Parallelized RDOQ Algorithm and Fully Pipelined Hardware Architecture for AVS3 Video Coding

Xiaofeng Huang, Ran Tang, Rui Pan, Haibing Yin, Zhao Wang, Shiqi Wang, Siwei Ma

***Abstract*—The rate-distortion optimized quantization (RDOQ) provides significant coding gain in the third generation of Audio Video coding Standard (AVS3). However, the high computational complexity and strong data dependency in RDOQ impede the hardware implementation. To address these issues, we propose a zig-zag scanline-level parallelized RDOQ algorithm and its fully pipelined hardware architecture for AVS3 video coding. For algorithm optimization, we update the run-level context for rate estimation in the inner zig-zag scanline and propose an efficient RD cost calculation form in the optimal coefficient level (OCL) decision step. In the last significant coefficient (LSC) position decision step, a greedy strategy based algorithm is proposed to optimize the determination process in parallel. Moreover, the proposed parallelized RDOQ algorithm is accelerated by single instruction multiple data (SIMD) on the Intel X86 platform. For hardware architecture design, a fully pipelined hardware architecture is proposed with nine pipeline stages. This design can process multiple transform units in parallel when the height is less than 32. Experimental results show that the proposed algorithm achieves 31.42%, 28.60%, and 28.58% time-saving by 0.23%, 0.24%, and 0.26% Bjøntegaard delta rate (BD-Rate) increase on average under all intra (AI), random access (RA), and low delay B (LDB) configurations, respectively. The hardware implementation achieves 32 coefficients per cycle, and the area consumption is 1223.2K logic gates when working at 471.2MHz. It is proven that the proposed algorithm and hardware architecture design achieve a good trade-off between coding efficiency and hardware throughput.**

***Index Terms*—RDOQ, AVS3, zig-zag scanline, parallelized algorithm, hardware architecture.**

I. Introduction

With the growing popularity of ultra-high definition (UHD) video content such as 4K and 8K, the amount of video data has shown an explosive increment. According to Cisco's report, video data accounted for 82% of all mobile data traffic in 2022, up from 59% in 2017 [1]. This has resulted in an urgent need for more advanced video compression coding tools that can provide high compression efficiency to alleviate the burden of video data transmission and storage. To address this challenge, several standard groups have established novel video coding standards. The Joint Video Exploration Team (JVET) formed by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) has developed several video coding standards [2]–[4], and the latest one is called Versatile Video Coding (VVC) [5]. The AVS working group officially initiated the next-generation video coding standard - AVS3 in March 2018 and finalized the first phase of AVS3 in 2019 [6].

The AVS3 standard is an improvement over its predecessor AVS2, which aims to provide better video compression capability. Several advanced coding tools have been adopted in AVS3. In block partitioning, AVS3 extends the quadtree partition in AVS2 to a more flexible partition mechanism, including quadtree, binary tree [7], and extended quadtree partition [8]. In intra-coding, new coding tools including intra-derived tree, intra-prediction filter, and two-step cross-component prediction mode are specially designed for AVS3. In inter coding, AVS3 introduces adaptive motion vector resolution, enhanced motion vector resolution, and affine motion mode [9]. In transform coding, more transform types like DST-VII and DCT-VIII are supported in AVS3 [10]. These advanced coding tools make AVS3 show a significant improvement over AVS2 in terms of compression efficiency.

In addition to the advanced coding tools mentioned above, AVS3 adopts rate-distortion optimized quantization (RDOQ) to improve coding efficiency further. RDOQ offers significant performance gains compared to conventional uniform scalar quantization (USQ). The coding gain is achieved by searching for the optimal coefficient level (OCL) and last significant coefficient (LSC) position in a rate-distortion (RD) optimal way. Specifically, to determine the OCL for each transform coefficient, RDOQ repeatedly computes the RD cost for all possible candidate levels. The rate cost is computed sequentially due to entropy coding with context model update. Similarly, the decision-making process for the LSC position is also sequential as all the quantized coefficients are searched one by one. The position with the minimum RD cost is chosen as the optimal one. The coefficients are accessed in zig-zag scan order during the RDOQ process, which yields memory access conflicts when trying to parallelize in hardware implementation. As a result, the high computational complexity of RDOQ and its sequential processing characteristics pose challenges for real-time video coding applications.

This work was supported in part by the National Natural Science Foundation of China under Grant 61901150, 61931008, and 61972123, and in part by the National Key R&D Program of China under Grant 2021ZD0109800.

Xiaofeng Huang and Haibing Yin are with the School of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China, and also with the Advanced Institute of Information Technology, Peking University, Hangzhou 311215, China (e-mail: {xfhuang; yhb}@hdu.edu.cn).

Ran Tang is with the School of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: rtang@hdu.edu.cn).

Rui Pan is with the School of Electronics and Information, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: 222040188@hdu.edu.cn).

Shiqi Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: shiqwang@cityu.edu.hk).

Zhao Wang and Siwei Ma are with the School of Computer Science, Peking University, Beijing 100871, China (e-mail: {zhaowang; swma}@pku.edu.cn).

Over the past decade, numerous researchers have attempted to simplify the RDOQ process in different ways to reduce its computational complexity [11]–[14]. These optimization methods can be classified into two main categories, bypass RDOQ and fast RDOQ. Bypass RDOQ skips the RDOQ process entirely by utilizing techniques such as all zero block (AZB) detection. In contrast, fast RDOQ seeks to simplify the computation process within RDOQ. From another perspective, existing methods can be classified as software or hardware-based, depending on the target optimization platform. The research on RDOQ is summarized in Fig. 1.

In the bypass RDOQ methods, pieces of research skip the redundant process by pre-detecting AZBs to reduce the computational complexity [15]–[18]. In [19], [20], these two works detected genuine all-zero block (GZB) and pseudo all-zero block (PZB) to early terminate the process. Work [19] introduced a sum of absolute difference (SAD) lower bound as a threshold to detect GZB and proposed an RD estimation method to detect PZB. To resolve the problem that distortion and rate estimation models for PZB determination tend to be sensitive to the initial values in work [19], work [20] proposed a novel PZB detection method, where rate distortion optimization (RDO) with adaptive RD estimation is employed. Similarly, works [21], [22] utilized the maximum transform coefficient in the transform block as the threshold for fast AZB detection. To improve the detection accuracy, multi-stage AZB detection methods were proposed in [23], [24]. For the coefficient-level RDOQ bypass method, works [25], [26] proposed an adaptive dead zone offset to adjust the quantization level from one to zero at the high-frequency domain in large transform blocks. These algorithms have shown significant time savings under specific conditions. However, their processing logic remains redundant when the conditions are not met. Consequently, these bypass RDOQ methods are of limited use for real-time hardware RDOQ applications, as the complexity is still the bottleneck when implemented in hardware.

In the fast RDOQ methods, they focus on the simplification of RD calculation. Rate estimation in RDOQ is usually derived by the entropy coding with context model update which impedes parallel computing. To resolve this problem, work [27] proposed a rate model through an analytical method to speed up the RDOQ, and it can predict the bitrate by different entropy coding methods. However, the method [27] is no longer suitable for HEVC as HEVC has more complex transform types. Therefore, work [28] proposed a low-complexity RDOQ optimization algorithm based on a hybrid Laplacian model. An alternative approach is to calculate the difference of RD cost between two candidate quantized levels, rather than calculating the two RD costs separately [29]. Similarly, work [30] built a low-complexity ΔJ model based on syntax elements involved in the rate estimation. Moreover, works [31] and [32] accelerated the RDOQ process utilizing deep learning methods. Although these algorithms have reduced computational complexity significantly, they have not adequately addressed the data dependency problem. The coefficients in RDOQ still require sequential processing, which hinders parallelized hardware acceleration.

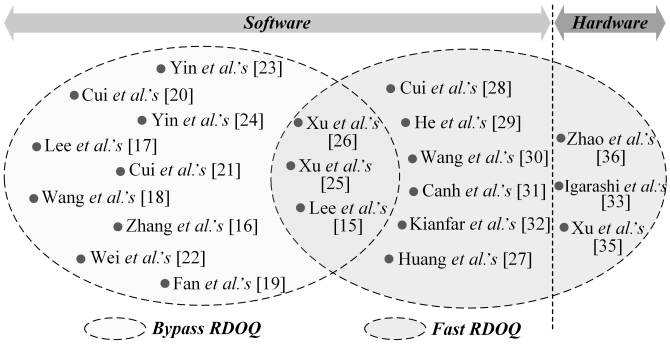


Fig. 1. Summary of the existing research on RDOQ.

From Fig. 1, it is evident that most of the current fast algorithms are developed for software platforms, while only a few are designed for hardware platforms. Work [33] parallelized the RD cost calculation by utilizing the entropy at the previous frame and utilized a bi-directional parallel scan to accumulate the RD cost for the LSC position search. It can achieve 4K@60fps real-time processing on GPU, but it brings a significant 2.6% increase in Bjøntegaard delta Rate (BD-Rate) [34]. In [35], a hardware-friendly fast RDOQ algorithm for AVS3 was proposed. It eliminates the data dependency during rate estimation, but the optimization of the LSC position search has not been addressed. In work [36], a zig-zag scanline-based RDOQ algorithm is proposed. However, to facilitate hardware implementation, it fixed the context during RDOQ processing, which resulted in some BD-Rate increase. These hardware-oriented methods take the data dependency problem into full consideration and utilize straightforward methods like fixed context. These methods achieve parallelization but at the cost of significant coding performance loss.

