PAPER

A VVC Dependent Quantization Optimization Based on the Parallel Viterbi Algorithm and Its FPGA Implementation

Qinghua SHENG[†], ***Member***, **Yu CHENG**[†], **Xiaofang HUANG**[††], **Changcai LAI**[†], **Xiaofeng HUANG**[†††], ***and* Haibin YIN**[†††a)], ***Nonmembers***

SUMMARY Dependent Quantization (DQ) is a new quantization tool introduced in the Versatile Video Coding (VVC) standard. While it provides better rate-distortion calculation accuracy, it also increases the computational complexity and hardware cost compared to the widely used scalar quantization. To address this issue, this paper proposes a parallel-dependent quantization hardware architecture using Verilog HDL language. The architecture preprocesses the coefficients with a scalar quantizer and a high-frequency filter, and then further segments and processes the coefficients in parallel using the Viterbi algorithm. Additionally, the weight bit width of the rate-distortion calculation is reduced to decrease the quantization cycle and computational complexity. Finally, the final quantization of the TU is determined through sequential scanning and judging of the rate-distortion cost. Experimental results show that the proposed algorithm reduces the quantization cycle by an average of 56.96% compared to VVC's reference platform VTM, with a Bjøntegaard delta bit rate (BDBR) loss of 1.03% and 1.05% under the Low-delay P and Random Access configurations, respectively. Verification on the AMD FPGA development platform demonstrates that the hardware implementation meets the quantization requirements for 1080P@60Hz video hardware encoding.

key words: *video coding, viterbi algorithm, dependent quantization (DQ), rate-distortion optimized quantization (RDOQ), FPGA*

1. Introduction

With the rapid development of the information society, HEVC [1] has become one of the most popular video coding standards. It has significantly improved video coding technology through enhanced flexibility and versatility. The newer video coding standard, VVC [2], has further doubled the compression efficiency by introducing new encoding tools to reduce data redundancy. Similar to HEVC, VVC follows a block-based hybrid video coding architecture. Experimental results have shown that in the random access configuration, the average coding bit rate is further reduced by 33.1% [3].

Quantization is one of the most time-consuming and complex parts of video coding, and designing an efficient and low-cost quantization algorithm is of great research interest. In most video coding algorithms, scalar quantization is commonly used to process transform coefficients [4]. In advanced encoders, the optimal quantization value is selected by calculating the distortion (D) and the bit rate (R) required to transmit the quantization value using a Lagrangian function [5]. This type of quantization is known as Rate-Distortion Optimized Quantization (RDOQ), and the function's value is called RD cost. RD cost consists of two parts: computational cost and cost optimization. The computational cost includes the calculation of D and R. Due to the sequential nature of bit rate calculation, all transform coefficients must be quantized and encoded in a predetermined order. Similarly, cost optimization is also performed sequentially, requiring all candidate quantization values to be considered, resulting in significant computational and time cost increases.

Hence, accelerating the quantization part has always been a critical optimization module for video encoders. In the software algorithm area, computational optimization is achieved mainly by speeding up iterative calculations and predicting all-zero blocks [6]. Traditional optimization approaches focus on algorithm structure [7] and implementation platforms [8], which have achieved certain results but are difficult to apply to hardware architecture. Parallel encoding of RDOQ on hardware has been implemented through high-level synthesis [9], but the parallel segmentation is uneven, resulting in low hardware circuit efficiency. In recent years, deep learning has also been explored to replace prediction search and decision-making parts of intra-predicted and inter-predicted in video coding [10], resulting in significant improvements. Kianfar et al. [11] considered using neural networks to replace traditional rate-distortion optimization and quantization parts, but the encoding efficiency was not ideal.

Vector quantization is a widely studied and mature technology in the field of coding, and its most widely used form the four-state encoding has been adopted by VVC. Vector quantization encoding has been extensively investigated in various coding applications, such as image coding [12] and neural network coding [13]. The latest VVC standard introduces dependent quantization (DQ) based on vector quantization [14], which is not the first application of vector quantization in video coding. The HEVC standard also includes a simple form of vector quantization called signed data hiding [15]. In the quantization process, DQ follows the principle of RDOQ and involves the sequential calculation

Manuscript received November 15, 2023.
Manuscript revised February 5, 2024.
Manuscript publicized March 4, 2024.

[†]The authors are with the School of Electronics and Information, Hangzhou DianZi University, Hangzhou, China.
[††]The author is with the School of Electronic Engineering, Hangzhou Dianzi University Information Engineering College, Hangzhou, China.
[†††]The authors are with the School of Communication Engineering, Hangzhou DianZi University, Hangzhou, China.
a) E-mail: yhb@hdu.edu.cn
DOI: 10.1587/transinf.2023EDP7243

of D and R. The key contribution of DQ lies in improving the quantizer itself, enabling the representation of values with fewer bits. In terms of accelerated coding for DQ, Niu et al. [16] had identified the characteristics of entirely zero quantization in advance to reduce the number of quantization calls. Wang et al. [17] has improved the accuracy of the RD cost of DQ, resulting in increased relative efficiency. Adhuran et al. [18] had further adjusted the quantization parameters (QP) of DQ for 3D video coding, expanding its application range. However, these optimization methods mentioned above are more traditional and not suitable for hardware implementation. Liu et al. [19] had established a dependency relationship between TUs, which improves coding efficiency but greatly reduces the feasibility of parallel computing between TUs. Therefore, while DQ is efficient in software coding, there is a lack of research papers specifically focusing on hardware-accelerated coding for DQ.

To address these shortcomings, this paper proposes a hardware quantization implementation for DQ based on a parallel Viterbi algorithm. The aim is to reduce the hardware cost of RD cost calculation by breaking the data dependency in traditional RDOQ and reducing the quantization cycle. Section 2 provides a brief introduction to the algorithm structure and implementation challenges of DQ in VTM. Section 3 presents the parallel design details of the proposed algorithm and its hardware implementation framework. Section 4 summarizes the experimental content of the paper and discusses the obtained results. Finally, Sect. 5 concludes the paper.

