レート歪み最適化による量子化プロセスのハードウェア化に関する一検討

森口 元気† 神戸 尚志‡ 藤田 玄‡‡

†近畿大学大学院 総合理工学研究科
‡近畿大学 理工学部 電気電子工学科
‡‡大阪電気通信大学 情報通信工学部

概要

フル HD を超える 4K/8K の超高精細テレビの登場により，動画圧縮技術の高圧縮化が求められている．H.264 および H.265 の標準化参照ソフトウェア (JM，HM) で用いられている量子化手法，レート歪み最適化による量子化 (RDOQ) は，圧縮率と量子化歪を最適化し符号化効率を向上させる有効な手法である．しかし，この手法は，依存性の高いデータの使用，演算量の増大と言った点からハードウェアへの実装が困難とされている．本論文では，ハードウェア向けに改良した RDOQ アルゴリズムに対する，C 言語ベースの回路合成システムである BachC を用いた設計と，パイプライン化等による高速化について述べる．

A Hardware Implementation of Rate-Distortion Optimized Quantization

Genki Moriguchi† Takashi Kambe‡ Gen Fujita‡‡

†Graduate School of Science and Engineering Research, Kinki University
‡Department of E & E, Faculty of Science and Engineering, Kinki University
‡‡Faculty of Information and Communication Engineering, Osaka Electro-Communication University

Abstract

With the advent of ultra-high-definition television(UHDTV) exceeding full HD, high compression methods of moving image coding technology have been required. Rate-distortion optimized quantization(RDOQ) is one of the most important technology in H.264/AVC and H.265/HEVC reference software(JM,HM) for improving encoding efficiency. However, hardware implementation of RDOQ algorithm is difficult due to problems of calculation load and high data dependency. We implement RDOQ algorithm to hardware using high-level synthesis tool BachC and apply parallelization and pipelining to accelerate it.

1 はじめに

現在，フル HD を超える 4K/8K の超高精細テレビの登場で，映像データ容量が膨大になり，動画圧縮符号化技術による高圧縮化が求められている．圧縮率向上のために予測処理や CABAC と言った手法が適用されてきたが，それによる演算量の増大がリアルタイム処理の実現における問題となっている．

本論文では，代表的な動画圧縮技術である H.264/AVC[1] で適用されている高圧縮化・高画質化の手法の一つである RDOQ を対象とし，高位合成ツールである Bach システム [2] を用いてハードウェア実装を行い，その高速化を図る．ハードウェア実装する RDOQ アルゴリズムは，筆者等が文献 [3] で改良を加えたものを用いる．

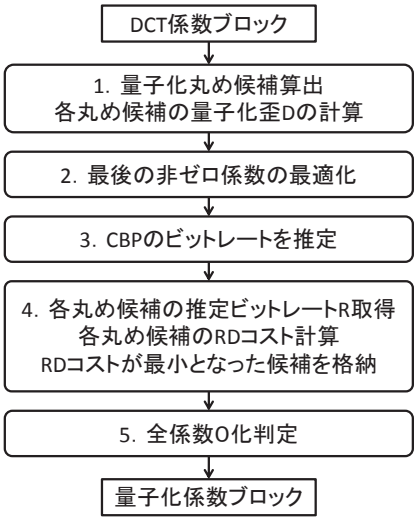

図 1: JM18.0 の RDOQ 処理フロー

2 レート歪み最適化による量子化 (RDOQ)

H.264 の参照ソフトウェア JM18.0 [4] における量子化処理にはレート歪み最適化手法 (以下 RDOQ)[5, 6] が導入されているが，実際にチップに実装された報告はまだない．RDOQ では処理ブロック内のレート歪みコスト (以下 RD コスト) の合計が最小となるように複数の丸め候補の中から量子化値を決定する．RDOQ の処理は，(1) 丸め候補の計算・各丸め候補の歪 D の計算，(2) 最後の非ゼロ係数の最適化，(3)CBP のビットレート推定，(4) 各丸め候補の推定ビットレート R の取得および RD コスト計算，(5) 全係数 0 化判定，の 5 つの手法を用いている (図 1)．

