システムＬＳＩ設計技術　108－23
(2003. 1. 29)

DRPでのウェーブレットフィルタの実装

出口　勝昭†　　山田　　裕†　　天野　英晴†

† 慶應義塾大学大学院理工学研究科
〒223–8522 横浜市港北区日吉 3–14–1
E-mail: †{deguchi,yamada,hunga}@am.ics.keio.ac.jp

あらまし　NEC が開発した Dynamically Reconfigurable Processor(DRP) は、Processing Element(PE) の配列から成る Tile の集合によりデータパスを形成する。DRP は 16 コンテキスト分をチップ内に保持し、これを Tile 単位で自由に切り替えることが可能である。この DRP のマルチコンテキスト機能を利用して、複数の基底数を扱うことが可能な Daubechies のウェーブレットフィルタを実装した設計事例について報告する。DRP 上に実装した基底数可変フィルタは、基底数固定フィルタの性能に比べると若干性能が低いが、最新の PC 上のソフトウェアによる実行性能に匹敵する性能を実現可能であることがわかった。

キーワード　マルチコンテキストデバイス、DRP、離散ウェーブレット変換、Daubechies フィルタ

Implementation of Discrete Wavelet Transform on DRP

Katsuaki DEGUCHI†, Yutaka YAMADA†, and Hideharu AMANO†

† Graduate School of Science and Technology, Keio University
Hiyoshi 3–14–1, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †{deguchi,yamada,hunga}@am.ics.keio.ac.jp

Abstract　NEC's Dynamically Reconfigurable Processor (DRP) is a multicontext reconfigurable device consisting of eight individually reconfigurable units called "Tile." Data path configuration mapped to each tile can be selected from on-chip repository of sixteen circuit configurations, or contexts. The context switching can be done with a clock cycle. Using this mechanism, Daubechies wavelet filter whose length of filter can be changed is designed and implemented. Evaluation results show that the performance degradation caused by the variable length mechanism is not so large compared with a fixed length filter, and its absolute performance is comparable to those of software execution on recent high performance PCs.

Key words　multicontext device, DRP, Discrete Wavelet Transfer, Daubechies

1. はじめに

2002 年になって、高速な動的再構成が可能なプロセッサアレイ構造を持つリコンフィギャブルデバイスの本格的な商用化が進んでいる。中でも 2002 年 10 月 NEC が発表した Dynamically Reconfiguarable Processor (DRP) [1] は、1 クロックで切り替え可能なコンテキストをチップ内に 16 セット保持することができるマルチコンテキストデバイスであり、広い分野での応用が可能であるが、とりわけストリームデータを扱う応用分野が有望とされている。そこで我々は、画像処理の圧縮や解析の有力な手段として注目されている離散ウェーブレット変換に着目し、マルチコンテキスト機能を生かして基底数を必要に応じて切り替えることのできる Daubechies ウェーブレットフィルタを提案する。そして、このフィルタを、DRP 上に実装しハードウェア要求量と動作周波数を評価した結果を報告する。

まず、2 節で最近のマルチコンテキストリコンフィギャラブルデバイスの検討を行い、DRP の詳細について述べる。次に 3 節でウェーブレット変換について紹介し、4 節に基底数可変の Daubechies フィルタを提案する。5 節で DRP 上での実装について述べ、5 節で処理結果と必要クロック数について述べる。

表 1　マルチコンテキストデバイスと他のデバイスの比較

	処理性能	柔軟性	電力効率	面積利用効率
Multi-Context	高い	非常に高い	高い	高い
FPGA	高い	高い	低い	高い
ASIC	非常に高い	低い	非常に高い	非常に高い
General Processor	低い	非常に高い	低い	低い

2. マルチコンテキストリコンフィギャラブルデバイス

2.1 最近のマルチコンテキストデバイスの特徴

マルチコンテキストリコンフィギャラブルデバイス [4] [5] はコンテキストを切り替えることによってハードウェアの構成を動的に変更できる。以降、このようなマルチコンテキストリコンフィギャラブルデバイスのことをマルチコンテキストデバイスと呼ぶことにする。

