためには一定のサイクルを必要とする．特に，DRP 上でストリーム処理を実行する場合，以下のような典型的な処理フローが考えられる．

(1) 処理するデータを分散メモリ (レジスタ) から読み出す
(2) 要求された処理を PE アレイ上で実行する
(3) 演算結果を分散メモリ (レジスタ) に書き戻す

ここで，これらの一連の処理を一つのステップと考える．通常，このようなステップは反復して実行され，ステップごとに少なくとも 1 クロックを要する．全てのステップが終了次第，出力データをチップ外に出力する．時分割多重実行では，ステップの処理は PE アレイからなるコンテキストにマッピングされ，ステップの遷移はコンテキストの切り替えに対応する．

このように，対象とするアプリケーションには逐次実行の制約があるが，並列性はそれぞれのステップに対して定義される．本論文では，各ステップにおける必要 PE 数をそのステップの並列性と定義する．**図 4** は，JPEG エンコーダにおける離散コサイン変換 (DCT) を DRP の PE アレイ上に実装した場合の各ステップにおける必要 PE 数を示す．この実装例では，無限個の PE と VMEM/HMEM が利用可能であると想定している．各ステップはアルゴリズムに従って

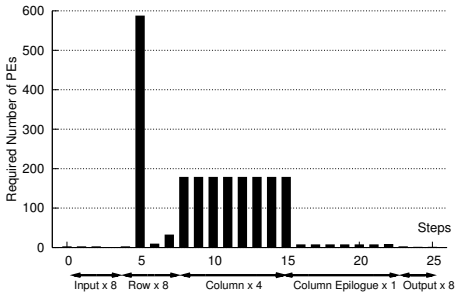

図 4 ステップ毎の必要 PE 数 (DCT)
Fig. 4 The number of required PEs for each step (DCT)

一定回数反復して実行される．

DCT は，主に 8×8 の画像データ上での行および列方向への積和・積差演算から構成される．この実装例では，それぞれ 8×8 の画像データを VMEM で構成される 8 個の独立した配列に格納している．これは，8 個の行方向のデータを同時に読み出し並列処理を行うためである．データの読み出しが終了した段階で，行方向の処理が 600 個近くの PE を使用して並列に処理される．なお，ここでの定数乗算は DRP コンパイラによって Shift & Add に自動的に展開される．一方，列方向の演算ステップでは，逐次的にデータを読み出す必要があることから並列性が低下する．

ここで，コンテキストサイズを一つのコンテキストあたりで利用可能な PE 数とし，N で表す．このとき，あるアプリケーションのステップの集合を S_N とし，ステップ $i \in S_N$ の利用 PE 数，反復回数，遅延時間をそれぞれ n_N^i, l_N^i, t_N^i とする．このときのアプリケーションの実行サイクル数と最大遅延時間をそれぞれ c_N, d_N とすると，

$$c_N = \sum_{i \in S_N} l_N^i, \qquad d_N = \max_{i \in S_N} \{t_N^i\}$$

となり，実行時間 e_N は $c_N d_N$ で与えられる．

ここで，図 4 の例のように，無限個の PE を想定 ($N = \infty$) する．このとき，S_∞ の要素の個数，すなわち総ステップ数の最小値 s_∞ は，アルゴリズム自身とデータ構造に依存することになる．よって，たとえ無限個の PE を想定しても，対象アプリケーションは少なくとも s_∞ ステップを必要とする．

ゆえに，c_∞ は，対象アプリケーションの DRP における実行クロック数の最小値を意味する．

本論文では，この無限個の PE を想定した評価結果を基に，コンテキストサイズを変えて時分割実行した場合に，性能とコストがどのように変化するかをモデル化し，理論式あるいは簡単なプログラムにより，実行時間と必要コンテキスト数を見積もる手法を示す．この手法により，無限個の PE を想定して評価した結果のみから，それぞれのコンテキストサイズでの性能とコストを見積もることができる．

