Title	動的リコンフィギャラブルプロセッサにおける並列タスクのデータ転送を隠ぺいするための効果的な処理法
Author(s)	荒木，光一；佐藤，幸紀；井口，寧
Citation	電子情報通信学会論文誌 D，J92-D(12)：2137-2146
Issue Date	2009-12-01
Type	Journal Article
Text version	publisher
URL	http://hdl.handle.net/10119/9171
Rights	Copyright (C)2009 IEICE. 荒木 光一，佐藤 幸紀，井口 寧，電子情報通信学会論文誌 D，J92-D(12)，2009，2137-2146. http://www.ieice.org/jpn/trans_online/
Description	

動的リコンフィギャラブルプロセッサにおける並列タスクのデータ転送を隠ぺいするための効果的な処理法

荒木　光一[†]　　佐藤　幸紀[††]　　井口　寧[††]

An Effective Processing Method for Hiding Data Transfer of Parallel Tasks on a Dynamically Reconfigurable Processor

Koichi ARAKI[†], Yukinori SATO[††], and Yasushi INOGUCHI[††]

あらまし　動的リコンフィギャラブルプロセッサ（DRP）においてデータ並列性が高いタスク（データ並列タスク）を高並列な SIMD 型回路により処理する場合，DRP の入出力ポートの制約により SIMD 型回路と外部モジュールの間でデータ転送を直接できないため，DRP 内の大容量メモリにデータをバッファリングした後に SIMD 処理する必要がある．しかし，全実行時間においてデータをバッファリングするためのデータ転送時間の割合が高い場合，データ転送時間がボトルネックとなり，DRP の性能を十分に発揮することができない．そこで，本論文では，データ転送を隠ぺいするためにデータ並列タスク処理とデータ転送をオーバラップさせる手法を提案する．データ転送の回路とデータ並列タスク処理の回路を別々に配置してパーシャルコンテクストスイッチをすることで，それらをオーバラップさせる．NEC エレクトロニクス社の DRP-1 で評価を行った結果，提案手法はバッファリングによる SIMD 処理と比較して最大で全実行クロック数を 57%削減し，使用リソース数も削減できた．また，回路の動作周波数も向上したことにより全実行時間を 1/4 以下に削減できた．

キーワード　動的リコンフィギャラブルプロセッサ，パーシャルコンテクストスイッチ，データ転送，データ並列性

1. まえがき

近年，System-on-a-Chip（SoC）における Turn Around Time と開発コストの問題を解決する動的リコンフィギャラブルプロセッサ（DRP）が注目されている [1]〜[5]．従来の SoC に搭載されている ASIC は特定のタスクのみハードウェア処理するが，DRP は柔軟性を利用することで様々なタスクをハードウェア処理できる．これにより，新しい技術が登場した場合，ASIC や SoC を再開発せずに対応できる．また，必要に応じてタスクに特化したハードウェアを実現することもできる [6]．

† 北陸先端科学技術大学院大学情報科学研究科，能美市
School of Information Science, Japan Advanced Institute of Science and Technology, Asahidai1–1, Nomi-shi, 923–1292 Japan
†† 北陸先端科学技術大学院大学情報科学センター，能美市
Center for Information Science, Japan Advanced Institute of Science and Technology, Asahidai1–1, Nomi-shi, 923–1292 Japan

中でもマルチコンテクスト方式の DRP は，内部に複数のコンテクスト（回路情報）を保存することによって 1 クロックでコンテクストスイッチ（再構成）を行う．したがって，動的にコンテクストスイッチを行い処理することで仮想的に巨大なハードウェアを実現でき，面積効率を向上させることができる．

JPEG の DCT 処理などに代表されるデータ並列性をもつタスク（データ並列タスク）は異なるデータに対して同じ処理するので，SIMD 型回路を実現することで高速に処理できる [7]．DRP でデータ並列タスクの処理を SIMD 処理する場合，DRP 上に実現した SIMD 型回路の並列度次第では，入出力ポートのビット数の制約により SIMD 型回路のすべての Processing Unit（PU）と外部モジュール間でデータを同時に転送できない可能性がある．したがって，DRP 内の大容量メモリにデータをバッファリングして，外部モジュールとのデータ転送を行う必要がある．しかし，この手法ではデータ並列タスク処理時間とは別にデータ転送時間が発生してしまう．そのため，全実行時間におい

てデータ転送時間の割合が高い場合，データ転送がボトルネックとなってしまう．

そこで，本論文では，データ並列タスクの処理におけるデータ転送を隠ぺいするためにオーバラップ法を提案する．オーバラップ法は，データ転送のコンテクストとデータ並列タスク処理を行うコンテクストを別々に配置して独自に（部分的に）コンテクストスイッチを行うことで，データ並列タスク処理とデータ転送をオーバラップさせる．これにより，データ転送を隠ぺいし，データ並列タスクの処理に必要な全実行クロック数と全実行時間を削減する．本論文で対象とするDRPは，NECエレクトロニクス社のDRP-1である．

以降の本論文の構成は，**2.** で問題点と関連研究を述べ，**3.** でDRP-1について述べる．**4.** では，提案手法であるオーバラップ法を説明し，**5.** で評価する．**6.** で本論文をまとめる．

2. 問題点と関連研究

2.1 データ転送のボトルネック

データ並列タスク処理は，データの並列性を利用するSIMD処理で高速化できる．SIMD処理は，入力データを外部モジュールからSIMD型回路の全PUへ一度に転送することで，データ並列タスクを並列処理する．そして，出力データを全PUから外部モジュールへ一度に転送する．

