平成 14 年度

学士学位論文

JPEG2000 用離散ウェーブレット変換のデータ駆動型実現法

A Data-Driven Implementation of Discrete Wavelet Transform for JPEG2000

1030274　　そね田　紘貴

指導教員　　岩田　誠

2003 年 2 月 24 日

高知工科大学 情報システム工学科

要 旨

JPEG2000 用離散ウェーブレット変換のデータ駆動型実現法

そね田　紘貴

近年、高圧縮率・高画質・各アプリケーションへの柔軟な対応が可能な静止画圧縮方式として JPEG2000 が標準化された。JPEG2000 はインターネット・印刷・スキャン・デジタル写真技術・モバイル・医療画像・電子商取引といった高品位な画質が求められる分野からモバイルまで幅広い分野での活用が期待されている。しかし JPEG2000 には

- タイリング時のブロック歪み
- EBCOT の高処理負荷
- 離散ウェーブレット変換 (DWT) の必要メモリの増大にともなう高処理負荷

など、いくつかの課題が指摘されている。

本研究では、JPEG2000 の高処理負荷の要因の 1 つである DWT を、超高速・省電力のデータ駆動型マルチメディアプロセッサ DDMP で高速に実現する手法を提案する。DDMP は、従来のノイマン型プロセッサとは異なる独自のアーキテクチャを持ち、取り分け映像処理においてその真価が発揮される。本手法では、DDMP の「2 世代化」と呼ばれる処理機構を利用し、処理負荷を軽減した。また、ノードに条件実効命令拡張子を指定し必要なメモリ量を削減した。

本稿では提案手法を用いた整数型 DWT の実装方法を提示すると共に、シミュレーションによりその性能を評価した。結果、本提案手法が従来の DSP による処理と比較して高速に実現可能であることを明らかにした。

キーワード　データ駆動，離散ウェーブレット変換，JPEG2000, 2 世代化

Abstract

A Data-Driven Implementation of Discrete Wavelet Transform for JPEG2000

Hirotaka SONEDA

JPEG2000 was recently standardized as an image compression scheme in place of JPEG. Then, various JPEG2000 applications, e.g., a digital photograph, a medical image, E-commerce, mobile terminal, and so on, might come into wide use because of its higher compression rate, quality, and flexibility. However, the JPEG2000 employed some complicated sub-algorithms such as embedded block coding optimized transform (EBCOT) and discrete wavelet transform (DWT).

In this research, a data-driven parallel implementation method of lifting-based integer DWT for JPEG2000 is proposed. The proposed method utilizes spatial and time parallelism inherent in the DWT. Furthermore, the method efficiently executes this parallel DWT program on a data-driven media processor (DDMP) because DDMP can realize the highly parallel processing and the low power consumption by virtue of its unique architecture. In the method, a pair of neighbor pixel data with each conditinals execution flag is concurrently operated by single machine instructions, so that the total processing load reduced to about 50 % without memory. Performance evaluation results shows that the proposed method is 1.4 times faster than the other DSP implementations under the equivalent clock frequency.

key words Data-Driven, Discrete Wavelet Transform, JPEG2000, Pair-Packet

目次

図目次

図目次

表目次

第 1 章

序論

近年、マルチメディアとインターネットアプリケーションの発展とともに、静止画像圧縮技術はインターネットでの表示や、デジタルカメラや個人端末での画像保存で利用されるだけでなく、リモートセンシングや医療画像・e コマースといった分野でも利用されるようになっている。

現在、一般に広く普及している静止画像圧縮技術に、JPEG や GIF がある。画像処理の特性から、JPEG は主に自然画像、GIF は主に抽象画像に用いられている。しかし、静止画像圧縮技術を利用するアプリケーションの増加に伴い、今までの方式では各アプリケーションへの対応が困難になった。そのため、様々なアプリケーションの要求に対応可能で、かつ現在の圧縮方式よりも高圧縮率を持ちながら高品質を保持可能な圧縮形式が望まれるようになってきた。

この要求に応えるべく考案されたのが，静止画像圧縮標準規格 JPEG2000[1] である。JPEG2000 には様々なアプリケーションの要求に対応可能な特徴が多数あり、インターネット・カラーファクシミリ・印刷・スキャン・デジタル写真技術・リモートセンシング・モバイル・医療画像・電子図書館/アーカイブおよび電子商取引といった高品位な画質が求められる分野からモバイルまで幅広い分野での活用が想定されている。これらの市場はすでにデジタル画像の市場として存在しており、現在 JPEG などの既存の符号化法が適用されている。しかし、JPEG よりも非常に圧縮率が高く、同等の画質を実現するのに JPEG2000 なら 50〜70 %のファイルサイズで足り、様々なアプリケーションの要求に対応可能、といった JPEG2000 の特徴を考えると、これから広く活用されることが期待できる。また、Motion-JPEG2000 という動画対応の JPEG2000 も策定されている。動画像圧縮方式とし

て MPEG が広く利用されているが、MPEG は時間軸方向の圧縮も行っているため、圧縮/伸張システムが大規模になるという問題がある。そこで時間軸方向の圧縮をすることなく、単位時間ごとに連続した静止画像を圧縮する Motion-JPEG2000 ならば MPEG と比較してシステムの小規模化を図ることができるという利点を得られる。

しかし、JPEG2000 は、JPEG に比べて圧縮/伸張に必要とされる演算処理量が増えるという問題点がある。　JPEG2000 は JPEG にくらべ、圧縮で 5 倍〜6 倍、伸張で 3 倍〜4 倍の処理能力が必要といわれており [2]、速度が重視される分野では大きな障害になる。この問題点は (1)EBCOT(Embedded Block Coding Optimized Transform) の高処理負荷、(2) 離散ウェーブレット変換 (Discrete Wavelet Transform) の必要メモリの増大にともなう高処理負荷、の 2 点がおもな要因だと考えられ、これらの技術課題が生じ、JPEG2000 の処理の高速化を困難にしている [3]。

そこで、本研究では JPEG2000 の高処理負荷の要因の 1 つである DWT を、超高速・省電力のデータ駆動型メディアプロセッサ DDMP(Data-Driven Media Processor)[4] で高速に実現する手法を提案する。

DDMP は、従来のノイマン型とは異なる独自のアーキテクチャを採用することで高速処理、低消費電力化を実現している。ノイマン型プロセッサがプログラムカウンタにより逐次的に命令を取り出して実行するのに対し、DDMP は行き先情報、世代識別子等を含むパケット形式のデータを取り扱う。これにより、データは独立してプロセッサ内を巡回することが可能となる。世代識別子は、フィールド、ライン、ピクセルの 3 次元で構成され、同じ識別子を持つデータ同士に対してのみ処理が実行されるという特性を持つ。これらの識別子により、各データに独立性が与えられ複数の処理をパイプライン並列に実行可能となり、取り分け映像処理においてその真価が発揮される。

本提案手法では (a) 空間的・時間的並列処理における負荷の軽減、(b) メモリ使用量の削減、を行う。(a) については、隣接する 2 つの画素データを 1 つのパケットに格納する「2 世代化」と呼ばれる処理機構の活用により入力データを半減し、処理負荷を軽減する。また (b) については、ノードに条件実行命令拡張子を指定し、必要なメモリ量を削減し、これに

対応する。

本稿では提案手法を用いた整数型 DWT の実装方法を提示すると共に、シミュレーションによりその性能を評価した。結果、本提案手法が従来の DSP による処理より高速に実現可能であることを明らかにした。

第 2 章では JPEG2000 の符号化手順の概要について述べると共に、各要素技術を述べる。そして、それを基に JPEG2000 の問題点を考察する。

第 3 章では、第 2 章で述べた問題点を解決する方法として DDMP による整数型 DWT の高速実現を提案する。整数型 DWT は加減算命令・シフト命令といった、少数の命令セットで実現可能であり、ハードウェア実現において低演算量と低消費電力化が期待できる。整数型 DWT を DDMP で作成するために、整数型 DWT の並列性から検証し、DDMP で実現するための要件と問題点を述べる。

第 4 章では第 3 章であげられた問題点を解決する方法として、(a)「2 世代化」と呼ばれる処理機構を用いて処理負荷を軽減する、(b) ノードに条件実行命令拡張子を指定し、必要なメモリ量を削減する、の 2 つの実現法を提案し、32bit DDMP 上で整数型 DWT のリフティング構成を実現したプログラム構成を提示する。

第 5 章ではシミュレータにより、第 4 章の提案について性能評価結果を示し、その有効性を明らかにする。提案手法の評価については市販のの DSP による整数型 DWT の処理時間と比較する。

第 6 章では、第 5 章までで得られた結果をもとに提案手法の実用性、残された課題について述べる。

第 2 章

JPEG2000 の符号化手順

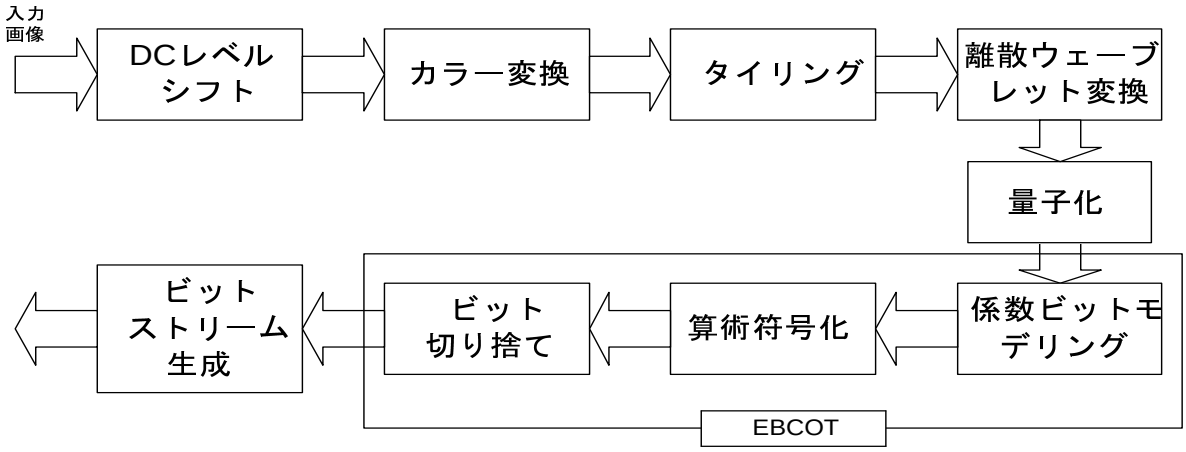


図 2.1　JPEG2000 の符号化手順

図 2.1 に JPEG2000 の符号化手順の概略を示す。以下、各要素技術について説明する [1,4-9]。

2.1　DC レベルシフト

入力画像は、まず DC レベルシフト部に入力される。DC レベルシフト処理は後に処理をする離散ウェーブレット変換を効率的に行うためであり、圧縮率の向上につながる。RGB 信号は一般に正の整数として与えられるので、ダイナミックレンジの半分を減算し、逆変換では各信号値にダイナミックレンジの半分を加算する処理を行う。なお、入力信号が符号つき整数の場合には、このレベルシフト処理は行われない。

例えば 1 画素 24bit の画像の場合、3 つのコンポーネント (R,G,B) が存在し、各々のコンポーネントは 0〜255 の 8bit 符号なし整数で表されるので、これを以下の式に示すように

128 減算することで-128〜127 にレベルシフトする。

$$R = R - 128, \quad G = G - 128, \quad B = B - 128$$

2.2　色変換

入力された RGB の色空間は輝度 Y と色差 Cr・Cb からなる YCrCb の色空間に変換する。人間の視覚は、輝度成分に敏感で色差成分に鈍感という特性を持つ。この特性を利用し、圧縮率の向上を可能とする。JPEG2000 では 2 種類の色変換が準備されている。非可逆符号化に対しては ICT(Irreversible Color Transform)、可逆符号化に対しては RCT(Reversible Color Transform) を使用する。これは ICT が非可逆であり、可逆符号化に利用できないためである。計算式を以下に示す。

- ICT

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.229 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.34413 & -0.71414 \\ 0.5 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix}$$

- RCT

$$\begin{bmatrix} Y^{'} \\ C_b^{'} \\ C_r^{'} \end{bmatrix} = \begin{bmatrix} \lfloor \frac{R+2G+B}{4} \rfloor \\ R - G \\ B - G \end{bmatrix}$$

$$\begin{bmatrix} G \\ R \\ B \end{bmatrix} = \begin{bmatrix} Y^{'} - \lfloor \frac{C_b^{'} - C_r^{'}}{4} \rfloor \\ C_b^{'} + G \\ C_r^{'} + G \end{bmatrix}$$