最近発表された DRP や ACM [10]、DAP/DNA [11] などのマルチコンテキストデバイスは粗粒度の構造をもち、マトリックス状に配置された要素間のデータパスを切り替えることにより、

柔軟に構造を変化させることができる。

表1はこのような粗粒度のマルチコンテキストデバイスの特徴と他のデバイスの特徴の比較を示したものである。それぞれの項目を以下に検討する。

a) 処理性能:

ハードウェアで処理を行うため、高速に処理を行うことが可能となる。ロジックにLUTは用いずに、ALU等の演算器を用いているものが多く、FPGAよりも高速に処理を行うことができる。多くの要素をマトリックス状に配置しているアーキテクチャでは、特に並列処理に対して高い性能を示している。

また、データの転送に関してはチップ内のメモリを異なる回路間で共有メモリとして見ることができる。I/O部分がボトルネックとならないため、高速にデータのやりとりを行うことが可能である。

b) 柔軟性:

動的に再構成が可能であり、静的にしか再構成のできないFPGAや、ASICよりも高い柔軟性を持っている。FPGAに比べハードウェア構成を短い時間で変更することが可能であり、処理に応じてハードウェアを変更することが可能となる。

c) 電力効率:

回路構成情報を格納しておくメモリの消費電力が必要となるが、ASICと同様に低周波数で高速な処理を実行できるため消費電力は低い。機能を排他的に使う場合は利用していない回路のリーク電流がかからないため、複数のハードウェアを用いる場合に比べ、消費電力を抑えることが可能だと考えられる。

d) 面積効率:

一つのチップに複数の回路を実現できるため、FPGAやASICに比べ、面積あたり様々の処理を行うことが可能である。また、利用効率の点でもFPGAと同様の効率を得ることができる。

複数の処理を単一のチップで排他的に処理を行うシステムにおいては、さらに面積を有効に利用できるシステムが設計できる。実際には一つのチップが実現できる回路の規模は搭載されたメモリに依存するが、動作中に外部のメモリから構成情報を内部のメモリにロードできるデバイスもある。構成要素にLUTを利用していないデバイスでは、構成情報のサイズを小さくでき、多くの構成情報を内部のメモリに格納できる。

2.2 Dynamically Reconfigurable Processor (DRP)

DRP[1]は、粗粒度のマルチコンテキストデバイスであり、全体の構造は図1に示す構造となっている。

DRPはCoreの周辺部に32ビットの乗算器を8セット、メモリモジュール、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。Core部分にはTileと呼ばれるリコンフィギャラブルユニットが4×2個配置されており、中央にチップ全体の状態遷移を制御するためのシーケンサであるCenteral State Transition Controller (CSTC)が配置されている。

各Tileは図2に示す通り、8×8のProcessing Element (PE)アレイ、状態制御を行なうシーケンサであるState Transition Controller (STC)から構成される。また、8bit×256エントリのメモリ8セットとこれらを制御するメモリコントローラ2セットを両側に持ち、8bit×8192エントリのメモリ4セットとメモリコントローラをTileの上部もしくは下部に持つ。DRP全体では8bit×256エントリのメモリを合計80セット、8bit×8192エントリのメモリを合計32セット持つ。

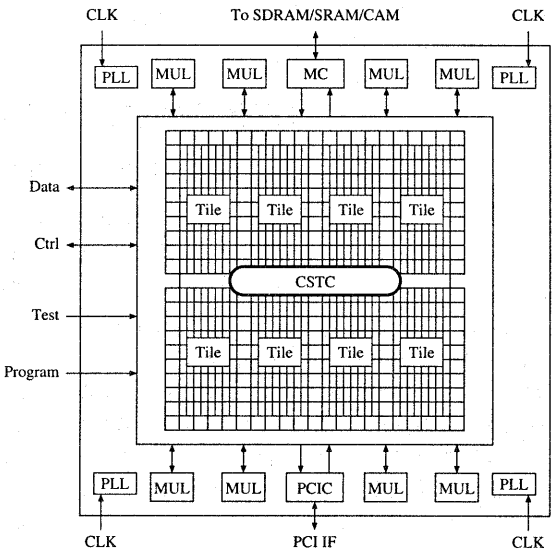

図1 DRPの構造

各PEは図3に示す通り、8bitのALU, シフトやデータ制御、簡単な論理演算を行なうData Management Unit (DMU)、8bitのFlip Flop、レジスタファイルから構成される。また、コンフィギュレーション時には命令データが命令メモリに書き込まれ、実行中にSTCが発行した命令ポインタを命令メモリが受け、利用する命令データをロードすることでハードウェア構成を変更する。

