社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2004－SLDM－116 (15)
2004／10／22

VGA動画に対応した小規模JPEG 2000コーデックコア

萩谷 太郎† 中村 創‡ 玄馬 哲‡ 桑原 隆司‡ 中山 寛†

† (株)富士通研究所 〒211-8588 神奈川県川崎市中原区上小田中 4-1-1
‡ 富士通デバイス(株) 〒224-0046 神奈川県横浜市都筑区桜並木 1-1

E-mail: † {hagiya, h.nakayama}@jp.fujitsu.com, ‡ {nakamura, genba, kuwahara}@fdi.fujitsu.com

あらまし 本稿では、回路規模の縮小とSDRAMアクセスの低減を図ったJPEG 2000コーデックコアを紹介する。本コーデックコアは、ウエーブレット変換部での乗算器の削減により回路を縮小し、画像劣化の生じない矩形領域分割を施した演算により外部SDRAMアクセスを大幅に低減した。さらにエントロピ演算器においては、コンテクストの算出と、MQ-CODERの演算をそれぞれ並列化することにより高速化を図った。本コアでは、SDRAMへのアクセス量をDWTの演算結果全て出力するものに比べ1/6程度に抑えた。さらに、演算速度向上によりVGA(640×480画素)動画を毎秒30フレームで符号化または復号可能とし、近年要求の高まっているMotion JPEG 2000にも対応した。

キーワード JPEG 2000, IP, VGA動画, 9×7フィルタ

JPEG 2000 codec IP with reduced circuit volume for VGA-video

Taro HAGIYA† Sou NAKAMURA‡ Akira GENBA‡ Takashi KUWAHARA‡

and Hiroshi NAKAYAMA†

† Fujitsu laboratories ltd. 4-1-1 Kamiodanaka, Nakahara-ku, Kawasaki, 211-8588 Japan
‡ Fujitsu devices inc. 1-1 Sakuranamiki, Tsuzuki-ki, Yokohama, 224-0046 Japan

E-mail: † {hagiya, h.nakayama}@jp.fujitsu.com, ‡ {nakamura, genba, kuwahara}@fdi.fujitsu.com

Abstract In this paper, we'll describe how we made the circuit volume decreased and computation time shorter to develop a low-power and high performance JPEG 2000 LSI. In DWT logic, the number of multiplier decreased by optimizing computational algorithm of the DWT and the quantity of SDRAM access decreased by means of limited size computation without tile boundary artifact. And also we made entropy computation optimized to realize high-speed codec. As the result, its SDRAM access is reduced to 1/6, and it is possible to encode or decode VGA (640x480 pixel) video 30 frames per second suitable for Motion JPEG 2000.

Keyword JPEG 2000, IP, VGA-video, 9x7 filter

1. JPEG 2000 概論

1.1. JPEG 2000 の利点

JPEG 2000[1]はISO/IECで標準化されている画像圧縮方式の一つである。JPEG,MPEGに代表される画像圧縮方式は、独立した矩形領域ごとに符号化を行うために、領域境界で歪みが発生しやすい。それに対しJPEG 2000は、画像全体に対してDWT (Discrete Wavelet Transform : 離散ウエーブレット変換)を適用するため、歪みが生じにくく圧縮効率が高いことで知られている。

また、圧縮されたデータのうち一部のデータのみを利用して、サイズ・質の異なる画像を再生できるため、異なる再生環境を持つメディアでも1つのデータから再生可能であるという特徴を持つ他、ウエーブレット係数にスクランブルを掛けることによる暗号化が可能であるなど、JPEGなどにはない特徴を持つ。

さらに、JPEG 2000によって圧縮された静止画を連続的に表示することにより実現されるMotion JPEG 2000は、MPEG, H264などの時間依存性を利用した圧縮形式に比べ圧縮効率は下がるものの、編集の容易さから動画の編集が必要とされる分野での利用が期待されている。

1.2. JPEG 2000 実装の難しさ

数々の利点を持ちながらも、現状としてJPEG 2000は広く使用されているとは言い難い。その大きな理由として、実装の難しさがある。JPEG 2000の実装を困難にしている原因として次のことが挙げられる。