2.3　タイリング

タイリングとは画像をいくつかのブロック (タイル) に分割し、以後、タイルを 1 枚の独立な画像として扱う。タイルのサイズは任意に選択可能であり、その最小単位は 1 × 1 である [7]。タイリングの例を図 2.2 に示す。

図 2.2　タイリング分割例 (分割数 4 × 5)

2.4　離散ウェーブレット変換

JPEG2000 では 2 次元信号である画像に対して、1 次元の 2 分割フィルタバンクを繰り返し使用して 2 次元 DWT を実現する。図 2.3 に 2 分割フィルタバンクを用いた 1 次元 DWT を示す。

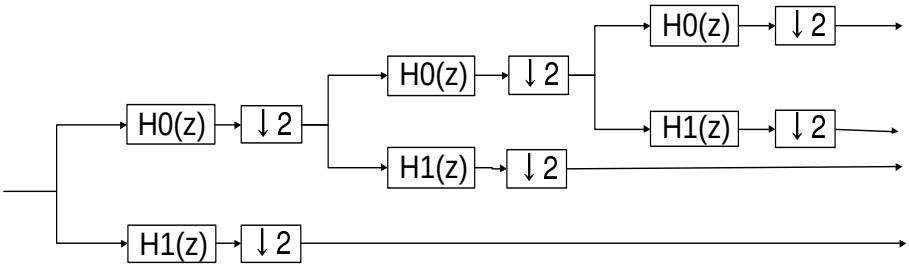


図 2.3　1 次元ウェーブレット分割

図中のＨ 0 はローパスフィルタ、Ｈ 1 はハイパスフィルタである。1 回の帯域分割の回数をウェーブレット変換の分解レベルと呼び、図 2.3 における分解レベルは 3 である。2 次元の画像信号に対しては、水平・垂直にこの処理を 1 レベルずつ実行する。図 2.4 は 2 回 2 次元 DWT を実行した場合の帯域成分を示した図である。図中の Ｌ は低域成分、Ｈ は高

域成分を示す。1 回目の 2 次元 DWT によって画像を 4 つの帯域 (LL,LH,HL,HH) に分解し、さらに最低域成分 (LL) を 4 つの帯域 (LLLL,LLLH,LLHL,LLHH) に分割している。JPEG2000 では 2 次元分解を 5 回まで繰り返す処理を基本としている。図 2.5 に実際の画像に 2 次元 DWT を 2 回行った結果を示す。 JPEG2000 は、DWT をリフティング構成を

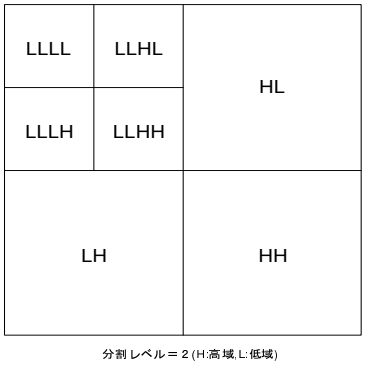


図 2.4　2 次元ウェーブレット分割 (2 レベル分割)

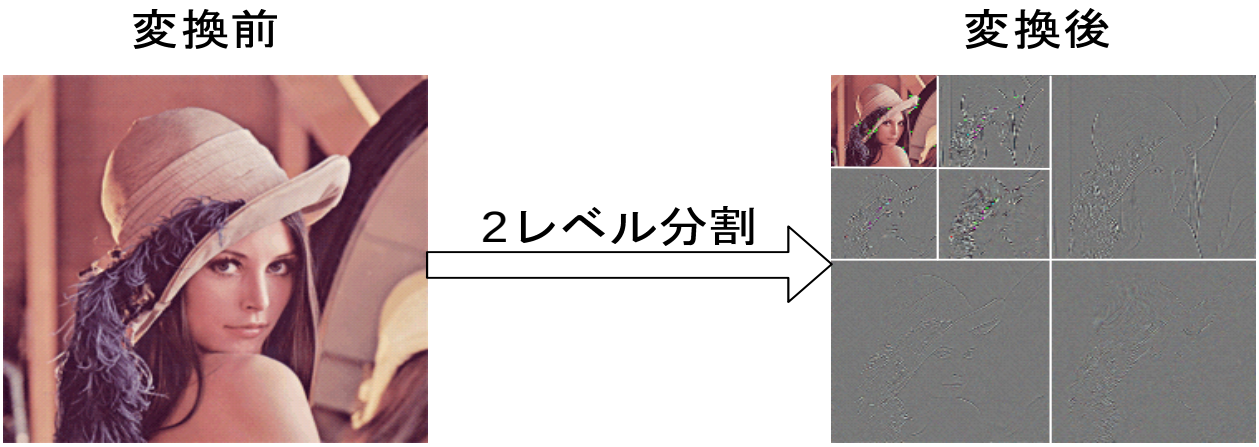


図 2.5　ウェーブレット変換による画像のウェーブレット分割例

基に実現することで可逆・非可逆符号化を可能としている。また JPEG2000 には、整数型 DWT と実数型 DWT の 2 種類の DWT が存在している。図 2.6、図 2.7 に示すリフティング構成は図 2.3 の 1 次元ウェーブレット分割の 1 レベルの処理に相当する。

2.4.1　整数型 DWT

整数型 DWT は以下に示す 5-3 フィルタと呼ばれるリフティング構成を水平方向・垂直方向に実行する。 図中の $P(z)$、$U(z)$ はそれぞれ以下の式で表される。

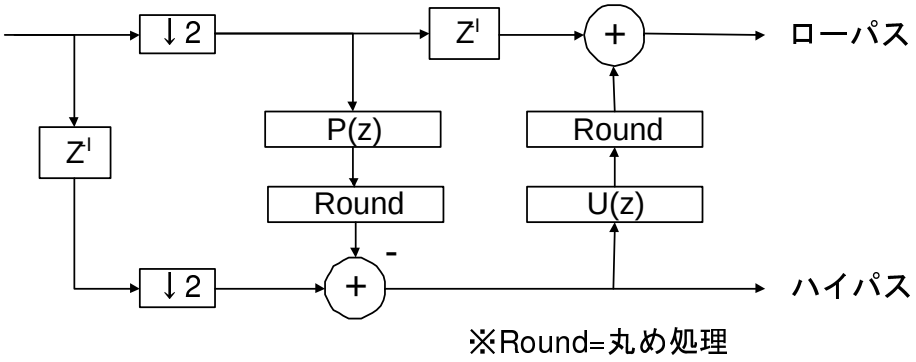


図 2.6　整数型 DWT のリフティング構成例

$$P(z) = \frac{1 + z^{-1}}{2} \qquad U(z) = \frac{1 + z^{-1}}{4}$$

整数型 DWT は Round 処理で小数第 1 位で四捨五入を行い、結果を整数化する。その際に発生した丸め誤差は逆変換時に排除することが可能となっており、図 2.6 の Round(丸め処理) を行うことで可逆符号化を実現することができる。また、整数型 DWT は加算と減算とシフト命令で実現することができ、ハードウェア実現において低演算量と低消費電力化が期待できる。

2.4.2　実数型 DWT

実数型 DWT は以下に示す 9-7 フィルタと呼ばれるリフティング構成を水平方向・垂直方向に実行する。 実数型 DWT は整数型 DWT に比べ、処理演算量は多いが、符号化効率が高い。図中のリフティング係数は

$\alpha = 1.586134342$

$\beta = -0.05298011854$

$\gamma = 0.8829110762$

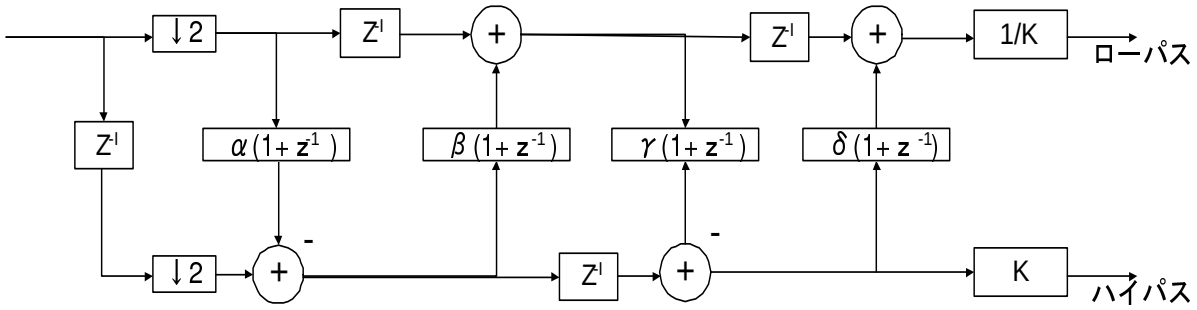


図 2.7　実数型 DWT のリフティング構成例

$\delta = 0.4435068522$

$K = 1.149604398$

となる。また、実数型 DWT では Round 処理は行われず、非可逆となっている。

2.4.3　実際の計算例

ここでリフティング構成による DWT の実際の計算を示す。整数型 DWT と実数型 DWT は共に原理が同じであるため、フィルタ数の少ない整数型 DWT を例に挙げて解説する。

元画像の有限長の信号に対し、対称性を持つように拡張する操作を対称拡張、さらに周期信号に拡張する操作を対称周期拡張と言う。また、この対称周期信号に対するフィルタ処理を対称周期たたみこみという。JPEG2000 では先に述べたように整数型 DWT をリフティング構成により実現する。したがって、対称周期たたみこみもリフティング構成に対して適用される。リフティング構成による整数型 DWT は次の 2 式で実現できる。

$$y(2n+1) = x_{ext}(2n+1) - \lfloor \frac{x_{ext}(2n) + x_{ext}(2n+2)}{2} \rfloor \quad (1)$$

$$y(2n) = x_{ext}(2n) - \lfloor \frac{y(2n-1) + y(2n+1) + 2}{4} \rfloor \quad (2)$$

2 式における x_{ext} は入力信号、y は出力信号である。(1) 式はハイパス出力に相当する。またこの結果を用いて (2) 式を計算するとローパス出力が得られる。図 2.8、図 2.9 に入力信号 $x(n) = (a, b, c, d, e, f)$ が与えられたときの演算を示す。入力信号長は偶数なので左に 2 点、右に 1 点、拡張を行う。拡張された信号を偶数信号、奇数信号に分解した後、ハイパス演算、ローパス演算を行う。