しかし，SoCを対象としているDRPの入出力ポートのビット数は小さいため，DRPでデータ並列タスクをSIMD処理する場合，全PUは外部モジュールとのデータ転送を直接できない．したがって，DRP内の大容量メモリをSIMD型回路と外部モジュール間のバッファメモリとして利用し，入出力データをバッファリングすることで，外部モジュールとデータ転送を行う必要がある．本論文では，この手法を従来手法としてバッファリング法と呼ぶ．また，バッファリングするためのDRP内の大容量メモリをバッファメモリと呼ぶ．

図1にバッファリング法の実行モデルを示す．バッファリング法は，Input Turn → SIMD Turn → Output Turnの順番で実行する．Input Turnで，バッファメモリに全PUへ転送する分の入力データを保存する．SIMD Turnでは，全PUがバッファメモリから同時に入力データを読み出し，コンテクストスイッチを行いながら処理する．出力データは，バッファメモリに再び保存される．そして，Output Turnにて外部モジュールへ出力データを転送する．入力データが残っている場合は，Input Turnから再度実行し，入力データがなくなるまで繰り返す．

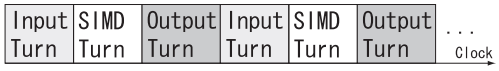

図1 バッファリング法（従来手法）の実行モデル
Fig. 1 Execution model of buffering method. (Typical method)

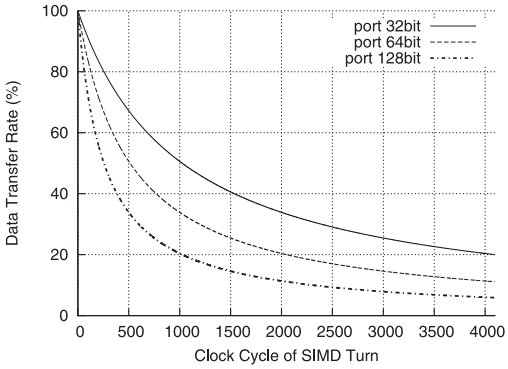

図2 データ転送クロック数の割合
Fig. 2 Rate of data transfer clock cycle.

図1から分かるように，バッファリング法はデータを転送するための時間が必要となる．Input TurnとOutput Turnのクロック数は，処理するデータ並列タスクの入出力データ量とデータのビット数によって決まるため固定となる．したがって，DRPの大規模化によりPU数が増加してSIMD Turnのクロック数が低下した場合，データ転送がボトルネックとなる．

例として，図2にJPEGのDCT処理におけるSIMD Turnのクロック数に対するデータ転送クロック数の割合を示す．入出力データは32 × 32サイズの画像で1データのビット数が16 bitである．SIMD Turnのクロック数が低下するにつれて，データ転送クロック数の割合が増加していることが分かる．また，入出力ポートのビット数を32 bit（port 32 bit）から128 bit（port 128 bit）に増加させても，SIMD Turnのあるクロック数から急激にデータ転送クロック数の割合が増加することも分かる．このことから，データ並列タスクの処理において，データ転送は今後無視することができないことが分かる．

2.2 関連研究

これまでにも，データ転送のボトルネックを解消することを目的とした研究はいくつか行われてきた[8], [9].

El-Araby らは，リコンフィギャラブルコンピュータの SRC-6E においてデータ転送とタスク処理のオーバラップさせる手法を提案し [8]，システムレベルでの並列性を解析することでパフォーマンスを向上させた．ホストメモリと FPGA 間にある FPGA 専用の大容量メモリは，FPGA 上の SIMD 型回路の全 PU で並列処理できるデータ量をバッファリングした後，FPGA 上の全 PU と並列にデータ転送し，同時に次のデータをバッファリングする．これにより，データ転送を隠ぺいした．しかし，DRP に彼らの手法を適応した場合，入出力ポートの制約により全 PU へのデータ転送とホストメモリとのデータ転送を同時に行うことができないため，結果的にバッファリング法と同様になる．

天野らは，DRP-1 でデータ転送を隠ぺいするためにダブルバッファリング機構を提案した [9]．DRP-1 内の大容量メモリをデータ転送用のメモリとタスク処理用のメモリの二つに分割し，バッファリングとデータ転送処理を同時に行う．データ転送とタスク処理が終了したら，データ転送用のメモリをタスク処理用のメモリに，タスク処理用のメモリをデータ転送用のメモリにスイッチングさせる．これにより，データ転送とタスク処理をオーバラップさせる．また，データ転送を独自のクロックで動作させるために，大容量メモリのスイッチやタスク処理との同期をとる配線などを大容量メモリの周辺に追加し，データ転送用のコンテクストを増設することで，タスク処理のクロックと独立させた．しかし，文献 [9] によれば，スイッチイングのための大容量メモリの拡張とデータ転送を独自にクロックで動作させるための機構の追加が必要であるため，簡単であるが DRP-1 を改良する必要がある．また，タスク処理同士のオーバラップに関しては，議論されていない．

これらの先行研究に対し，本論文のオーバラップ法は DRP-1 を現状のままデータ転送を隠ぺいすることが可能である．更に，DRP-1 のシステムとデータ並列タスクの両面から解析することで，タスク処理同士をオーバラップさせることができる．また，データ転送されてきた入力データは順次処理されるので，入出力データをバッファリングする必要がない．

3. DRP-1

3.1 DRP-1 の構成

DRP-1 は，NEC エレクトロニクス社が開発した DRP のプロトタイプであり，タスク処理を行う DRP コアと DRP コアの制御回路で構成されている．図 3 に DRP コアの構造を示す．0.15 μm の CMOS プロセスで設計されている DRP コアは，DRP である Tile，全体のコンテクストスイッチの制御を行う Central State Transition Controller（CSTC），各 128 bit の入出力ポートから構成されている．

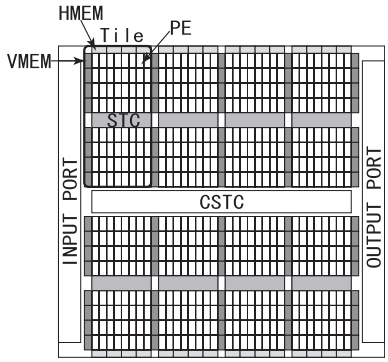

図 3　DRP コアのアーキテクチャ
Fig. 3　Architecture of DRP core.