2. Implementation and Complexity Analysis of DQ in VVC

In the VVC reference software platform VTM, there are three optional quantization methods, and the scanning process is depicted in Fig. 1. The encoding configuration file allows the selection of scalar quantization, RDOQ, or DQ. The coefficients of the TU undergo sequential processing using Zig-zag scanning. In the case of DQ and RDOQ, further calculation of the quantization cost caused by each coefficient is necessary to obtain a better quantization value.

2.1 Scalar Quantization

Scalar quantization initially utilizes the QP provided by the rate control algorithm to obtain the quantization step Q_{step} according to Eq. (1) as

$$Q_{step} = 0.625 \times 2^{\frac{QP}{6}} \tag{1}$$

The theoretical calculation process of scalar quantization is described by Eq. (2) as

$$l_i = floor\left(\frac{t_i}{Q_{step}} + f\right) \quad f = \begin{cases} 1/3 & \text{, Intra-Frame} \\ 1/6 & \text{, Inter-Frame} \end{cases} \tag{2}$$

In Eq. (2), l_i represents the quantization value, $floor$ is the rounding function, t_i denotes the TU coefficient, and f is the bias parameter of the coding mode.

However, due to the presence of a decimal coefficient in Eq. (2), it is not suitable for hardware implementation. The method employed by the actual encoder utilizes Eq. (3) as

$$\begin{aligned} l_i &= floor\left(\frac{t_i \cdot MF}{2^{qbits+T_{shift}}} + f\right) \\ &= (t_i \cdot MF + f') >> (qbits + T_{shift}) \end{aligned} \tag{3}$$

$$MF = 0.625 \times 2^{\text{qbits}+\frac{Rem}{6}} \tag{4}$$

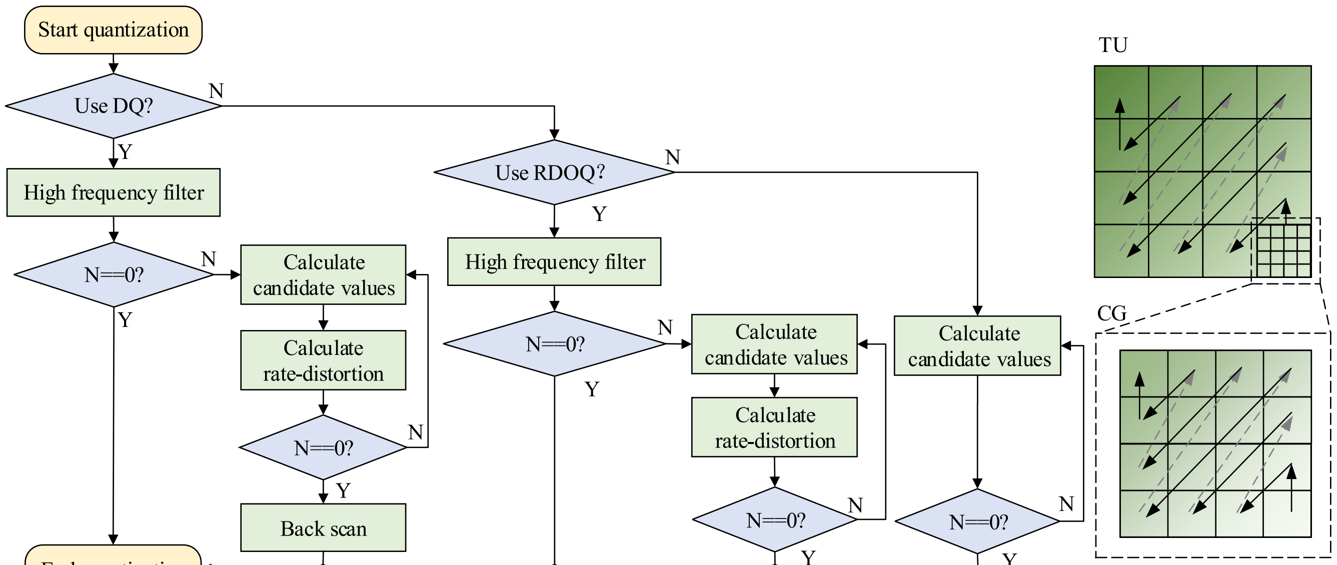


Fig. 1 Quantization methods and scanning process in VVC

where Eq. (4) defines MF as the continuous value in a single cycle. Here, Rem represents the remainder of QP to 6, $qbits$ is the amplification of all values of the number of left shifts required by an integer, T_{shift} denotes the quotient of QP to 6 and represents the number of cycle changes, and f' represents the amplification value of the left shift digit of f.

The calculation is decomposed into three steps in VTM. First, the amplified integer value MF and f' are calculated in the initial cycle. Then, the division of t_i by the Q_{step} is replaced by t_i multiplied by MF. Finally, f' is added to the result obtained in the previous step, followed by a right shift to eliminate the enlarged calculation bits. This approach allows for the conversion of complex decimal operations into simple fixed number multiplication and shift calculations, thereby reducing the computational cost.

2.2 Rate-Distortion Optimized Quantization

The efficiency of each quantized value is determined by the rate-distortion resulting from the encoding R and D. This calculation process can be described by Eq. (5) as

$$J_i = \min\left(D\left(t_i, l_i\right) + \lambda \cdot R\left(l_i\right)\right) \tag{5}$$

where J_i represents the RD cost of the current coefficient, D represents the distortion of the current quantization value compared to the original coefficient value, B represents the encoding bit rate, and λ is the Lagrange multiplier used to adjust the data weighting of the formula.

RDOQ still utilizes the Zig-zag scanning process of scalar quantization, as it is based on scalar quantization. Nonetheless, RDOQ continuously traverses the standard quantization value of the scalar quantizer, 0, and uses the rate-distortion Eq. (5) to calculate the RD cost between candidate values. The optimal quantization value is then selected based on achieving a practical application value closest to the theoretical efficiency in terms of the entropy-encoded bit rate and encoding-induced distortion, ultimately improving the encoding efficiency.

When it comes to HEVC encoding, Sze et al. [20] indicated that without calculating the rate-distortion of video encoding leads to an efficiency loss of 10–15% for intra-frame coding and 40% for inter-frame coding. However, calculating rate-distortion comes at a significant cost, mainly due to the computational cost and clock cycles required to traverse candidate values and calculate their RD costs. Moreover, the involvement of rate-distortion calculation introduces a correlation between quantized values, making it challenging for RDOQ to achieve the same clock cycle consumption as scalar quantization while maintaining high data throughput.