DRPは、NECが1998年に開発したDRL[2]同様、1クロックでコンテキスト切り替えが可能なマルチコンテキストデバイスで、Tile単位の部分再構成が可能である。一方で、DRLがLook Up Table (LUT)を構成要素とした細粒度のリコンフィギャラブルデバイスであったのに比べ、DRPは8bitのPEを構成要素としたリコンフィギャブルなプロセッサである点が全く異なっている。その特徴をまとめると以下のようになる。

a) マルチコンテキスト機能:

DRPは内部のメモリに最大16コンテキスト分の情報を蓄え

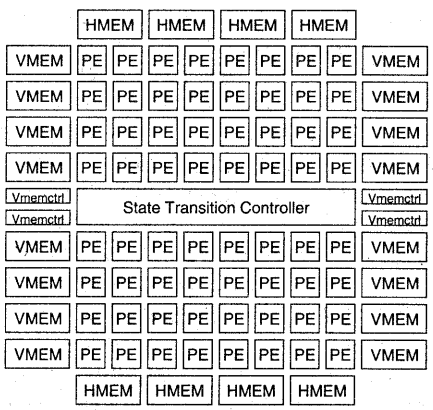

図2 Tileの構造

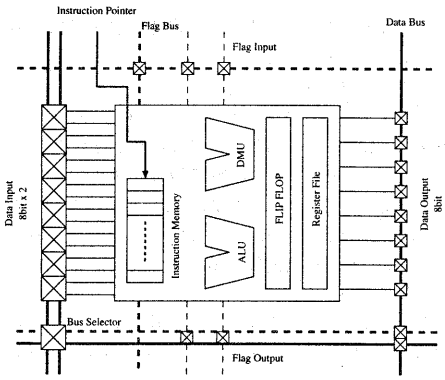

図3　PE の構造

ることができ、最大で 5 コンテキストの中から次のコンテキストを選択し、動的に再構成をすることができる。次のコンテキストの選択は STC に信号を送ることで行なう。さらに、動作中に、切り替えの対象外のコンテキストに対してコンフィギュレーション ロードが可能である。この特徴により、仮想ハードウェアや動的適応ハードウェアの効率的な実現が可能となる。

b）プロセッサアレイ構成:

DRP は FPGA 等の LUT によりロジックを実現する細粒度構成とは異なり、データパスを再構成するデバイスである。このため、8bit を基本単位とした効率的なデータフローの制御および多桁演算の高速実装が可能である。LUT を書き換えるのに比べて高速な再構成が可能であり、全く無駄なクロックなしに、切り替えと同時に動作を行なうことができる。

また、PE がチップ全体に配置されているため、並列処理を効率的に行うことが可能である。

c）パーシャルリコンフィギャラビリティ:

DRP はコンテキスト切り替えは Tile 単位で行うことができ、コンフィギュレーション データのロードは PE 単位で行なうことができる。これにより、Tile ごとに異なるコンテキスト構成を取ることができる。

d）メモリ拡散配置型のアーキテクチャ:

DRP は各 PE が Flip Flop およびレジスタファイルを独立に持ち、Tile の両側に 8bit×256 エントリの小規模なメモリを 8 セット、Tile の上側もしくは下側に 8bit×8192 エントリのメモリを 4 セットずつ持つ。これらの記憶要素の分散により、パイプライン構成、データ駆動型制御、フィードバック構成が簡単かつ高速に実現することができる。

3. ウェーブレット変換

ウェーブレット変換は、画像信号の圧縮や解析の有力な手段として注目されている。従来のフーリエ変換が、三角関数を基底とした直交変換であるのに対し、ウェーブレット変換では、局所化された関数から作られる相似関数系を基底とする。これにより、時間-周波数の同時分解が可能になる。