In this work, we propose a novel parallelized RDOQ algorithm and its fully pipelined hardware architecture for the AVS3 video encoder. Our primary objectives are to address the following three challenges: 1) achieving a high degree of data parallelism while minimizing coding loss, 2) resolving data access mismatch caused by zig-zag order in RDOQ, 3) designing a parallelized hardware architecture to support UHD applications. Compared with the vanilla HPM-4.0 software [37], the parallelized RDOQ algorithm reduces 31.42% RDOQ calculation time with a 0.24% BD-Rate loss. The proposed fully pipelined hardware architecture takes advantage of the parallelized RDOQ algorithm, which can achieve parallelism with 32 coefficients per cycle and have a higher throughput with less area consumption than previous work [36]. The main contributions of this work are summarized as follows.

- In the OCL decision step, the run-level context dependency for the rate estimation is optimized to the inner zig-zag scanline level, and an efficient RD cost form is proposed to optimize bit-width and rate cost calculation. It simplifies the computational complexity of OCL decision and parallelized the calculation at the scanline level with only a slight BD-Rate loss.
- In the LSC position decision step, we propose a greedy strategy based parallelized algorithm, which searches the sub-optimal LSC position i_k^* at a scanline k, and

determines the optimal LSC position i^* from these i_k^* candidates. It enables parallelized computation without any performance degradation for the LSC position decision.

- In the whole RDOQ flow, the scan order of the RDOQ is changed from zig-zag order to column-first order, which can address the memory conflict problem caused by zig-zag scan. Moreover, the proposed algorithm is further accelerated by single instruction multiple data (SIMD) on the Intel X86 platform.
- The proposed fully pipelined hardware architecture design is implemented with nine pipeline stages. The parallelism is 32 coefficients per cycle, and it can process multiple transform units (TUs) when the height is smaller than 32. The results show that our work achieves a good trade-off between coding efficiency and hardware throughput.

The rest of this paper is organized as follows. The RDOQ process in AVS3 and its hardware design challenges are briefly described in Section II. Section III presents the proposed parallelized RDOQ algorithm, and Section IV gives the hardware architecture design and timing diagrams. The experiment results and comparisons under the common test conditions are shown in Section V, followed by the conclusion and future work in Section VI.

II. Rate-Distortion Optimized Quantization in AVS3

In this section, we will first provide an overview of the AVS3 RDOQ process, followed by an analysis of the challenges that arise during hardware design.

A. AVS3 RDOQ Overview

RDOQ is a soft quantization decision technique that aims to find out the optimal quantized coefficient level considering a trade-off between rate and distortion [36], [38]. The flow of the RDOQ in AVS3 is shown in Fig. 2. This flow is similar to RDOQ in HEVC [28] but differs in two crucial ways. One is that the TU level zig-zag scan order is adopted by AVS3, instead of the coefficient group (CG) scan in HEVC. The other is that AVS3 supports rectangular blocks such as 16×8 and 64×8. The detailed AVS3 RDOQ flow in Fig. 2 consists of four internal components, which are pre-quantization, OCL decision, LSC position decision, and uncoded coefficient zero assignment.

In the pre-quantization step, each transform coefficient c_i located at the zig-zag scan index i in the TU undergoes a scalar quantization which is formulated as,

$$l_i = Round\left(\frac{|c_i|}{Q_{step}} + f\right), \tag{1}$$

where Q_{step} is the quantization step size determined by the quantization parameter (QP), f is the rounding offset, and l_i is the initial scalar quantization level.

In the OCL decision step, the optimal quantization level l_i^* is determined by comparing the RD cost of available candidate levels L_i. The derivation of L_i is listed in Table I. The level

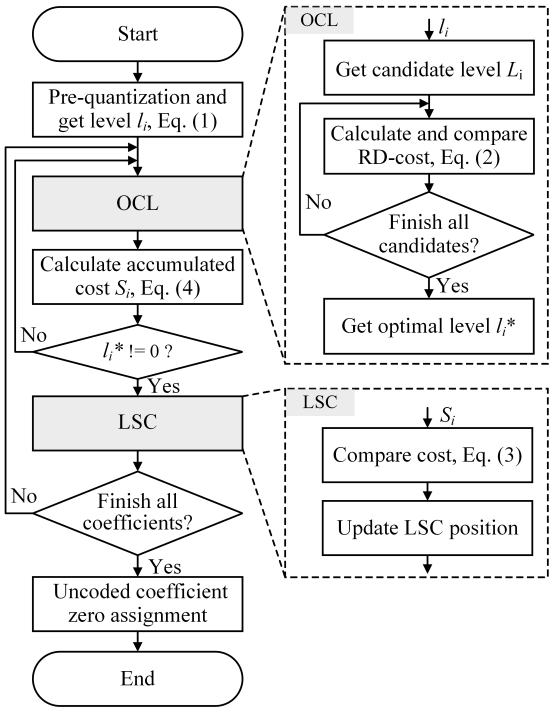


Fig. 2. Flowchart of the RDOQ in AVS3.

with the minimum cost is selected as the optimal choice. The OCL process is detailed as follows,

$$l_i^* = \arg\min_{L_i} J, \quad where \quad J = D + \lambda \cdot R, \tag{2}$$

where R and D are the rate and distortion, λ is the Lagrangian multiplier associated with the QP, and J is the calculated RD cost, respectively. Specifically, the rate R is derived through entropy coding with a context model update. In the AVS3 reference software HPM-4.0, the rate R is estimated using a look-up table, where the table index is determined based on the run-level context update [39], [40].

Then in the LSC position decision step, the optimal LSC position i^* is derived based on the sum of RD cost. The algorithm scans through each coefficient position i in zig-zag order and determines the optimal position by the following formula,

$$i^* = \arg\min_{i \in \{0,\ldots,W\times H-1\}} \left(S_i + E_{i,lsc}\right), \tag{3}$$

where W and H represent the width and height of the TU, $E_{i,lsc}$ denotes the rate cost of the synatx elemenet $coeff\_last$

TABLE I
DERIVATION OF CANDIDATE LEVELS

l_i	candidate levels (L_i)
0	N/A
1	0, 1
2	0, 1, 2
...	...
N	0, N-1, N

with the assumption that i is the LSC position, and S_i denotes the accumulated cost of coefficient positions from zero to i, respectively. The S_i is formulated as follows,

$$S_i = \sum_{j=0}^{i} (\Delta J_j), \ where \ \Delta J_j = J_{j,l_j^*} - J_{j,0} + E_{j,non\_lsc} \tag{4}$$

where J_{j,l_j^*} is the RD cost when the coefficient level at position j is quantized to l_j^*, $J_{j,0}$ is the distortion cost when the coefficient level is quantized to zero at position j, and E_{j,non_lsc} is the rate cost of the syntax $coeff\_last$ when the position j is not the LSC position, respectively. The value of E_{j,non_lsc} is set to zero when j is equal to i or l_j^* is equal to zero.

Finally, all non-zero coefficients after the LSC position should be set to zero in the uncoded coefficient zero assignment step.

B. Hardware Design Challenge Analysis

RDOQ determines the optimal quantization level based on an RD optimal perspective, which can achieve higher compression efficiency compared to USQ. The process of the RDOQ in AVS3 is shown in Fig. 2, which consists of four steps. The hardware design challenges for these steps are analyzed in this subsection.

The pre-quantization and uncoded coefficient zero assignment steps in Fig. 2 are well suitable for hardware implementation. This is because they do not involve any data dependency. The pre-quantization step can be implemented in parallel since all coefficients can be quantized independently using Eq. (1). Similarly, The uncoded coefficient zero assignment process can also be parallelized, where the coefficient level at location i is set to zero when i is larger than i^*. Moreover, this step can be merged into the inverse quantization process, which can lead to a reduction in pipeline latency [41].

In the OCL decision step, the RD calculations using Eq. (2) are executed to get the optimal level l_i^*. It can be parallelized as the input context for rate estimation is identical for all candidate levels in L_i. However, it must be processed serially between coefficient positions i and $i+1$. It is because the run-level context for the next coefficient $i+1$ depends on the optimal level l_i^*. Thus, the next coefficient cannot start until the l_i^* is decided and the context is updated. Similarly, the LSC position decision step cannot be implemented in parallel as the optimal i^* derived from Eq. (3) is searched serially for all coefficient positions in zig-zag order. The serial processing hinders the throughput of RDOQ and thereby reduces its overall efficiency.

Based on the above analysis, the data dependencies in the OCL decision and LSC position decision steps pose challenges to the parallel hardware implementation of RDOQ. Assuming that the pixel parallelism is 1 coefficient/cycle, processing a 32×32 TU will require a total of 1024 clock cycles. This is far from pixel parallelism in the hardware implementation of discrete cosine transform (DCT), where the parallelism can be 32 coefficients/cycle [42]–[44]. This pixel parallelism mismatch will reduce the throughput of the pipelined hardware design of the mode decision in video coding, where RDOQ is an essential stage in the mode decision pipeline [45]–[48].

Several methods have been proposed to address the data dependency problem in RDOQ [33], [35], [36]. They primarily aim to address the dependency issue caused by context update-based rate estimation, utilizing approaches such as fixed context. This allows the OCL decision process to be implemented in parallel between different coefficient positions, at the expense of some coding performance loss. Despite this, it is still challenging to parallelize the LSC position decision process due to two factors. One is that the zig-zag scan order searching in Eq. (3) hinders the parallelism. The transform coefficients output by the previous stage DCT in mode decision pipeline is column first style, which means the coefficients at positions (0, 2, 3, 9, ...) are output in one cycle [43]. If we try to parallelize the coefficients at positions from i to $i+3$ in one cycle, the coefficients c_i need to be reshaped from column-first style to zig-zag first style. This reshaping process will consume additional clock cycles and memory resources. The other is that the calculation of the accumulated cost S_i impedes the parallelism. If we try to achieve parallelism with 4 coefficients/cycle, S_i, S_{i+1}, S_{i+2}, and S_{i+3} need to be derived in one cycle. It is obvious that the computational logic is too heavy, and it will become the critical path resulting in worse timing and lower working frequency.