2.3 Dependent Quantization

DQ utilizes the correlation between digits to reduce the bit rate and employs the Viterbi algorithm to select the optimal quantization value. The quantization structure of DQ is based on an even-odd quantizer. Specifically, the even quantizer Q_0 quantizes even multiples such as 0, 2, 4, 6, etc.,

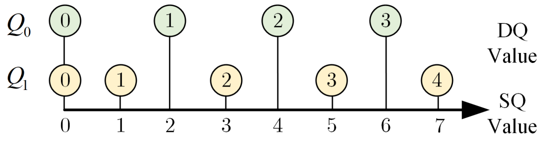


Fig. 2 Mapping relationship between DQ and scalar quantization values

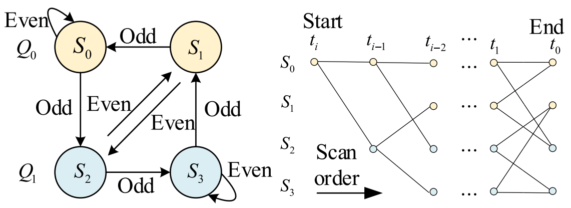


Fig. 3 DQ conversion relationship and scanning order

while the odd quantizer Q_1 quantizes odd multiples such as 0, 1, 3, 5, 7, etc. Both quantizers include 0 as a special value. Figure 2 presents the mapping relationship between the quantization values of DQ and scalar quantization.

The scalar quantization values obtained from Eq. (3) are continuously added by 3 and distributed to Q_0 and Q_1, generating four candidate quantization values. These four candidates are then divided into four states, $S_0 \sim S_3$, based on the parity of the encoding bits and the difference in quantizer. For instance, S_0 represents quantization values in the even quantizer Q_0 with an even number of ‘1’ in their binary encoding, such as 0, 6, 10, etc. S_1 represents quantization values in the even quantizer Q_0 with an odd number of ‘1’ in their binary encoding, such as 2, 4, 8, etc. The encoding rules for Q_1 are similar to those for Q_0 and are responsible for encoding the values in the odd quantizer. The parity of the quantization value’s bits and the parity of the quantizer together form a vector that is used to index to the next quantization node. Figure 3 illustrates the transition relationships and scanning order between states.

where t_i represents the transform coefficient and the starting state of the TU is fixed as S_0.

2.4 Analysis of Difficulty in DQ Hardware Implementation

Regarding the encoding end, each coefficient in the TU after quantization has M candidate values. While the probability algorithm-based rate-distortion method employed by DQ can determine the encoding cost between these candidate values, traditional traversal methods require multiple traversals to find the optimal path among M candidate values, resulting in high complexity. Therefore, the Viterbi algorithm is used in the VTM to convert the global optimal solution to the local optimal solution for each candidate value. By connecting the local optimal solutions of each candidate value in sequence, the globally optimal result can be obtained through a single traversal, significantly reducing computational and time costs during encoding.

In DQ, the transition between vector quantizers can be converted using the principle illustrated in Fig. 3. For each transform coefficient, the quantization value needs to select the node with minimum distortion. The RD cost calculated for each coefficient is assigned to all connections between grid nodes. Consequently, the optimal encoding of the quantization value can be represented by minimizing the path with the total RD cost. In DQ, each coefficient produces four non-zero candidate values, resulting in the need to calculate eight path costs. Compared to RDOQ, which has only two non-zero candidate values, the computational cost of rate-distortion optimization significantly increases. Additionally, as video encoding resolution continues to rise, there is a notable increase in the number of pixels being quantized. Therefore, improving the single-cycle processing pixel capability and simplifying the rate-distortion calculation process is crucial to accelerate DQ.

3. Design and Implementation of Hardware Parallel-Dependent Quantization Algorithm

3.1 Parallel Viterbi Coding

The ideal Viterbi algorithm is designed for selecting paths with a fixed number of nodes. However, the amount of quantized data in video coding is determined by various coding algorithm parameters. In video coding, the data passed from TU is arranged in sequential order from low frequency to high frequency using Zig-zag scanning. Due to the operations performed by intra-frame and inter-frame prediction algorithms, small discontinuous residual coefficients are present at the end of the sequence. This leads to the utilization of Bayesian algorithms in the VTM for high-frequency coefficient filters to compress the data volume before Viterbi algorithm processing, thereby reducing the encoding cycles. Nevertheless, when Q_{step} is too small or the residual coefficients are too large, the number of high-frequency coefficients that can be filtered by the threshold reduces. Consequently, more coefficient needs to be processed in the post-processing stage, resulting in significant fluctuations in the quantization cycles of TU.

To investigate the impact of high-frequency filters on the quantization cycles of the TU, this paper conducted experimental statistics on the percentage distribution of intra-predicted TU quantization cycles, as shown in Fig. 4. The darker the color, the greater the number of cycles consumed by DQ, and the larger the area of the color block indicates a higher actual proportion under that parameter. The experimental results reveal that a considerable number of TUs experience large fluctuations in the number of quantization cycles for small QP values, which is not conducive to hardware implementation. To address the aforementioned issues, it is crucial to control the number of cycles consumed by the DQ within a stable limit under various parameter configurations. To achieve this, this paper introduces high-frequency filters and designs the Viterbi algorithm used in VTM into several large parallel processing blocks for the entire data segment. The data blocks are internally iterated according to the original Viterbi algorithm, enabling parallel processing during quantization. In order to explore the impact of secondary segmentation on quantization efficiency, this paper investigates the distribution of quantization efficiency of frame-intra prediction under different secondary segmentation steps for various feature sequences, as depicted in Fig. 5. The figure demonstrates that as the length of the segmentation increases, the coding efficiency gradually increases and reaches its peak growth at approximately 128.