・ 演算量
演算量はJPEGの20倍とも言われ、LSI実装の際には多くの演算器が必要となる。
・ 演算時間
エントロピ演算器においては、直前のビットの演算結果が次のビットの演算に影響を及ぼすため並列処理が難しく、多くの処理時間が必要となる。
・ 記憶領域
DWT演算において画像全体に対して2次元フィルタを適用すること、エントロピ符号化・復号がビットスライス毎の処理であることに起因し、多くの中間データを記憶する領域が必要となる。

これらの理由から、JPEG 2000のLSI実装は、回路規模が増大し、コスト・消費電力の増大につながっていた。

2. LSI化のための改良

これらの問題が生じる理由の詳細と、本稿で紹介するコーデックコアではそれらの問題をどのように解決したかを以下に示す。

2.1. アーキテクチャ

まず、JPEG 2000コーデックコアのアーキテクチャを示す。

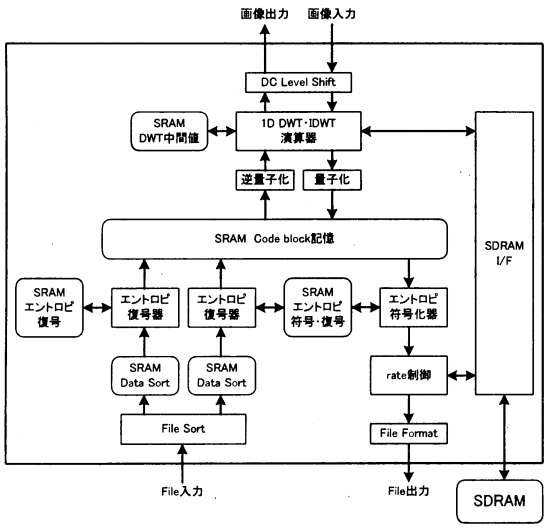


図 1 JPEG2000 LSIアーキテクチャ

図 1の上から下の方向が符号化処理の流れであり、下から上の方向が復号処理の流れである。回路縮小のため多くの演算器・内部RAM(SRAM)を符号化・復号の演算処理の中で共有化している。

本稿では、JPEG 2000回路の効率化のための要素技術として以下のものを順に紹介する。

・ DWT演算器の乗算器削減
・ DWT演算時のSDRAMアクセス低減
・ エントロピ符号化の高速化
・ エントロピ復号の高速化

2.2. DWT演算器

2.2.1. DWT演算器の乗算器削減

DWT演算器は、JPEG 2000の中でも最も演算量が多い。さらに、DWT演算器として、ビットシフトと加算器のみで演算可能な5×3フィルタより、実数の乗算が必要な9×7フィルタを使用した方が、圧縮効率が高いことが知られている。ところが、この9×7フィルタを単純に回路に落とすと、次のような演算フローになる。

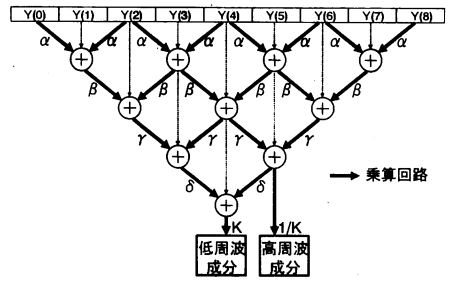


図 2 従来の9×7フィルタ演算フロー

図中の太線が乗算を表し、脇に書かれている文字は乗数(実数)を表す。

低周波成分と高周波成分を1つずつ出力するためには、同じ乗算を共通化しても、16回の乗算が必要となる。既に発表されているLSIの1つ[3]ではこの乗算器を削減しているが、この回路でも12個の乗算器が存在する。

図 2中の演算は入力画像データYが2画素ずつシフトしながら連続的に行われる。その時、同じ乗算を連続的に複数回行っていることに着目すると、一度計算された乗算結果を、隣の画素に対する演算に対して再度利用することが可能である。この考えをαからγの4つの乗算器に対して同様に適用することにより、図3の回路で、図 2の9×7フィルタの演算器と全く同じ演算結果が得られる。