DRP コアに 8 個搭載されている Tile には，8 bit の ALU などからなる基本要素の Processing Element（PE）が 8 × 8 個のアレー上に配置されており，その中央に State Transition Controller（STC）が配置されている．各 PE には 16 個のコンテクストを保存でき，STC からの命令ポインタによって実現するコンテクストを決定する．これにより，1 クロックでコンテクストスイッチができる．PE アレーの周囲には，2 kbit（8 bit × 256）の 2-port の VMEM と 64 kbit（8 bit × 8192）の 1-port の HMEM が配置されている．

3.2 パーシャルコンテクストスイッチ

各 Tile は独自に STC をもつので，DRP コア全体から見れば部分的にコンテクストスイッチを行うことができる．部分的にコンテクストスイッチを行うことをパーシャルコンテクストスイッチ（PCS）と呼ぶ．DRP-1 では，マルチプロセスと呼ばれる実行法で PCS を実現できる．マルチプロセスは，複数のプロセスからなるタスクを Tile 単位にプロセスを割り当てることでストリーム処理をする．

図 4 にマルチプロセスによる PCS の例を示す．各 Tile 間のデータ転送は，VMEM を FIFO として利用することで行われる．FIFO となる VMEM は，データを送信側の Tile と受信側の Tile を一つだけ指定できる．したがって，一つの VMEM から複数の Tile へデータの転送はできない．また，1 クロックで転送で

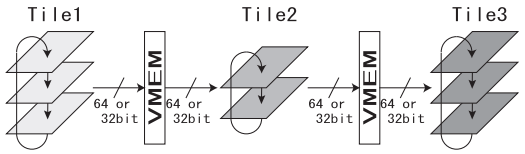

図 4　マルチプロセスによるパーシャルコンテクストスイッチ
Fig. 4　Partial context switch using multi-process.

きるビット数は 32 bit，または，64 bit である．

4. オーバラップ法

本論文では，データ並列タスクの処理において発生するデータ転送のボトルネックを解決するために，データ転送を隠ぺいするオーバラップ法を提案する．オーバラップ法は，データ転送のコンテクストとデータ並列タスク処理を行うコンテクストを Tile 単位で分割して PCS することで，データ転送とデータ並列タスク処理をオーバラップさせる．また，バッファリング法の SIMD 処理の間は，入出力データ転送を一時停止する必要があるが，オーバラップ法は入力データと出力データの片方，若しくは，両方の転送を連続的に行う．データ並列タスク処理の 1 ブロックのデータに対する処理を本論文では 1 タスクと呼ぶ．

4.1 オーバラップ法の構成と動作

オーバラップ法は，Input Stage（IS），Output Stage（OS）と Processing Stage（PS）の三つのステージに DRP コアを Tile 単位で分割する．バッファリング法の Input Turn と Output Turn では外部モジュールとバッファメモリ間のデータ転送を行うが，オーバラップ法の IS と OS は外部モジュールと PS 間のデータ転送を行う．PS は，データ並列タスク処理を行い，複数の Processing Tile（PT）で構成する．各 PT はデータ駆動で実行し，1 タスク分の処理をする．したがって，全 PT は同じ処理を行うので，回路構成，コンテクスト数とタスク処理の実行クロック数はすべて同じである．バッファリング法の SIMD Turn の回路は，DRP コア内の PE を可能な限り利用して SIMD 型回路を構成するが，オーバラップ法の PT は IS と OS 以外の Tile を利用して Tile 単位で構成する．

各 Stage は Tile 単位で分割されるので，IS，OS と PS の各 PT は PCS を行うことができる．したがって，ある PT がタスク処理を行っている間に，他の PT，IS，OS は独自にコンテクストスイッチを行いデータ転送，タスク処理を行うことができる．

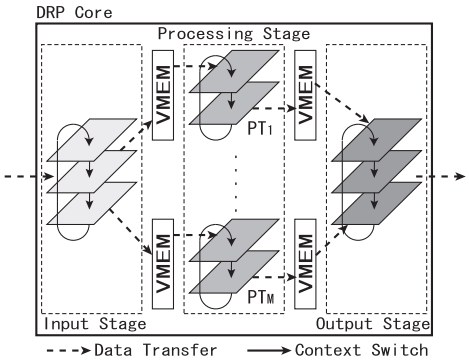

図 5　オーバラップ法の構成と動作
Fig. 5　Structure and behavior of overlap method.