3.2 コンテキストスケジューリング手法

これまで述べたように，時分割多重実行では，各ステップはあるコンテキストにマッピングされ，ステップの遷移はコンテキストの切り替えに対応する．しかし，コンテキストサイズ N と，利用可能コンテキスト数には限りがあるため，計算資源の利用効率を高めるために，実際には以下に示す操作が必要となる．

3.2.1 コンテキスト分割

各ステップが要求する利用 PE 数が，コンテキストサイズを越える場合，すなわち $n^i_\infty > N$ となる場合，そのステップ i をさらに複数のステップに小さく分割する必要がある．本論文では，これをコンテキスト分割と呼ぶ．**図 5** の A の領域のステップは，必要 PE 数がコンテキストサイズを越えているため，さらに小さく分割する必要がある．

コンテキストサイズ N が与えられたとき，コンテキスト分割により必要ステップ数の最小値 s_N は，

$$s_N = \sum_{i \in S_\infty} \left\lceil \frac{n^i_\infty}{N} \right\rceil \geq s_\infty$$

と定義される．また，このとき実行クロック数は c_N は，

$$c_N = \sum_{i \in S_N} l^i_N = \sum_{i \in S_\infty} \left(\left\lceil \frac{n^i_\infty}{N} \right\rceil l^i_\infty \right) \geq c_\infty$$

である．

実行クロック数は，コンテキストサイズが明らかになれば，上記の式で見積もり可能である．遅延時間に関しては，分割により一定の割合で削減されると仮定すれば，実行時間を見積もることができる．とはいえ，コンテキスト分割により，通常はデータパス自体を分割するため，遅延時間は削減されるが，実際には，分割の方法によって遅延時間に与える影響は異なる．この影響については実際のアプリケーションの評価時に議論する．

また，コンテキスト分割の損失として，分割箇所にデータ共有のためのレジスタが必要である．さらに，定数乗算のように，これ以上の分割が難しい場合には，分割にフラグメントが生じて無駄が増大すると同時に，最大遅延の増大を招く．この点についても実際のアプリケーションの評価を通して議論する．

3.2.2 コンテキスト統合

図 5 の B の領域のステップのように，並列性が少なく必要 PE 数の少ないステップが存在する場合，これらのステップを単一のコンテキストにそれぞれ割り当てると，PE の利用効率が極めて悪い．そこで，一つのコンテキストに利用 PE 数の少ない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．

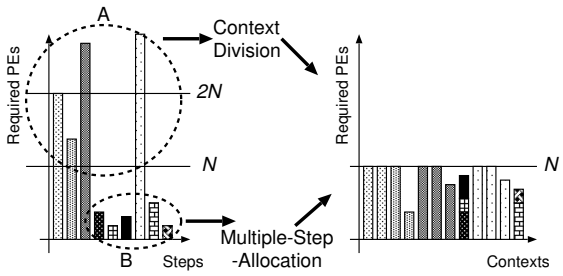

図 5 コンテキスト分割とコンテキスト統合
Fig. 5 Context Scheduling Technique

コンテキスト統合の概念を式で表すことは困難だが，コンテキストサイズ N の範囲内で統合するステップを決定するため，必要コンテキスト数 x_N は単純なプログラムで見積ることが可能である．なお，DRP の場合，同一コンテキストに統合可能なステップ数は最大で 4 までに制限されるが，全ステップがコンテキスト統合により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通常の実行方式に対応する．

コンテキスト統合は，確かに PE の利用効率を高め，必要コンテキスト数を削減するが，実行時間を改善するものではない．その主な理由は，コンテキスト統合を行っても，s_N および c_N が変化しないためである．