III. Proposed parallelized RDOQ Algorithm

In this section, we will present a parallelized RDOQ algorithm to speed up the process. In the OCL decision step, the run-level context for rate estimation is updated in the inner zig-zag scanline, and the RD cost formula presented in Eq. (2) is optimized to a more efficient form. In the LSC position decision step, a greedy strategy based algorithm is proposed, which enables the implementation of Eq. (3) in parallel between zig-zag scanlines. Finally, we summarize the parallelized RDOQ algorithm and accelerate the entire process using a SIMD approach.

A. Parallelized OCL decision

As described in subsection II-B, the run-level context update during rate estimation hinders the parallelized implementation of the OCL decision step. Specifically, this dependency arises from the use of variables $prev\_level$ and run in HPM-4.0. Here, $prev\_level$ represents the former non-zero optimal quantization level, and run denotes the number of zeros between two non-zero quantization coefficients. For example, if the optimal coefficient levels from positions 0 to 4 are (15, 1, 0, 0, 0), the values of $prev\_level$ and run for position 5 are 1 and 3, respectively. In this subsection, we propose a simplified derivation of these two variables in order to enable the parallelization of the OCL decision step.

Before illustrating the algorithm, we analyze the zig-zag scan in Fig. 5(a). It is observed that the zig-zag scan follows two different directions, either the down-left scan or the up-right scan. These two different scans have different properties in terms of deriving the variables $prev\_level$ and run. For coefficients located on the down-left scanline, the calculation

of these two variables depends on the coefficients with a column index greater than the current index. On the other hand, for coefficients located on the up-right scanline, the calculation depends on the coefficients with a column index less than the current one. As illustrated in subsection II-B, the transform coefficients are output from the DCT in the column-first order, meaning that coefficients with smaller column indices are output first. Therefore, different strategies are required to adapt to the two different scan directions.

For the value of variable $prev\_level$ at a zig-zag scan index i, the simplified derivation is formulated as below,

$$prev\_level_i = \begin{cases} 1, & if\ (u == 0)\ or\ (v == H^{'}) \\ l_i, & elif\ ((u+v)\&1 == 1)\ and\ (!!l_i) \\ l^*_{i-1}, & elif\ ((u+v)\&1 == 0)\ and\ (!!l^*_{i-1}) \\ 1, & else \end{cases}, \tag{5}$$

where the (u, v) is the coordinate index of the zig-zag scan position i, $H^{'}$ is the TU height H minus one, l_i is the pre-quantization level at position i, l^*_{i-1} is the OCL at previous position $i-1$, and the label ! represents the not operation, respectively.

Similarly, the derivation of the variable run at index i for the up-right scanline ($(u+v)\&1$==0) is formulated as Eq. (6). The derivation for the run_i at the down-left scanline ($(u+v)\&1$==1) is analogous to the Eq. (6), except that it is calculated based on l^*_{i+1} and run_{i+1}.

$$run_i = \begin{cases} 0, & if\ ((!!u)\ or\ (v == H^{'}))\ and\ (!!l_i) \\ 1, & elif\ ((!!u)\ or\ (v == H^{'}))\ and\ (!l_i) \\ 0, & elif\ (!!l^*_{i-1}) \\ run_{i-1}+1, & else \end{cases}. \tag{6}$$

Based on the above simplification, it is observed that the derivation of variables $prev\_level_i$ and run_i exhibits the data dependency within a scanline, and it relies on the coefficients where the column index u is smaller than the current column. This implies that the OCL decision process for coefficients at different rows v in one column can be executed in parallel, and the coefficients at different columns should be processed serially. This method can not only enhance the parallelism and preserve the context update for rate estimation but also ensure that the access aligns with the column-first order from DCT.

Furthermore, the RD cost calculation in Eq. (2) is optimized to a more efficient form, which is expressed below,

$$J_f = \frac{D}{\lambda} + R, \tag{7}$$

where the D is the distortion that requires multiplication by the scaling factor err_scale in HPM-4.0. The err_scale is correlated with the QP and TU size and remains constant for a TU. Therefore, the err_scale and λ can be combined to optimize the division operation in Eq. (7) to a multiplication-shift operation. The adoption of J_f brings two benefits. One is that it requires a lower bit-width for the representation of RD cost, which is beneficial for the chip area. In our design, the bit-width for J_f is limited to 34. The other is that it eliminates the multiplication operation for the rate cost. Compared to the independent calculation for distortion, there are still some data dependencies in the rate estimation according to Eq. (5) and Eq. (6). This form benefits the hardware design of rate calculation.

B. Greedy strategy based parallelized LSC position decision

As shown in Fig. 2 and Eq. (3), the optimal last LSC position i^* is determined by searching the minimum accumulated RD cost. The cost is accumulated in the zig-zag scan order, and its hardware design challenges are illustrated in subsection II-B. To overcome these challenges, a greedy strategy based LSC position decision algorithm is proposed. In this algorithm, the sub-optimal LSC position i^*_k at a scanline k is derived first as follows,

$$\bigcup_{k=0}^{n} i^*_k = \underset{i\in\{b(k),\ldots,e(k)\}}{\arg\min} \left(\widetilde{S}_i + E_{i,lsc}\right), \tag{8}$$

and

$$\widetilde{S}_i = \sum_{j=b(k)}^{i} \left(\Delta J_j\right), \tag{9}$$

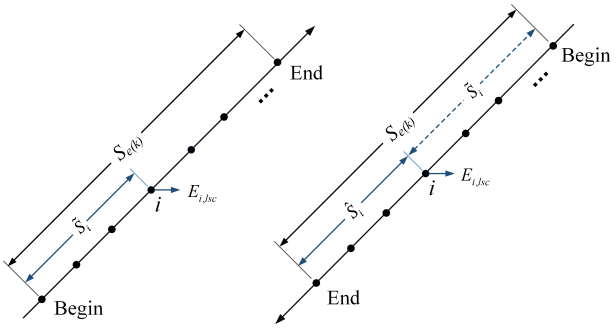
where n is the total number of scanlines in the TU which is equal to $W+H-1$, $\widetilde{S}_i$ is the accumulated cost similar to Eq. (4), and $b(k)$ and $e(k)$ represent the begin and end position on the k-th scanline, respectively. For example, the $b(k)$ and $e(k)$ are 6 and 9 for the 3-rd scanline in Fig. 5(a), respectively.

Then, the optimal i^* is derived from the candidate positions i^*_k, and the formula is shown in Eq. (10). The decision by Eq. (3) can be converted to Eq. (10), where the optimal position decision in the whole TU is divided into comparisons among local scanline optimal positions. And, the optimal last coefficient derived by Eq. (3) is the same as Eq. (10). This is because the RD cost J is calculated with inner-scanline dependency and does not have any inter-scanline dependency resulting from the OCL decision optimizations in subsection III-A.

$$\begin{aligned} i^* &= \underset{i\in\{0,\ldots,W\times H-1\}}{\arg\min} \left(S_i + E_{i,lsc}\right) \\ &= \underset{i\in\left\{\bigcup_{k=0}^{n} i^*_k\right\}}{\arg\min} \left(S_i + E_{i,lsc}\right). \end{aligned} \tag{10}$$

Eq. (8) and Eq. (10) are the keys to our greedy strategy based LSC position decision. It indicates that the optimal i^* is derived from the sub-optimal LSC position i^*_k. Furthermore, the derivation of i^*_k can be parallelized across different scanlines, as the J, $E_{j,lsc}$, and E_{j,non_lsc} in Eq. (8) and Eq. (9) are calculated independently between scanlines.

$$\begin{aligned} i^*_k &= \underset{i\in\{b(k),\ldots,e(k)\}}{\arg\min} \left(\widetilde{S}_i + E_{i,lsc}\right) \\ &= \underset{i\in\{b(k),\ldots,e(k)\}}{\arg\min} \left(\sum_{j=b(k)}^{i} \left(\Delta J_j\right) + E_{i,lsc}\right) \\ &= \underset{i\in\{b(k),\ldots,e(k)\}}{\arg\min} \left(\sum_{j=b(k)}^{e(k)} \left(\Delta J_j\right) - \sum_{j=i}^{e(k)} \left(\Delta J_j\right) + E_{i,lsc}\right) \\ &= \underset{i\in\{b(k),\ldots,e(k)\}}{\arg\min} \left(\widetilde{S}_{e(k)} - \hat{S}_i + E_{i,lsc}\right). \end{aligned} \tag{11}$$



(a) Up-right scan (b) Down-left scan

Fig. 3. The illustration of the accumulated cost variables S.

There is a problem with the derivation of the i_k^* for the down-left scanline, as it requires accessing the coefficients c_i in right-to-left column order. However, the coefficients are generated in inverse order from the DCT. To resolve this problem, the i_k^* derivation for the down-left scanline is formulated as Eq. (11). In this equation, the $\widetilde{S}_{e(k)}$ is the accumulated cost for the entire scanline, which is a constant value for all coefficients in the scanline. Thus, the $\widetilde{S}_{e(k)}$ can be eliminated for i_k^* calculation. With this optimization, Eq. (11) uses cost J for coefficients from left to right order, which aligns with the order of the DCT output. To help understand these variables, an illustration is shown in Fig 3 for the up-right scanline and down-left scanline, respectively.