図 5 に DRP コアにおけるオーバラップ法の構成と動作を示す．外部モジュールから転送されてきた入力データは，IS を経由して PS の PT_1 から PT_M へと順に転送される．IS は，1 タスク分の入力データを PT_i（$1 \leq i < M$）に転送した後，PT_{i+1} へ入力データを転送する．PT はデータ駆動であるので，入力データを一つでも受け取り次第，PT はコンテクストスイッチを行いながらタスク処理を開始する．各 PT の出力データは，OS を経由して外部モジュールへ転送される．IS と PS 間，PS と OS 間のデータ転送は，FIFO となる VMEM を経由する．

PS の全 PT は同じ処理を行うので，PS の全 PT のコンテクスト数は同じである．しかし，IS と OS はデータ転送を行うので，各 Stage の回路は異なる．したがって，図 5 のように PS の PT，IS と OS のコンテクスト数は異なる．IS はデマルチプレクサであり，OS はマルチプレクサであるので，IS と OS の回路は異なる．したがって，IS と OS のコンテクスト数も異なる．

4.2 オーバラップ法の実行モデル

オーバラップ法は，入力データと出力データの片方，若しくは，両方を連続的に転送する．図 6 (a) に無限の Tile 数をもつ DRP コアで，n タスクからなるデータ並列タスクを処理する場合の実行モデルを示す．1 タスク分の入力データ転送クロック数 C_{in} と出力データ転送クロック数 C_{out} は同じ（$C_{in} = C_{out}$）とする．PS の PT はデータ駆動の処理をするので，IS から一つでも入力データを受け取り次第，タスク処理を開始する．そのため，IS のデータ転送と PT のタスク処理が同時に実行する．また，PT は一つでも出力データを生成したら即座に OS へ転送するので，PT のタスク処理と OS の出力データ転送が同時に行われる．

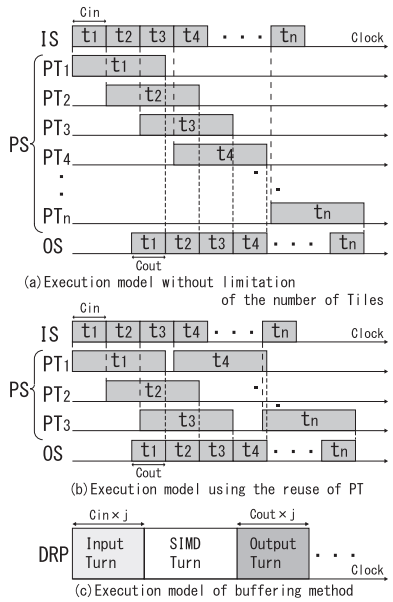

図 6　オーバラップ法とバッファリング法の実行モデル ($C_{in} = C_{out}$)
Fig. 6　Execution model of overlap method and buffering method. ($C_{in} = C_{out}$)

この場合，Tile 数は無限なので，PS にタスク数と同等の n 個の PT を構成できる．したがって，IS のデータ転送を各 PT へ連続的に行うことができるので，データ転送とデータ並列タスク処理をオーバラップさせることができる．また，各 PT のタスク処理同士もオーバラップできる．出力データが各 PT から転送されてくる間隔は，C_{in} の各 PT へ入力データを転送する間隔と等しいので，OS のデータ転送も連続的に行うことが可能である．

しかし，t_3 の入力データ転送中に PT_1 で処理されていた t_1 が完了するので，t_4 を PT_1 で処理できる．図 6 (b) に t_4 を PT_1 で処理する実行モデルを示す．このように，PT を再利用することによって，このデータ並列タスクは三つの PT で IS と OS のデータ転送を連続的に行い，データ転送とデータ並列タスク処理，PT のタスク処理同士をオーバラップさせることができる．PS の PT 数 M は，IS と OS のデータ転送と PT の再利用を考慮することで決定できる．

図 6 (c) に同じデータ並列タスクをバッファリング法で処理する実行モデルを示す．まず，j タスク分 ($1 \leq j \leq n$) の入力データをバッファメモリに保存した後，SIMD Turn で SIMD 処理を行う．そして，バッファメモリに保存されている j タスク分の出力データを外部モジュールへ転送する．全タスクを処理していない場合 ($j < n$)，Input Turn から再び行う．

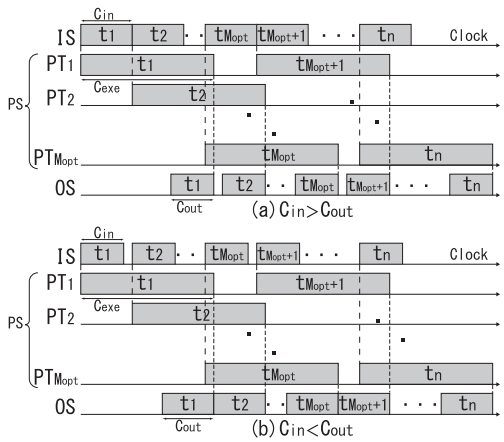

図 7　オーバラップ法の実行モデル
Fig. 7　Execution model of overlap method.

バッファリング法はデータ転送を行っている間，DRP 上にデータ転送を行う回路しか実現されていない．したがって，図 6 (c) から分かるように，データ転送とデータ並列タスク処理が別々になっている．一方，オーバラップ法は，図 5 のようにデータ転送を行う回路とデータ並列タスク処理を行う回路が別々配置されているので，図 6 (b) のようにデータ転送とデータ並列タスク処理をオーバラップさせることができる．

4.3　オーバラップ法の実行モデルの解析

データ転送を連続的に行うための PS の PT 数 M_{opt} は，データ転送クロック数と 1 タスク処理の実行クロック数 C_{exe} に依存する．$C_{in} = C_{out}$ では，図 6 (b) のように IS と OS のデータ転送を連続的に行うことができる．

一方，$C_{in} \neq C_{out}$ では，1 タスク分のデータ転送の間隔が異なるため，図 7 のように IS と OS の片方だけ連続的に転送する．$C_{in} > C_{out}$ の場合，あるタスクの出力データ転送完了時点で次のタスクの出力データがないため，図 7 (a) のように IS のデータ転送のみ連続的に行われる．$C_{in} < C_{out}$ の場合，図 7 (b) のように OS のデータ転送のみ連続的に行われる．これは，2 回目以降の PT_1 への入力データ転送開始時点で PT_1 は前のタスク処理を完了していないため，IS と PS 間の VMEM にその入力データを保存する必要がある．したがって，VMEM の容量を満たした時点で入力データ転送を停止する必要がある．PS と OS 間の VMEM への出力データの保存も考えられるが，結