一方，コンテキスト統合により，単一コンテキストのデータパス全体が平面的に拡大するため，配線遅延が増大する傾向にある．また，単一コンテキスト内の複数のステップ間でデータを共有する場合に限り，マルチプレクサを追加する必要があり，場合によっては必要 PE 数を増加させ，最大遅延時間を増大させる．このため，不必要なコンテキスト統合を行わないように注意が必要であるが，最大遅延パスを含むステップは，通常はそれ自身が多くの PE を消費するため，コンテキスト統合の対象ステップとなることはない．よって，多くの場合，コンテキスト統合による性能およびコストへの影響は少ない．ここでは，理論的に見積もりを行う場合は，コンテキスト統合での遅延の増加は生じないと仮定する．

3.3 コンテキストサイズのトレードオフ

これまで述べたように，コンテキストサイズ N はコンテキストあたりの利用可能 PE 数を意味する．DRP の場合，1 つの Tile 中に 64 個の PE とそれに応じた VMEM/HMEM を装備しており，コンテキストサイズの変更は Tile 単位で行なわれる．よって，N は 64 の倍数となる．実際には，全 PE を用いた場合，配置配線が困難となることから，利用 PE 数は N の 80% 程度に抑えることが望ましい．

N を大きく取れば，多くの PE による並列処理により，性能を向上させることが可能である．しかし，多くのアプリ

ケーションにおいて，各ステップにおける並列性 n_N^i は一様ではないため，N を大きくしすぎると PE の利用効率が悪化し，面積あたりの性能が低下する．さらに，面積が増加することで，配線遅延と消費電力の増大を招く．また，PE の利用効率を高めるために，不必要なコンテキスト統合を行うと，最大遅延時間の増大により，却って性能を悪化させてしまう可能性がある．

一方，コンテキストサイズを小さくすると，PE の利用効率は向上し，面積効率と電力効率は共に向上する．しかし，コンテキスト分割により絶対性能が低下することから要求性能を満足できない可能性がある．また，コンテキストサイズが小さすぎると，必要ステップ数および必要コンテキスト数の増加とともにコンテキストスイッチの頻度が高まり，付加的な消費電力の増加を招くことが予想される．

4. 評価および理論モデルとの比較

4.1 実アプリケーションによる評価

前節で示したモデルと実際のアプリケーションの挙動の差を調べるため，評価用アプリケーションとして，以下に示すストリーム処理を選んだ．

- JPEG エンコーダにおける離散コサイン変換 (DCT)
- MP3 デコーダにおける変形逆離散コサイン変換 (IMDCT)
- 高速フーリエ変換 (FFT)
- 誤り訂正符号 Viterbi デコーダ (Viterbi)
- 共通鍵ブロック暗号 Advanced Encryption Standard (AES)
- 一方向ハッシュ関数 Secure Hash Algorithm 1 (SHA-1)

これらのアプリケーションのアルゴリズムは BDL で記述し，DRP コンパイラを用いて合成した．

ここで，時分割多重実行のトレードオフを評価するため，それぞれのアプリケーションをコンテキストサイズ N を変えながら実装した．具体的には，まず $N = \infty$ を想定してアルゴリズムの並列性を測定し，8 Tile で実装できる範囲で並列性を高めるようにデータ構造等の工夫を行った．次に，アルゴリズムは変えずに，コンテキストサイズ N を変化させ，それぞれのコンテキストサイズで性能が最も高くなるようにチューニングを行った．ここで，コンテキスト分割および統合は DRP コンパイラが自動的に行う方法と，人手で行う方法が選択可能である．本評価では，コンテキスト分割は最も性能が高くなるように可能な限り人手で最適化を行った．一方，コンテキスト統合は，あまり性能，コストに影響を与えないため，例外を除き DRP コンパイラに委ねることにした．

なお，N に制約を設けない場合，利用する PE 数が 8 Tile に実装可能な範囲を越えてしまうと，DRP-1 上に配置配線を行うことができない．この場合は，配置配線前の結果を利用している．配置配線後の結果と比べ，利用ステップ数，コンテキスト数に違いはないが，配置段階で回路の最適化が行なわれるため，利用 PE 数は多めに算出される傾向にある．