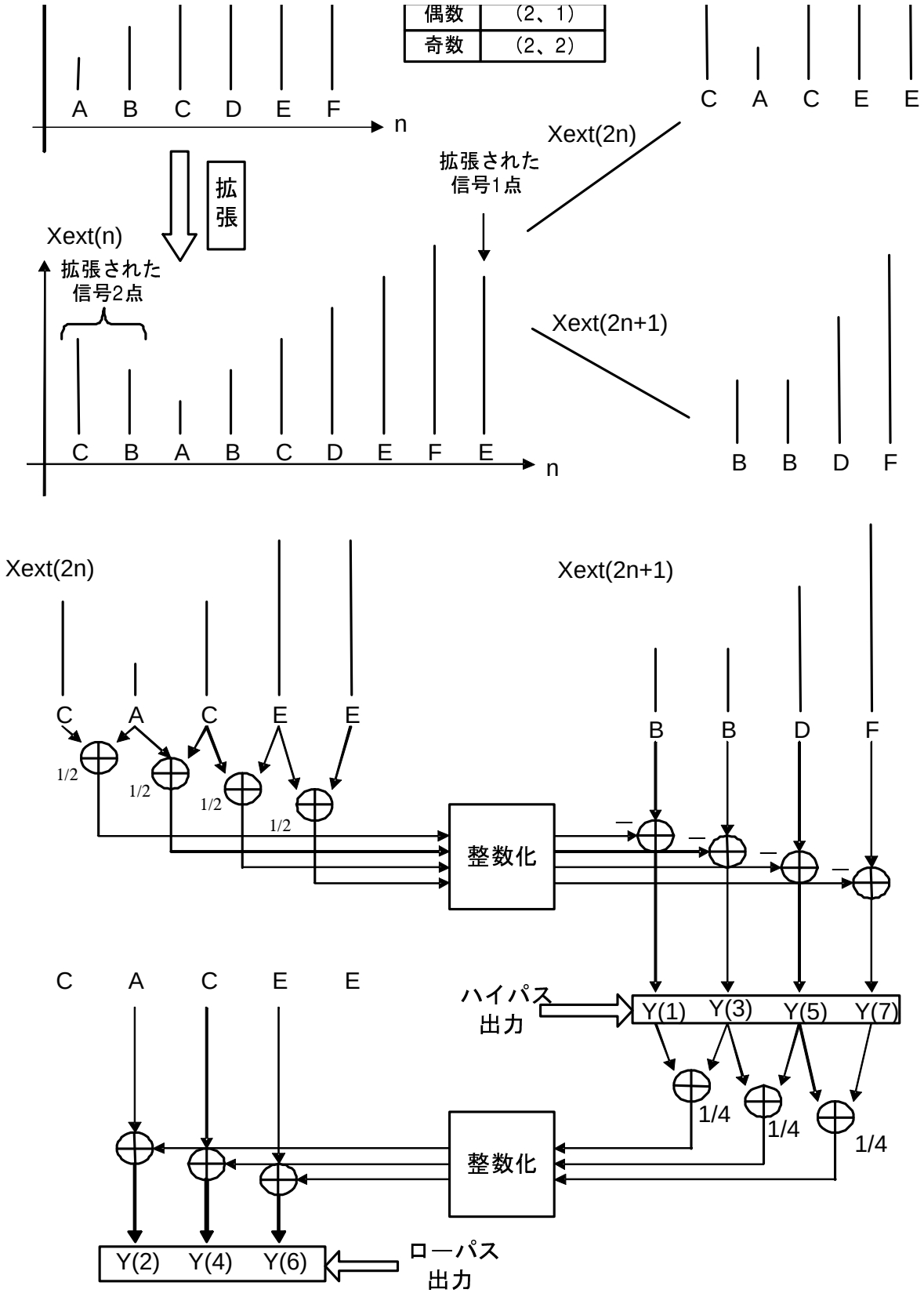


図 2.8　リフティング構成による整数 DWT の実際の計算

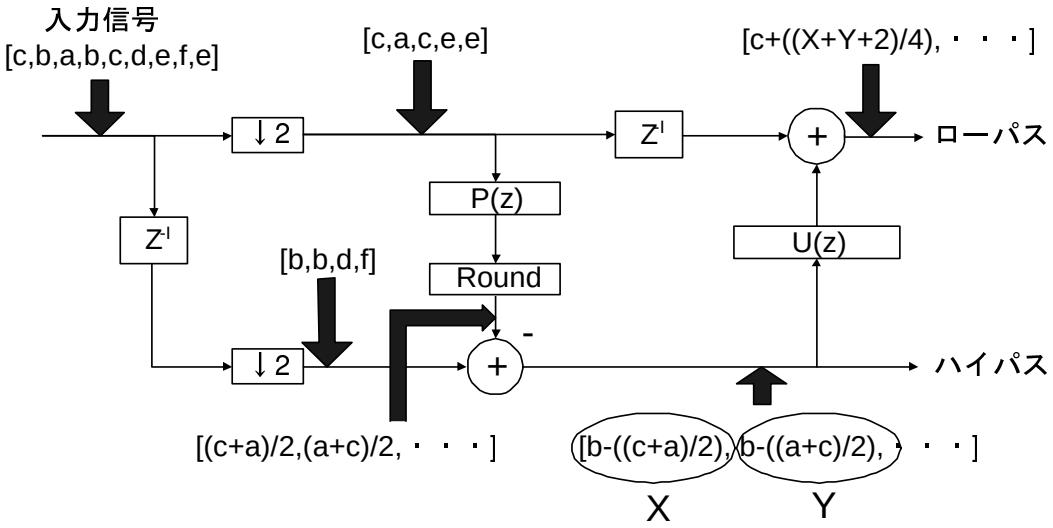


図 2.9　整数型 DWT のリフティング構成の演算例

2.5　量子化

JPEG では DCT(Descrete Cosine Transform) 係数を必ず量子化するが、JPEG2000 の量子化は実行/不実行を選択できる。前述した DWT において、可逆である整数型 DWT を行った場合、量子化を行わないことで可逆圧縮を可能とする。また、圧縮データ量の制御を算術符号化の後に符号の切り捨ても可能である。ウェーブレット係数を量子化する場合にはスカラ量子化を行う。スカラ量子化の量子化ステップサイズは次の式で表現される。

$$\Delta_b = 2^{R_b - \varepsilon_b}(1 + \frac{\mu_b}{2^{11}})$$

Δ_b はサブバンド b の量子化ステップ、R_b はサブバンド b のダイナミックレンジである。ε_b と μ_b は逆量子化の際に利用される、量子化ステップ表現の値である。スカラ量子化のステップサイズを決める方法として JPEG2000 には Implicit と Explicit の 2 種類がある。Explicit は Implicit に比べ複雑ではあるが、より最適なステップサイズを選択できる。

- Implicit

  DWT 係数の最低域のステップサイズから他のバンドのステップサイズを決定する。

- Explicit

  すべてのサブバンドごとに計算を行い、ステップサイズを決定する。

2.6 EBCOT

DWT 後の変換係数は EBCOT(Embedded Block Coding with Optimized Truncation) と呼ばれるアルゴリズムにより符号化される。EBCOT は Embedded 符号化アルゴリズムのひとつである [7]。JPEG2000 では受信済みの符号化データへ追加データを送ることによって、画質を向上させる SNR スケーラビリティが EBCOT の使用により容易に実現されている。EBCOT の処理手順を図 2.10 に示す。

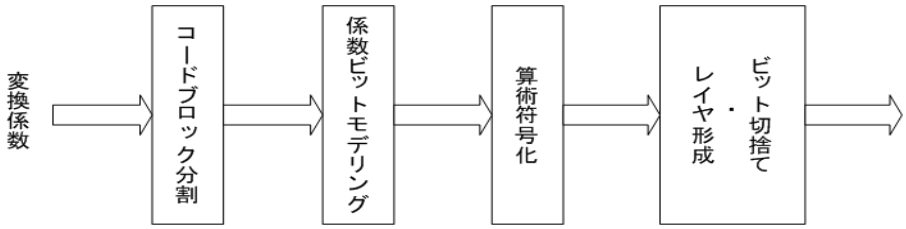


図 2.10 EBCOT の処理手順

2.6.1 コードブロック分割

入力された各サブバンド係数はコードブロックと呼ばれる矩形領域に分割される。EBCOT ではこのコードブロックが処理単位となる。コードブロックのサイズは面積が 16 画素から 4096 画素の範囲内ならば自由に決めることが可能である。

2.6.2 係数ビットモデリング

分割された各コードブロックに対し、まずビットプレーン分解を行う。コードブロックが 4×4 のサイズの場合、16 画素の量子化係数が表現される。この 16 個の係数のうち、絶対値が最大のものが $13(1101_{(2)})$ とすると、このコードブロックは 4 つのビットプレーンに分解される。そして係数ビットを判定し、それぞれのビットプレーンを 3 つの符号化パスに分解する。

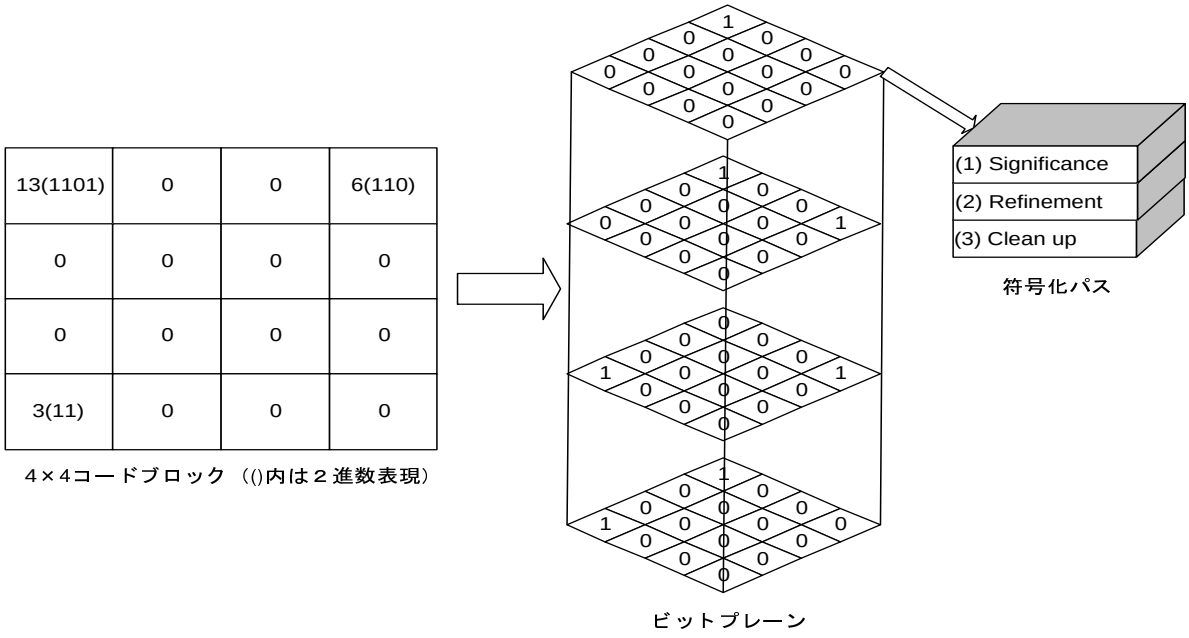


図 2.11　ビットプレーン分解

2.6.3　算術符号化

JPEG2000 では、2 値算術符号化である MQ コーダが採用されている。この符号化は効率的に画像を圧縮するという目的のほかに、階層的に圧縮データを構成可能にする。

2.6.4　レイヤ形成・ビット切り捨て

レイヤとは再生画像における SNR 向上に対応する符号化データのまとまりである。上位レイヤのデータに対して下位レイヤのデータを追加送信することで段階的に再生画像の向上が可能となる。また、与えられたビットレートを超えるデータに対しては切り捨て処理を行う。

2.7　問題点

JPEG2000 はリフティング構成による DWT と EBCOT の使用により、多くの優れた特徴を実現している。しかしこれらの技術は同時にいくつかの技術問題を生じさせている。

2.7 問題点

DWT は、処理に要するメモリが画像サイズの大きさに依存する。そのため高解像度の画像ほど、処理に必要なメモリは増加してしまう。解決方法として、タイリングの使用で DWT のメモリ使用量を削減可能であるが、高圧縮時にタイル間のブロック歪みが発生してしまう。また、DWT は多くの演算量を必要としているため、高速化が困難となっている。

EBCOT における算術符号化は 2 値算術符号化器である MQ コーダを用いている。MQ コーダは学習型の算術符号化器であり、学習の際には、46 の状態からなる状態遷移表を用いる。この符号化方式は大量の確率データの保存、更新が必要であり、演算量が多いため、高処理負荷となっている。

第 3 章

DWT のデータ駆動型実現法の要件

第 2 章では JPEG2000 の符号化手順を述べるとともに、JPEG2000 の問題点を考察した。前述したように DWT は加減算とシフト命令といった、少数の命令セットで実現可能であり、ハードウェア実現において低演算量と低消費電力化が期待できる。そこで本研究では JPEG2000 の高処理負荷の一つである DWT を DDMP によって高速実現する手法を提案する。

3.1 DWT の概要

図 3.1 はある入力画像における DWT を表したものである。DWT のアルゴリズムはまず、入力画像の対称周期拡張を行い、水平方向に 1 次元 DWT を行い、低域成分と高域成分に分解する。次に、その低域成分と高域成分に対し 1 次元 DWT を行うことで、低域：低域、高域：低域、低域：高域、高域：高域といった成分分解を行う。

JPEG2000 には、整数型 DWT と実数型 DWT の 2 種類の DWT が存在している。実数型 DWT は整数型 DWT のリフティング構成にいくつかのフィルタの追加とフィルタ係数の変更のみで実現可能である。そこで本稿では整数型 DWT の高速実現法を考案する。

3.2 DWT の並列性

独立したデータについて処理を行う場合、その部分を並列に実行したほうが処理レートが向上することから、DWT の高速化を図るためには DWT 処理に内在する並列性を十分に抽出し、ハードウェアにより並列処理するのが望ましい。