ウェーブレット変換には、連続ウェーブレット変換と離散ウェーブレット変換がある。本稿では、データ圧縮や解析等に用いられる離散ウェーブレット変換を扱う。

3.1 DWT とサブバンド符号化

DWT は一種のサブバンド符号化であり、低周波成分について分割をツリー状にくりかえす分割により表すことができる。このツリーの深さをオクターブと呼ぶ。図 4 に 3 オクターブ分割の例を示す。これらは、ローパスフィルタ (LPF)、ハイパスフィルタ (HPF)、ダウンサンプラ、によって構成される。

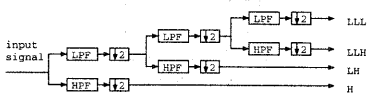

図4　1 次元ウェーブレット変換の構成 (3 段階)

3.2 DWT の構成方法

DWT は構成方法として基底数が 2、3 のときの変換式をそれぞれ式 1、式 2 に示す。H(z) が LPF をあらわし、G(z) が HPF をあらわす。$h_N[i]$、$g_N[i]$ は、基底数が N のときのスケーリング係数列をあらわしている。これらの係数列は基底数によって独立であり、基底数が N のとき、数列の長さは 2N となる。この係数列と入力信号の内積によって DWT を行うことができる。

$$\begin{aligned} H(z) &= h_{N=2}[0] + h_{N=2}[1]z^{-1} + h_{N=2}[2]z^{-2} + h_{N=2}[3]z^{-3} \\ G(z) &= g_{N=2}[0] + g_{N=2}[1]z^{-1} + g_{N=2}[2]z^{-2} + g_{N=2}[3]z^{-3} \end{aligned} \quad (1)$$

$$\begin{aligned} H(z) &= h_{N=3}[0] + h_{N=3}[1]z^{-1} + h_{N=3}[2]z^{-2} + \\ &\quad h_{N=3}[3]z^{-3} + h_{N=3}[4]z^{-4} + h_{N=3}[5]z^{-5} \\ G(z) &= g_{N=3}[0] + g_{N=3}[1]z^{-1} + g_{N=3}[2]z^{-2} + \\ &\quad g_{N=3}[3]z^{-3} + g_{N=3}[4]z^{-4} + g_{N=3}[5]z^{-5} \end{aligned} \quad (2)$$

本稿では、DWT を構成するために、直交基底を作る連続かつサポート・コンパクトなウェーブレットである Daubechies のウェーブレットを用いる [15]。

3.3 Systolic 型 DWT

最初に過去に提案された切り替えることのできる効率的なアーキテクチャについて述べる。初期の DWT のアーキテクチャとして提案された [12] [13] は特定の基底数とオクターブ数に特化しており、オクターブ数や基底数の拡張性に欠けていた。しかし、1995 年に Vishman らが提案した systolic array を用いた方法 [14] は、2 次元 routing network を使い、オクターブ数やフィルタ基底数の拡張が容易である。

この systolic array アーキテクチャは図 5 に示すようにハイパスフィルタとローパスフィルタの二つのフィルタユニットから構成される。両方とも基底数個の MAC ユニットをもつ linear systolic array である。ローパスフィルタの結果は systolic routing network で実装されているメモリに格納される。

systolic array アーキテクチャで行なわれる処理を以下に示す。奇数クロックでは、外部からの入力を直接受け取り 1 オクターブ目の計算を行なう。ハイパスからの出力は出力ポートに送られ、ローパスからの出力は routing network におくられる。2 オクターブ目以降の処理は偶数クロックで行われる。この場合、入力は routing network から取ってくる。routing network は 3 つの異なるタイプのセルによって構成される。R_l はローパスフィルタの出力を受け取り、S に値を格納する。これらのセルは値を保持し、R_r がクロックイベントを発生させると上か横の近傍に値を渡す。このスケジューリングはフィルタとデータの適切な組合せになるように行なわれる [14]。