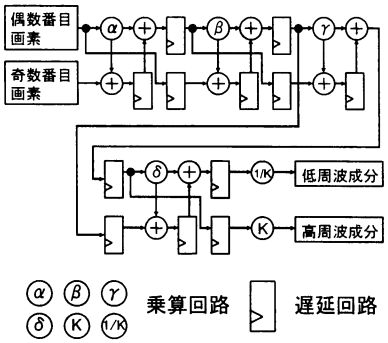


図 3　乗算器削減 9×7 フィルタ回路

この回路中には乗算器は 6 個しか存在せず、回路的に大きな乗算器を大幅に削減した。

IDWT (Inverse DWT : 逆ウエーブレット変換)に関しても同様に 6 個の乗算器から構成した。さらに、DWT と IDWT では使用する乗数が同じであるため、乗算器を共有した。この結果、コーデック全体で乗算器は 6 個のみである。

2.2.2. 外部 SDRAM アクセス低減演算方法

DWT の結果は、複数回 2 次元フィルタをかけることによって算出される。まず 1 段階目として、画面全体に垂直方向にフィルタをかけ、次に水平方向にフィルタをかける。その後 2 段階目として、1 段階目で低周波成分として出力された部分に対して再度、垂直・水平の 2 次元フィルタを適用する。これを 5 回ほど繰り返すことによって DWT の演算が終了する。

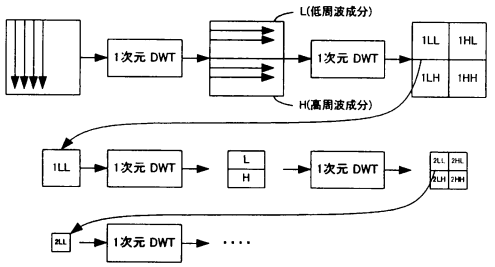


図 4　DWT の演算手順

つまり、DWT の演算結果、および演算中間データを演算する都度、全て外部の SDRAM へ出力すると、画面サイズの数倍のデータを読み書きする必要があり、外部 SDRAM へのアクセスが膨大な量になってしまう。

本コーデックコアでは、SDRAM へのアクセスを低減するために、DWT を矩形領域に分割して効率的に内部 RAM を利用して演算を行えるようにした。以下にその演算手順を示す。

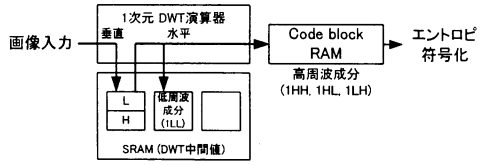


図 5　DWT 演算(1 段階目)

まず、外部から入力されたデータを垂直方向にフィルタをかけ、中間値記憶用の内部 RAM に格納する。その後、内部 RAM から読み出し、水平方向フィルタをかけ高周波成分(1HH, 1HL, 1LH)は内部の Code block RAM へ書き込み、そのまま SDRAM に出すことなくエントロピ符号演算を行う。低周波成分(1LL)は、中間値記憶 RAM に格納する。

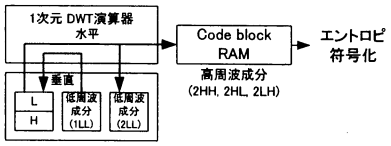


図 6　DWT 演算(2 段階目)

2 段階目として、内部に低周波成分(1LL)が十分に蓄積された後、低周波成分(1LL)に対して垂直、水平方向にフィルタを適用し、低周波成分(2LL)はやはり内部 RAM に記憶し、高周波成分(2HH, 2HL, 2LH)は Code Block RAM に記憶され、そのままエントロピ符号化演算を行う。

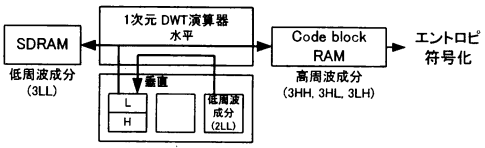


図 7　DWT 演算(3 段階目)