果的に同様の対応をする必要がある．入力データ転送の間隔を C_{out} にすることで，VMEM へのデータ保存を回避できる．

PS の PT 数 M_{opt} は，C_{in}，C_{out} と C_{exe} で決まる．t_k を PT_1 で処理するためには，t_{k-1} の入力データ転送中に，PT_1 で処理されている t_1 が完了すればよい．したがって，PS に k-1 個の PT を実現することで，入力データと出力データの片方，若しくは，両方を連続的に転送できる．連続的にデータ転送を行うための PS の PT 数 M_{opt} は，式 (1) で求めることができる．

$$M_{opt} = \left\lceil \frac{C_{exe}}{\max(C_{in}, C_{out})} \right\rceil \tag{1}$$

また，PS に使用される PT 数が M_{opt} 以上の場合の実行モデルの理論的な全実行クロック数 C_{total} は，式 (2) で求めることができる．

$$C_{total} = (N-1) \cdot \max(C_{in}, C_{out}) + C_{exe} \tag{2}$$

N はデータ並列タスクのタスク数である．式 (2) の要素には PT 数 M がないため，全実行クロック数には PT 数が関係していない．したがって，M_{opt} 以上 PT 数を増やしても全実行クロック数を削減できない．M_{opt} を DRP コアに実現すれば十分であるといえる．

5. 性能評価

5.1 評価条件

DRP-1 に JPEG2000 の DC レベルシフトと RCT，JPEG エンコードの 1D-DCT を実装してオーバラップ法を評価した．DC レベルシフトと RCT は RGB 画像の 1 画素に対して，1D-DCT は YCC 色空間に変換された画像を 8×8 サイズのブロックに対して処理を行うデータ並列タスクである．DC レベルシフトと RCT は，それぞれ独立したデータ並列タスクであるが，本論文では一つのデータ並列タスクの処理として実装した．バッファリング法とオーバラップ法で実装した DC レベルシフトと RCT の処理を式 (3)，(4) に示す．同様に 1D-DCT 処理を式 (5)，(6) に示す．

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} R-128 \\ G-128 \\ B-128 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} Y' \\ Cb' \\ Cr' \end{bmatrix} = \begin{bmatrix} \left\lfloor \frac{R'+2G'+B'}{4} \right\rfloor \\ R'+G' \\ B'+G' \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \begin{bmatrix} X(0)+X(7) \\ X(1)+X(6) \\ X(2)+X(5) \\ X(3)+X(4) \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} = \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & b \end{bmatrix} \begin{bmatrix} X(0)-X(7) \\ X(1)-X(6) \\ X(2)-X(5) \\ X(3)-X(4) \end{bmatrix} \tag{6}$$

表 1 に各データ並列タスクの入出力データを示す．DC レベルシフトと RCT の入力と出力のデータビットは同じなので，C_{in} と C_{out} は等しい．一方，1D-DCT は出力のデータビットが入力のデータビットの 2 倍なので，C_{out} は C_{in} の 2 倍となる．

両手法ともにプログラム内にコンテクストの分割を明示する手動スケジューリングモードでコンパイルした．本来，DRP-1 の入出力ポートは，各 128 bit であるが，テストボードの影響により各 64 bit である．

オーバラップ法の PT は，1 タスク分を処理する PS 内の回路である．PT は，Tile 単位で構成されており，Tile 内の PE を利用して構成する．PE は，ALU などで構成されている DRP コアの基本要素である．バッファリング法の Processing Unit（PU）は SIMD 型回路であり，Tile の境目に関係なく，DRP 全体の PE を可能な限り利用して構成する．オーバラップ法の PT とバッファリング法の PU の処理単位は同じである．

5.2 バッファリング法（従来手法）の実装

バッファリング法は，DRP コア全体をコンテクストスイッチしながら処理するシングルプロセスで実装

表 1 入出力データの詳細
Table 1 Detail of input/output data.

データ並列タスク	入力		出力	
	入力データ	データビット数	出力データ	データビット数
DC Level Shift and RCT	256×256 の 24 bitRGB 画像	符号なし 8 bit	256×256 の YCC 画像	符号付き 8 bit
1D-DCT	256×256 の Y 成分画像	符号付き 8 bit	256×256 の DCT 係数	符号付き 16 bit

表 2　バッファリング法の実行結果
Table 2　Execution result of buffering method.

データ並列タスク	入力データ転送（クロック数）	SIMD（クロック数）	出力データ転送（クロック数）	OH_{all}（クロック数）	全実行クロック数	SIMD の並列度
DC Level Shift and RCT	769×32（42.6%）	262×32（14.5%）	772×32（42.7%）	98（0.2%）	57794	5
1D-DCT	32×256（20.7%）	52×256（33.8%）	67×256（43.5%）	770（1.9%）	39426	4

した．

(a)　DC レベルシフトと RCT

全 VMEM と VMEM の 2-port の読出しを利用して 5 並列（5PU）の SIMD 型回路として “DC レベルシフトと RCT” を実装した．VMEM の 2-port の読出しの影響により DRP の全 PE を使用できなかったため，使用した PE 数は 255 個となった．1PU 当り 8 画素の処理を行うので，SIMD 型回路全体では 40 画素の処理を同時に行う．