このアーキテクチャは以下のような利点をもつ。

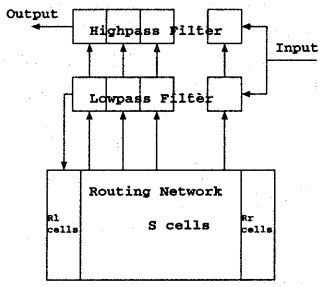

図5　systolic array Architecture

（1）　基底数N×最大オクターブ数Jのメモリセルですむ。
（2）　入力データの長さに無関係である。
（3）　routing network と linear array に行と列を追加するだけで、異なるフィルタの基底数とオクターブ分割数の異なるフィルタを作ることができる。

4. 基底数可変 DaubechiesDWT の提案

Daubechies N のフィルタは、基底数 N によって異なる周波数特性をもつフィルタを得られる。Daubechies のウェーブレットは、N が増加するとともに遮断特性のよいフィルタを得ることができる。[15] その反面、N が大きいほど、計算量が大きくなり消費電力も大きい。このため、基底数を動的に切り替えることができれば、状況の変化に応じて適切な構造を取ることが可能となる。

本稿では、設計し直すことなく基底数 N を変更することが可能な DWT フィルタを提案する。上述の systolic 型の DWT では、routing network と linear array に同じ構造をもつ行と列を追加することで、容易に基底数の異なる設計が行えたが、本稿での DWT では、設計をし直すことなく、外部からの値の設定のみで基底数の異なる DWT として動作可能なフィルタを提案する。このフィルタは DRP の持つマルチコンテキスト機能を利用すれば低コストで実現が可能である。ただし、systolic 型 DWT のようなオクターブ分割数への工夫は、現状では特に行なっていない。

4.1 実行アルゴリズム

3.2 節より、基底数 N の Daubechies フィルタでは、2N 回の乗算をおこなっていることがわかる。したがって、乗算二つを一まとまり (以下、1 セグメント) にし計算し必要に応じてセグメントの計算を繰り返すことで、基底数可変のフィルタが実現可能となる。例えば、N=2 では、2 セグメント、N=3 では 3 セグメントを計算すれば良いことになる。

今回提案するフィルタはこのセグメントの計算を繰り返すことで、スケーリング係数列を格納するメモリの許す限り基底数 N をいくらでも大きくすることが可能である。(ただし、今回は基底数を 1〜10 の範囲として設計した。)

このセグメントの計算の繰り返しを DRP のマルチコンテキストリコンフィギュラブル機能を使うことで効率的に実装することが可能である。

4.2 マルチコンテキスト化

基底数可変の Daubechies フィルタを実現するために、計算結果の書き込みや乗算回数を制御するコントロールステージ、一つのコンテキストで、二組の係数とデータの読みだしと乗算をおこなうセグメントステージを 2 つ分、計 3 つのコンテキストを使う。それぞれのコンテキストの動作の詳細は以下のようになっている。

（1）　コントロールステージ
このステージでは、セグメントステージ 1、セグメントステージ 2 の繰り返しによって得られた計算結果 *sum* の値をメモリに格納する。また、基底数 N 分の乗算の設定を行う。次のセグメントステージ 1 で乗算する Daubechies 係数 2 つ $k1, k2$ とデータ 2 つ $d1, d2$ を共有レジスタに格納する。セグメントステージ 1 のコンテキストに状態遷移する。

（2）　セグメントステージ 1
このステージでは、1 セグメントの計算をおこなう。共有レジスタ $k1, k2, d1, d2$ から $k1*d1+k2*d2$ を計算する。計算結果を保持するレジスタ *sum* に $sum = sum + k1*d1 + k2*d2$ を格納する。次のセグメントステージ 2 で使う係数を $k3, k4$ に、データを $d3, d4$ に格納する。基底数分の乗算が終了している場合は、コントロールコンテキストに遷移し、すべき計算が残っている場合には、セグメントステージ 2 のコンテキストに遷移する。

（3）　セグメントステージ 2
このステージでは、セグメントステージ 1 と同様に 1 セグメントの計算をおこなう。共有レジスタ $k3, k4, d3, d4$ から $k3*d3+k4*d4$ を計算し、*sum* に値を追加する。次のセグメントステージ 1 で使う係数を $k1, k2$ に、データを $d1, d2$ に納する。基底数分の乗算が終了している場合は、コントロールコンテキストに遷移し、すべき計算が残っている場合には、セグメントステージ 1 のコンテキストに遷移する。