(1) の処理では，DCT 係数と量子化ステップから 3 つの丸め候補 (ゼロ (l_i^0)，切り下げ丸め値 (l_i^{floor})，切り上げ丸め値 (l_i^{ceil})) を算出し，各候補の量子化歪みを計算する．(2) の処理では，ジグザグシーケンスの順で最後に位置する非ゼロ係数 (以下，LNZ 係数) の位置を RD コストが最小となる様に最適化する．(3) の処理では，CBP(Coded Block Pattern) と呼ばれるフラグのビットレートの推定値を取得する．(4) の処理では，各丸め候補のビットレートの推定値 R を取得し，RD コスト計算 (式 (1)) を行う．最後に (5) の処理では，最適化された RD コスト合計値と全係数が 0 の場合の RD コスト合計を比較し，全係数を 0 にするか判定する．

$$J_i^j = err_i^j + \lambda \times bits_i^j \qquad (1)$$

ここで，添え字 i はジグザグシーケンス上の DCT 係数位置，j は丸め候補 (0,floor,ceil) を示す．RDOQ は従来の量子化に比べ，より高い符号化効率を得ることが可能であるが，量子化係数毎に最適な量子化値を探索するため，演算量が多く，計算複雑度が高い．また，ビットレート取得における係数間の依存性は，ハードウェア設計における並列処理による高速化を困難にしている．

3 RDOQ のハードウェア設計

本章では RDOQ のハードウェア設計について述べる．本研究では，高位合成ツールの一つである Bach システムを用いて，文献 [3] で提案した RDOQ 高速化アルゴリズム (以下提案 RDOQ) をハードウェア実装し，その高速化を行う．ここで，RDOQ への入力データはオンチップの RAM に置く．また，本研究で設計する RDOQ の回路は 4x4 ブロックの処理を対象とする．図 2 に設計後の提案 RDOQ のブロック図と提案手法との対応を示す．

提案 RDOQ では，並列処理のサポート，演算回数の削減等，ハードウェア化に適した手法を提案している．従来の RDOQ では困難であったが，提案 RDOQ により可能となったハードウェア高速化手法として，3.1 節では RD コスト計算の並列化，3.2 節では丸め候補計算・歪計算の並列化について述べる．これらの並列化により，RDOQ でボトルネックである全係数の量子化歪み・RD コスト計算の逐次処理を高速に行うことが可能になる．また，3.3 節では LNZ 係数探索処理のループパイプライン化について述べる．このループパイプライン化では，LNZ 係数の最適化処理の逐次処理を高速に行うことを目的としている．さらに，3.4 節では 1 サイクルで LNZ 係数探索範囲決定処理を実現するプライオリティエンコード化について，3.5 節では，RDOQ を複数ステージに分け，同期通信を用いたパイプラインを行う同期パイプライン化について述べる．

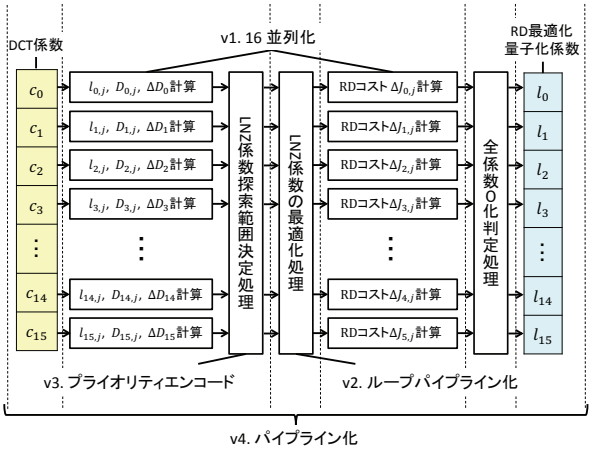

図 2: 提案 RDOQ のブロック図

3.1 RD コスト計算の並列化

本節ではまず，RD コスト計算の並列化について述べる．RD コスト計算では，量子化歪み D に対し，R は推定ビットレートを用いる．量子化係数のビットレートは，significant_coeff_flag，last_significant_ coeff_flag，coeff_greater_one_flag，coeff_abs_level_minus2 の主に 4 つ要素に関連している．ここで，coeff_greater_one_flag，coeff_abs_level_minus2 とは，coeff_abs_level_minus1 を 2 つに分割して考えた要素であり，coeff_greater_one_flag は，量子化レベル l_i の絶対値 $|l_i|$ が 1 か否かを示すフラグ，coeff_abs_level_minus2 は，量子化レベル $|l_i|$ から 2 を引いた値である．この 4 つのシンタックス要素のビットレートを推定し，それぞれを合計した値が R となる (式 (2))．

$$
\begin{aligned}
&if(significant\_coeff\_flag == 0)\ Bits_i = B_{sc} \\
&else\ if(l_i < 2)\ Bits_i = B_{sc} + B_{ls} + B_{cg} \\
&else\ Bits_i = B_{sc} + B_{ls} + B_{cg} + (l_i - 2) \times B^1_{ca} + B^0_{ca}
\end{aligned} \tag{2}
$$