3.2　DWT の並列性

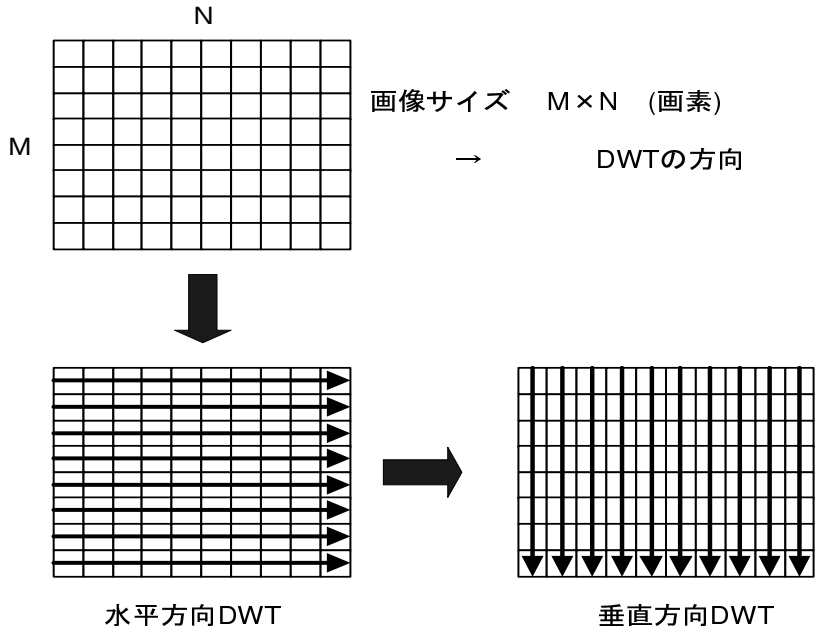


図 3.1　DWT 処理アルゴリズム

DWT の変換単位はライン単位で処理される。各ラインはそれぞれ独立した領域を表しており、互いに影響することがない。また、水平 DWT/垂直 DWT では水平 DWT が 3 ライン処理することができれば垂直 DWT を実行できる。各画素もそれぞれ独立した値として扱うことができる。タイリング処理によってタイル化されたデータもそれぞれ独立しており、互いに影響がない。以上のことにより、DWT には以下の 3 つの並列性があげられる。図 3.2 にライン単位の独立性とピクセル単位の独立性を示す。

- ライン単位での並列性

  本実現法における各ラインはそれぞれ他のラインの画像情報と独立して扱える
- 各ピクセル単位の並列性

  各ピクセルにおいてピクセルデータは互いに独立しており、それぞれのデータが独立に処理できる。
- タイル単位の並列性

  タイリングとは画像をいくつかのブロック (タイル) に分割し、以後、タイルを 1 枚の独立な画像として扱う。タイルは互いに独立しており、並列に処理することが可能で

ある。

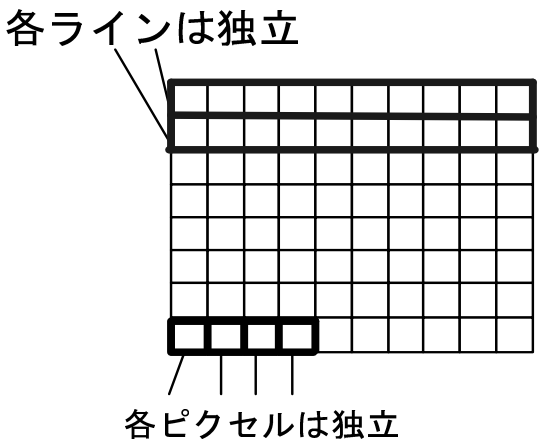


図 3.2　DWT の並列性

3.3　整数型 DWT の並列アルゴリズム

図 3.2 の画像は、左上の画素から順に水平方向入力される。DWT の処理アルゴリズムから、各ピクセル/ラインの並列性より、入力とほぼ同時に水平方向 DWT を行える。次の垂直 DWT は、縦 3 ラインが入力されないと実行できないため、それまでデータを一時メモリに退避させておく必要がある。水平 DWT が 3 ライン分入力されれば垂直 DWT を水平 DWT と同時に行う。以上のことを考慮した上でフローグラフを作成すると図 3.3 のようになる。

水平方向 DWT/垂直方向 DWT は共にリフティング構成を基に処理を行う。本稿ではこの整数型 DWT のリフティング構成を実現する。

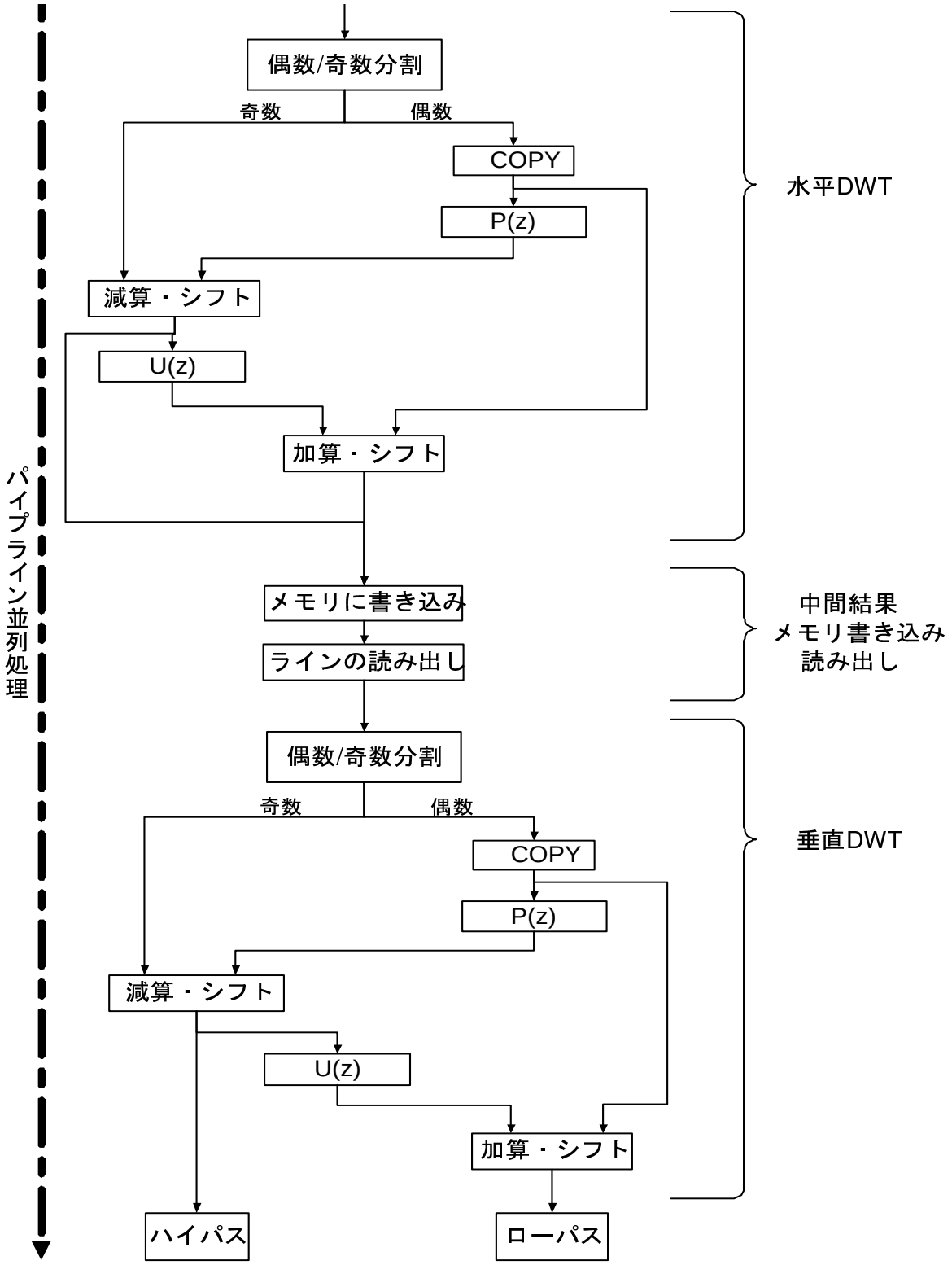
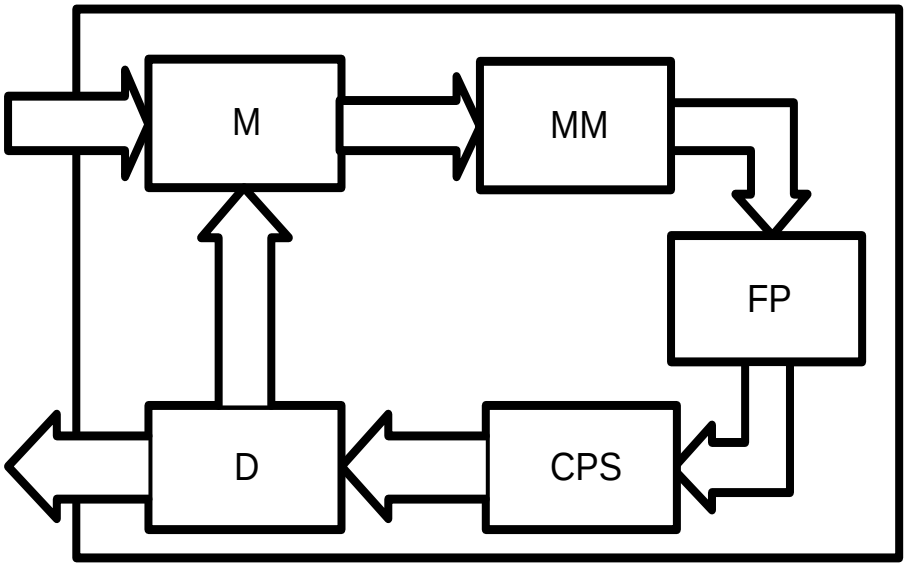


図 3.3　整数型 DWT 処理フローグラフ図

3.4 DDMP

3.4.1 DDMP の基本アーキテクチャ

DDMP は、従来のノイマン型プロセッサとは異なる独自のアーキテクチャを採用することで、高速処理・低消費電力を実現している。ノイマン型プロセッサがプログラムカウンタにより逐次的に命令を取り出して実行するのに対し、DDMP は行き先情報、世代識別子等を含むパケット形式のデータを取り扱う。したがって、パイプライン処理の効率的な実現が可能である。



MM :Matching Memory(待ち合わせメモリ)
FP :Functional Processing Unit(演算部)
CSP :Cache Program Storage(プログラム記憶部)
M :Flow Merging Module(合流部)
D :Flow Diverting Module(分岐部)

図 3.4 DDMP のナノプロセッサ構成図

図 3.4 に DDMP のナノプロセッサ構成を示す。パケット内の行き先情報は図 3.4 の分岐部 D においてデータの行き先の決定に使用される。行き先情報は処理内容に応じて CPS 部で変更され、これによりデータは独立してプロセッサ内を巡回する。世代識別子は、フィールド・ライン・ピクセルの 3 次元で構成される。DDMP は、同じ行き先情報・世代識別子を

持つデータ同士に対してのみ実行する特性を持つ。図 3.4 の MM 部に処理対象となるデータのうち先着のものを格納し、同一の識別子を持つデータが MM 部に到達したとき、格納していたデータを読み出し FP 部で演算処理する。これらの識別子により、図 3.5 に示すように各データに独立性が与えられて空間的・時間的な並列処理が可能となる。

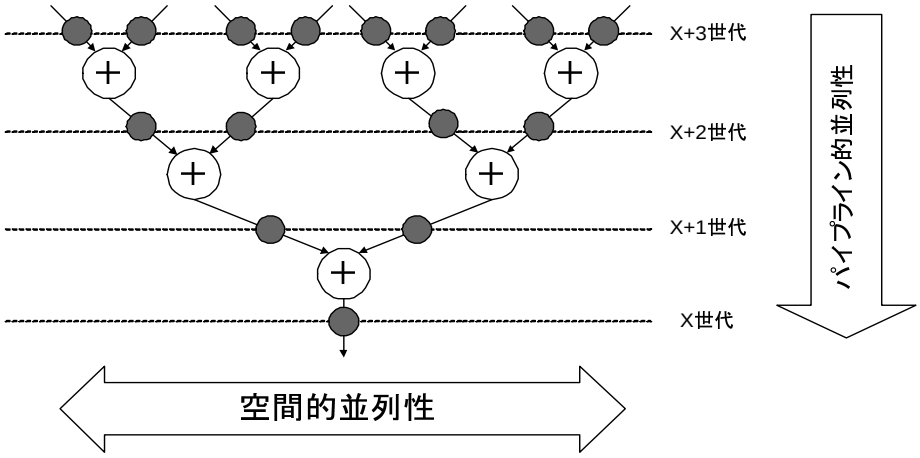


図 3.5　DDMP の並列概念図

DDMP のパイプラインは自己タイミング型パイプライン機構となっている。これは各々のデータが自己タイミングするため、処理に必要な部分のみを動作させることができ、省電力化できる。また、複数のパイプラインを並列に動作させ、かつ、連結して巨大なパイプラインを形成できるため、柔軟な処理変更が可能であり、スループットを向上できる。