3 段階目も同様に、内部に蓄積された低周波成分(2LL)に対して垂直、水平方向にフィルタをかけるが、低周波成分(3LL)は外部の SDRAM へ出力する。SDRAM に出力する理由は、3 段階目の低周波成分(3LL)のデータ量は少なく、3 段目用の内部 RAM を増設したとしても、それよって削減できる SDRAM アクセスは少ないからである。さらに、4 段階目以降も同様に演算結果を SDRAM に出力することにより、DWT 演算の段数がいくつの場合であっても、同様の演算手順で演算可能であるからである。

2.2.3. DWT 中間演算データ記憶

2.2.2で示した手法の様に、演算を矩形領域ごとに分割して演算を行うことにより SDRAM アクセスを低減することが可能である。ところが JPEG などと同様に、

矩形領域で独立して演算を行うと領域境界で歪が発生する。これを回避するために、次の様な手順で演算を行う。

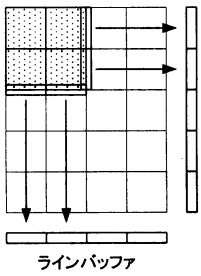


図 8　DWT 中間データ保存

領域境界で演算途中の値をラインバッファに記憶し、次に隣の領域を演算するときに記憶していた中間値を利用して演算を続ける。この演算方法によって画像が劣化することはない。さらに、演算量・演算結果共に矩形領域ごとに演算を区切らなかった場合と全く変わらない。

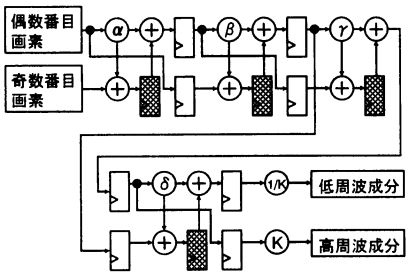


図 9　DWT 中間データ記憶位置

このブロック分割演算は、図 9の演算器遅延回路のうち、網掛けのデータを中間データとして、ラインバッファに記憶することにより実現できる。

2.3. エントロピ演算器

JPEG 2000 の核となる技術として、DWT とエントロピ演算がある。DWT は多くの乗算器が存在し演算回路が多かったのに対し、エントロピ演算器中の演算器回路は DWT 演算器ほど多く必要としない。しかし、エントロピ演算器の処理は 1 ビットずつ行われるため、処理時間が長くなってしまう問題がある。

以下に、高速化のために並列化したエントロピ演算器の演算手順を示す。

2.3.1. エントロピ符号化器の並列化

JPEG 2000 で採用されているエントロピ演算は、コンテクスト依存の算術符号化を行っている。エントロピ演算の第 1 ステップとして、ビットモデリング演算を行い、それぞれのビットに対するコンテクストを算出する。このビットモデリング演算の処理単位は図 10 に示すように、1 つの画素を 1 ビットずつ分解し、同じ深さのビットを集めたビットプレーンである。さらに、そのビットプレーンも 1 ビットずつ、3 つのパス (SP パス, MR パス, C パス)のいずれかに分類され、それぞれのパスごとに処理が行われるために、通常の 1 ビットずつ行う処理では非常に時間がかかる。

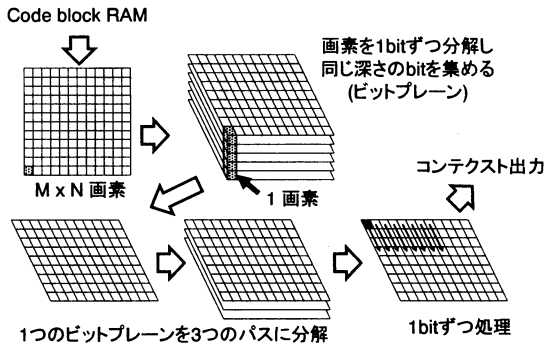


図 10　ビットモデリング演算手順

このビットモデリング演算を高速化するために、図 11に示すように 3 つのパスにおける 4bit を同時に処理する。これにより、各パスに属するビット数に大きく左右されることなく安定してコンテクストを出力することが可能となった。