状態遷移を表すと以下の図 6 ようになっている。

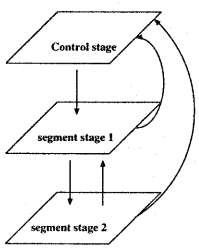

図6　基底数可変 Daubechies フィルタの状態遷移

また、この基底数可変のフィルタを DRP で実装することの特徴として以下のようなことが挙げられる。

（1）　ASIC でこのフィルタを設計すると、乗算器やメモリの周辺で複雑なセレクタを必要になるが、DRP では各コンテキストに必要な機能を分割しておくことで複雑なセレクタを用意する必要がない。

（2）　ASIC では、複雑なセレクタの制御をさけるために、乗算器の数を増やすとそれだけで大きな面積を消費することになるが、DRP では回路情報を必要に応じて切り替えることで多くの乗算器を面積を増やすこと無く使えることになる。

（3）　コンテキストの切り分けによる回路の単純化はクリティカルパスの減少につながる。

（4）　面積と消費電力をおさえることができる。

5. DRP 上での 2 次元 DWT フィルタの実装

本稿で提案する DWT フィルタをもつ画像データの圧縮に対応した 2 次元 Daubechies フィルタを設計し実装した。

5.1 仕　　様

- 最大で 80×80 の画像サイズに対応。
- 扱う画像データの形式は 8bit のグレースケール PGM とする。
- 一データあたりのデータは整数 16 ビット、少数部 16 ビット計 32 ビットとして扱う。

5.2 動　　作

本稿で設計した 1 オクターブ分の DWT の処理の流れは以下のようになる。

（1） フィルタの基底数についての情報及び画像サイズについての情報を読み込む。

（2） 画像データを DRP 内部の HMEM に読み込む (扱う画像形式は 1 画素 8bit のグレースケール PGM 形式の画像データとする。)

（3） 画像データに対して水平方向にウェーブレット変換を行う。

（4） 変換した画像データを転置する。

（5） そのデータに対して、水平方向にウェーブレット変換を行う。

（6） データを転置する。

以上の処理をおこなうことで、図 7 のように元の画像を LL、LH、HL、HH の成分に分解した画像を得ることが出来る。

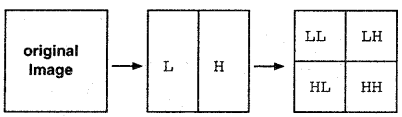

図 7　2 次元における 1 オクターブ分の DWT

5.3 フィルタ全体の状態遷移

上の動作を実現するために、以下の図 8 ような状態遷移を考え、各コンテキストに切り分けた。

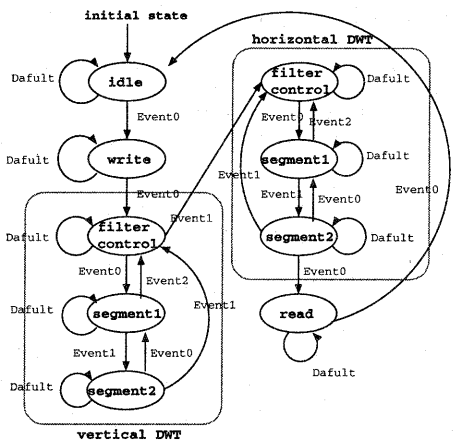

図 8　フィルタの状態遷移

5.4 各コンテキストの概要

- context0 idle state: カウンターやフラグを初期化するコンテキストで wstart 信号によって、次の状態に移行する。
- context1 read state: 画像データや、Daubechies の係数を内部の HMEM に読み込む。また、使うフィルタの設定値や画像サイズ分のデータを読み込む。
- context2～context4 上述の基底数可変フィルタの 3 つのコンテキストに対応する。水平方向の DWT をおこなう。
- context5～context7 上述の基底数可変フィルタの 3 つのコンテキストに対応する。垂直方向の DWT をおこなう。
- context8 変換後のデータを外部に出力する。すべてのデータを外部に出力すると、context0 にもどる。