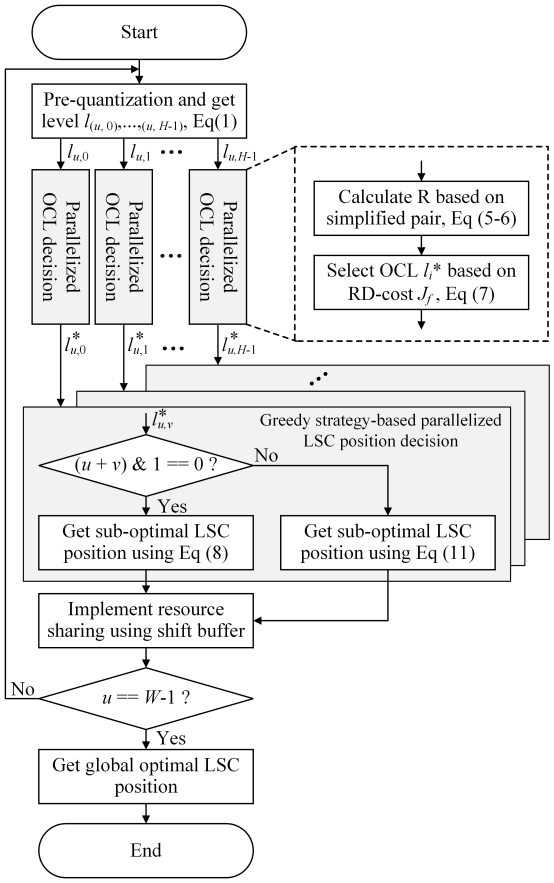


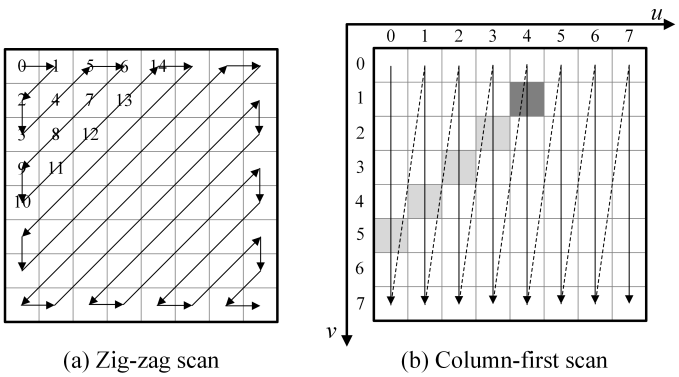
Fig. 4. The flowchart of the proposed parallelized RDOQ algorithm.

For the optimal i^* derived based on Eq. (10), it will need to buffer all suboptimal i_k^* and then compare their accumulated cost to decide the best. Since there are a total of $W+H-1$ scanlines, this means that it will need to buffer $W+H-1$ times resources. To save the buffer, an on-the-fly decision approach is proposed. When the LSC position decision processes the coefficients at column u, the i^* for all previous $u-1$ scanlines is derived simultaneously. The accumulated cost for the optimal i_{u-1}^* can be calculated using Eq. (12). Then, the optimal i^* for all $u-1$ scanlines can be derived by comparing with the minimum accumulated cost. It is noted that the remaining $H-1$ scanlines should be added to the comparison when all W scanlines finish the on-the-fly process. With this optimization, it will need to only buffer H times resources. The H times resources are shared by all $W+H-1$ scanlines and are implemented using a shift buffer.

$$J_{i_{u-1}^*} = \left(\sum_{k=0}^{u-2} \widetilde{S}_{e(k)} + \widetilde{S}_{i_{u-1}^*} + E_{i_{u-1}^*,lsc} \right). \tag{12}$$

C. Proposed parallelized RDOQ algorithm

As described previously, we propose a parallelized RDOQ scheme that consists of two components: a parallel OCL decision and a greedy strategy based parallel LSC position decision. An overview of the flowchart is shown in Fig. 4. In the parallel OCL decision, we calculate the simplified $(run, prev\_level)$ pair to remove partial dependency for rate estimation and propose an efficient RD cost form J_f to reduce rate cost computation and save chip area. In the greedy strategy based parallel LSC position decision, the optimal i^* is derived based on the suboptimal i_k^* for each scanline k, where the derivation of i_k^* can be executed in parallel for each scanline. Moreover, a shift buffer is utilized to save the buffer. With these optimizations, the scan order of the RDOQ can be changed from zig-zag order to column-first order, which aligns with the output of DCT. The data dependency only exists within the scanline, which means that all coefficients in the same column can be processed in parallel. This is illustrated in Fig. 5. We provide a detailed description of the proposed parallelized RDOQ algorithm in Algorithm 1.



(a) Zig-zag scan (b) Column-first scan

Fig. 5. Diagram of the scan order conversion from (a) zig-zag order to (b) column-first order. In (b), the dark gray coefficient depends on the previous light gray coefficients in the inner zig-zag scanline.

Algorithm 1 The proposed parallelized RDOQ algorithm

Input: $c_{0,..,W\times H-1}$, W, H, λ.
Output: $l^*_{0,..,W\times H-1}$, i^*.

TEMP BUFFER: $S_{best}[H]$, $E_{best}[H]$, $S_{all}[H]$, $i^*_{best}[H]$;
for all $u = 0, 1, \cdots, W-1$ **do**
  for all $v = 0, 1, \cdots, H-1$ **do**
    Get c_i, zig-zag index i is derived based on (u, v);
    Calculate l_i based on Eq. (1);
    Calculate $prev\_level_i$ and run_i based on Eq. (5) and Eq. (6), respectively;
    Calculate R, where R = f ($prev\_level_i$, run_i, l_i);
    Get optimal level l_i^* by $\arg\min_{L_i}(J_f)$.
    if $u == 0$ or $v == W-1$ **then**
      Initial: $\widetilde{S}_{best}[v] \leftarrow \Delta J_v$, $E_{best}[v] \leftarrow E_{v,lsc}$, $S_{all}[v] \leftarrow 0$, $i^*_{best}[v] \leftarrow i$;
    else
      if $(u+v)$ & 1 **then**
        $\hat{S}_i$ = $S_{all}[v]$ + ΔJ_v; // down-left scanline
        if $-\hat{S}_i + E_{i,lsc} < -S_{best}[v] + E_{best}[v]$ **then**
          $S_{best}[v] \leftarrow \hat{S}_i$, $E_{best}[v] \leftarrow E_{i,lsc}$, $i^*_{best}[v] \leftarrow i$;
        end if
        $S_{all}[v] \leftarrow \hat{S}_i$;
      else
        $\widetilde{S}_i$ = $S_{all}[v]$ + ΔJ_v; // up-right scanline
        if $\widetilde{S}_i + E_{i,lsc} < S_{best}[v] + E_{best}[v]$ **then**
          $S_{best}[v] \leftarrow \widetilde{S}_i$, $E_{best}[v] \leftarrow E_{i,lsc}$, $i^*_{best}[v] \leftarrow i$;
        end if
        $S_{all}[v] \leftarrow \widetilde{S}_i$;
      end if
    end if
  end for
  Select the optimal i^* for u scanlines according to Eq. (12);
  for all $v = 1, 2 \cdots, H$ **do**
    Implement resource sharing using shift buffer: $S_{best}[v-1] \leftarrow S_{best}[v]$, $E_{best}[v-1] \leftarrow E_{best}[v]$, $S_{all}[v-1] \leftarrow S_{all}[v]$, $i^*_{best}[v-1] \leftarrow i^*_{best}[v]$;
  end for
end for
for all $t = 1, 2, \cdots, H-1$ **do**
  Compare the remaining cost to determine the global optimal i^*;
end for

D. SIMD-based fast RDOQ

Although the algorithm proposed in subsection III-C can be run in parallel for all coefficients within the same column, it may take longer execution time on the Intel X86 platform as each coefficient is processed without skipping. A SIMD-based approach is adopted to address this issue, and the implementation and design are explained below.

During the pre-quantization step, each transform coefficient is processed independently. In the SIMD optimization, we execute four coefficients simultaneously by utilizing intrinsics.

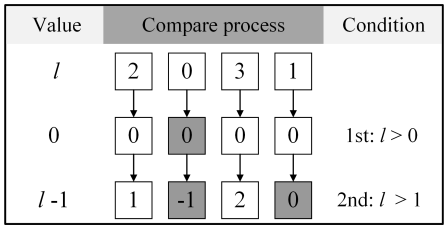


Fig. 6. An example of OCL decision process using SIMD.

Similarly, the parallelism is four coefficients per instruction in the OCL decision step, consistent with the parallelism of the pre-quantization step. For the decision of l_i^*, there may have different numbers of candidate level L_i for each of the four parallelized coefficients, which will impact the parallelization by intrinsic due to its irregularity. To address this issue, all coefficients will calculate the RD cost for three candidate levels as shown in Fig. 6. It should be noted that the candidate levels for the gray color block will not participate in the optimal level l_i^* comparison. For the intrinsic of minimum cost comparison, the PCMPGTD instruction is used and the result is bit-inverted by the VPANDN instruction.

In the LSC position decision step, the pixel parallelism is four coefficients similar to the previous steps. The shift buffer is adopted to save the buffer as illustrated in subsection III-B. Rotation instruction is needed to implement this behavior. We use the VPSRLDQ, Extract, and Insert operations to rotate the data. VPSRLDQ instruction shifts the data in the 128-bit channel by a specified byte, the extract operation extracts the data in the channel, and the insert operation inserts the data into a specified location, respectively.