また，N に制約を設けない場合の実装結果より，節 3 で示した時分割多重実行モデルを用いて，利用 PE 数，コンテキスト数，実行時間を見積もり，実装結果と比較する．

4.2 実 装 結 果

表 1 に各アプリケーションの実装結果を示す．左から想定 Tile 数，全コンテキストのうちの最大利用 PE 数，最大遅延時間，最大動作周波数，必要クロック数，消費電力を示している．なお，括弧内の数値は前節のモデルより求めた理論的な見積もり値を示す．$N = \infty$ は，無限の PE アレイサイズを想定して，コンテキスト分割および統合を行っていない状態での測定結果であり，理論的な見積もりの基盤として用いられる．

$N = \infty$ での最大利用 PE 数は，アプリケーションの最大並列性を意味する．DCT, IMDCT, Viterbi, AES-CBC は FFT, SHA-1 に比べて相対的に並列性の高いアプリケーションであるといえる．また，最大遅延時間は DRP コンパイルフローにおける配置配線後のレポートから算出した値である．必要クロック数は，入力ストリームの 1 ブロックデータあたりの実行クロック数である．

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と，利用 PE 数や最大動作周波数などの実装結果より電力値を算出しており，実チップから測定した場合と異なる可能性がある．また，対象とする消費電力は 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，ここでは算出される電力値はサイズを 8 Tile に固定しているため，電力プロファイラの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算出している．

評価結果より，前節のモデルで予測した通り，N の減少に伴ないコンテキスト分割により必要コンテキスト数は増加し，最大利用 PE 数は減少している．前節で示したモデルから得られる理論的な見積りからも同様の傾向が確認できる．しかし，理論値はコンテキスト分割および統合のロスを考慮していないため，場合によっては実測値と比較するとある程度の差が生じている．また，理論式はコンテキスト統合による遅延時間の増大を無視しているため，FFT や SHA-1 などの並列性の低いアプリケーションでは，可能な限りコンテキスト統合を行ってコンテキスト数を減らした場合の設計となっている．しかし，実際には各サイズで性能を最適化する設計を行い，統合によって遅延時間が伸びて性能に影響を与える場合，これを避けている．このため，理論式からの見積もりに比べ，最大利用 PE は少なめ，利用コンテキスト数は多めになっている．

一方，必要クロック数は，見積もり値に示す傾向にしたがって，N の減少に伴ない増加している．しかし，DCT および Viterbi など並列性の高いアプリケーションで，見積もりとの差が大きくなっている．これは，並列性の高いアプリケーションでは，一定以上分割すると，レジスタの挿入や，これ以上分割が難しい演算などにより，分割のオーバヘッドが大きくなるためと考えられる．