3.4.2　DDMP と映像処理の親和性

前述したように、DDMP では世代識別子と呼ばれる 3 次元の識別子を用いる。映像処理においては、図 3.6 に示すようにピクセル番号で水平方向、ライン番号で垂直方向、フィールド番号で時間方向を識別する。これにより、各画素は独立性を持ち並列処理が可能となる。

また、このアーキテクチャの有効性を高めるために DDMP には世代識別子をアドレスとしてアクセス可能なメモリを搭載している。DDMP-10A には 32bit × 2K words のデータを格納できる内部メモリを 10 個と 64M words のデータを格納することのできる外部メモリを 1 つ搭載している。これらのメモリを有効に利用し、プロセッサ内を巡回するパケット

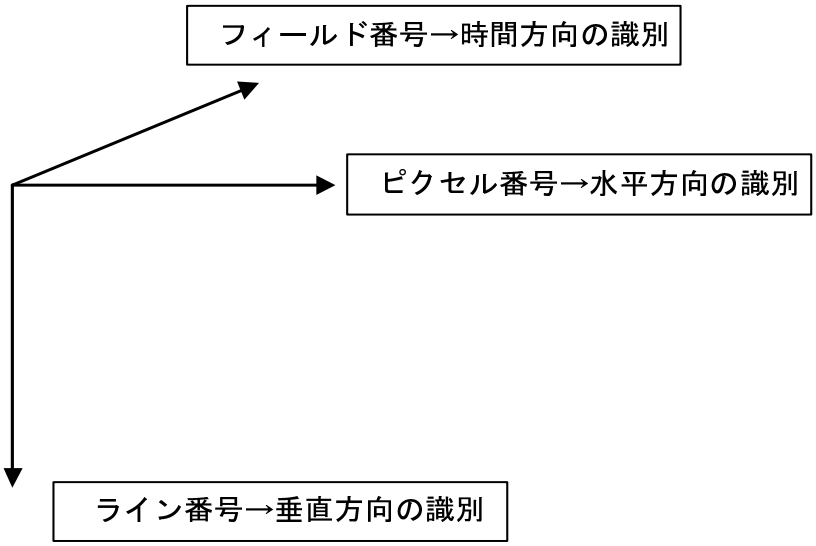


図 3.6　映像処理における世代識別子の割り当て方

数を極小化することで処理性能の向上が可能となる。

3 次元の世代識別子とそれを使ってアクセス可能なメモリを搭載することで、単純なアドレス指定型線形メモリを搭載したプロセッサに比べ、DDMP は映像処理との親和性を高めている。

3.5　整数型 DWT 実現の課題

図 3.8 は整数型 DWT のリフティング構成をデータフローグラフ化したものである。入力信号数は偶数とする。入力信号が偶数の場合、最後の信号に対しての周期拡張は 1 点のみであるため画像全体への影響は低いと考える。そのため、対象周期拡張ははじめの信号に対しては行うが、最後の信号に関しては周期拡張を行わない。これにより命令数の削減が可能となる。

まず画像が入力され、入力データのピクセル番号の偶数番号と奇数番号で分岐を行う。分岐された奇数世代に対しては世代に加算を行う。偶数番号に対してはパケットの複製を行う。複製された偶数世代パケットはメモリにデータを格納し、破棄するものと、自分のパケットの次の偶数世代と加算するものに利用される。世代番号が 0 のデータだけ廃棄してい

るのは対称周期拡張のためである。世代に加算を行った奇数世代のデータから、偶数同士の加算を行ったデータを減算し、高域成分を出力する。高域成分の出力パケットの世代はすべて奇数世代なので、出力する際には世代番号から 1 減算し、偶数世代にして出力する。

次に出力された高域成分をもとに低域成分を出力する。まず出力された高域成分のパケットを複製する。複製されたデータは自分のパケットのひとつ手前の世代番号のパケットデータと加算するために利用する。世代番号 1 のデータに対してのみ処理が異なるのは対称周期拡張のためである。加算されたデータはさらに 2 を加算した後、自分のパケットの世代番号をアドレスとしてメモリアクセスしメモリに格納していたデータと加算を行う。結果、低域成分を出力することが可能となる。図 3.7 に示すように偶数番号の世代には低域成分のデータを、奇数番号のデータには高域成分を格納する。

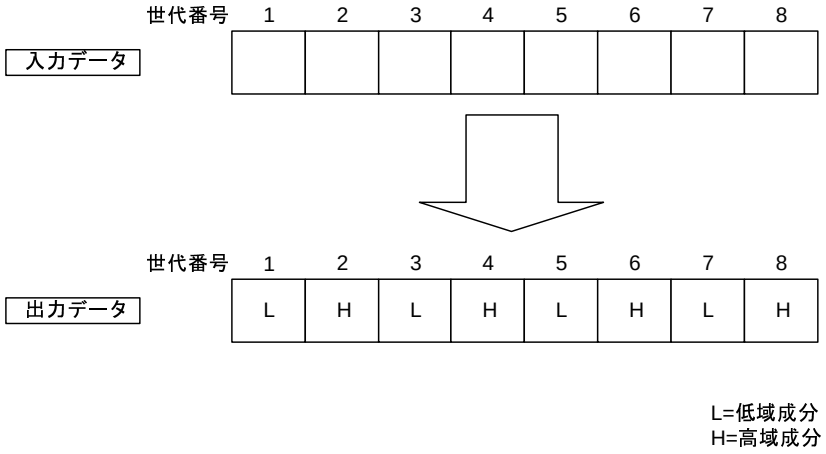


図 3.7　入出力結果の概念

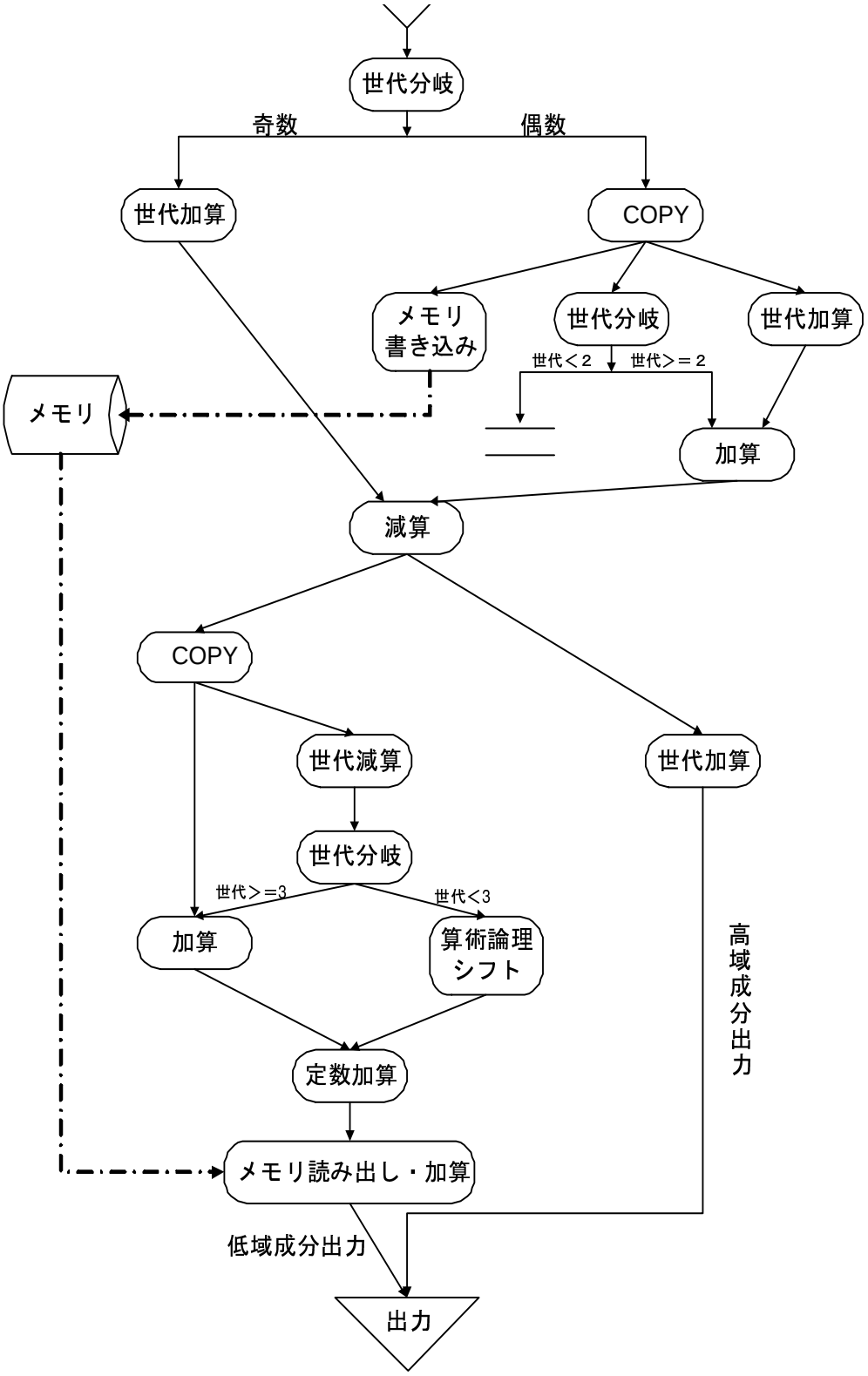


図 3.8　整数型水平 DWT データフローグラフ図

この実現法には処理時間の問題がある。フローグラフのパイプライン段数が多いため、一つのパケットの処理に時間がかかってしまい、高速化が困難となっている。表 3.1 に処理時間の比較結果を示す。比較対象として、Analog Devices 社の DSP「ADSP-21535」、Texas Instruments 社の「TMS320C64x」の処理時間データを用いた。この問題を解決するためには空間的・時間的処理における負荷の軽減を行う必要がある。また、メモリ使用量も削減することが望ましい。

表 3.1　性能評価 (data size = 256)

	動作周波数	処理時間
DDMP	140MHz	7.8 μ sec
ADSP-21535	300MHz	2.6 μ sec
TMS320C64x	300MHz	1.7 μ sec

第 4 章

DWT 処理データ駆動並列の最適化

第 3 章では DDMP で整数型 DWT のリフティング構成を実装した場合の問題点を示した。本章ではそれらの問題点の解決に以下の手法を提案する。

- 「2 世代化」と呼ばれる処理機構で入力データを処理負荷を軽減する
- ノードに条件実行命令拡張子を指定し、必要なメモリ量を削減する

4.1 2 世代化による処理負荷の軽減

4.1.1 2 世代化

DDMP は 2 世代化と呼ばれる、1 つのパケットに 2 つのデータを格納し、それらを同時に処理する方式を可能としたハードウェア機構を備えている。それぞれのデータの世代番号は、偶数世代と奇数世代であり、画像処理においては隣接する画素データを 2 世代パケットに格納する。2 個のデータはそれぞれ、偶数世代を第 0 データ、奇数世代を第 1 データと呼ぶ。32bit DDMP の入力パケットは 2 世代フラグ操作 (V) によって 2 世代/1 世代の選択が可能となっている。