IV. FULLY PIPELINED HARDWARE ARCHITECTURE DESIGN

It is a fact that the RDOQ algorithm shows high computational complexity and strong data dependency, which poses significant challenges for real-time video coding. In the previous section III, a parallelized RDOQ algorithm is proposed to resolve these problems. Building upon that, this section focuses on the development of a fully pipelined and high-throughput hardware architecture for the parallelized algorithm.

A. Overall Architecture

The TU size in the AVS3 standard can range from 4×4 to 64×64. However, to reduce the coefficient coding for large TU size, only the first 32 rows/columns coefficients are kept and the high-frequency coefficients are set to zero. Consequently, the width W and height H of the TU for RDOQ must not exceed 32. To accommodate this constraint, the RDOQ hardware architecture is designed with a pixel parallelism of 32, which means that a maximum of 32 coefficients within the same column can be processed simultaneously.

The overall architecture of the hardware design is shown in Fig. 7. It comprises nine pipeline stages, with the pre-quantization step occupying stages $S1$ and $S2$, the OCL decision step occupying stages $S2$, $S3$, and $S4$, and the LSC

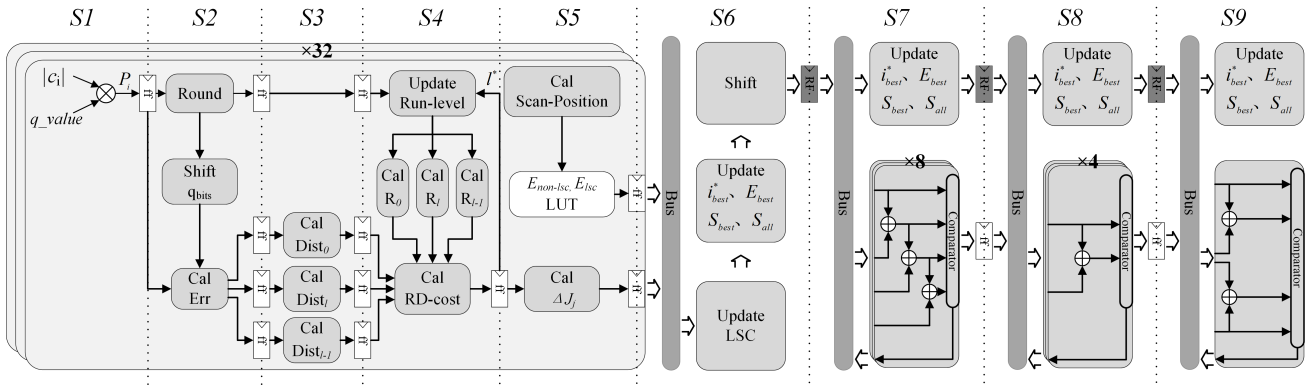


Fig. 7. The hardware architecture design of the parallelized RDOQ algorithm.

decision step occupying the remaining five stages. Stages $S1$ to $S6$ are on-the-fly processing for each column, which implements the for-loop from line 2 to 31 in Algorithm 1. Stages $S7$ to $S9$ decide the final optimal i^*, which implements the for-loop from line 32 to 34 in Algorithm 1. The comparator is employed to get the optimal i^*. To accommodate various TU sizes, stage $S7$ utilizes eight four-input comparators, stage $S8$ utilizes four two-input comparators, and stage $S9$ employs one four-input comparator.

The space-time in Fig. 8 illustrates the time scheduling for different TU sizes, including $W \times 4$, $W \times 8$, $W \times 16$, and $W \times 32$. It is shown that the stages from $S1$ to $S6$ are seamlessly pipelined, and stages from $S7$ to $S9$ are sparsely scheduled when all W columns are processed. To enhance parallelism for TUs with a height H less than 32, multiple TUs are combined and processed simultaneously by reusing hardware resources. Specifically, the architecture can simultaneously process eight $W \times 4$ TUs, four $W \times 8$ TUs, and two $W \times 16$ TUs, respectively. The width W needs to be consistent across the parallelized TUs. Following the on-the-fly decision approach outlined in subsection III-B, the LSC decision process for the first W scanlines requires $(6 + W - 1)$ cycles. Furthermore, the LSC decision for the remaining $H - 1$ scanlines commences once the last column of the current TU enters stage $S7$. Therefore, the RDOQ process for one TU requires a total of $(6+W-1+3)$ clock cycles.

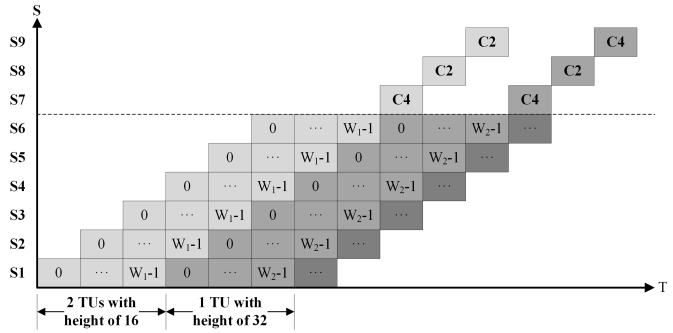


Fig. 8. The space-time diagram of the pipelined RDOQ design.

B. OCL decision

The strong run-level context dependency between neighboring coefficients poses a challenge for parallelized OCL hardware implementation and hinders real-time processing. However, this data dependency problem is partially resolved by the parallelized OCL algorithm in subsection III-A. The proposed algorithm segregates the context dependency from TU-level to zig-zag scanline line level, which makes the data dependency exist within one scanline. The scanline-level dependency exists for the derivation of the variables pre_level_i and run_i, and their derivation process is shown in Eq. (5) and Eq. (6). From these two formulas, it indicates that the pre_level_i and run_i depend on the optimal coefficient level l^*_{i-1} or l^*_{i+1} for the up-right and down-left scanline, respectively. However, determining the optimal coefficient level involves complex RD cost calculations, in which the rate cost, in turn, depends on pre_level_i and run_i. As a result, this dependency increases the complexity of the hardware design. To resolve the above problem, the pre_level_i and run_i derivation and optimal coefficient level decision are implemented in a single cycle, and then utilizing a data-forwarding method to implement this dependency [49]. Furthermore, to improve the timing of the design, the RD cost calculation is divided into three distinct pipeline stages, spanning from $S2$ to $S4$. The pipeline stage for distortion calculation is allocated prior to the rate calculation, spanning from $S2$ to $S3$, while the rate calculation and RD cost comparison are scheduled at stage $S4$. This is because the distortion is calculated without context dependency for all candidate levels.

Considering that the maximum number of candidate levels would not surpass three as listed in Table. I, the rate calculation design and the distortion calculation design are instantiated in parallel for all three distinct candidate levels. As shown in Fig. 7, $Dist_0 \& R_0$, $Dist_{l-1} \& R_{l-1}$ and $Dist_l \& R_l$ represent the distortion and rate design of different candidate levels. Following the optimization of the RD cost calculation mentioned in Eq. (7), the calculation of distortion is determined by the following formulas,

$$\begin{cases} Err_0 = P_i \times S_f >> 20, \\ Err_{l-1} = (P_i - (|l-1| << q_{bits})) \times S_f >> 20, \\ Err_l = (P_i - (|l| << q_{bits})) \times S_f >> 20, \end{cases} \quad (13)$$

$$Dist_j = Err_j^2, j = 0, l-1, l, \tag{14}$$

where q_{bits} is the shift precision, P_i is the multiplication result at stage $S1$, l is the initial scalar quantization level, j is the value of the candidate levels, S_f is the scaling factor with the λ optimization in subsection III-A. To improve timing performance, the distortion calculation is distributed from stage $S2$ to $S3$, where each stage has a multiplication operation.

Due to that the RD cost is optimized to a more efficient form in subsection III-A, there is no multiplication operation in stage $S4$. The calculation of R_0 is relatively simple. The rate calculation architecture for R_l and R_{l-1} is divided into three datapaths, as shown in Fig. 9. In this figure, table est_level is derived from entropy coding, ctx_level represents the minimum value between pre_level_i and 5, and L_i is the value of the candidate level. When L_i is equal to one, the rate cost is derived from table est_level indexed by ctx_level. For values of L_i greater than eight, the rate cost calculation becomes more intricate. In addition to considering the lookup value from table est_level, it also takes the Golomb coding of $|L_i|-9$ into account. For the remaining value of the candidate level, the rate cost also depends on the lookup value from the table est_level. Furthermore, an integral component of the rate cost calculation involves multiplying the lookup value with the value of $|L_i|$. To minimize area consumption and enhance timing performance, all multiplication operations involving est_level and the limited range of candidates (between one and eight) are implemented using separate add and shift operations,

As mentioned earlier, the calculation of rate cost and the comparison of RD cost are executed within a single cycle due to context dependency. Once all rate costs have been calculated, the minimum RD cost is identified and selected as the optimal one. The corresponding candidate is then designated as the optimal coefficient level l_i^*. Subsequently, this optimal coefficient level is transmitted back to update the run-level context.