表 1 評価アプリケーションの実装結果
Table 1 Implementation Results of Target Applications

Application	Tile ($N/64$)	Context (x_N)	Max PEs	Delay[ns] (d_N)	Freq.[MHz] ($1/d_N$)	Clocks (c_N)	Power[mW] (p_N)
DCT	∞	26	588	87.7	11.4	73	-
	8	16 (10)	259 (379)	82.0	12.2	92 (81)	345.4
	6	25 (14)	186 (304)	76.0	13.2	153 (81)	269.0
	4	34 (16)	148 (200)	67.3	14.9	168 (89)	217.6
	2	64 (27)	73 (98)	69.2	14.4	332 (145)	113.1
IMDCT	∞	16	888	160.6	6.2	1347	-
	8	13 (9)	360 (333)	95.5	10.5	3681 (3773)	248.3
	6	14 (11)	280 (301)	102.4	9.8	3791 (3837)	180.1
	4	20 (14)	183 (198)	83.7	11.9	3928 (3901)	164.7
Viterbi	∞	3	893	60.0	17.6	2	-
	8	7 (4)	320 (364)	27.2	36.8	8 (4)	1136.0
	6	8 (5)	204 (298)	27.7	36.1	10 (4)	908.1
	4	9 (7)	160 (179)	31.0	32.2	11 (6)	611.8
AES-ECB	∞	8	514	29.1	34.3	10	-
	8	6 (4)	448 (357)	27.0	36.2	20 (20)	1092.5
	6	7 (6)	224 (257)	22.2	45.0	29 (20)	1080.0
	4	8 (7)	224 (172)	27.6	37.1	29 (30)	627.3
FFT	∞	14	61	40.1	24.9	49050	-
	8	5 (4)	101 (123)	35.6	28.1	49050 (49563)	731.4
	4	6 (4)	87 (123)	37.2	26.9	49050 (49563)	373.2
	2	7 (5)	73 (83)	38.4	26.1	49050 (49563)	225.3
	1	16 (7)	33 (51)	23.0	43.4	73125 (60827)	186.9
SHA-1	∞	14	61	35.1	28.5	93	-
	8	8 (4)	61 (217)	27.6	36.2	93 (93)	806.8
	6	8 (4)	61 (217)	27.2	36.8	93 (93)	624.1
	4	8 (4)	61 (156)	29.5	33.9	93 (93)	394.8
	2	8 (5)	61 (94)	29.6	33.8	93 (93)	213.8
	1	10 (10)	53 (46)	31.2	32.1	94 (135)	114.3

伝搬遅延時間の見積もりは前節のモデルは分割によって一定の割合で減少し，統合によっては変化しないと考えたが，実際の設計ではアプリケーションによる依存性が大きい．SHA-1 と FFT は両者共に並列性は小さいが，FFT はコンテキスト統合がさほど遅延時間に影響しないため，N が大きい範囲でコンテキスト数を削減できている．一方で，SHA-1 は遅延時間の増大を抑えるためには，コンテキスト統合を行うことができず，N が大きくてもコンテキスト数は減っていない．モデル通り，コンテキスト分割により，最大遅延時間は減少する傾向にあるが，Viterbi や SHA-1，FFT では，かえって増加するケースがみられた．これは，N の減少により，PE の利用密度が増加し配置配線の自由度が低下したことで配線長が増大したためである．

消費電力も前節のモデルでは予測ができない要素である．基本的な傾向として，消費電力はコンテキストサイズの減少に伴ない減少することがわかった．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 の総数が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，単純な回路による実験から平均 5% 程度であることがわかった．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ことがわかった．

4.3 トレードオフの検討

4.3.1 トレードオフの指標

本節では，前節で示した評価アプリケーションにおいて，各コンテキストサイズに対する性能，面積，消費電力のトレードオフの検討する．

まず，ある N に対する各アプリケーションの性能は，実行時間 $e_N = c_N d_N$ を指標とする．

次に，各アプリケーションに対する面積コストを，使用した Tile の総面積と定義する．面積コストには，使用した PE と VMEM/HMEM, 各 PE の命令メモリの面積が含まれる．DRP では，VMEM/HMEM は，64PE から成る Tile 単位に対して装備されるため，コストは Tile の個数に対して線形に増加することになる．このとき，あるコンテキストサイズ N に対する面積コスト f_N を以下で定義する．

$$f_N = (1 + \gamma x_N) \left\lceil \frac{N}{64} \right\rceil$$

ここで，γ は，1 つの PE のデータパス部の面積に対する命令メモリ 1 コンテキスト分の占める面積の割合を示す．

ここで，γ は，1 つの PE の面積に対する命令メモリ 1 コンテキスト分の占める面積の割合を示す．DRP-1 における面積の割合は公表されていないが，16bit の PE を用いた動的リコンフィギャラブルプロセッサ ADRES[9] では，32 コンテキスト分の命令メモリの面積が 1 PE にほぼ等しくこの値は，0.031 である．DRP-1 は 8bit の PE であるため，や