6. 評　　価

実際の実装は DRP 上の Tile に対応したシミュレーション環境で実装した。図 9 の入力画像に対して、図 10 変換画像 (N=2) を得ることができ、設計したフィルタが正しく動作していることがわかる。

図 9　変換前の画像

図 10　変換後の画像

6.1 処理に必要な clock 数

最初に、変換にかかるクロック数について検討する。

画像サイズを n^2、基底数を N とすると、変換に 1 データあたり $(N+1)$ のコンテキストの切り替えが必要であり、また、各コンテキストでは、2 クロックずつ消費するため、画像全体の変換に $2(N+1)n^2$ クロックかかる。したがって、必要なクロック数 clock(N,n) は、式のようにもとめられる。式 4 の右辺の項は順に、水平方向の DWT、垂直方向の DWT にかかるクロック数を表している。

$$clock(N,n) = 2(N+1)n^2 + 2(N+1)n^2 \qquad (3)$$

となる。この式をもとに各基底数でのクロック数を表 2 にしめす。

表 2　クロック数

clock(N,n)	クロック数	clock(N,n)	クロック数
clock(1,80)	51200	clock(6,80)	179200
clock(2,80)	76800	clock(7,80)	204800
clock(3,80)	102400	clock(8,80)	230400
clock(4,80)	128000	clock(9,80)	256000
clock(5,80)	153600	clock(10,80)	281600

また、比較のために、基底数 N=2 に特化した DWT フィルタを設計し、実装した。この基底数では、必要な乗算器が LPF で 4 つ HPF で 4 つの合計 8 個であり、DRP の 1 つのコンテキストに搭載されている乗算器がちょうど 8 個であるので、N=2 にもっとも効率的な設計ができた。また、基底数可変のフィルタにくらべて、余計なコントロールが必要無いので、クロック数は削減され、以下のクロックの計算式となる。

$$clock_{static2}(n) = 2(1+1)n^2 + 2(1+1)n^2 \qquad (4)$$

この式 5 より、80×80 の画像の処理に必要なクロック数は、

51200 であり、前述の基底数可変の DWT フィルタの基底数 2 の場合の 76800 にくらべて早いことがわかる。基底数 2 のみの比較なので、

$$\frac{clock(n,2)}{clock_{static2}(n)} = 1.4 \qquad (5)$$

より、基底数 2 では専用に設計した場合の方が 1.4 倍ほど処理が速い。

6.2 ソフトウェアとの比較

現段階では、DRP 上の実装は充分な最適化が行なわれておらず、また、実機での動作周波数の測定は行なっていない。このため、今回は本稿での DWT フィルタが PC 上のソフトウェアでの処理速度と同等の処理速度を得るためにには DRP でどれだけの周波数を最低限必要とするかを調べることにする。

処理速度を表すために 80 × 80 サイズの画像の FPS を用いる。使用した CPU は、Pentium4 1.7GHz、Pentium3 800MHz、Athlon 1GHz である。

ソフトウェアでの各 CPU の FPS は図 11 のグラフのようになった。

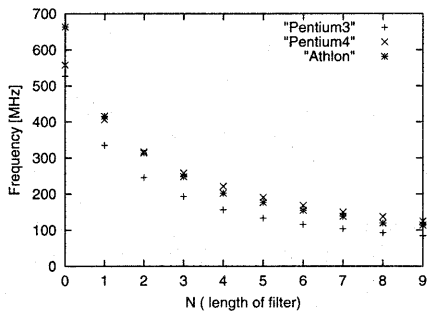

図 11　各 CPU での FPS

本稿で実装した基底数可変の DWT フィルタがこの FPS を得るために必要な動作周波数は以下の図 12 のグラフのようになる。

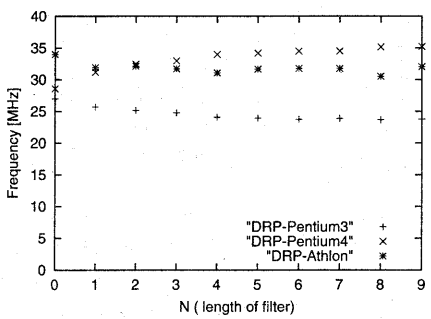

図 12　各 CPU の FPS を得るための必要動作周波数

図 12 より、本稿のフィルタがもっとも高速な Pentium4 と同様もしくは、それ以上の性能を得るためには、36MHz 以上の動作周波数を得られればよいことになる。DRP の最大動作周波数は 120MHz であり、今までの実装実績 [9] を考えると、この動作周波数は充分達成可能な値である。