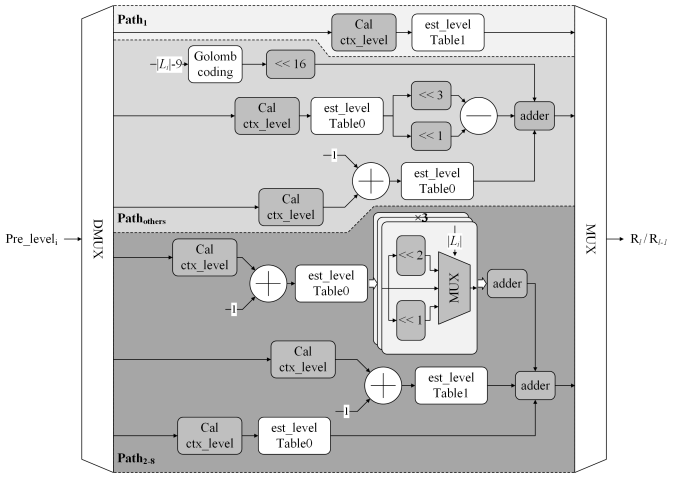


Fig. 9. Three datapaths of the rate calculation design for R_l and R_{l-1}.

C. LSC position decision

The hardware architecture for the LSC position decision is scheduled from stage $S5$ to $S9$, as shown in Fig. 7. This architecture can be divided into three key components, each serving a distinct purpose.

The first component is following the determination of the minimum RD cost as the optimal choice. The calculation of ΔJ_j, E_{non_lsc}, and E_{lsc}, as listed in Eq. (4), is scheduled at stage $S5$, preparing for the subsequent calculation of the optimal LSC position.

The second component is dedicated to making on-the-fly sub-optimal scanline LSC position decisions, which is used to calculate the optimal LSC position for the previous W scanlines. This calculation is scheduled at the pipeline stage $S6$. In this stage, a shift buffer is incorporated to align the temporary calculate results of each scanline, which corresponds to the for loop from line 28 to 30 in Algorithm 1. Besides, the on-the-fly algorithm as described in subsection III-B results in a sub-optimal position of an entire scanline being obtained in each cycle. Subsequently, a comparison is carried out between the obtained results and the previous scanlines, enabling the determination of the optimal position.

The remaining component focuses on calculating the optimal LSC decision for the remaining $H-1$ scanlines. This process is distributed across stages $S7$ to $S9$. Once the final column of a given TU reaches stage $S7$, all sub-optimal positions for each scanline are obtained. At this point, the computation process for determining the optimal LSC decision of the remaining $H-1$ scanlines begins. Considering that the maximum height of the TU is 32, carrying out a one-by-one sequential comparison of sub-optimal positions will require 32 cycles. To minimize latency and improve efficiency, the comparison of sub-optimal positions is executed in parallel, reducing the number of cycles to three. As an example, for a TU with a block size of 8×8, the sub-optimal 7 scanlines results enter the stage $S7$. After parallel comparison in stage $S7$, the two sub-optimal positions will be decided and get the final optimal LSC in stage $S8$. Besides, the bypass operation is adopted in $S9$ to align the output. Hence, the optimal LSC position of TU with a block size of 8×8 is pipelined with a total of 16 $(6 + W - 1 + 3)$ cycles.

V. Experimental Results

In this section, software and hardware-related results are carried out and analyzed. Firstly, the experimental environment settings are given. Then, we present and analyze the software-related results such as coding performance and computational complexity. Finally, we provide the hardware-related results including area and throughput.

A. Experimental Settings

To validate the accuracy and efficiency of the proposed scheme, we implement the parallelized RDOQ algorithm in the AVS3 reference software HPM-4.0. The coding performance of the proposed algorithm is evaluated under all intra (AI), random access (RA), and low delay B (LDB) configurations based on the AVS3 common test conditions (CTC) [50].

TABLE II
BD-RATE OF PARALLELIZED RDOQ IN HPM-4.0 UNDER AI, RA, AND LDB CONFIGURATIONS.

Class	Sequences	AI			RA			LDB		
		BD-Rate	TS_Q	TS_{Enc}	BD-Rate	TS_Q	TS_{Enc}	BD-Rate	TS_Q	TS_{Enc}
UHD	Tango2	0.18%	33.36%	10.21%	0.28%	29.96%	7.35%	0.28%	30.12%	8.03%
	Parkrunning3	0.08%	30.12%	8.10%	0.17%	27.04%	6.29%	0.17%	27.64%	6.67%
	Campfire	0.12%	31.42%	8.49%	0.34%	28.71%	7.02%	0.36%	27.92%	6.82%
	DaylightRoad2	0.08%	30.89%	8.31%	0.20%	27.95%	6.75%	0.19%	29.05%	7.15%
	Average	0.12%	31.45%	8.78%	0.25%	28.42%	6.85%	0.25%	28.68%	7.17%
HD	Cactus	0.34%	30.75%	8.22%	0.21%	28.03%	6.47%	0.29%	27.91%	6.63%
	BasketballDrive	0.45%	30.13%	8.12%	0.23%	27.12%	6.51%	0.34%	27.38%	6.45%
	MarketPlace	0.18%	33.01%	9.97%	0.44%	30.69%	7.73%	0.22%	30.05%	7.29%
	RitualDance	0.12%	31.29%	8.34%	0.31%	29.01%	7.13%	0.11%	29.89%	7.15%
	Average	0.28%	31.30%	8.66%	0.30%	28.71%	6.96%	0.24%	28.81%	6.88%
720p	City	0.36%	33.11%	10.02%	0.44%	29.55%	7.25%	0.53%	30.10%	8.11%
	Crew	0.36%	31.52%	8.56%	-0.19%	28.88%	6.50%	0.41%	28.17%	6.39%
	vidyo1	0.14%	30.43%	8.19%	0.34%	28.14%	6.44%	0.20%	27.35%	6.32%
	vidyo3	0.25%	29.52%	7.98%	0.13%	27.36%	6.27%	-0.04%	27.64%	6.46%
	Average	0.28%	31.15%	8.69%	0.18%	28.48%	6.61%	0.28%	28.32%	6.82%
240p	BasketballPass	0.26%	31.59%	8.65%	0.14%	28.04%	6.38%	0.53%	27.82%	6.53%
	BQSquare	0.13%	32.56%	9.51%	0.50%	29.93%	7.28%	0.04%	29.05%	7.02%
	BlowingBubbles	0.35%	29.15%	7.84%	0.22%	27.03%	6.10%	0.39%	27.14%	6.27%
	RaceHorses	0.28%	33.91%	10.28%	0.10%	30.12%	8.01%	0.21%	30.06%	8.08%
	Average	0.23%	31.80%	9.07%	0.24%	28.78%	6.94%	0.29%	28.52%	6.98%
Summary	**Overall**	**0.23%**	**31.42%**	**8.80%**	**0.24%**	**28.60%**	**6.84%**	**0.26%**	**28.58%**	**6.96%**

The test sequences consist of four categories, namely 240p (416×240), 720p (1280×720), HD (1920×1080), and UHD (4320×2160). The quantization parameters are set to 27, 32, 38, and 45. The coding performance is measured by BD-Rate and negative values represent performance gains [34]. The computational complexity of the proposed parallelized RDOQ algorithm (with SIMD optimization) is measured by the actual encoding time. The test platform is Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz. The time-saving TS_Q of the RDOQ process and the time-saving TS_{Enc} of the entire encoding process are obtained as follows,

$$TS_Q = \frac{T_{Org\_Q} - T_{Pro\_Q}}{T_{Org\_Q}} \times 100\%, \tag{15}$$

$$TS_{Enc} = \frac{T_{Org} - T_{Pro}}{T_{Org}} \times 100\%, \tag{16}$$

where T_{Org_Q} and T_{Pro_Q} represent the RDOQ time for the original and proposed SIMD-based parallelized schemes, respectively. Similarly, T_{Org} and T_{Pro} represent the total coding time for the original and proposed parallelized algorithm, respectively.

B. RD Performance and Computational Complexity Evaluation

The overall experimental results that compared our parallelized RDOQ with vanilla HPM-4.0 are presented in Table II. Both the RD performance and time-saving results are evaluated for each sequence under the AI, RA, and LDB configurations from Table II. It can be observed that the BD-Rate increments of our proposed scheme are 0.23%, 0.24%, and 0.26% under the AI, RA, and LDB configurations, respectively. The coding performance loss is insignificant, and the BD-Rate increment is relatively consistent across each sequence.

We give the time-saving results for the proposed parallelized RDOQ algorithm based on SIMD in Table II. It can be observed that the time of the RDOQ process is reduced by 31.42%, 28.6%, and 28.58% under AI, RA, and LDB configurations on average, respectively. And, it can save 8.80%,

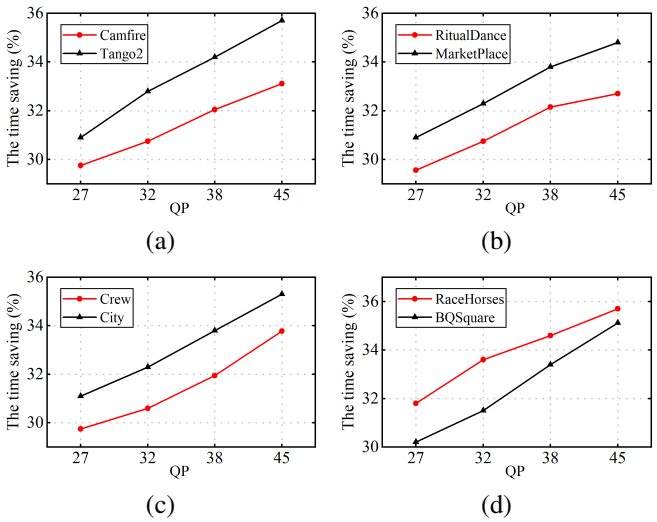


Fig. 10. The time-saving results in the four sequence cases including (a) UHD, (b) HD, (c) 720p, and (d) 240p under AI configuration.