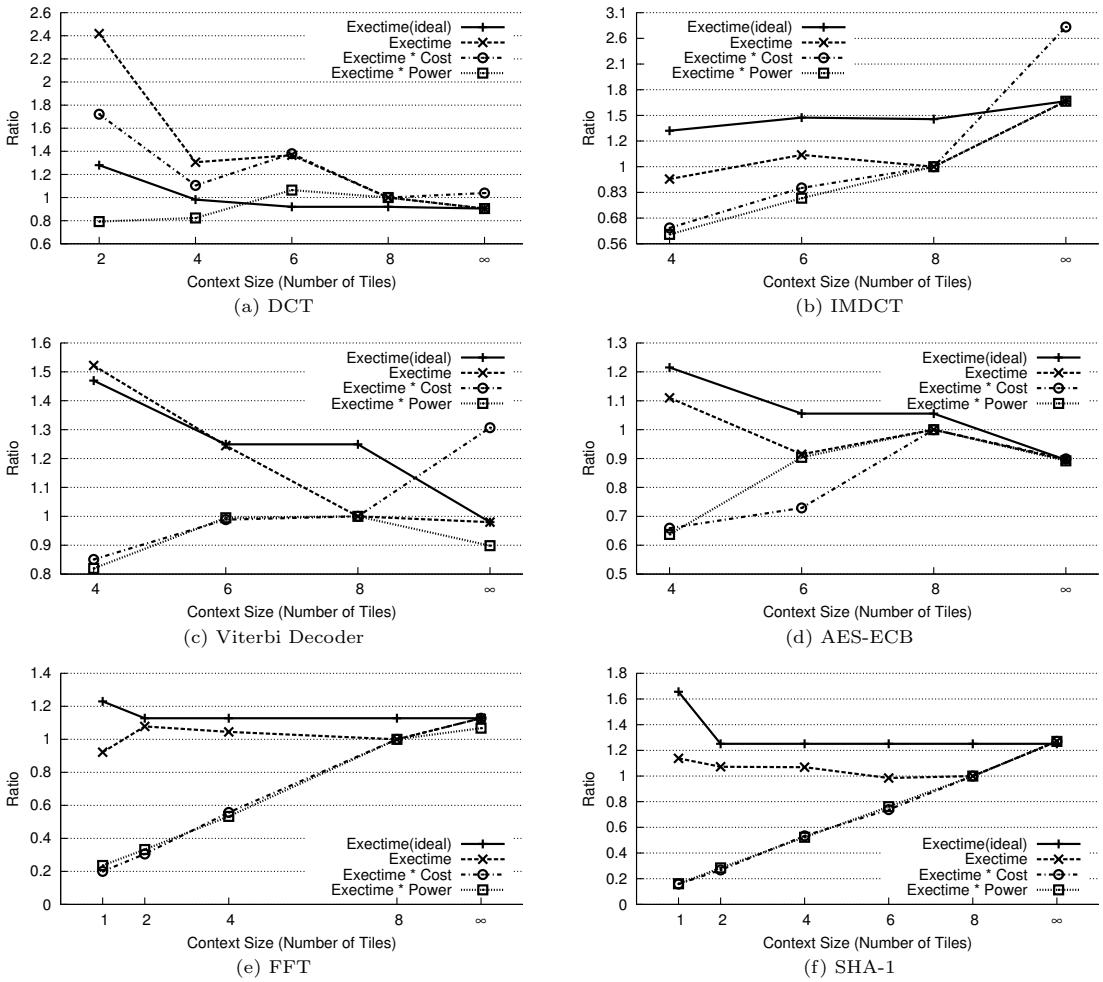

図 6 コンテキストサイズに対する性能と面積，消費電力のトレードオフ
Fig. 6 Performace, Area Cost, and Power Consumption v.s Context Size