また、今回実装した DWT フィルタでは、セグメントステージ 1、セグメントステージ 2 ともに、DRP 上の 8 つの乗算器のうち、4 つのみを利用している。残っている乗算器を使うと有効利用すれば、処理時間をさらに向上させることができる。

7. ま と め

本稿では NEC が開発した DRP 上に基底数可変の Daubechies 基底の DWT フィルタを実装した。このフィルタの大きな特徴として、設計し直すことなく外部からの値の設定で基底数を切り替えて使うことがあげられる。また、本稿で提案したフィルタをつかった 2 次元データを圧縮する DWT フィルタの性能は最近の CPU と同等の性能が得られるということがシミュレーションの結果からわかった。

8. 謝　　辞

DRP 及び開発環境を提供して頂き、本研究に対する多くのアドバイスをしてくださった NEC エレクトロニクスおよび NEC ラボラトリーズの方々に深く感謝します。

文　　献

[1] M.Motomura:"A Dynamically Reconfigurable Processor Architecture," Microprocessor Forum, Oct. 2002.

[2] Fujii, T., Furuta, K., Motomura, M., Nomura, M., Mizuno, M., Anjo, K., Wakabayashi, K., Hirota, Y., Nakazawa, Y., Ito, H. and Yamashina, M.: A Dynamically Reconfigurable Logic Engine with a Multi-Context/ Multi-Mode Unified-Cell Architecture, *Proc. Intl. Solid-State Circuits Conf.*, pp. 360–361 (1999).

[3] M.Yamashina, M.Motomura, "Reconfigurable Computing: Its concept and practical embodiment using newly developed DRL LSI," Proc. ASP-DAC 2000, pp.329–332, (2000).

[4] X.-P. Ling, H. Amano, "WASMII: A Data Driven Computer on a Virtual Hardware" Proc. FCCM '93, pp. 33–42, 1993.

[5] S. Trimberger, D. Carberry, A. Johnson and J. Wong "A Time-Multiplexed FPGA" Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines, pp.22-28, (1997).

[6] N.Kaneko, H.Amano, "A General Hardware Design Model for Multicontext FPGAs," Proc. of FPL2002, pp.1037-1047, 2002.

[7] Shibata, Y., Uno, M., Amano, H., Furuta, K., Fujii, T. and Motomura, M.: A virtual hardware system on a dynamically reconfigurable logic device, Proc. FCCM pp. 295–296, (2000).

[8] 宇野、天野、"DRL チップを用いたリコンフィギャブルテストベッドにおける WASMII システム," 信学報 CPSY2001-82, 2002 年 1 月

[9] 山田、出口、金子、天野、"マルチコンテキストリコンフィギャラブルデバイス上でのデータドリブンアプリケーションの実装," FPGA/PLD Conf. Exhibit. ユーザプレゼンテーション予稿、2003 年 1 月

[10] QuickSilver Technology, http://www.quicksilvertech.com/.

[11] IP FLEX DAP/DNA, http://www.ipflex.com/.

[12] K.K. Parhi, T. Nishitani, "VLSI architectures for discrete wavelet transforms," IEEE Trans. on VLSI Systems, Vol1, No.2, pp. 191–202, June 1993 4.1

[13] A.S. Lewis, G. Knowles "VLSI architecture for 2D Daubechies wavelet transform without multipliers," Electronics Letters, Vol. 27, No.2, pp. 171–173, Jan. 1991 4.1

[14] M. Vishwanath R.M. Owens, M.J. Irwin, "VLSI Architecturesfor the Discrete Wavelet Transform," IEEE Trans. on Circuits and Systems-11: Analog and Digital SP Vol. 42, pp. 305-316, 5. May 1995 4, 4.1, 4.1, 3, 5.8

[15] 榊原晋," ウェーブレットビギナーズガイド", 東京電気大出版