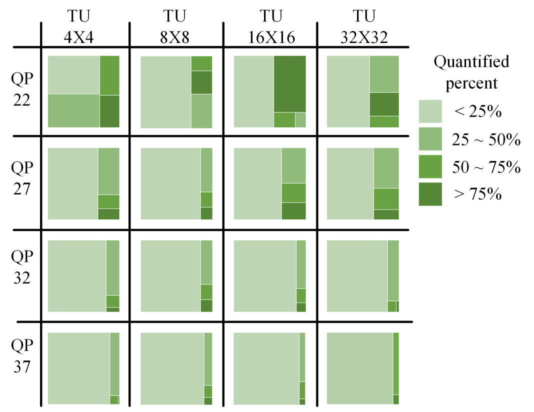


Fig. 4 The actual percentage of quantization cycles consumed under different parameters

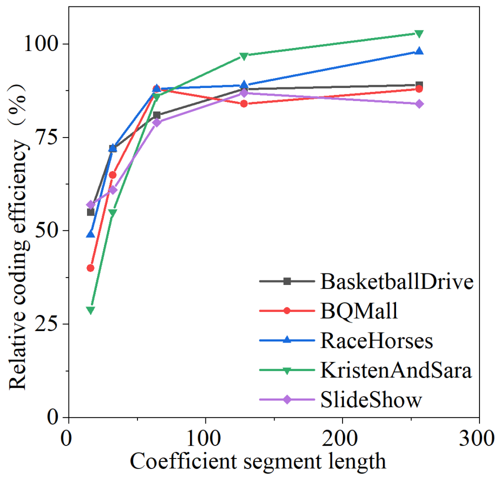


Fig. 5 Efficiency distribution of different sequences under quadratic Viterbi coding

3.2 Parallel Segment Concatenation

During the quantization process of transform coefficients arranged in the Zig-zag scan order, each coefficient segment is independently quantized from the starting state and converges to one of the four ending states. However, there is a lack of necessary judgment connection between the starting and ending states of different coefficient segments. As a result, termination occurs after the first coefficient segment when outputting in reverse scan order. In regular quantization, the Nth coefficient always serves as the starting node of parallel processing, and the $(N + 1)$th coefficient acts as

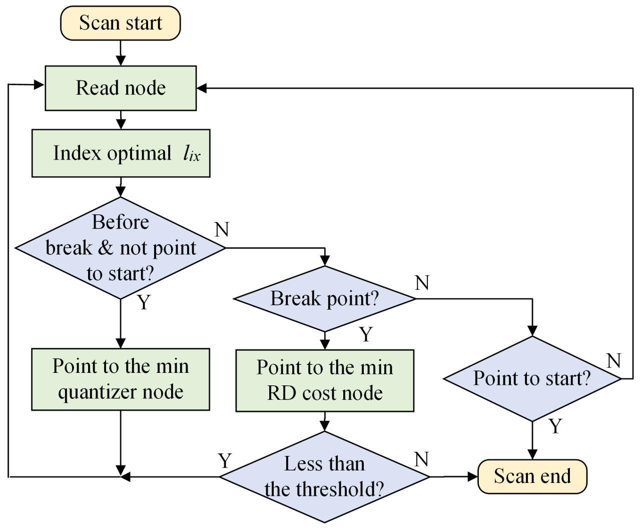


Fig. 6 Parallel segmentation concatenation state machine

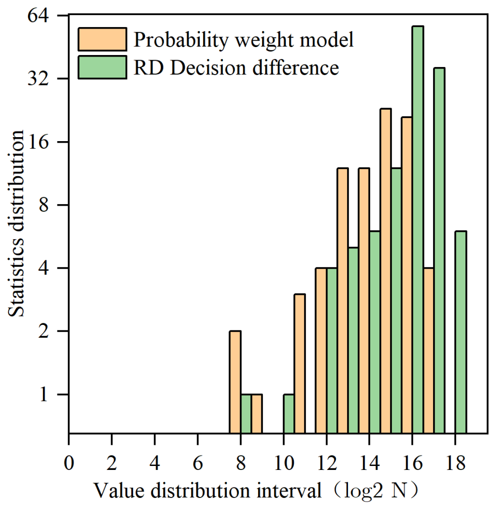


Fig. 7 Statistical distribution of probability weight values in rate-distortion decision-making

the ending node of the next parallel processing for a parallel segment. Since the starting state of the Nth coefficient is fixed as S_0, the only possible connections are S_0 to S_2, resulting in the fixed connection of S_0 to S_2 and S_1 to S_3 for the $(N+1)$th to $(N+2)$th coefficients. However, the actual connection between the Nth and $(N+1)$th coefficients should be freely selected based on the RD cost of both the previous and upcoming encoding.

To address this problem, this paper proposes a method for connecting parallel segments, as depicted in Fig. 6. During the scanning process, DQ reads the quantization value of the current node and the next index based on the previous index until the end of the path. In order to connect the quantization results of different segments, when reaching the end of a segment, DQ terminates quantization if the next index points to the endpoint, which is the coding start; At segment boundaries, the fixed index pointing to the endpoint is disregarded. Instead, the node with lower quantization distortion is manually selected as the last node of the segment. The RD cost of this node is then compared with the first node of the next segment. Based on a cost threshold, the decision is made to either terminate quantization or index the next node.

3.3 Streamlined Rate-Distortion Decisions

In the video coding process, the RD cost of each quantization path needs to be calculated using the rate-distortion Eq. (5). This formula quantifies the impact of each path on the actual encoding results as a specific value, enabling the determination of the optimal quantization path by comparing the cost values. Compared to the traditional RDOQ, which only has two or three candidate values, DQ needs to calculate the RD cost of at least eight candidate values simultaneously. This results in a significant increase in computation time and hardware cost. The ideal calculation of RD cost requires the introduction of detailed probability weights for calculation. In VTM, the calculation weights range from 200 to 200, 000, which can lead to significant logic delay and resource consumption in hardware implementation. Additionally, the RD cost impact generated by each coefficient is accumulated iteratively and added to the calculation of the RD cost of the next coefficient. This accumulation of hundreds or thousands of RD costs can result in a large carry-chain in hardware implementation, causing serious logic delay.

On the other hand, the efficiency of the quantization candidate values needs to be compared based on their RD costs. However, this process does not consider the specific numerical values, which cause data redundancy. To further investigate the differences in numerical calculation of RD cost, this paper statistically analyzes the distribution of probability weight model values used in rate-distortion calculation based on YUV, as shown in Fig. 7. The yellow part in the figure represents the calculated values of probability weight, while the green part represents the RD cost difference between the path selection during encoding. The distribution of numerical values in Fig. 7 indicates that the more values are distributed to the right of the horizontal axis, the more redundancy exists between the RD costs of the candidate values. Therefore, reducing the redundancy between the difference and the numerical values is crucial for simplifying the RD cost calculation.