や大きめに見積もり，本論文では，$\gamma = 0.1$ と仮定した．

さらに，本評価では，x_N が命令メモリに保持可能なコンテキスト数以下であることが条件となる．現在の DRP-1 は，16 コンテキストを保持可能であるが，次世代のチップでは，アーキテクチャを大きく変えることなく，64 コンテキスト程を保持可能であると考えられる．このため，本論文では，$x_N \leq 64$ とした．

以上より，時分割多重実行におけるコスト性能比 (性能に対して必要なコスト) の指標として，$r_N = f_N e_N$ を定める．

消費電力の性能指標は、バッテリーに対する負荷を考える必要があることから，それぞれのアプリケーションを実行するのに必要な総エネルギー量とした．必要な総エネルギー量 j_N は，実行時間と消費電力の積 $e_N p_N$ で表される．

4.3.2 結果および理論モデルとの比較

各アプリケーションに対する評価結果を**図 6** に示す．なお，それぞれの値は，想定 Tile 数 8 ($N = 512$) の実装結果を基準として正規化している．指標のうち、実行時間とコスト性能比は，理論モデルから予想可能であるが，グラフの混乱を避けるため，ここでは実行時間のみを Ideal として示した．なお，図中では，実行時間 e_N を Exectime，コスト性能比 r_N を Exectime*Cost，必要エネルギー量 j_N を Exectime*Power と表記している．

実行時間の評価結果より，ほとんどのアプリケーションに対して，並列性の許す限りは性能的には改善することが可能であることがわかる．特に並列性の高いアプリケーションでは，実行サイクル数が大幅に削減されるため，その効果が大きい．一方，並列性の低い FFT と SHA-1 では，1,2 Tile 以上にコンテキストサイズを増加させても，実行クロック数を大きく削減できないため，実行時間も横ばいである．

また，実行時間の理論的な見積りと実測値を比較すると，DCT や Viterbi など並列性が高いアプリケーションでは，先に検討したようにコンテキスト分割および統合のロスが顕在化する場合には両者の差が大きいが，性能の向上が限界にあたる Tile 数は正しく見積ることが可能である．例えば，DCT の場合には，理論値では 4 Tile 以上のサイズでは性能は横ばいである．実測値では 2 Tile から 4 Tile にサイズを増やすことで大きく性能が向上するが，それ以降では理論値の見積りと同様，大きな性能向上は達成していない．むしろ，

4 Tile と 6 Tile のケースを比較するとわかるように，遅延時間の増大により却って性能を損ねることがある．

Viterbi の場合，理論値の見積りでは，6 Tile と 8 Tile では性能に変化はみられないが，8 Tile 以上のサイズを設けることで性能向上が期待できる．実際に，実測値では 6 Tile から 8 Tile にサイズに増やすことで性能向上が達成されており，Viterbi は 8 Tile 以上に N を増やすことで性能向上が期待できるアプリケーションの例であるといえる．AES-ECB は，Viterbi と同様，理論値から 8 Tile 以上のサイズを設けることで性能向上が期待できるが，その度合はあまり大きくなく，実測値と同様 6 Tile に性能向上の限界が存在するといえる．IMDCT では 4 Tile，FFT と SHA-1 では 2 Tile 程度のサイズが性能向上の限界であり，理論値からも同様の傾向が確認できる．

次に面積効率に関しては，DCT の場合では，実行時間の改善がみられない 6 Tile のケースが効率が悪いが，8 Tile に増やすことで面積効率を高めることが可能であることがわかる．その他のアプリケーションに関しては，実装した中で最も小さいコンテキストサイズのケースが最も面積効率が高い結果となった．これは，コンテキストサイズを増加した場合の実行時間の改善がコストの増加割合を下回ったためであり，理論的な見積もりからも同様の結果が得られている．