しかし，4 種のシンタックス要素の内，coeff_greater_one_flag，coeff_abs_level_minus2 は一つ前の係数の同シンタックス要素の値に依存している．そのため，RD コスト計算は全係数を逐次で行う必要があった．提案 RDOQ ではマクロブロックあたりで使用する推定ビットレートをシンタックス要素毎に統一することで，依存関係の解消と推定ビットレートの計算量・データ量の削減を行なう．これにより，推定ビットレート R をブロック内の全係数で共有でき，依存関係もないため RD コスト計算の並列化が可能となる．

3.2 丸め候補計算・歪計算の並列化

次に，丸め候補計算・歪計算の並列化について述べる．丸め候補計算・歪計算時に LNZ 係数の探索範囲を決めるレベル (noLevels=1,2,3) 分け処理が存在する．この時，ジグザグスキャン順で最後に出てくる $noLevels = 3$ からそれ以降の $noLevels = 2$ の間を探索範囲とするが，処理係数が LNZ 係数の探索範囲であるかどうかのフラグ管理の際に係数間に依存関係が発生する．このため，丸め候補計算・歪計算を全係数で逐次に行う必要があり，RDOQ のボトルネックとなっていた．図 3 に LNZ 係数の探索範囲の例を示す．

本研究では，この係数間の依存関係を次のようにアルゴリズムを変更して解消する．丸め候補を算出した際に，noLevels=2 であるか否かを示す Start フラグおよび noLevels=3 であるか否かを示す Stop フラグの 2 つのフラグを係数毎に与えておき，処理ブロックの歪計算終了後に，逆ジグザグシーケンス順で各係数の Start フラグおよび Stop フラグの値をビット連結する．そのフラグを用いて探索範囲の開始位置，終了位置を決定する．このアルゴリズムの変更によって丸め候補計算・歪計算時における係数間の依存が解消されるため，処理ブロックの全係数における丸め候補計算・歪計算の並列処理による高速化が可能となる．

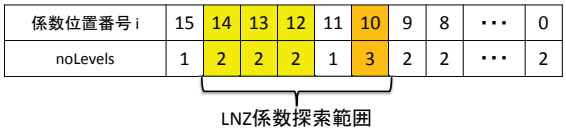

図 3: LNZ 係数の探索範囲の例

3.3 LNZ 係数探索処理のループパイプライン化

次に，LNZ 係数探索処理のループパイプライン化について述べる．LNZ 係数の最適化処理は，探索範囲内の各係数を LNZ 係数であると仮定し，各 LNZ 係数パターンの探索範囲内の RD コスト合計を比較し，コストが最小となったパターンの LNZ 係数を使用することで，符号化効率を向上する手法である．この LNZ 係数の探索の際に，LNZ 係数パターン間にデータの依存が発生しており，各パターンを並列に処理するといった高速化手法がとれない．また，一つのパターンに対し，最大 4 回の RD コスト計算を行うため，演算負荷も大きい．特に，$\lambda \times R_i$ の演算はハードウェアでは 1 度に 3 サイクルを要する．しかし，提案 RDOQ ではマクロブロックあたりの推定ビットレートを統一している．そのため，提案 RDOQ では $\lambda \times R_i$ の演算を LNZ 係数最適化処理のループ外で行うことで，演算結果を各パターンで共有でき，演算負荷を削減する．

本研究では，LNZ 係数最適化処理ループにループパイプライン化を適用する．Bach システムでは，throughput プラグマにスループット値を指定することで，そのスループット値でループをパイプライン化する．$\lambda \times R_i$ を LNZ 係数最適化ループ外で演算することで，1 ループ 11 サイクルから 5 サイクルとなり，さらにスループット値'1' を指定することで，5 ステージ 1 サイクルのパイプライン化による高速化を実現する．