In addition, this paper also examines the impact of numerical bit width reduction and iteration variable bit-width, used in RD cost calculation, on video coding efficiency, as shown in Fig. 8. The influence of numerical bit width on encoding efficiency aligns with the data statistics analysis in Fig. 7. When the bit width of the weight reduction is less than 10, there is still a difference between the rate-distortion weights, and the encoding efficiency is not significantly affected. When the data bit width is further reduced, the difference between rate-distortion weights gradually approaches zero and the encoding efficiency starts to decrease rapidly. Therefore, the final simplified RD cost calculation comprises low-bit width iteration variables and rate-distortion numeri-

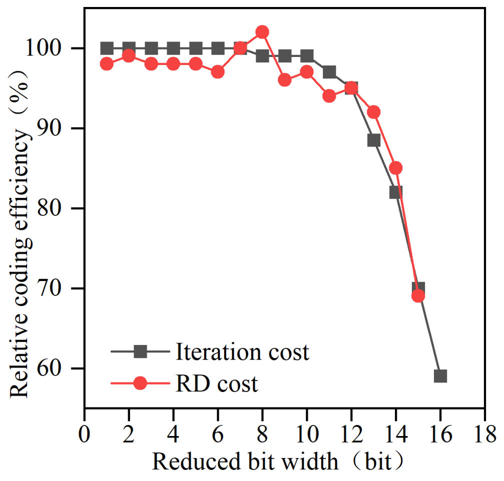


Fig. 8 Effect of rate-distortion calculation on encoding efficiency under different bit width precision

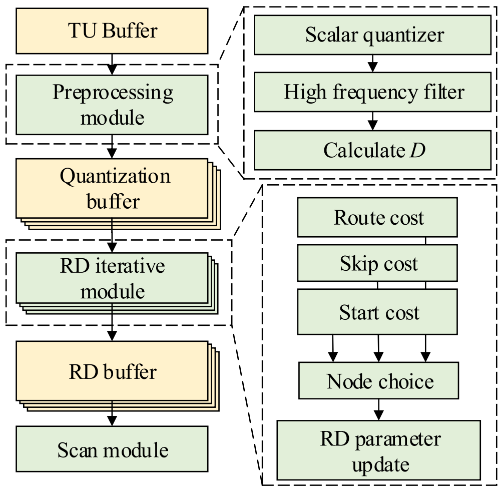


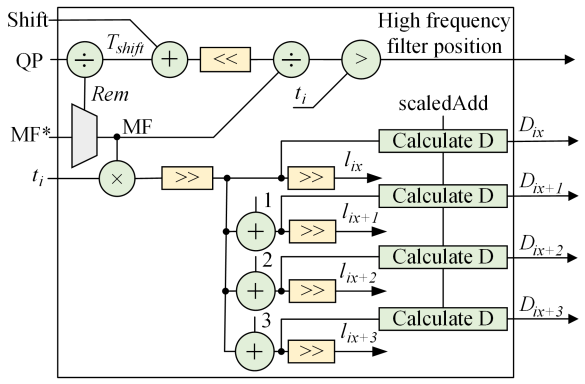
Fig. 9 Hardware parallel-dependent quantization algorithm hardware overall architecture

cal calculations.

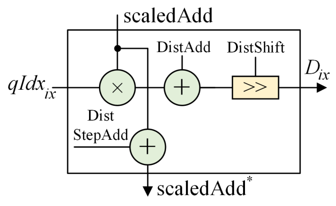
3.4 Hardware Parallel-Dependent Quantification Implementation

Although DQ in VTM has a processing sequence, the actual processing capacity of each step is not balanced. To solve this problem, this paper splits each step of DQ and stores the data of the front and rear modules through Buffer. To verify the feasibility of this algorithm in hardware implementation, this paper proposes a hardware parallel-dependent quantization pipeline processing structure. The architecture consists of three main modules: preprocessing, RD iteration, and scan module, as illustrated in Fig. 9.

The quantization starts from the TU coefficients, which undergo preprocessing to generate quantization candidate values l_{ix} through high-frequency coefficient threshold filtering and scalar quantizer. l_{ix} and corresponding quantization distortion costs D_{ix} are stored in the quantization buffer.



(a) Architecture for quantization value calculation



(b) Architecture for quantization distortion calculation

Fig. 10 Hardware architecture of the preprocessing module

This stage will consume the number of coefficients plus a pipeline delay of 2 clock cycles. When the input coefficient t_i exceeds the filtering threshold, the current coefficient position is stored and awaits further rate-distortion iteration processing. In the RD calculation part, the algorithm sequentially reads from the corresponding quantization buffer, starting from the high-frequency threshold address, and calculates the RD costs, writing it into the RD buffer for the current coefficient. Simultaneously, the current rate-distortion parameter is iterated for the cost calculation of the next coefficient until the last coefficient is processed. This stage will consume a maximum of a coefficient number of clock cycles, which decreases as the degree of parallelism increases. Once all coefficients are quantized, the scanning module utilizes the scanning concatenation algorithm in Fig. 6 to output in reverse order until the quantization endpoint is reached.

This algorithm connects preprocessing, RD iteration, and scan module through pipelines and buffers, reducing the handshake delay between modules and improving the coding efficiency of the algorithm on smaller TU. The modular design and implementation platform enhances flexibility and adaptability, facilitating future improvements and practical applications of the algorithm.

3.4.1 Implementation of Preprocessing Module

The hardware architecture for the preprocessing module is illustrated in Fig. 10. The design of the quantization distortion calculation in the hardware architecture is depicted in Fig. 10 (a). The original data t_i is input to the preprocessing module in the order of transform data arranged by Zig-zag scanning. Initially, the scalar quantizer is designed according to Eq. (3), and the four non-zero quantization values l_{ix} are obtained by continuously adding 1 and shifting. Simultaneously, the high-frequency filtering threshold is calculated based on the current configuration parameters provided by the rate control algorithm. Additionally, the position of the first coefficient is output through the threshold.