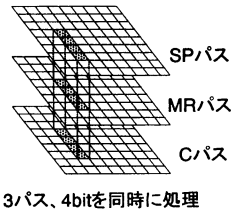


図 11　ビットモデリング並列処理

ところがこの演算方法は高速である反面、3 つのパスのデータを同時に出力することになり、各パスのコンテクストを記憶する必要性が発生する。なぜなら、ビットモデリングの後の演算である MQ-CODER は、1 つのビットプレーン中の SP パスの演算が全て終わってから MR パスの演算に移り、MR パスの演算が全て終わってから C パスの演算に移るからである。図 12 のように単純に 3 つのパスそれぞれに内部 RAM を用意し、パスごとにコンテクストを振り分けて書き込み、読み込み時は SP→MR→C の順番に RAM を切り替えることによっても実現可能である。しかしながら、それぞれのパスに属するコンテクストの数は、コードブロック・処理ビットプレーンによって可変であるため、

それぞれの RAM の容量は最大になるときに合わせて用意しなければならない。

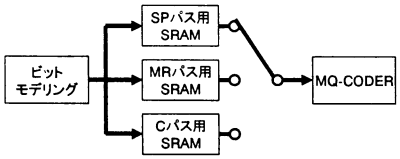


図 12 ビットモデリング後の RAM

このままでは容量が大きいので、3 つに分かれている内部 RAM を一つに纏めたいが、それぞれのパスに属するビット数は浮動であり、1 つの RAM をパスごとに記憶領域を分割して使用することは困難である。しかし、3つのパスのうちCパスとSPパスのビット数は、現在処理しているビットプレーンの処理結果に依存するのに対し、MR パスに属するビットの数だけは、前のビットプレーンの処理が終わった時点で明らかになる。この点に着目し、図 13のように"MR パス領域"と、"SP,C パス共通領域"に分割し、SP パスと C パスは両端から書き込むような構成にする。また内部 RAM のサイズは 1 ビットプレーンのコンテクスト数が最大となるときのコンテクストが記憶できる容量にする。この構成では、パスが SP パスと C パスのコンテクストの数の合計は共通領域を超えることはないので、2 つのデータ領域が重なることはない。このように書き込むことにより、内部 RAM の容量を約 1/3 に抑えることができる。

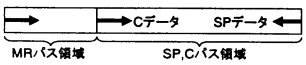


図 13 コンテクスト記憶領域(書き込み)

この手法で記憶した場合の読み込む順序を図 14に示す。まず右端から SP パスの領域を読み込み、次に左端から MR パス、C パスと読み込みこむことにより、MQ-CODER に必要な順序である SP→MR→C の順にデータを取り出すことが可能である。

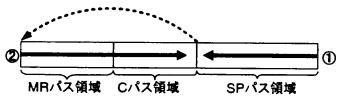


図 14 コンテクスト記憶領域(読み込み)

2.3.2. エントロピ復号器の高速化

エントロピ復号の演算は、符号化の演算よりさらに処理の高速化が困難である。図 15に示すように、周辺のビット情報から 1 つのコンテクストが決定し、そのコンテクストを利用して MQ-CODER によって 0,1 のビット(シンボル)が復号される。その復号されたシンボルを利用し始めて次のビットにおけるコンテクストが決定する。したがって、符号化演算で行ったように、あらかじめまとめてコンテクストを出力し、並列でシンボルを復号することが困難である。

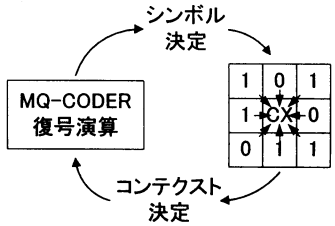
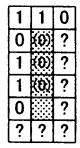


図 15 エントロピ復号演算手順

それを解決するために次のような演算を行う。

図 16は、網掛けの部分の 4 つのビットにおけるコンテクストを同時に求める例である。4 つのビットのうち、一番上のビットにおけるコンテクストは確定するが、それ以外のビットにおけるコンテクストは上のビットの復号結果が未定であるため確定しないことがある。上のビットの値が未定の時には、復号結果が '0' であると仮定して通常通りにコンテクスト演算を行う。このとき、上が'0'であると仮定して出力されたコンテクストは予測コンテクストとして扱う。