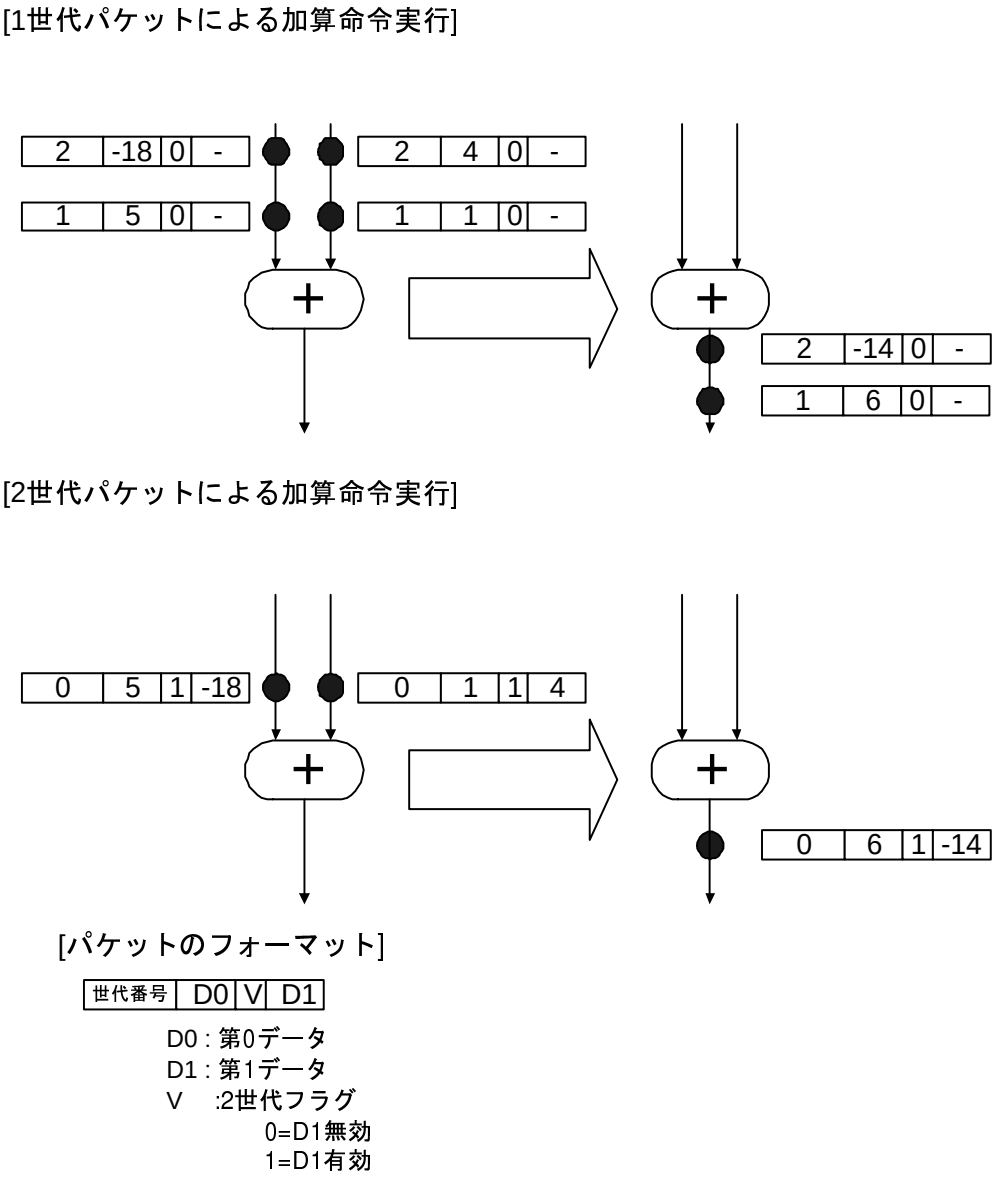


図 4.1　2 世代化

4.1.2　整数型水平 DWT への応用

この 2 世代化を用いて実装することで、入力パケットの量を半減させるだけではなく、フローグラフの命令数を減らすことが可能となる。図 4.2 に 2 世代化を用いたデータフローグラフの一部 (整数型水平 DWT の高域成分出力部) を示す。

入力パケットの D0 領域には偶数世代のデータ、D1 領域には奇数世代のデータを格納す

4.1　2 世代化による処理負荷の軽減

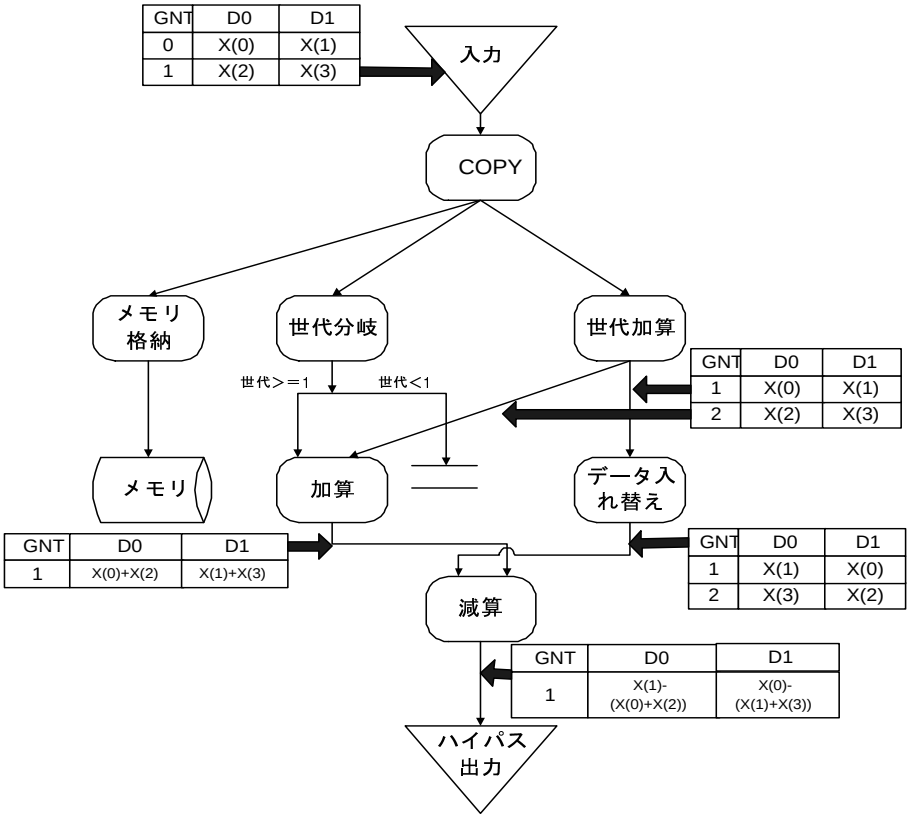


図 4.2　2 世代化を用いた高域出力

る。パケットが入力されると世代分岐命令と世代加算命令に利用するために複製する。世代分岐命令では対称周期拡張のために世代 0 のパケットを廃棄する。世代分岐命令で出力されたデータと、世代加算命令で出力されたデータを加算することで、自分のパケットの次の偶数世代のデータと加算することが可能となる。次に、そのデータと世代加算されたデータの D0 と D1 を入れ替えたデータと減算することで、高域成分がパケットの D0 領域に格納され出力が可能となる。2 世代化を利用した場合、以下のような利点が生まれる。

- 入力パケット量の削減

  2 世代化を行わない場合の入力パケット量の半分の量で実現可能である。これにより総処理時間の短縮が可能となる。

- 命令数の削減

  2 世代化を行い、偶数世代は D0 領域に、奇数世代は D1 領域に格納する。これにより、

データフローグラフ内での偶数/奇数世代の分岐命令の削減が可能となる。

これらの利点によって空間的・時間的並列処理における負荷の軽減を行った。

4.2　条件実行命令拡張子の使用によるメモリ使用量の削減

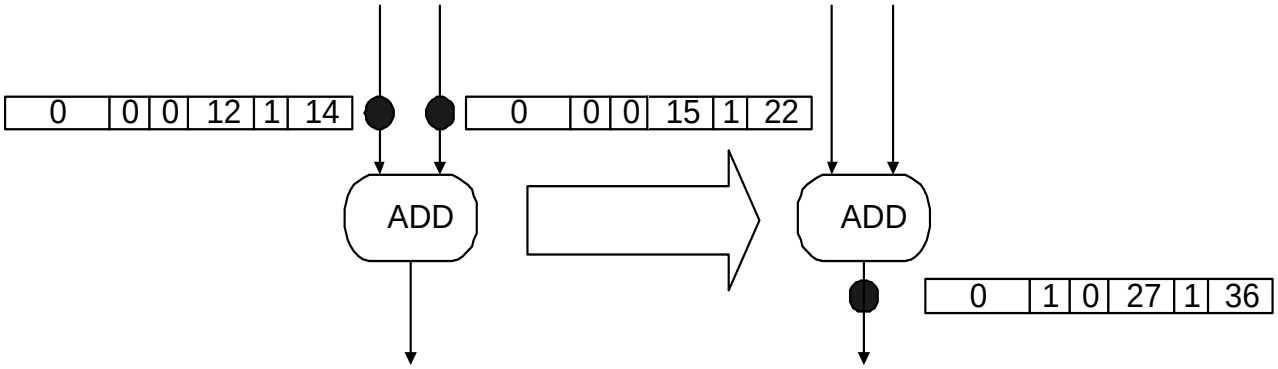
4.2.1　フラグビットによる条件実行

DDMP10-A には命令を記述する際に、条件実行命令拡張子の指定が可能である。条件実行命令拡張子を指定するとフラグビットと組み合わせて、演算の実行/不実行を制御することができる。2 世代化を行っている場合はこの指定によって D0・D1 毎に演算の実行/不実行の制御が可能である。表 4.1 に条件実行命令拡張子とフラグビット (FB) による命令実行/不実行の動作例を示す。また、図 4.3 に具体的な動作を示す。条件実行命令拡張子を使用しない加算実行命令では FB の値に関係なく、第 0 データ、第 1 データ共に加算が実行される。条件実行命令拡張子のひとつである [IFNOT] を使用した場合、第 0 フラグ (FB0) が 0 の時第 0 データに対して演算が実行され、第 1 フラグ (FB1) が 1 の時第 1 データに対しては演算が行われず、左入力の第 1 データが出力される。

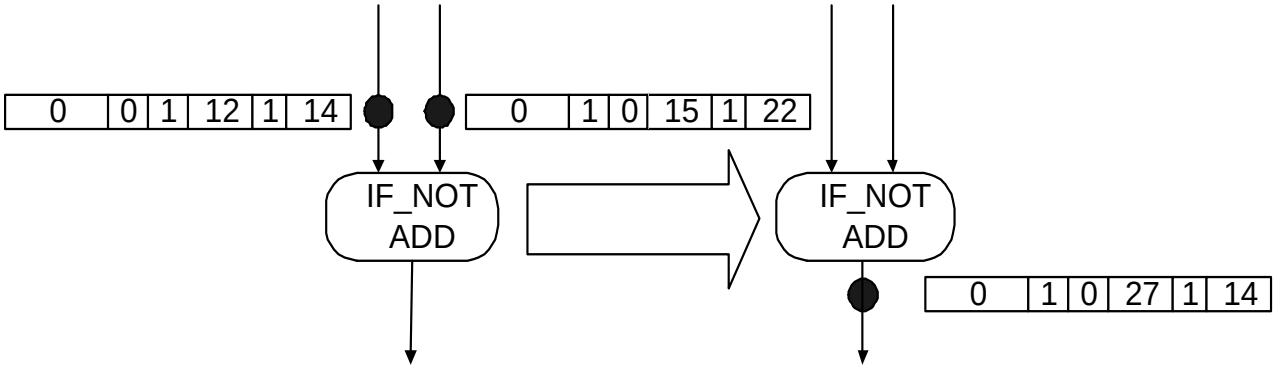
表 4.1　条件実行命令拡張子とフラグビットによる命令の実行/不実行の動作

条件実効命令拡張子	Data0 に関する演算	Data1 に関する演算
指定なし	必ず実行	必ず実行
IFNOT	左 FB[0]=0 のとき実行、 演算結果を出力する 左 FB[0]=1 のとき不実行、 左 Data0 を出力する	左 FB[1]=0 のとき実行、 演算結果を出力する 左 FB[1]=1 のとき不実行、 Data1 を出力する

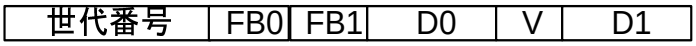
[条件実行命令拡張子を使用しない加算命令実行]



[条件実行命令拡張子を使用する加算命令実行]



[パケットのフォーマット]