To reduce the complexity of rate-distortion iteration calculation, this architecture simultaneously calculates D_{ix} during the preprocessing stage. Specifically, based on the cost in the rate-distortion Eq. (5), it is necessary to calculate the distortion cost. However, there is no dependency between distortion costs before and after, and it is only related to the parameters provided by the rate control algorithm and the differences generated by the addition of quantization values. The hardware architecture for the quantization distortion calculation is shown in Fig. 10 (b), and the calculation of D and the high-frequency filtering threshold is defined as:

$$D = (qIdx * scaleAdd + DistAdd) >> DistShift \quad (6)$$

where $qIdx_{ix}$ represents the quantized value before l_{ix} shifting, and $scaleAdd, DistAdd, DistStepAdd$, and $DistShift$ are the distortion weights determined by the rate control algorithm. As $qIdx$ gradually increases, the distortion weight $scaleAdd$ also increases gradually and is used for the calculation of D_{ix+1}.

$$Thres = (14 - T_{shift} - TransformShift)/MF \quad (7)$$

where $Thres$ represents the high-frequency filtering threshold, and $TransformShift$ represents shift caused by TU.

3.4.2 Implementation of Rate-Distortion Iterative Module

The hardware architecture for the rate-distortion iterative module is depicted in Fig. 11. This module is responsible for calculating the RD cost of the current coefficient. It sequentially utilizes the preprocessed quantization value l_{ix},

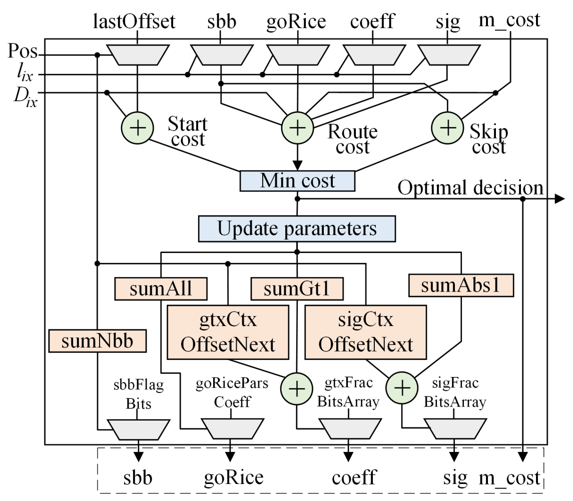


Fig. 11 Hardware architecture of the rate-distortion iterative module

quantization distortion cost D_{ix}, and coefficient position Pos from the quantization buffer. These are combined with dependent parameters such as position weight ($lastOffset$), coefficient coding weight ($sbb, goRice, etc.$), and iterative cost (m_cost) according to the optional quantization method. The start cost, route cost, and skip cost of the current coefficient are calculated based on these inputs and can be described by Eq. (8) as

$$\begin{aligned} Startcost &= lastOffset + D_{ix} \\ Routecost &= sbb + goRice + coeff + sig + m_{cost} \\ Skipcost &= sbb + m_{cost} \end{aligned} \quad (8)$$

Once the minimum cost is determined, it is stored temporarily in the RD buffer. Additionally, the weight parameters used for rate-distortion are updated based on the current optimal decision. For coefficients with RD costs lower than the start cost, their parameters remain fixed at the initial values. When a coefficient is quantized as a non-start decision, the weight parameters used are updated based on the current quantization value and cost. The corresponding parameters are extracted from the original weight parameter array ($sbbFlagBits, goRiceCoeff, etc.$) for the calculation of the RD cost of the next coefficient. This process continues until all the quantized data has been processed.

4. Experimental Results and Analysis

To validate the efficiency of the hardware-parallel dependent quantization algorithm and the hardware implementation, as well as compare it with the efficiency of traditional scalar quantization hardware circuits and the coding efficiency of traditional DQ algorithms, we conducted encoding efficiency tests in the VVC reference platform VTM. In addition, we used Verilog HDL to design the circuit and simulated and synthesized it in the Vivado 2022.2 environment.

4.1 Comparison of Software Algorithm Coding Efficiency

We integrated the proposed method into the VVC reference platform VTM 14.2 and conducted tests according to the Common Test Conditions (CTC) of the Joint Video Exploration Team (JVET). The QP values were set to 22, 27, 32, and 37, and we collected and statistically analyzed the encoded results to evaluate the coding efficiency. Negative values $\Delta BDBR$ indicate coding efficiency gains. We tested the coding gain of both Low-delay P (LP) and Random Access (RA) configurations for recommended test sequences from Class A to Class F. To analyze the algorithm efficiency, we evaluated the relative coding efficiency loss of the proposed algorithm, defined as:

$$\Delta BDBR = BDBR_{HDL} - BDBR_{DQ} \quad (9)$$

where $\Delta BDBR$ represents the coding gain of the algorithm compared to scalar quantization, $BDBR_{HDL}$ represents the coding gain of the proposed algorithm under a given configuration environment, and $BDBR_{DQ}$ represents the coding

Table 1 Efficiency comparison of the proposed algorithm under Low-delay P and Random Access configurations

Sequences		LP $\Delta BDBR$	RA $\Delta BDBR$
	Campfire	0.82	1.23
	FoodMarket4	-0.83	0.10
Class A1	Tango2	0.39	-0.03
3840×2160	CatRobot	0.59	0.78
	DaylightRoad2	1.41	0.33
Class A2	ParkRunning3	1.55	1.64
3840×2160	BasketballDrive	0.98	0.68
	BQTerrace	1.64	2.60
	Cactus	0.90	1.00
Class B	MarketPlace	0.39	0.50
1920×1080	RitualDance	0.22	0.24
	BasketballDrill	1.23	1.35
	BQMall	1.73	1.67
Class B	PartyScene	2.09	1.89
832×480	RaceHorses	1.83	1.64
	FourPeople	1.15	0.94
	Johnny	2.56	1.49
Class E	KristenAndSara	0.55	0.95
1280×720	KristenAndSara	0.55	0.95
	ArenaOfValor	1.04	0.91
Class F	BasketballDrillText	0.84	1.33
	SlideEditing	0.62	0.95
	SlideShow	0.86	1.12
	Class A1	0.13	0.44
	Class A2	1.19	0.92
	Class B	0.83	1.00
	Class C	1.72	1.64
	Class E	2.02	1.13
	Overall	**1.05**	**1.03**
	Class D	1.42	1.98
	Class F	0.84	1.08

gain of VVC under the same configuration environment. The experimental results in Table 1 show that the proposed algorithm has an average loss of 1.05% and 1.03% compared to the VTM algorithm in all test sequences under Low-delay P and Random Access configurations respectively. The coding gains of each individual sequence in the YUV overall coding gain are shown as well. It can be seen that similar experimental results are observed for the proposed method under both Low-delay P and Random Access configurations. Therefore, the proposed algorithm has good versatility and flexibility, which can reduce the complexity of video hardware encoding while ensuring video quality. The experimental results show that the proposed algorithm achieves a good trade-off between coding efficiency and computational complexity, and has stable gains under different environmental parameter configurations.