6.84%, and 6.96% under AI, RA, and LDB configurations on average for the total encoding time. Furthermore, we conduct experiments to evaluate the time-saving TS_Q on various QPs. As shown in Fig. 10, it is found that more time can be saved as the QP increases. The reason mainly lies in that higher QP corresponds to a higher probability that the quantization coefficient will become zero, resulting in a higher chance that all the four parallelized coefficients in Fig. 6 become zero in the TU. When we accelerate the SIMD-based OCL decision process, we can skip the decision process if all four coefficients to be executed are zero.

As analyzed above, the proposed scheme effectively reduces coding complexity while incurring insignificant RD performance loss compared to the original RDOQ. Furthermore, computational complexity has been significantly reduced. These results demonstrate that our proposed parallelized RDOQ method can achieve a good trade-off between coding performance and computational complexity.

C. Ablation Study

In order to evaluate the effectiveness of the proposed method, we conduct ablation experiments in this subsection. In terms of the ablation study on coding performance, the performance changes of the simplified run-level context, the improved RD cost form based on Eq. (7), and the greedy strategy based LSC position decision are presented in Table III. It is shown that the simplified run-level context gives a 0.23%, 0.26%, and 0.30% BD-Rate increase under the AI, RA, and LDB configurations, respectively. The improved RD cost form eventually brings a 0.00% (0.23%-0.23%), -0.02% (0.24-0.26), and -0.04% (0.26%-0.30%) BD-Rate increase under the AI, RA, and LDB configurations, respectively. As for the greedy strategy based LSC position step, it shows no influence on the coding performance. Therefore, it is concluded that the performance loss is primarily attributed to the simplified run-level context. And the improved RD cost form has a minimal effect, with some sequences even showing slight improvements. This is because the simplified run-level context remains the dependency for only the up-right scan lines, which impacts the rate calculation. Whereas, the improved RD cost form is affected only by a decimal precision error in the calculation of D/λ. The greedy strategy based LSC location decision has no coding efficiency impact because the dependency in the TU is moved to the scanline level in the OCL decision step. The optimized greedy strategy based LSC decision will not impact the final result from this point of view.

In terms of the ablation study on time-saving, the results for the three stages using SIMD optimization are tabulated in Table IV. The results show that the pre-quantization step using SIMD can achieve 21.53%, 20.35%, and 19.92% time savings under the AI, RA, and LDB configurations, respectively. For the OCL decision step, an 8.94% (30.47% - 21.53%), 7.01% (27.36% - 20.35%) and 7.14% (27.06% - 19.92%) time savings can be obtained, respectively. For the LSC location decision step, a 0.88% (31.35% - 30.47%), 1.27% (28.63% - 27.36%) and 1.5% (28.56% - 27.06%) time saving is obtained, respectively. The results indicate that the pre-quantization step has the most significant time savings primarily due to its independent calculation process, which is more amenable to SIMD optimization. The reason for the poor optimization process of the LSC process is that the data in the look-up table needs to be exchanged between register and memory when calculating the rate. And for the LSC location decision step with the least time-saving, the reason for this derives from two factors. One is that the parallelized algorithm needs to decide the sub-optimal i_k^* for different scanlines and to calculate multiple temporary variables as shown in Algorithm 1. The other is that when we optimize if and $else\ if$ statements in Algorithm 1 based on SIMD, it needs the help of mask operation to achieve this function. The mask operation leads to a reduction in efficiency.

D. Performance Comparison with Other Methods

To further verify the priority of our proposed scheme, the results compared with the five other existing methods are shown in Table V. It is worth noting that these methods have been implemented on two different reference software platforms, namely HM (HEVC) and HPM (AVS3). For the work proposed on HM, we use the results from the original work for comparison directly. Since the parallel algorithm proposed in this paper is implemented on HPM-4.0, we re-implement the HPM-4.0 platform-based algorithms and get the results for comparison.

As shown in Table V, work [15] and work [26] both speed up for the software platform, and they can achieve a time-saving ranging from 12.9% to 15.8% with a negligible BD-Rate loss. Compared with [15] and [26], our parallelized algorithm can achieve more time saving with closely matched coding efficiency and focuses on the fully pipelined hardware

TABLE III
BD-RATE RESULT OF ABLATION STUDY.

Method combination	BD-Rate		
	AI	RA	LDB
Simplified run-level context	0.23%	0.26%	0.30%
Simplified run-level context + Optimized RD cost	0.23%	0.24%	0.26%
Simplified run-level context + Optimized RD cost + greedy strategy based LSC position decision	0.23%	0.24%	0.26%

TABLE IV
TIME-SAVING RESULT OF ABLATION STUDY.

SIMD-based method combination	TS_Q		
	AI	RA	LDB
Pre-quantization	21.53%	20.35%	19.92%
Pre-quantization + OCL decision	30.47%	27.36%	27.06%
Pre-quantization + OCL decision + LSC position decision	31.35%	28.63%	28.56%

TABLE V
COMPARISON WITH OTHER WORKS.

Existing work	Reference software	Optimized for	AI		RA		LDB	
			BD-Rate	TS_Q	BD-Rate	TS_Q	BD-Rate	TS_Q
Lee *et al.'s* [15]	HM-11.0	Software	0.09%	14.4%	0.11%	15.8%	0.08%	14.9%
Xu *et al.'s* [26]	HM-16.15	Software	0.00%	14.4%	0.04%	12.9%	0.12%	12.9%
Igarashi *et al.'s* [33]	HM-16.0	Hardware	N/A	N/A	2.51%	N/A	N/A	N/A
Xu *et al.'s* [35]	HPM-4.0	Hardware	N/A	N/A	0.86%	27.6%	0.64%	30.6%
Zhao *et al.'s* [36]	HPM-4.0	Hardware	0.82%	N/A	0.46%	N/A	0.36%	N/A
Proposed	HPM-4.0	Hardware	0.23%	31.4%	0.24%	28.6%	0.26%	28.58%

implementation. Work [33] proposed a parallelized RDOQ algorithm for the HEVC encoder. It achieves 4K@60fps real-time encoding on GPU with a significant 2.51% BD-Rate loss. However, our proposed RDOQ algorithm achieves parallel processing with only 0.24% BD-Rate loss, which provides a better trade-off between coding efficiency and hardware implementation. Work [35] introduces all-zero block skipping and an optimized rate estimation algorithm. It is shown that it obtains 27.6% and 30.6% time savings in the RA and LDB configurations with 0.86% and 0.64% coding performance loss, respectively. Our coding efficiency loss is smaller than work [35]. In addition, we add our previous work [36] into comparison as well, and it is shown that the improved parallelized algorithm in this paper has better coding performance than in [36].

E. Hardware Implementation Results

Currently, there are limited studies focusing on hardware implementations of the RDOQ algorithm for AVS3 encoders. This is primarily due to its strong context dependency and computational complexity. The proposed RDOQ hardware architecture in our research is based on the parallelized algorithm. This implementation allows for parallel processing of multiple TUs with high parallelism, while minimizing resource utilization.

To demonstrate the advancements of our work, we conduct a comprehensive comparison on Field-Programmable Gate Array (FPGA) platform with Zhao et al.'s [36], and the results are tabulated in Table VI. Our proposed design was implemented using Verilog Hardware Description Language (HDL) and synthesized using Synplify Pro 2019.03 for the Xilinx UltraScale+ MPSoCs 7ev.

TABLE VI
COMPARISION OF FPGA HARDWARE RESOURCE CONSUMPTION.

Design	Zhao *et al.'s* [36]	Proposed
LUTs	109759	72017
FFs	29576	23440
DSPs	481	224
Parallelism	4-32	32
Cycles	538	40
Frequency	200MHz	116MHz
Throughput	4K@45fps	8K@89fps

As shown in Table VI, our proposed design achieves a 34.39% reduction in LUTs $((109759 - 72017) \div 109759)$ and over 50% reduction in DSPs compared to Zhao et al.'s [36]. Moreover, apart from minimizing area consumption, our design has a significantly higher processing throughput compared to their design. The processing of a TU with the size of 32×32 only requires 40 cycles at a frequency of 116MHz, which can meet the real-time encoding of 8K@89fps $((32 \times 32 \times 116 \times 10^6) \div (7680 \times 4320 \times 40))$. While their method operates at a higher frequency of 200MHz, it needs a considerably large number of clock cycles, resulting in lower overall throughput. Furthermore, the proposed design has been synthesized by Design Compiler with the UMC 28nm cell library, the gate count is 1,223.2K, the chip area is $0.67mm^2$, and the maximum working frequency is 471.2MHz.

VI. CONCLUSION AND FUTURE WORK

In this paper, a parallelized RDOQ algorithm and its fully pipelined hardware architecture design are proposed for the AVS3 video coding standard. For the parallelized algorithm, the run-level context for the rate estimation is updated in the inner zig-zag scanline and an efficient RD cost form is adopted in the OCL decision step. In the LSC position decision step, a greedy strategy based algorithm is proposed to achieve the LSC position decision compute in parallel. Moreover, the parallelized algorithm is accelerated based on SIMD. The experimental results show that the BD-Rate of the parallelized algorithm is increased by 0.23%, 0.24%, and 0.26% under the AI, RA, and LDB configurations, respectively. The computational time is reduced by 31.42%, 28.60%, and 28.58% compared to the vanilla HPM-4.0, respectively. For the hardware architecture, we design a nine-stage fully pipelined architecture that achieves 32 coefficients per cycle. It can process multiple TUs when the height is less than 32. The area consumption is 1223.2K gate count ($0.67mm^2$) when working at 471.2MHz. It proves that our design achieves a good tradeoff between coding efficiency and hardware throughput. In our future study, we will extend the proposed parallelized algorithm to RDOQ in HEVC and explore novel

methods to speed up the dependent quantization computation in VVC.