D0 : 第0データ
D1 : 第1データ
V　:2世代フラグ
　　0=D1無効
　　1=D1有効
FB0 : 第0フラグ
FB1 : 第1フラグ

図 4.3　条件実行命令拡張子を指定した記述例

4.2.2　整数型水平 DWT のへの応用

2 世代化を用いて高域成分を出力する際、高域成分は D0 領域に格納される。この時、D1 領域には何も格納されないため、これを有効利用できれば、処理負荷を軽減可能となる。図 4.4 に 2 世代化と条件実行命令拡張子の指定を行ったデータフローグラフの一部 (整数型水平 DWT の高域成分出力部) を示す。

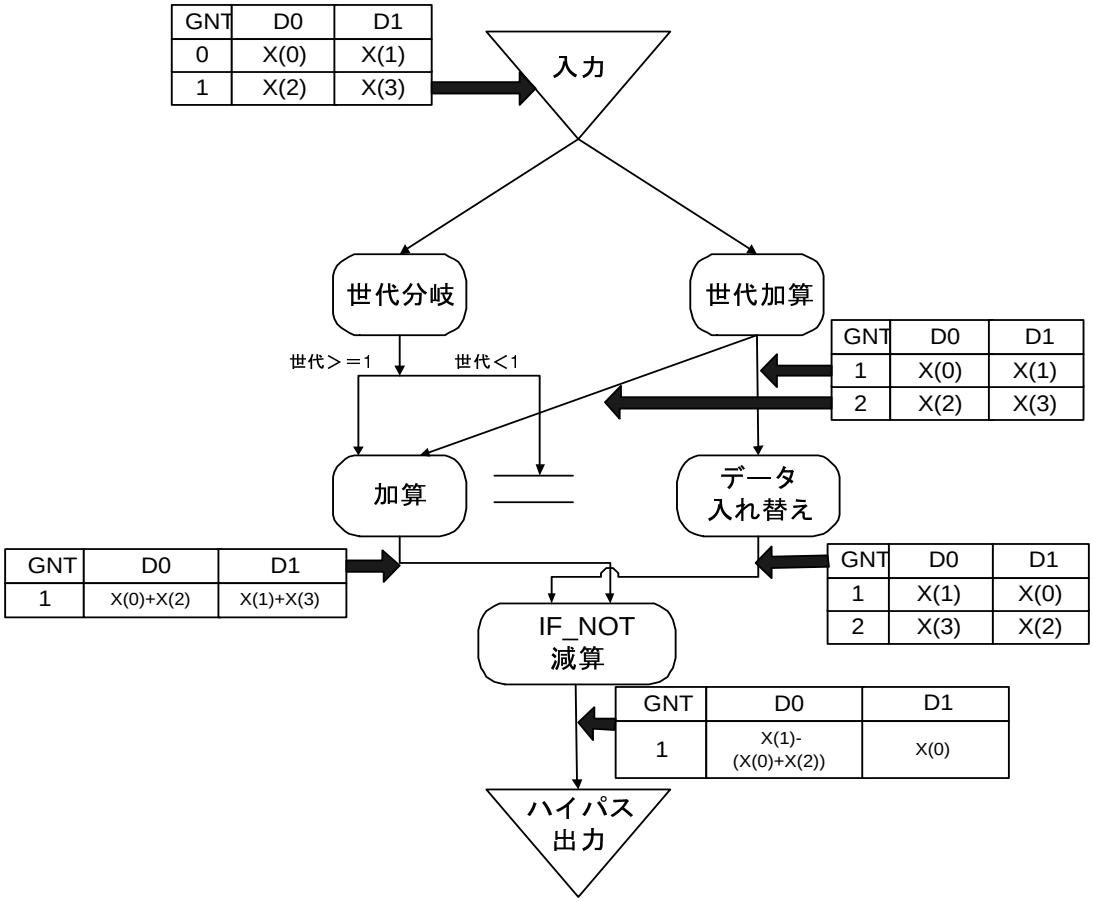


図 4.4　2 世代化と条件実行命令拡張子の指定を用いた高域出力

条件命令実行拡張子を指定した減算を使用することで、高域成分を出力する際に D0 領域には高域成分を D1 には低域成分を演算するために必要な偶数世代のデータを格納する。これにより、データをメモリに書き込む必要がなくなると同時に、命令数の削減が可能となる。

4.3　整数型水平 DWT の作成

図 4.5 はこれまでに述べた解決法を用いた整数型垂直 DWT のデータフローグラフである。まず画像が入力され、入力パケットの D0 領域には偶数世代のデータ、D1 領域には奇数世代のデータを格納する。パケットが入力されると、世代分岐命令と世代加算命令に利用するために複製する。世代分岐命令では対称周期拡張のために世代 0 のパケットを廃棄する。世代分岐命令で出力されたデータと、世代加算命令で出力されたデータの加算で、自分のパケットの次の偶数世代のデータとの加算が可能となる。次に、そのデータと世代加算されたデータの D0 と D1 を入れ替えたデータと減算することで高域成分がパケットの D0 領域に格納され出力が可能となる。この減算で条件実行命令拡張子を指定することで、D1 領域に低域成分の演算に必要な偶数世代のデータの保持が可能となる。

次に出力された高域成分をもとに低域成分を出力する。まず出力された高域成分のパケットを複製する。複製されたデータは自分のパケットのひとつ手前の世代番号のパケットデータと加算するために利用する。世代番号 1 のデータに対してのみ処理が異なるのは対称周期拡張のためである。加算されたデータはさらに 2 を加算した後、自分のパケットの世代番号をアドレスとしてメモリアクセスし、メモリのデータと加算を行う。結果、低域成分が出力される。なお、出力 0 に高域成分を、出力 1 には低域成分を出力する。

本章で述べた手法を用いたデータフローグラフは 3 章で示したものと比較して命令数を 3 命令削減することに成功した。また、クリティカルパス長を 11 命令から 8 命令に削減した。本手法ではメモリの使用なしにすべての処理を実行可能である。

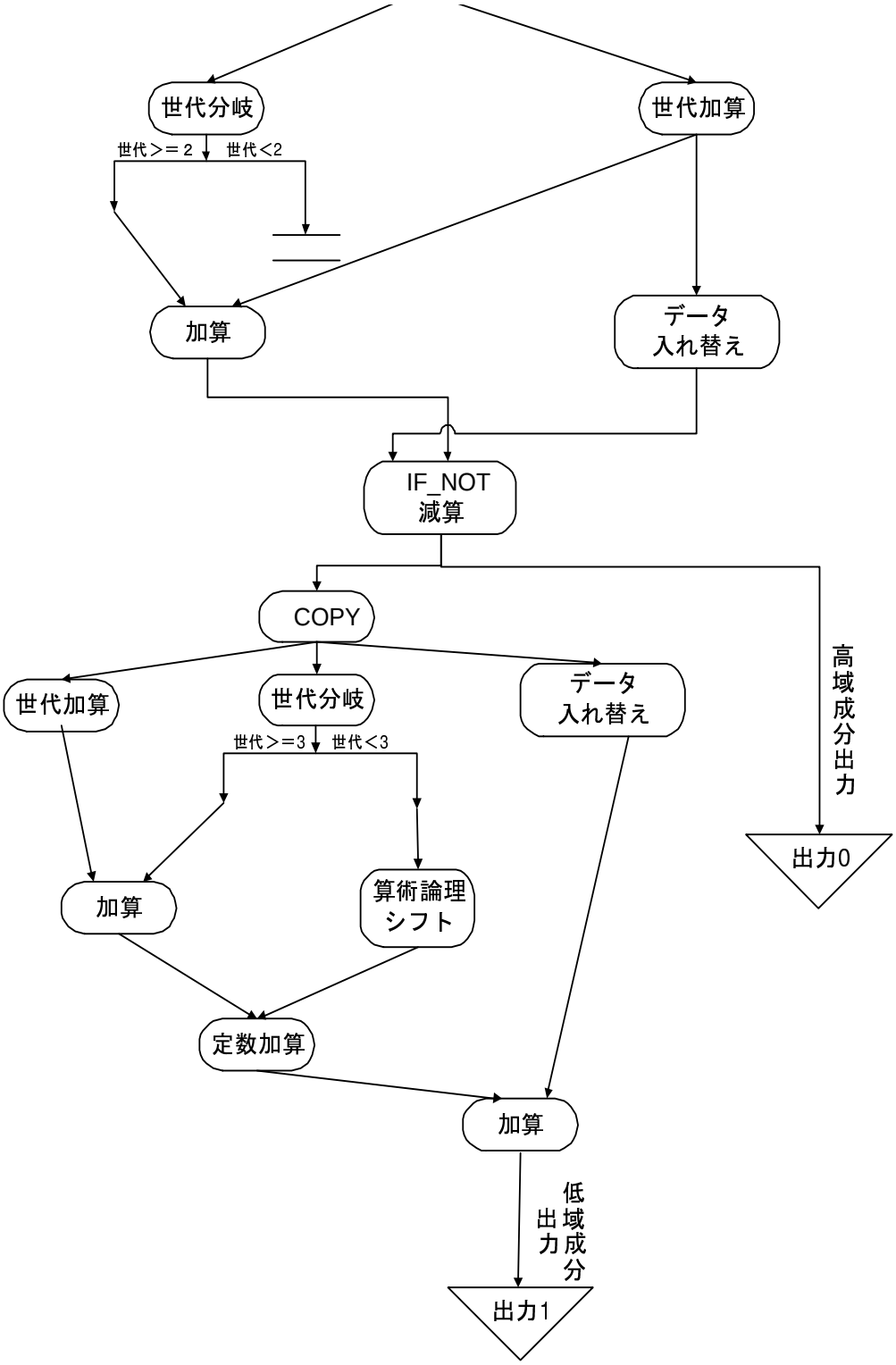


図 4.5　整数型水平 DWT データフローグラフ図 (新方式)

第 5 章

性能評価

5.1 シミュレータの原理

本研究のシミュレーションには、32bit DDMP のフローグラフシミュレータ DADT(DDMP Application Designer's Toolkit) で行う。このシミュレータは DDMP チップの構成を基にした、フローグラフシミュレーションツールである。プログラムは、データの流れるフローグラフを作成し、シミュレータがこれを解析して入力データを演算する。入力データは情報処理対象となる 1 つまたは 2 つのデータと横並びに、データに関するタグ (世代番号、ノード番号、命令コードなど) を持つパケットデータである。本研究では 1 パケットのデータを 2 入力信号のデータとしている。これにより、2 世代化をデータの入力開始時より行える。

5.2 DADT(DDMP Application Designer's Toolkit)

図 5.1 にシミュレーション作業の流れを示す。図中の各作業のついて解説する。

- プログラムソース記述

  プログラムソースファイルの記述を行う。。32bit DDMP のプログラムはノードとアークからなるデータフローグラフで作成され、フローグラフエディタを用いて記述する。

- プログラムアセンブル

  作成したデータフログラフを 32bit DDMP で稼動させたときの各 PE に対する処理負荷を定量的に見積もり、最適化を行う。その後に作成したデータフローグラフをプログ