4.2 Comparison of Hardware Resource Consumption

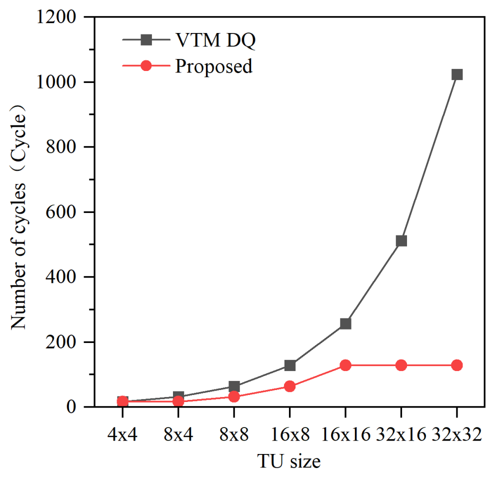
The proposed method was implemented on the AMD XCZU19 platform using Vivado 2022.2 as the development tool. Table 2 presents the resource consumption and comparison. The hardware resource consumption data of VTM's DQ algorithm is estimated based on the VTM algorithm and existing computing frameworks. The experimental data in this paper are further optimized and improved based on VTM's experiments for comparison purposes. Additionally, as a comparison of quantization algorithms, the comparison data of scalar quantization is sourced from the quantization part of the HEVC video hardware encoding core open-sourced by the VIP laboratory of Fudan University, which represents the mainstream parallel scalar quantization solution. Furthermore, another paper implemented a more complex RDOQ parallel encoding scheme through high-level language synthesis [9].

Table 2 Compared with the results achieved in other literature

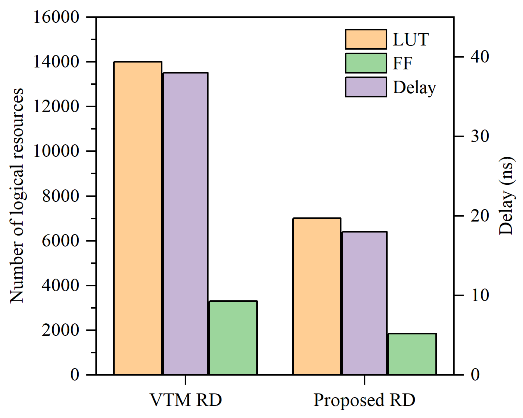
Document	VIP's	Zhao's [9]	VTM-DQ	Proposed
FPGA model	XCZU19	ALVEO U250	XCZU19	XCZU19
LUT	7709	109759	19954	60586
FF	2104	29576	4087	15301
DSP	32	481	42	42
FMAX	119.4M	200M	14M	55.6M
Parallelism	32	4-32	1	1-8

Table 2 shows the comparison of the proposed algorithm with the results achieved in other literature. The FPGA model used for all comparisons is XCZU19. The amount of LUTs, FFs, DSPs, and the maximum frequency achieved (FMAX) are shown for each algorithm. Additionally, the parallelism level is indicated for each algorithm.

The experimental data demonstrate that the proposed algorithm exhibits higher efficiency compared to other rate-distortion hardware encoding modules within the same logical magnitude. Moreover, in comparison to various scalar quantization hardware solutions with lower efficiency, the hardware cost of rate-distortion calculation does not increase significantly. The proposed algorithm performs second-segmented parallel processing on the size of TU, and limits the bit width of probability weight and iteration cost used in the rate-distortion calculation process. Figure 12 shows the comparison of the quantization cycle and hardware resource consumption with VTM's native rate-distortion calculation quantization cycle and hardware resource consumption. Considering coding efficiency and hardware feasibility, according to the amount of data processed, the proposed algorithm saves an average quantization cycle of 56.96%. In the rate-distortion part, the proposed algorithm reduces 74.8% of LUT usage, 64.6% of FF usage, and 64% of delay compared to VTM's native rate-distortion calculation. In the video encoding of 1080P@60Hz, there are at most $1920 \times 1080 \times 60 = 124.4$M coefficients that need to be quantized per second. According to the distribution of high-frequency filtering, the actual average number of encoding cycles is 43M clocks, which is much lower than the actual operating frequency of 55.6M clocks of the proposed algorithm. Therefore, the proposed hardware architecture can meet the practical encoding requirements of 1080P@60Hz.



(a) Comparison of quantization cycle consumption



(b) Comparison of hardware resource consumption

Fig. 12 DQ hardware resource consumption comparison

5. Conclusion

This paper presents a novel parallel hardware implementation of dependent quantization designed specifically for the VVC standard, aiming to address the computational complexity associated with traditional DQ computation. The key contribution of this work is the reduction of quantization cycles in TU through the use of a high-frequency coefficient filter and parallel Viterbi encoding. Furthermore, numerical redundancy is leveraged in the RD cost calculation during the quantization process, resulting in reduced computational weights in terms of bit width and overall computational complexity per operation. The output is generated using a reverse scanning path node. By employing a pipeline that encompasses preprocessing, RD iteration, and scan module, the proposed algorithm significantly improves the hardware efficiency for processing parallel-quantization with fewer coefficients. Experimental results demonstrate that compared to VTM, the proposed algorithm achieves a $\Delta BDBR$ loss reduction of 1.05% and 1.03% in Low-delay P and Random Access configurations, respectively, as well as an average reduction of 56.96% in quantization cycles. Furthermore, compared to other similar hardware quantization schemes, the proposed algorithm achieves higher compression efficiency while consuming fewer logic resources.