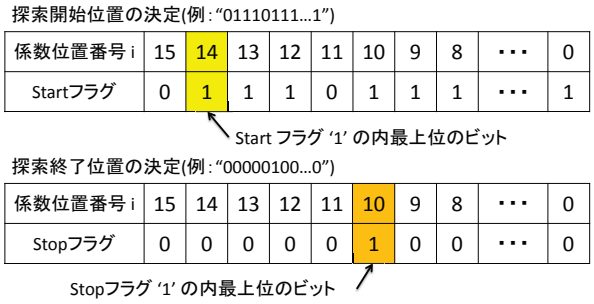

図 4: LNZ 係数探索範囲決定の例

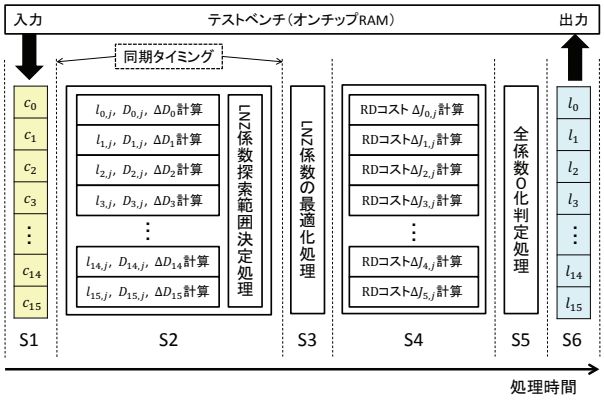

図 5: RDOQ のパイプラインステージ

3.4 LNZ 係数探索範囲決定処理のプライオリティエンコード化

次に，LNZ 係数探索範囲決定処理のプライオリティエンコード化について述べる．3.2 節で述べた様に，本研究では丸め候補計算・歪計算の並列化のために，丸め候補計算・歪計算と同時に行っていた探索範囲決定処理を，歪計算後に行うようにアルゴリズムを変更する．

本研究では，この探索範囲決定処理をプライオリティエンコードで実現することで高速化する．LNZ 係数の探索範囲は最終的に，ジグザグシーケンス順で最も後ろに位置する Start フラグ=’1’ の係数が探索開始位置として，最も後ろに位置する Stop フラグ=’1 の係数が探索終了位置として決定される．つまり，ジグザグシーケンス順で後ろの係数を優先して，対応する係数位置番号を出力する処理である．プライオリティエンコード化するために，各係数の Start フラグおよび Stop フラグを，最も後に位置する係数から順にビット連結し，連結したビットを入力として最も上位ビットに位置する’1’ と対応する係数位置番号を出力する様に動作を記述する．図 4 に提案する LNZ 係数探索位置の決定例を示す．これにより，LNZ 係数の探索開始位置，終了位置の決定処理をを合わせて 1 サイクルで実現する．

3.5 同期パイプライン化

本節では，関数 (ステージ) 間で同期信号を使ってパイプライン化する同期パイプライン化について述べる．

H.264 の圧縮符号化はマクロブロック (16x16 ブロック) 単位で行われるが，本研究の RDOQ 回路は 4x4 ブロック単位で行うため，マクロブロック毎に 16 個の 4x4 ブロックの RDOQ をパイプラインで処理することで高速化する．ただし，イントラ予測時は 4x4 ブロック間に依存関係があるため，インター予測時のみこの手法を適用する．

本研究では，RDOQ の処理を図 5 の様に 6 ステージ (S1～S6) に分割し，ステージ間で同期することでパイプライン化する．具体的には，ステージ間でフロー依存のあるデータを同期レジスタに格納し，それを次のステージの処理開始時に読み込む．これを各ステージがループして行うことでパイプライン処理となる．この時，各ステージ間で同期するデータは 1 つとし，フロー依存のあるデータをビット連結している．ただし，v3 のプライオリティエンコード化の適用した時点では，S1 の入力部でオンチップ RAM から入力されるデータが多く，他のステージと比べボトルネックとなっている．そこで，複数の入力データをビット連結し，入力データ数を削減することでこのボトルネックを解消する．