必要な総エネルギー量も，面積効率の評価結果と同様，全てのアプリケーションにおいて，実装した中で最もコンテキストサイズの小さいケースが最も少ない必要エネルギー量となり，最も電力効率が高い結果となった．DCT の場合には，2 Tile から 4 Tile に増やすことで実行時間を大きく削減することができるが，それよりも消費電力の増加割合いが高いため，2 Tile のケースの方が必要エネルギー量が少ない．

評価全体を通して，面積効率と電力効率の間には相関関係があり，コンテキストサイズの増減に対して同様の変化の傾向を示すことがわかった．また，面積効率を最適化するコンテキストサイズを選択することで，多くの場合において電力効率も最適化できることがわかった．

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ける，時分割多重実行の性能，コスト，消費電力のトレードオフを議論した．性能とコストについては，無限のコンテキストサイズを想定してアプリケーションの並列性を評価することで，それぞれのコンテキストサイズで実行時間とコストがどのように変化するかをモデル化し，それぞれの値を見積もる方法を提案した．

次に，実際にいくつかの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，見積もりが正しいかどうかを検討した．各アプリケーションを異なるコンテキストサイズを想定して設計し，面積効率，電力効率を最適化するコンテキストサイズを評価した．その結果，理論モデルからの見積もりは，かなりのケースで実際の設計からの値を予想することが可能だが，いくつかの場合で差が大きくなることが明らかになり，その原因はコンテキストの分割と統合に関する遅延時間の変化を正確に予測することが難しいこと，コンテキスト分割ロスの見積もりが正確でないことが主である点が明らかになった．

また，評価結果より，コンテキストサイズの増減に対して面積効率と電力効率の間には相関関係があることがわかった．また，面積効率を最適化するコンテキストサイズを選択することで，多くの場合において電力効率も最適化できることがわかった．

以上の結果から，アプリケーションに対するコンテキストサイズの決定の方法には以下の方法が考えられる．(1) 無限のコンテキストを想定してアプリケーションの並列性を測定する．(2) 本論文で提案したモデルを用いて実行時間を見積もる．(3) 実行時間が要求仕様を満足するために必要な最も小さな Tile を用意することで面積効率および電力効率は最大になる．

ただし，今回の評価は，アプリケーションの数も限定されており，さらに PE の粒度も 8bit に固定されている．今後は，さらに広い範囲でトレードオフを検討していく必要がある．一方，FPGA におけるアーキテクチャ評価はすでに多数報告されており，近年では特に高電力効率を実現するアーキテクチャがすでに検討されている．今後は，Xilinx 社の Spartan FPGA など，組み込み分野への応用を想定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．

謝辞 本研究を進めるにあたり DRP-1 とその開発環境を提供していただいた 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の皆様に感謝致します．

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 Transaction on Information & System*, Vol. E87-D, No. 8, pp. 1997–2003 (2004).
4) Trimberger, S., Carberry, D., Johnson, A. and Wong, J.: A Time-Multiplexed FPGA, *Proc. of FCCM97*, pp. 22–28 (1997).
5) Gonzalez, R. E.: Xtensa: A Configurable and Extensible Processor, *IEEE Micro*, Vol. 20, No. 2, pp. 60–70 (2000).
6) 宮森: コンフィギャラブルプロセッサ MeP とその SoC 応用開発事例, 電子情報通信学会技術研究報告 CPSY2004-81, pp. 31–36 (2005).
7) Arnold, J.: S5: The Architecture and Development Flow of a Software Configurable Processor, *Proc. of FPT2005*, pp. 120–128 (2005).
8) 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, 信学技報 VLD2003-118, pp. 23–28 (2004).
9) Veredas, F., Scheppler, M., Moffat, W. and Mei, B.: Custom Implementation of the Coarse-Grained Reconfigurable ADRES Architecture for Multimedia Purposes, *Proc.of FPL2005*, pp. 106–111 (2005).