“DC レベルシフトと RCT” の処理は，最初にバッファサイズの制約により 2048 画素分のデータを 769 クロックかけて VMEM へ転送する．次に，各 PU が VMEM から 51 回データを読み出して “DC レベルシフトと RCT” の処理を行う．52 回目は，1PU のみが VMEM からデータを読み出して処理する．1 回当りの “DC レベルシフトと RCT” の処理には，5 クロックかかる．また，SIMD 型回路による処理の開始時に VMEM から入力データを読み出すための設定に 1 クロック，終了時に VMEM 内に入力データが残されているかを確認するために 1 クロックかかる．したがって，2048 画素を 5 クロック×52 回＋1 クロック＋1 クロック ＝ 262 クロック かけて処理する．出力データは，再度 VMEM に保存される．そして，2048 画素分の出力データを 772 クロックかけて VMEM から転送する．この実行フローを 32 回繰り返す．

入力と出力のデータ量とデータビット数は同じなので，データ転送クロック数は理論的に同じになる．しかし，出力データは複数の VMEM に保存されているため，出力データ転送時に VMEM の切換を行う必要があり，この切換時に 1 クロックの OH_{trans} が発生する．VMEM の切換を 3 回行うので，出力データ転送クロック数は 772 クロックとなる．

(b)　1D-DCT

DRP コアにある基本要素 PE すべてと VMEM の 2-port の読出しを利用して 4 並列（4PU）の SIMD 型回路として “1D-DCT” を実装した．1PU 当り 8×8 サイズの Y 成分画像である 1 ブロックを処理するので，SIMD 型回路全体で 4 ブロックの処理を同時に行う．“1D-DCT” の処理は，コンテクスト数の制約により 4 ブロック単位でデータ転送と処理を行う．まず，4 ブロック分のデータを 32 クロックかけて VMEM へ転送し，4PU で 50 クロックかけて “1D-DCT” の処理を 1 回行う．また，“DC レベルシフトと RCT” と同様に SIMD 型回路による処理の開始時の VMEM の設定と終了時の入力データの確認に各 1 クロックかかるので，合計で 52 クロックかかる．VMEM に保存されている 4 ブロック分の出力データを 67 クロックかけて転送する．この実行フローを 256 回繰り返す．出力データ転送では，“DC レベルシフトと RCT” と同様の理由により OH_{trans} が発生する．VMEM の切換が 3 回行われるので，出力データ転送クロック数は 67 クロックとなる．

表 2 にバッファリング法の実行結果を示す．OH_{all} は，DRP-1 の制約により実行時に発生したオーバヘッドである．入出力データ転送クロック数の割合は “DC レベルシフトと RCT” で 42.6% + 42.7% = 85.3%，“1D-DCT” で 20.7%＋43.5% = 64.2% なので，入出力データ転送がボトルネックであることが分かる．特に，“DC レベルシフトと RCT” は，簡単な処理であるため，SIMD 回路全体で 3 クロックで 40 画素の処理ができる．このため，データ転送クロック数の割合が非常に高く，データ転送がボトルネックであることが分かる．

5.3　オーバラップ法（提案手法）の実装

オーバラップ法は PCS を行うので，マルチプロセスで実装した．オーバラップ法による各データ並列タスクの実装は，以下のフロー行った．

（1）　1PT に転送するデータ量の決定

（2）　PT の設計

（3）　1PT 当りの実行クロックから PS の PT 数 M_{opt} を算出

（4）　IS と OS の設計

オーバラップ法の PS の 1PT 当りのデータ並列タスクの処理単位は，バッファリング法の SIMD 型回路の 1PU と同じ処理単位とした．したがって，“DC レベルシフトと RCT” は 1PT 当り 8 画素の処理を行い

表 3　オーバラップ法の実行結果と理論的な全実行クロック数
Table 3　Execution results and theoretical total clock cycles of overlap method.

データ並列タスク	IS の入力データ転送（C_{in}）	PS の処理（C_{exe}）	OS の出力データ転送（C_{out}）	M_{opt}（PT 数）	理論的な全実行クロック数（C_{total}）	OH_{all}	実装による全実行クロック数
DC Level Shift and RCT	3×8192	4×8192	3×8192	2	24577	133	24710
1D-DCT	8×1024	48×1024	16×1024	3	16416	1032	17448

8192 タスクを処理する．“1D-DCT” では 1PT 当り 1 ブロックの処理をし，1024 タスクを処理する．

表 3 に各データ並列タスクの実行結果と理論的な全実行クロック数を示す．“1D-DCT” は，入力と出力のデータ量は同等であるが，出力データのビット数が入力データのビット数の 2 倍であるので，C_{out} が C_{in} の 2 倍となる．各データ並列タスクの 1 タスク処理には，“DC レベルシフトと RCT” が 4 クロック，“1D-DCT” が 48 クロックかかる．“DC レベルシフトと RCT” のバッファリング法では，1 タスクと同等の 8 画素に対して 5 クロックで処理をしているので，1 タスク処理の処理を 20%削減している．また，“1D-DCT” ではバッファリング法が 50 クロックかかるので，4%削減している．各データ並列タスクの PS は，式 (1) で求められた M_{opt} の PT 数で実装した．“DC レベルシフトと RCT” の M_{opt} は 2 なので，PS は二つの PT で，“1D-DCT” の M_{opt} は 3 なので，PS は三つの PT で実装した．理論的な全実行クロック数 C_{total} は，C_{in}，C_{exe}，C_{out} とタスク数を式 (2) に適応して求めた．

5.4　オーバラップ法の評価

5.4.1　実行クロック数の比較

図 8 にオーバラップ法，バッファリング法とオーバラップ法の理論値の全実行クロック数の比較を示す．オーバラップ法は，バッファリング法と比較して全実行クロック数を “DC レベルシフトと RCT” では約 57%，“1D-DCT” では約 55%削減できた．この結果より，オーバラップ法はデータ並列タスクの処理の全実行クロック数を大幅に削減できることが分かる．