0,1 : 確定済
(0) : 0と仮定
? : 未定

図 16 予測コンテクスト算出

このようにして、出力された(確定した)コンテクストと予測コンテクストを連続的に MQ-CODER に入力する。確定しているコンテクストはそのまま演算を行えば問題がないが、予測コンテクストの場合では上の復号結果によって、コンテクストを変更する必要がある。上のビットの復号結果が '0'であるときには、予測コンテクストの予測はそのまま正しいコンテスクトとして利用できるが、'1'であるときには次のコンテクスト変換テーブルによって変換する必要がある。

表 1　予測コンテクスト変換テーブル

LL, LH subband		
pCX		CX
8	→	8
7	→	7
6	→	7
5	→	7
4	→	-
3	→	4
2	→	3
1	→	3
0	→	3

HL subband		
pCX		CX
8	→	-
7	→	8
6	→	8
5	→	8
4	→	7
3	→	7
2	→	6
1	→	6
0	→	5

HH subband		
pCX		CX
8	→	8
7	→	7
6	→	7
5	→	5
4	→	5
3	→	4
2	→	2
1	→	2
0	→	1

表の左側の値(pCX)は予測コンテクストであり、上のビットの復号結果が'1'であったときには、コンテクストを表の右側の値(CX)に変換することによって正しいコンテクストを得ることができる。表中の網掛けの部分は予測コンテクストとして出力されない値である。

この表は、予測コンテクストから確定したコンテクストへの変換は一意に変わる単純なものであり、上のビットが確定してから再度コンテクスト算出するビットモデリング演算器より遥かに単純であることを示している。従来シンボルが確定してから1クロックサイクルかけて次のビットに対するコンテクスト演算していたが、このコンテクスト変換テーブルを利用することにより、シンボルが決定したのと同じサイクルに次のビットにおけるコンテクストを確定することが可能になり、コンテクスト算出のために待たされる時間が短縮される。

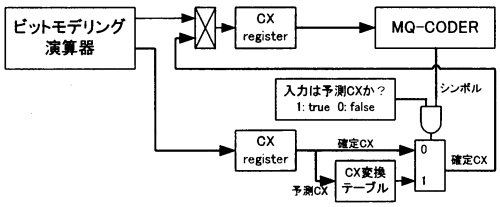


図 17　コンテクスト変換テーブルを利用した回路

図 17はビットモデリング演算器から 2 つのコンテスクトが同時に出力される、コンテスクト変換テーブルを利用した回路の例である。2 つ目のコンテスクトは確定していないことがあるので、上のビットの復号結果によって、適宜変換する必要がある。

3. まとめ

このコア性能は次の表に示す通りである。

表 2　コーデックコア性能

画像サイズ	≦4000×2000
処理フレーム数	30 フレーム/秒 (VGA 画像処理時)
動作周波数	54MHz
消費電力	400mW (推定)
ゲート規模	650K BC (推定)
タイルサイズ	≦4000×2000
DWT フィルタ	9×7 フィルタ
Code Block サイズ	≦32×32

この JPEG 2000 コーデックコアの性能は、近年高く要求の高まっている Motion JPEG 2000 をターゲットとしている。

また、本コアの最大の特徴は、SDRAM へのアクセス量を大幅に削減したことである。DWT の演算結果、および演算途中のデータを全て SDRAM に書き出すことをすれば、画像と同じか、それ以上のデータを SDRAM に書き込み、エントロピ符号化の前に読みこむことになる。これを、直接エントロピにデータを渡すことにしたために、SDRAM アクセスは、レート制御にかかわるアクセスも含めて約 1/6 に押さえている。

文　献

[1] ISO/IEC 15444-1 Information technology - JPEG 2000 image coding system - Part1: Core coding system, 2000.

[2] 野水泰之, "次世代符号化方式 JPEG2000", トリケップス, 2001.

[3] H.Yamauchi et al., "A 1440x1080 Pixels 30Frames/s Motion-JPEG2000 Codec for HD Movie Transmission," ISSCC Dig. Tech. Papers, San Francisco, Feb. 2004.