3.6 ハードウェア設計結果

表 1 に，本研究のハードウェア設計環境を示す．また，オンチップメモリへのアクセス時間は 5ns と設定した．

提案 RDOQ をシーケンシャルのままハードウェア化した回路 (v0) をベースとして，高速化手法を適用した設計結果を表 2 に示す．

v0 の回路と比較し，提案した高速化手法を全て適用した v3 の回路では処理時間は約 5.6 倍の高速化となった．適用した高速化手法の内，大きく高速化に貢献したのは v1 の 16 並列化であり，5.6 倍の高速化の内，約 77%は 16 並列化によるものである．この理由としては，RDOQ の処理内で最も複雑性の高く処理時間の大部分を占める「丸め候補算出・歪計算部」「RD コスト計算部」に対して，16 並列化を適用できたことが大きい．一方，ループパイプライン化およびプライオリティエンコード化は，v1 で並列化できなかった RDOQ の処理に適用したものである．ループパイプライン化は RDOQ において上記の次に複雑性の高い「LNZ 係数最適化処理部」に適用した．v1 の並列化により，処理内でネッ

表 1: 設計環境

動作合成システム	Bach System ver3.6
RTL シミュレーション	ModelSim(Mentor Graphics, Inc)
論理合成ツール	Design Compiler 2008(Synopsys, Inc)
動作周波数	100 MHz
セルライブラリ	日立 0.18μm セルライブラリ
処理ブロック数	6336 個 (4x4 ブロック)

表 2: 設計結果

バージョン	回路規模 [gate]	処理時間 [μs]
v0(シーケンシャル)	158,734	98,551
v1(16 並列化)	778,654	30,654
v2(ループパイプライン化)	794,006	28,556
v3(プライオリティエンコード化)	785,035	28,049
v4(同期パイプライン化)	設計中	設計中

クとなった LNZ 係数最適化処理に対してループパイプラインを適用し，高速化することで，RDOQ の各処理の処理時間のバランスを取ることができた．これは，v4 の同期パイプライン化による高速化に有利である．また，プライオリティエンコード化は「探索範囲決定処理」に適用している．探索範囲決定処理は，並列化の際に丸め候補算出・歪計算部から切り離された処理であるが，プライオリティエンコードによる効率の良い処理に変更することで，回路規模を維持しつつ 1 サイクルで実現でき，ハードウェアに効果的な手法だと言える．v4 の同期パイプライン化を適用した回路の設計結果については発表時に述べる．

4 おわりに

本研究では文献 [3] で提案した RDOQ のハードウェア実装を行った．提案 RDOQ で提案したハードウェアに適したアルゴリズムの改良を活用し，並列化，ループパイプライン化，プライオリティエンコーダ化といったハードウェア高速化手法を適用した回路は，シーケンシャルな提案 RDOQ 回路に対し，処理時間を約 82%削減した．また，入力部の高速化を含め，同期パイプライン化を適用した場合，プライオリティエンコード化適用後の回路に対し，さらに約 80%の処理時間削減を見込んでいる．

今後の課題として，パイプラインステージの分割数を増やし，さらなる高速化を図る．また，H.265/HEVC [7, 8] には H.264 とほぼ同様な RDOQ が用いられているため，H.265 への提案手法の適用，ハードウェア実装を検討する．

参考文献

[1] 大久保榮, 角野眞也, 菊池義浩, 鈴木輝彦, “ 改定三版 H.264/AVC 教科書,”インプレス R&D，東京，2009.

[2] K.Okada, et al.: ”Hardware Algorithm Optimization Using Bach C,” IEICE Trans. Fundamentals vol.E85-A, No.4, pp835-841, 2002.

[3] 森口元気，神戸尚志，藤田玄: ”レート歪み最適化による量子化プロセスの高速化手法とその評価”, 電子情報通信学会信学技報 113(454): 67-72, 2014.

[4] JVT reference software version 18.0 http://iphome.hhi.de/suehring/tml/download.

[5] Marta Karczewicz，Yan Ye，Peisong Chen: “ Rate Distortion Optimized Quantization ”，JVT-AA026.

[6] Limin Liu, and Alexis Tourapis: ”Rate distortion optimized quantization in the JM reference software, ” JVT-AA027, April 24-29, 2008.

[7] 村上篤道，浅井光太郎，関口俊一: ”高効率映像符号化技術 HEVC/H.265 とその応用，”オーム社，2013.

[8] 大久保榮，鈴木輝彦，高村誠之，中條健: ”高効率映像符号化技術 H.265/HEVC 教科書，”株式会社インプレスジャパン，2013.