オーバラップ法の実装値と理論値の比較では，実装上のオーバヘッドにより若干実装値が上回っている．“DC レベルシフトと RCT” では，主にデータ入力開始のオーバヘッドにより差が発生した．DRP-1 の制約上，全入力データを連続的に転送することができず，途中で入力データ転送を停止する必要があるため，オーバヘッドが発生した．“1D-DCT” では，ある PT の出力データ転送から次の PT の出力データ転送へ変更するときに OS において 1 クロックのオーバヘッド

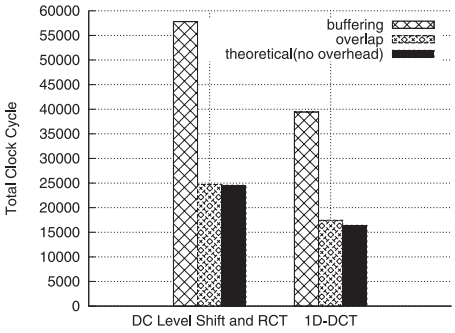

図 8　全実行クロック数の比較
Fig. 8　Comparison of the total clock cycles.

が発生するため，8 × 8 サイズの DCT 係数を出力するたびに OS のデータ転送は 1 クロックの停止をする．これにより，出力データを連続的に転送することができず，理論値との差が発生した．

5.4.2　実行時間の比較

図 9 にオーバラップ法とバッファリング法の全実行時間の比較を示す．Place & Route 後の動作周波数は，“DC レベルシフトと RCT” のバッファリング法が 21 MHz であり，オーバラップ法が 36 MHz であった．“1D-DCT” では両手法とも 16 MHz であった．全実行クロック数の大幅な削減により，オーバラップ法の全実行時間は “DC レベルシフトと RCT” が 4 倍，“1D-DCT” が 2.2 倍バッファリング法より高速に処理できた．両手法で動作周波数が同じである “1D-DCT” では，全実行クロック数の大幅な削減による効果が分かる．

5.4.3　使用リソースの比較

表 4 にオーバラップ法とバッファリング法の使用リソースの比較を示す．PE 数は，各コンテクスト番号の使用 PE 数とコンテクスト間の最大使用 PE 数である．理論上の最小 Tile 数は，実装後の最大使用 PE 数から算出された Tile 数である．オーバラップ法の PS の PT 数は M_{opt} であり，“DC レベルシフトと RCT” が 2，“1D-DCT” が 3 である．各データ並列タスクの両手法において Tile は，8Tile（DRP コア上の全 Tile）使用し，HMEM は使用しなかった．

バッファリング法の両データ並列タスクの実装の結

表 4　使用リソースの比較
Table 4　Comparison of the used resources.

	Context No.	DC Level Shift and RCT				1D-DCT			
			オーバラップ法				オーバラップ法		
		バッファリング法	IS	PT（1PT 分）	OS	バッファリング法	IS	PT（1PT 分）	OS
PE 数	1	34	15	7	11	5	20	8	12
	2	1	10	5	6	1	28	5	15
	3	56	21	27	22	6	63	27	56
	4	1	-	40	-	26	-	60	-
	5	255	-	40	-	81	-	60	-
	6	4	-	8	-	237	-	119	-
	7	142	-	-	-	345	-	124	-
	8	196	-	-	-	512	-	124	-
	9	15	-	-	-	385	-	-	-
	10	19	-	-	-	469	-	-	-
	11	19	-	-	-	5	-	-	-
	12	19	-	-	-	10	-	-	-
	13	19	-	-	-	10	-	-	-
	14	33	-	-	-	10	-	-	-
	15	-	-	-	-	8	-	-	-
	MAX	255	21	40	22	512	63	124	56
実装時の Tile 数（IS+PT×$Mopt$+OS）		8	8（2+2×2+2）			8	8（1+2×3+1）		
理論上の最小 Tile 数（IS+PT×$Mopt$+OS）		8	4（1+1×2+1）			8	8（1+2×3+1）		
VMEM 数		80	32			48	48		
HMEM 数		0	0			0	0		

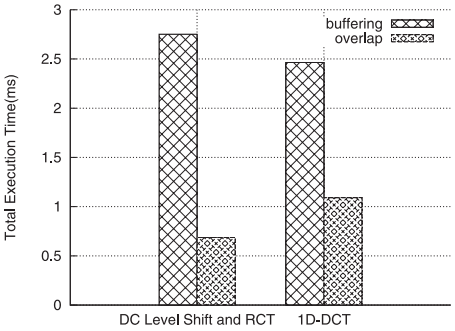

図 9　全実行時間の比較
Fig. 9　Comparison of the total execution time.

果，最大使用 PE 数は，“DC レベルシフトと RCT” が 255 個，“1D-DCT” が 512 個となった．1Tile 当り 64 個の PE が配置されているので，“DC レベルシフトと RCT” では 4Tile で，“1D-DCT” では 8Tile 使用することで SIMD 型回路を実現できる．しかし，“DC レベルシフトと RCT” では，VMEM を 80 個（DRP コア上の全 VMEM）使用しているので，VMEM と SIMD 型回路間の配線のために 8Tile 使用する必要がある．したがって，バッファリング法の両データ並列タスクの理論上の最小 Tile 数は，8Tile である．

オーバラップ法は，データ転送する IS と OS，データ並列タスク処理する PT 単位でパーシャルコンテクストスイッチを行うので，各ステージごとに Tile 数が異なる．“DC レベルシフトと RCT” は IS を 2Tile，PT を 2Tile（PS では 2Tile×2PT=4Tile），OS を 2Tile で実装した．実装の結果，最大使用 PE 数は IS が 21 個，PT が 40 個，OS が 22 個となり，すべて 1Tile で実現できるため，理論上の最小 Tile 数は 4Tile となる．