ラムオブジェクトファイルへ変換する。

- 入力パケットファイル作成

  プログラムに対する入力データパケットの情報をテキストエディタ等を用いて記述する。

- シミュレーション実行

  オブジェクトファイルと入力パケットファイルを指定して実行する。

- シミュレーション実行結果の確認

  シミュレーション実行の結果、出力パケットファイルが作成される。

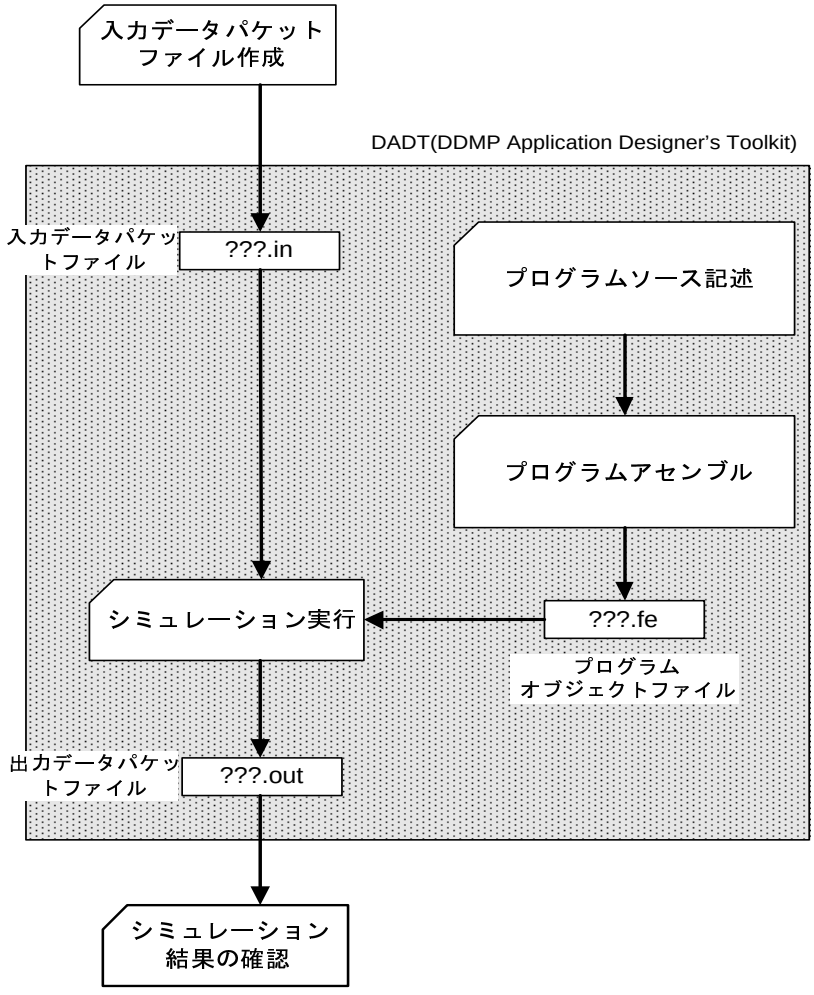


図 5.1 シミュレータ使用時のプログラム開発イメージ

5.3　シミュレーション手法

実験には 256 信号× 8 ラインの入力データを用いた。この入力信号長は性能比較の対象とする DSP の入力信号長と同じサイズである。DDMP は処理の終了した命令については実行されないため、プログラム中の負荷は実行されている命令だけに限定されてしまう。つまり、入力データ量が少ない場合、プログラム中の負荷が最大に達する前に処理が終了してしまう可能性がある。そこで 8 ライン分の入力データを用い、プログラム中の負荷が最大になる状態を作り出す。負荷が最大となり、出力データの間隔が一定になっている状態の 1 ラインの処理時間を適用する。測定方式は以下のものとする。

1. 総命令数

   データフローグラフ中の命令数 (個)。データフローグラフ中の命令数がプログラムの実行速度を左右するため、本研究のプログラム評価指標のひとつとする

2. 処理時間

   255 信号のデータに対して水平整数型 DWT を行うのに必要な時間。

   1 ラインのデータがすべて出力されるまでのサイクル数× 1 サイクルあたりの時間 (単位　μsec)

3. クリティカルパス長

   データフローグラフ中でデータが通過する経路の中で最も長い経路の命令数

5.4　旧方式水平方向整数型 DWT との比較

旧方式水平方向整数型 DWT とは、2 世代化・条件命令実行拡張子の指定を行わずに水平方向整数型 DWT を 32bit DDMP で実装した場合の水平方向整数型 DWT のことである。

比較結果を表 5.1 に示す。改良水平方向整数型 DWT は 2 世代化・条件命令実行拡張子の指定を行い、入力データ量の削減・命令数の削減によるクリティカルパスの短縮化を行った。そのため、データフローグラフ中の入力データの処理時間の大幅な短縮が可能となった。

表 5.1　旧方式との比較

	総命令数	クリティカルパス	処理時間
旧方式水平方向整数型 DWT	16 命令	11 命令	7.8 μ sec
改良水平方向整数型 DWT	13 命令	8 命令	2.7 μ sec

5.5　現行 DSP(Digital Signal Processer) との比較

シミュレーション結果と現行の DSP との処理時間の比較を図 5.2 に示す。比較対象として、Analog Devices 社の DSP「ADSP-21535」、Texas Instruments 社の「TMS320C64x」の処理時間データを用いた。なお、32bit DDMP の動作周波数を 300MHz とし、処理時間を 1.2μ sec とすることで現行 DSP よりも高速に実現可能である。

表 5.2　性能評価 (data size = 256)

	動作周波数	処理時間
32bit DDMP(2 世代化)	140MHz	2.7 μ sec
32bit DDMP(2 世代化)	300MHz	1.2 μ sec
32bit DDMP	140MHz	7.8 μ sec
ADSP-21535	300MHz	2.6 μ sec
TMS320C64x	300MHz	1.7 μ sec

第 6 章

結論

本研究では JPEG2000 の高速実現として、DWT のデータ駆動型実現法について述べた。

第 2 章では JPEG2000 の符号化方法について述べ、その特徴と問題点について触れた。JPEG2000 は高圧縮を行っても高画質を保持可能という大きな特徴があるが、(1)EBCOT の高処理負荷 (2) 離散ウェーブレット変換 (DWT) の必要メモリの増大にともなう高処理負荷といった技術課題が生じている。そのため JPEG2000 の符号化処理は JPEG にくらべ、圧縮で 5 倍〜 6 倍、伸張で 3 倍〜 4 倍の処理能力が必要となっている。

第 3 章では、DWT 処理の概要と内在する並列性を述べ、DWT の具体的な並列アルゴリズムをモデル化した。そして DWT の内在する並列性から、DDMP 処理方式に基づいた水平方向 DWT アルゴリズムを検証し、処理の高速化を測る上で、フローグラフ化の問題点として、メモリ使用による処理負荷、ひとつのパケットの処理時間の遅延、を挙げた。JPEG2000 には整数型 DWT と実数型 DWT の 2 種類が存在する。実数型 DWT は整数型 DWT の応用であるため、整数型 DWT の高速化手法は実数型 DWT の高速化手法としても利用できる。

第 4 章では、水平方向整数型 DWT を DDMP で実装する場合の最適化手法を検証し、(a) 空間的・時間的並列処理における負荷の軽減、(b) メモリ使用量の削減による高速化を提案した。(a) については、隣接する 2 つの画素データを 1 つのパケットに格納する「2 世代化」と呼ばれる処理機構で入力データを処理負荷を軽減する。また (b) については、ノードに条件実行命令拡張子を指定し、必要なメモリ量を削減する。この手法を用いて新たなデータフローグラフを示した。

第 5 章では、提案した手法を用いた水平方向整数型 DWT の比較評価を行った。比較対

象は、提案した手法を用いない水平方向整数型 DWT である。評価項目はデータフローグラフ中の総命令数、クリティカルパス、処理時間である。この評価によって提案手法を用いた場合、大幅な性能向上が可能となる事を証明した。また提案手法を用いた水平方向整数型 DWT と市販の DSP による水平方向整数型 DWT の処理時間比較も行い、DSP よりも高速に実現可能なことを明らかにした。

本研究の成果として、提案手法によって水平方向整数型 DWT を実現した場合、市販される DSP と比較評価すると 1.4 倍高速であることを提示した。また、本提案手法はリフティング構成の実現に有効な手法であり、同じ構成で実現される実数型 DWT にも応用可能といえる。本手法の採用により、JPEG2000 の DWT を DDMP で高速に実現可能なことを明らかにした。

一方、垂直方向 DWT は水平方向 DWT の処理が 3 ライン終了するたびに 2 ライン分の垂直方向 DWT を処理可能である。この並列性の活用により、水平方向 DWT の終了と同時に垂直方向 DWT の処理を終えることができ、さらなる性能向上が見込める。しかし、この並列性を活かすためには、画像データを保存するために多量のメモリ領域が必要となる。32bit DDMP の内蔵メモリは 1 PE あたり 2048 画素しか格納することができないため、メモリ量の不足が問題となる。解決法として、(1) 内蔵メモリの使用 PE 数を増やす、(2) 外部メモリの利用、が想定される。しかし、(1) は最大 20480 画素まで格納可能であるが、PE を複数使用するために命令数を増やさなければならず、これは処理速度の低下を招く恐れがある。一方、(2) は外部メモリは少数の命令で使用可能であり、メモリサイズも 64M word と豊富であるが、メモリアクセス速度が内部メモリに比べて遅いため、メモリアクセスを頻繁に行わないようなアルゴリズムの提案が必要である。

そこで今後の課題として、複数内蔵メモリまたは外部メモリを利用した、整数型 DWT における水平方向/垂直方向処理の並列実現が残されている。また本研究では DWT の基本である整数型のみに着目したが、本手法を応用した実数型水平 DWT のデータ駆動型実現法が考えられる。今回はシミュレーションによる評価のため消費電力での比較は行っていない。本稿で述べたように DDMP は、従来のノイマン型とは異なる独自のアーキテクチャを採用

することで省電力化を実現している。今後、本提案手法を実機で評価し、消費電力における現行 DSP との比較が必要である。

謝辞

本研究に於いて、懇切なる御指導、御鞭撻を賜った 岩田 誠 教授に心より感謝の意を表します。

本研究の基礎としているデータ駆動型アーキテクチャを提唱され、様々なご示唆を賜った寺田 浩詔 教授に心より感謝の意を表します。

本研究の副査を引き受けていただいた 清水 明宏 教授、福本 昌弘 助教授に心より感謝の意を表します。

本研究を進めるにあたり、様々な御助言、御指導を賜った 大森 洋一 助手に心より感謝の意を表します。

日頃から暖かい御支援を頂いた大学院博士課程　林　秀樹　氏に心より感謝の意を表します。

御多忙な中、本研究の御指導、御助言に貴重な時間を費やして頂いた岩田研究室大学院生の 橋本 正和 氏に心から感謝の意を表します。

日頃から御支援、御助言を頂いた岩田研究室大学院生の 別役 宣奉 氏、森川 大智 氏に心から感謝の意を表します。

御多忙な中、多くの御助言、多大なる御支援を頂いた岩田研究室大学院生の 中村 勲二 氏に心から感謝の意を表します。

日頃から暖かい御支援を頂くとともに、多くの御助言を頂いた岩田研究室大学院生の方々、小倉 通寛 氏、三宮 秀次 氏、志摩 浩 氏に心から感謝の意を表します。

日頃から多くの御意見、御支援を頂いた岩田研究室の方々、岩井 秀樹 氏、大石 祐子 氏、西山 直人 氏、宮崎 康徳 氏、山岡 正明 氏に心から感謝の意を表します。

日頃から御意見、御支援を頂いた岩田研究室後輩の方々、朝日山 輝久 氏、大沢 有紀　氏、小笠原 新二 氏、白根 裕太 氏、千頭 裕子 氏、長野 祐介 氏、濱田 康裕 氏、山本 真弘 氏に心から感謝の意を表します。

参考文献

[1] Charilaos Christopoulos, Athanassios Skodras, Touradj Ebrahimi, “The JPEG2000 still image coding sysytem:an overview,” *IEEE Trans. on Consumer Ele.*, 46(4), 1103-1127，2000.

[2] 部流雅茂, “画像処理システムの要素技術,” CQ 出版社, Interface, pp.44-45，Jan.2002.

[3] 貴家仁志, “JPEG2000 の特徴とその機能拡張,” 信学技報,DSP2002-26,May.2002.

[4] H. Terada, S. miyata, and M. Iwata, “DDMP’s: self-timed super-pipelined data-driven multimedia processors,” *Proc. of the IEEE*, 87(2), 282–296，1999.

[5] 貴家仁志, “JPEG2000 を中心とした画像圧縮技術の流れ,” CQ 出版社, Interface, pp.46-58，Jan.2002.

[6] 貴家仁志, 小林弘幸, “整数ウェーブレットの設計とそのリフティング構成,” 信学技報,DSP98-123,Nov.1998.

[7] 貴家仁志, 渡邊修, “JPEG2000 符号化アルゴリズムの要素技術,” CQ 出版社, Interface, pp.59-71，Jan.2002.

[8] 福原隆浩, “JPEG2000/Motion-JPEG2000 の技術概要と応用,” CQ 出版社, Interface, pp.131-137，Nov.2002.

[9] 野水泰之, “JPEG2000 最新動向,” 画像電子学会誌, VOl.30,No.2,pp.167-175,2001.

[10] 中野宏毅, 山本鎭男, 吉田靖男, “ウェーブレットによる信号処理と画像処理,” 共立出版株式会社 (1999)

[11] 橋本正和, 岩田誠 “形状適応 DCT のデータ駆動型並列実現法,” 高知工科大学　情報システム工学科　学士学位論文 (2000)

[12] 中村勲二, 岩田誠 “ピクセルベース動画圧縮方式とそのデータ駆動型実現法,” 高知工科大学　情報システム工学科　学士学位論文 (2001)