Acknowledgements

This research was funded by “Pioneer” and “Leading Goose” R&D Program of Zhejiang Province under Grant 2022C01068, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LDT23F01014F01, in part by the National Natural Science Foundation of China under Grant 61901150 and 62031009, and in part by the National Key R&D Program of China under Grant 2021ZD0109802.

References

[1] G.J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” Circuits and Systems for Video Technology (CSVT), IEEE Circuits and Systems Society, vol.22, no.12, pp.1649–1668, 2012.

[2] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan, and J.-R. Ohm, “Overview of the versatile video coding (VVC) standard and its applications,” IEEE Trans. Circuits Syst. Video Technol., vol.31, no.10, pp.3736–3764, 2021.

[3] A. Mercat, A. Mäkinen, J. Sainio, A. Lemmetti, M. Viitanen, and J. Vanne, “Comparative rate-distortion-complexity analysis of vvc and hevc video codecs,” IEEE Access, vol.9, pp.67813–67828, 2021.

[4] T. Wiegand and H. Schwarz, “Video coding: part II of fundamentals of source and video coding,” Foundations and Trends® in Signal Processing, vol.10, no.1–3, pp.1–346, 2016.

[5] E.-h. Yang and X. Yu, “Soft decision quantization for H.264 with main profile compatibility,” IEEE Trans. Circuits Syst. Video Technol., vol.19, no.1, pp.122–127, 2008.

[6] M. Wang, X. Fang, S. Tan, X. Zhang, and L. Zhang, “Low complexity quantization in high efficiency video coding,” IEEE Access, vol.8, pp.145159–145170, 2020.

[7] J. He, F. Yang, and Y. Zhou, “High-speed implementation of rate-distortion optimised quantisation for H.265/hevc,” IET Image Processing, vol.9, no.8, pp.652–661, 2015.

[8] H. Igarashi, F. Takano, T. Takenaka, H. Inoue, and T. Moriyoshi, “Parallel rate distortion optimized quantization for 4k real-time gpu-based hevc encoder,” 2018 IEEE Visual Communications and Image Processing (VCIP), pp.1–4, IEEE, 2018.

[9] J. Zhao, F. Yang, X. Huang, G. Xiang, P. Zhang, L. Zhao, and W. Yan, “Scanline-based fast algorithm and pipelined hardware design of rate-distortion optimized quantization for avs3,” 2023 IEEE International Conference on Consumer Electronics (ICCE), pp.1–6, IEEE, 2023.

[10] H. Wang, S. Yu, Y. Zhang, Z. Kuang, and L. Yu, “Hard-decision quantization algorithm based on deep learning in intra video coding,” 2019 Data Compression Conference (DCC), pp.607–607, IEEE, 2019.

[11] D. Kianfar, A. Wiggers, A. Said, R. Pourreza, and T. Cohen, “Parallelized rate-distortion optimized quantization using deep learning,” Oct. 21 2021. US Patent App.17/070, 589.

[12] K. Sühring, M. Schäfer, J. Pfaff, H. Schwarz, D. Marpe, and T. Wiegand, “Trellis-coded quantization for end-to-end learned image compression,” 2022 IEEE International Conference on Image Processing (ICIP), pp.3306–3310, IEEE, 2022.

[13] H. Kirchhoffer, P. Haase, W. Samek, K. Müller, H. Rezazadegan-Tavakoli, F. Cricri, E.B. Aksu, M.M. Hannuksela, W. Jiang, W. Wang, S. Liu, S. Jain, S. Hamidi-Rad, F. Racapé, and W. Bailer,

"Overview of the neural network compression and representation (nnr) standard," IEEE Trans. Circuits Syst. Video Technol., vol.32, no.5, pp.3203–3216, 2021.
[14] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand, "Hybrid video coding with trellis-coded quantization," 2019 Data Compression Conference (DCC), pp.182–191, IEEE, 2019.
[15] J. Wang, X. Yu, D. He, F. Henry, and G. Clare, "Multiple sign bits hiding for high efficiency video coding," 2012 Visual Communications and Image Processing, pp.1–6, IEEE, 2012.
[16] W. Niu, X. Huang, H. Yin, Y. Lu, Y. Zhou, and C. Yan, "Fast all zero block detection algorithm for versatile video coding," Multimedia Tools and Applications, pp.33693–33718, 2023.
[17] M. Wang, S. Wang, J. Li, L. Zhang, Y. Wang, S. Ma, and S. Kwong, "Low complexity trellis-coded quantization in versatile video coding," IEEE Trans. Image Process., vol.30, pp.2378–2393, 2021.
[18] J. Adhuran, G. Kulupana, C. Galkandage, and A. Fernando, "Multiple quantization parameter optimization in versatile video coding for 360° videos," IEEE Trans. Consum. Electron., vol.66, no.3, pp.213–222, 2020.
[19] K. Liu, D. Liu, L. Li, and H. Li, "Context-adaptive inverse quantization for inter-frame coding," IEEE Open Journal of Circuits and Systems, vol.2, pp.660–674, 2021.
[20] V. Sze, M. Budagavi, and G.J. Sullivan, "High efficiency video coding (hevc)," Integrated Circuit and Systems, Algorithms and Architectures, p.40, Springer, 2014.

Qinghua Sheng received the M.S. degree from Xidian University in 2003. He is currently pursuing the Ph. D. degree in Hangzhou DianZi University. His research interests include FPGA acceleration, electronic system integration, and video coding.

Yu Cheng received the B.S. degree from Hangzhou DianZi University in 2021. He is currently pursuing the M.S. degree in Hangzhou DianZi University. His research interests include FPGA acceleration and video coding.

Xiaofang Huang received the M.S. degree from Hangzhou Dianzi University in 2009. Her research interests include video coding and electronic system integration.

Changcai Lai received the Ph. D. degree from Northwestern Polytechnical University in 2007. His research interests include video coding and chip architecture design.

Xiaofeng Huang received the Ph. D. degree from Peking University in 2016. His research interests include video coding and chip architecture design.

Haibin Yin received the Ph. D. degree from Shanghai Jiao Tong University in 2007. His research interests include video coding and chip architecture design.