REFERENCES

[1] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (vni) complete forecast update, 2017–2022," *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pp. 1–30, 2018. I

[2] ITU-T and ISO/IEC JTC 1, *Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video*, ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2 Video), version 1, 1994. I

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003. I

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012. I

[5] 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 Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021. I

[6] J. Zhang, C. Jia, M. Lei, S. Wang, S. Ma, and W. Gao, "Recent development of avs video coding standard: Avs3," in *2019 picture coding symposium (PCS)*. IEEE, 2019, pp. 1–5. I

[7] Y.-W. Huang, C.-W. Hsu, C.-Y. Chen, T.-D. Chuang, S.-T. Hsiang, C.-C. Chen, M.-S. Chiang, C.-Y. Lai, C.-M. Tsai, Y.-C. Su *et al.*, "A vvc proposal with quaternary tree plus binary-ternary tree coding block structure and advanced coding techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 5, pp. 1311–1325, 2019. I

[8] M. Wang, J. Li, L. Zhang, K. Zhang, H. Liu, S. Wang, S. Kwong, and S. Ma, "Extended quad-tree partitioning for future video coding," in *2019 Data compression conference (DCC)*. IEEE, 2019, pp. 300–309. I

[9] K. Fan, Y. Cai, X. Gao, W. Chen, S. Wu, Z. Wang, R. Wang, and W. Gao, "Performance and computational complexity analysis of coding tools in avs3," in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2020, pp. 1–6. I

[10] J. Zhang, C. Jia, M. Lei, S. Wang, S. Ma, and W. Gao, "Recent development of avs video coding standard: Avs3," in *2019 picture coding symposium (PCS)*. IEEE, 2019, pp. 1–5. I

[11] H. B. Yin, E.-H. Yang, X. Yu, and Z. Xia, "Fast soft decision quantization with adaptive preselection and dynamic trellis graph," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 8, pp. 1362–1375, 2014. I

[12] M. Wang, X. Xie, H. Fan, S. Wang, J. Li, S. Dong, G. Xiang, and H. Jia, "Fast rate distortion optimized quantization method for hevc," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4. I

[13] H. Yin, H. Wang, X. Huang, and H. Yin, "Efficient hard-decision quantization using an adaptive deadzone offset model for video coding," *IEEE Access*, vol. 7, pp. 151 215–151 229, 2019. I

[14] 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 Transactions on Image Processing*, vol. 30, pp. 2378–2393, 2021. I

[15] H. Lee, S. Yang, Y. Park, and B. Jeon, "Fast quantization method with simplified rate–distortion optimized quantization for an hevc encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 107–116, 2015. I, V-D, V-D, V

[16] Y. Zhang, R. Tian, J. Liu, and N. Wang, "Fast rate distortion optimized quantization for hevc," in *2015 Visual Communications and Image Processing (VCIP)*. IEEE, 2015, pp. 1–4. I

[17] B. Lee, J. Jung, and M. Kim, "An all-zero block detection scheme for low-complexity hevc encoders," *IEEE Transactions on Multimedia*, vol. 18, no. 7, pp. 1257–1268, 2016. I

[18] M. Wang, X. Xie, J. Li, H. Jia, and W. Gao, "Fast rate distortion optimized quantization method based on early detection of zero block for hevc," in *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*. IEEE, 2017, pp. 90–93. I

[19] H. Fan, R. Wang, L. Ding, X. Xie, H. Jia, and W. Gao, "Hybrid zero block detection for high efficiency video coding," *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 537–543, 2016. I

[20] J. Cui, R. Xiong, X. Zhang, S. Wang, S. Wang, S. Ma, and W. Gao, "Hybrid all zero soft quantized block detection for hevc," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 4987–5001, 2018. I

[21] J. Cui, R. Xiong, F. Luo, S. Wang, and S. Ma, "An adaptive and low-complexity all-zero block detection for hevc encoder," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4. I

[22] H. Wei, W. Zhou, X. Zhang, X. Zhou, and Z. Duan, "All zero block detection for hevc based on the quantization level of the maximum transform coefficient," *Multimedia Tools and Applications*, vol. 78, pp. 363–387, 2019. I

[23] H. Yin, H. Cai, E. Yang, Y. Zhou, and J. Wu, "An efficient all-zero block detection algorithm for high efficiency video coding with rdoq," *Signal Processing: Image Communication*, vol. 60, pp. 79–90, 2018. I

[24] H. Yin, H. Yang, X. Huang, H. Wang, and C. Yan, "Multi-stage all-zero block detection for hevc coding using machine learning," *Journal of Visual Communication and Image Representation*, vol. 73, p. 102945, 2020. I

[25] M. Xu, T. N. Canh, and B. Jeon, "Simplified rate-distortion optimized quantization for hevc," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2018, pp. 1–6. I

[26] M. Xu, T. Nguyen Canh, and B. Jeon, "Simplified level estimation for rate-distortion optimized quantization of hevc," *IEEE Transactions on Broadcasting*, vol. 66, no. 1, pp. 88–99, 2020. I, V-D, V-D, V

[27] T.-Y. Huang and H. H. Chen, "Efficient quantization based on rate–distortion optimization for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 6, pp. 1099–1106, 2015. I

[28] J. Cui, S. Wang, S. Wang, X. Zhang, S. Ma, and W. Gao, "Hybrid laplace distribution-based low complexity rate-distortion optimized quantization," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3802–3816, 2017. I, II-A

[29] 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. I

[30] J. Wang, H. Yin, Z. Gao, and X. Zhang, "Improved rate distortion optimized quantization for hevc with adaptive thresholding," in *2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2016, pp. 1–4. I

[31] T. N. Canh, M. Xu, and B. Jeon, "Rate-distortion optimized quantization: A deep learning approach," in *IEEE High Performance Extreme Computing Conference*, 2018. I

[32] D. Kianfar, A. Wiggers, A. Said, R. Pourreza, and T. Cohen, "Parallelized rate-distortion optimized quantization using deep learning," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–6. I

[33] H. Igarashi, F. Takano, T. Takenaka, H. Inoue, and T. Moriyoshi, "Parallel rate distortion optimized quantization for 4k real-time gpu-based hevc encoder," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2018, pp. 1–4. I, II-B, V, V-D

[34] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *ITU SG16 Doc. VCEG-M33*, 2001. I, V-A

[35] J. Xu, G. Xiang, Y. Yan, Y. Wen, X. Huang, P. Zhang, and W. Yan, "Hardware-friendly fast rate-distortion optimized quantization algorithm for avs3," in *Fourteenth International Conference on Digital Image Processing (ICDIP 2022)*, vol. 12342. SPIE, 2022, pp. 593–600. I, II-B, V, V-D, V-D

[36] 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," in *2023 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2023, pp. 1–6. I, I, II-A, II-B, V, V-D, V-D, V-E, VI, V-E

[37] "AVS3 Software Repository." Accessed. [Online]. Available: https://gitlab.com/AVS3software/hpm/-/tags/HPM-4.0. I

[38] X. Huang, W. Niu, Q. Zhang, H. Yin, G. Xiang, S. Wang, and S. Ma, "Rate-distortion optimization-based learned fractional interpolation filter design for hevc," *IEEE Transactions on Broadcasting*, 2023. II-A

[39] J. Wang, X. Wang, T. Ji, and D. He, "Transform coefficient coding design for avs2 video coding standard," in *2013 Visual Communications and Image Processing (VCIP)*. IEEE, 2013, pp. 1–6. II-A

[40] Z. Wang, R. Wang and K. Fan, *The coding method of the index of last non-zero coefficient in AVS3*, AVS document M4649, 2019. II-A

[41] L. Braatz, B. Zatt, D. Palomino, L. Agostini, and M. Porto, "High-throughput and low-power integrated direct/inverse hevc quantization

hardware design," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5. II-B

[42] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer dct architectures for hevc," *IEEE Transactions on Circuits and systems for Video Technology*, vol. 24, no. 1, pp. 168–178, 2013. II-B

[43] Y. Fan, Y. Zeng, H. Sun, J. Katto, and X. Zeng, "A pipelined 2d transform architecture supporting mixed block sizes for the vvc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 3289–3295, 2019. II-B, II-B

[44] Z. Hao, H. Sun, G. Xiang, P. Zhang, X. Zeng, and Y. Fan, "A reconfigurable multiple transform selection architecture for vvc," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 5, pp. 658–669, 2023. II-B

[45] H. Sun, D. Zhou, L. Hu, S. Kimura, and S. Goto, "Fast algorithm and vlsi architecture of rate distortion optimization in h. 265/hevc," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2375–2390, 2017. II-B

[46] Y. Zhang and C. Lu, "Efficient algorithm adaptations and fully parallel hardware architecture of h. 265/hevc intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 11, pp. 3415–3429, 2018. II-B

[47] X. Huang, K. Wei, H. Yin, C. Zhu, H. Jia, and D. Xie, "Three-level pipelined multi-resolution integer motion estimation engine with optimized reference data sharing search for avs," *Journal of Real-Time Image Processing*, vol. 15, pp. 43–55, 2018. II-B

[48] Y. Zhang and C. Lu, "A highly parallel hardware architecture of table-based cabac bit rate estimator in an hevc intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1544–1558, 2018. II-B

[49] D. Harris and S. Harris, *Digital design and computer architecture*. Morgan Kaufmann, 2010. IV-B

[50] J. Chen, "AVS3-P2 Common Test Conditions V9.0," *AVS-Doc, N2813*, 2020. V-A