オーバラップ法の “1D-DCT” は，IS を 1Tile，PT を 2Tile（PS では 2Tile×3PT=6Tile），OS を 1Tile として実装した．その結果，最大使用 PE 数は IS が 63 個，PT が 124 個，OS が 56 個となったので，IS と OS は 1Tile で，PT は 2Tile で実現できるため，理論上の最小 Tile 数は 8Tile となる．

PE 数から算出された理論上の最小 Tile 数と VMEM の使用量をもとにバッファリング法と比較して，オーバラップ法は，“DC レベルシフトと RCT” では Tile 数を 50%削減し，VMEM 数も 60%削減した．“1D-DCT” では，Tile 数と VMEM 数ともに同等であった．

6. む す び

本論文は，DRP におけるデータ並列タスクの処理に発生するデータ転送のボトルネックを問題点とし

て，データ転送を隠ぺいするためにデータ転送とデータ並列タスク処理をオーバラップさせる手法を提案した．本手法は，データ転送のコンテクストとデータ並列タスク処理のコンテクストを別々に配置して，それぞれが独自にコンテクストスイッチをすることでオーバラップを実現させる．また，タスク処理同士もオーバラップするので，データ並列タスクの特性に合わせた柔軟な構成が可能となる．

DRP 内の大規模メモリにデータをバッファリングして SIMD 処理する手法を従来手法として，JPEG2000 の DC レベルシフトと RCT，JPEG エンコーダの 1D-DCT を DRP-1 に実装して評価した．その結果，本手法は従来手法と比較して最大で全実行クロック数を 57%削減し，動作周波数も向上したことで全実行時間を 1/4 以下に削減した．また，高速化に伴うリソースの増加はなく，DC レベルシフトと RCT では，Tile 数を約 50%，VMEM 数を 60%削減できた．これらの評価から本手法は，非常に有効であることが分かった．

本手法では，データ転送時間に対して一つのタスク処理の時間が非常に長い場合，PT 数の急激な増加，若しくは，性能向上率の低下が発生してしまう．このことから，今後は，PT 数の増加と性能向上率の低下の防止のために，データ並列タスク処理のパイプライン化と本手法の併用について研究を行う予定である．

謝辞　本研究を行うにあたって DRP-1 と開発環境を提供して頂き，多くのアドバイスをして頂いた NEC エレクトロニクス社の皆様に深く感謝致します．

文　　献

[1] M. Motomura, “Dynamically reconfigurable processor architecture,” Microprocessor Forum, Oct. 2002.

[2] T. Sato, H. Watanabe, and K. Shiba, “Implementation of dynamically reconfigurable processor DAPDNA-2,” Proc. IEEE 2005 Int. Sym. VLSI Design, Automation and Test, 2005, pp.323–324, April 2005.

[3] 津田野賢伸，高田雅士，秋田庸平，田中博志，佐藤真琴，伊藤雅樹，“ディジタルメディア向け再構成型プロセッサ FE-GA の概要,” 信学技報，RECONF2005-65, Dec. 2005.

[4] V. Baumgarte, G. Ehlers, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, “PACT XPP a self-reconfigurable data processing architecture,” J. Supercomputing, vol.26, pp.167–184, 2003.

[5] S. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. Taylor, and R. Laufer, “PipeRench : A coprocessor for streaming multimedia acceleration,” Proc. Int. Sym. on Computer Architecture, pp.28–39, May 1999.

[6] 天野英晴，“ダイナミックリコンフィギャラブルプロセッサの研究開発動向,” 信学技報，ICD2003-130, Oct. 2003.

[7] S. Hauck and A. Dehon, Reconfigurable Computing: The Theory And Practice of FPGA-Based Computation, Morgan Kaufmann Publishers, 2008.

[8] E. El-Araby, M. Taher, K. Gaj, T. El-Ghazawi, D. Caliga, and N. Alexandridis, “System-level parallelism and throughput optimization in designing reconfigurable computing applications,” Proc. Int. Sym. Parallel and Distributed Processing, 2004, pp.136, April 2004.

[9] 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-,” Proc. Int. Conf. Field Programmable Logic and Applications, 2005, pp.347–352, Aug. 2005.

（平成 21 年 1 月 21 日受付，7 月 13 日再受付）

荒木　光一　（学生員）

2006 鳥取大・教育地域科・地域科学課程卒業．2008 北陸先端科学技術大学院大学情報科学研究科博士前期課程了．現在，同大同研究科博士後期課程在籍．リコンフィギャラブルシステムに関する研究に従事．

佐藤　幸紀　（正員）

平 13 東北大・工・機械知能卒．平 15 同大大学院情報科学研究科前期課程了．平 18 同大学院情報科学研究科後期課程了，博士（情報科学）．同年ファインアーク（株）入社．入社後も東北大学大学院情報科学研究科大学院研究生，民間等共同研究員研究として組込みプロセッサシステムの低電力化に関する研究開発に従事．平 19 年 4 月より北陸先端科学技術大学院大学助教（情報科学センター）．計算機アーキテクチャ，高性能計算システムやその応用に関する研究に従事．

井口　　寧　（正員）

1991 東北大・工・機械卒．1994〜1997 日本学術振興会特別研究員．1997 北陸先端科学技術大学院大学情報科学研究科博士後期課程了．現在，同大情報科学センター助教授．また，2002 から 2006 まで科学技術振興事業団さきがけ研究 21（機能と構成）に参加し研究に従事．2008〜南フロリダ大学上級客員研究員．この間並列システム，ウェーハスタック集積システムに関する研究を行う．IEEE，情報処理学会各会員．