On a Highly Efficient RDO-Based Mode Decision Pipeline Design for AVS

Chuang Zhu, Huizhu Jia, Shanghang Zhang, Xiaofeng Huang, Xiaodong Xie, and Wen Gao, *Fellow, IEEE*

***Abstract*—Rate distortion optimization (RDO) is the best known mode decision method, while the high implementation complexity limits its applications and almost no real-time hardware encoder is truly full-featured RDO based. In this paper, first, a full-featured RDO-based mode decision (MD) algorithm is presented, which makes more modes enter RDO process. Second, the throughput of RDO-based MD pipeline is thoroughly analyzed and modeled. Third, a highly efficient adaptive block-level pipelining architecture of RDO-based MD for AVS video encoder is proposed which can achieve the highest throughput to alleviate the RDO burden. Our design is described in high-level Verilog/VHDL hardware description language and implemented under SMIC 0.18-μm CMOS technology with 232 K logic gates and 85 Kb SRAMs. The implementation results validate our architectural design and the proposed architecture can support real time processing of 1080P@30 fps. The coding efficiency of our adopted method far outperforms (0.57 dB PSNR gain in average) the traditional low-complexity MD (LCMD) methods and the throughput of our designed pipeline is increased by 11.3%, 19% and 17% for I, P and B frames, respectively, compared with the existed RDO-based architecture.**

***Index Terms*—AVS, mode decision, pipeline, RDO, throughput.**

I. Introduction

VIDEO coding, which bridges the crucial gap between the user's high-quality visual demands and the limited capabilities of transmission networks and storage devices, has enabled a host of new 'multimedia' applications including digital television, digital versatile disk (DVD) movies, streaming Internet video, home digital photography and video conferencing [1]. Massive efforts have been made to improve the coding performance and several video coding standards like H.264 [2], MPEG-4 [3] and AVS [4] are developed to promote video coding applications in multimedia. RDO-based mode decision (MD) [5] is adopted in many video coding standards with the increasing demands for high quality video applications, while the high computation complexity is unbearable in many video coding applications.

A number of strategies have been developed in recent years to alleviate the unbearable computational burden of RDO-based MD. To achieve this goal, some papers [6]–[9] have presented fast MD methods trying to reduce the candidate modes. These algorithms firstly study the spatial or temporal features of a block and then skip the unnecessary candidate modes based on experimental results. In [6], Pan *et al.* established a local edge map direction histogram based on Sobel operator to predict the most probable intra-prediction direction, and skipped the unnecessary candidate modes (the intra-prediction directions with less probability) directly. Wu *et al.* [9] studied the spatial homogeneity and the temporal stationarity of a macroblock to reduce the inter candidate modes, and the proposed algorithm can save about 30% of coding time with ignorable coding performance degradation. Other papers [10]–[13] did not skip any candidate mode and focused on reducing the complexity of calculating each R-D cost value. Paper [10], [11] utilized linear combinations of *Tc*, *To* and *Tz* to estimate bit-rate in MD process, where *Tc* denotes the numbers of the total nonzero coefficients, *To* denotes the trailing ± 1 values of quantized transform block, and *Tz* represents the number of zero coefficients. Paper [12] proposed a distortion measurement obtained in transform domain. In [13], Zhao *et al.* deduced a novel weighted sum of quantized transform coefficients and utilized it as an efficient rate estimator; the authors also proposed a new transform-domain distortion (TDD) estimation method by using the discarded lower bits in the quantization process. The reported coding performance in [13] is comparable with the original RDO implementation.

Most of the works above were not hardware-oriented and they cannot be implemented easily in hardware. It is very challenging to design the architecture for the RDO-based MD in hardware because of the abundant intra-inter modes and the block-level reconstruction loop caused by intra prediction which induces the bubble cycles and decreases the hardware usage efficiency and throughput [14]. Typically, hardware MD algorithms can be classified into two categories. In the first category, works like [15]–[18] proposed similar RDO-based methods as the above strategies, which have been developed to alleviate the computational burden, and paid great efforts to make them hardware-oriented. Work [15] proposed a regular spatial domain filtering technique to compute the dominant edge strength (DES) to reduce the possible predictive modes and designed a corresponding effective VLSI circuit. The proposed method is an improved version of [6] and the complicated procedures of determining the edge angle in [6] is avoided to make it possible for VLSI design. However, it suffered from the PSNR degradation. Paper [16] proposed an approach to reduce the computational costs by the approximation of entropy coding bit amount in inter block size decision. The reported PSNR degradation of the proposed method in comparison to full-featured RDO is 0.1 dB in the worst case. Other hardware-oriented RDO-based mode decision methods appear

Manuscript received July 31, 2012; revised December 06, 2012 and March 21, 2013; accepted May 19, 2013. Date of publication September 05, 2013; date of current version November 13, 2013. This work was supported in part by grants from the Chinese National Natural Science Foundation under contract No. 61035001 and No. 61176139, and Development Program of China (863 Program) under contract No.2012AA011703. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yiannis Andreopoulos.

The authors are with the Institute of Digital Media, Peking University, Beijing 100871, China (e-mail: hzjia@pku.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2013.2280446

in works [17], [18] recently. Paper [17] utilized the features of different intra and inter modes which have different probabilities to be selected to reduce some candidate modes. The authors also proposed intelligent pipeline scheduling (6-stage block level pipelining) mechanism to partly break the data dependency in intra prediction. On the one hand, the proposed method also suffers from PSNR degradation due to the reduction of candidate modes; on the other hand, the throughput of the developed block level pipelining is not sufficient. In the second category, in order to alleviate the computational burden and avoid the reconstruction loop, the low-complexity mode decision (LCMD) algorithm as in the standard reference software was adopted in [19]–[21]. In encoder chip design [19], sum of absolute difference between original pixels and predicted pixels (SAD) was used in judging the best mode. Hadamard transform was adopted in [20] to improve the coding performance and in these works sum of absolute transform difference (SATD) was applied. In work [21], the authors realized an HDTV1080p H.264/AVC encoder chip and used different schemes in inter and intra mode decision, respectively. The work performed a low-complexity inter mode decision based on the computation of SATD of the residues; in intra mode decision, it used edge-based method in [6] and presented a corresponding VLSI implementation. The main problem of the second category is that the coding performance is much worse than that of the RDO-based method.

To the best of our knowledge, there is no proposed scheme in the literature that performs genuine or full-featured RDO-based mode decision in the hardware encoder design. Different from the above two categories of hardware-oriented MD algorithms, this work adopts nearly genuine RDO-based MD method without skipping any candidate mode or simplifying the R-D cost value calculation to obtain high coding performance, which outperforms all the methods of the above two categories. This work aims at designing high throughput MD pipeline architecture to address the problem of large computational burden. Recent work [22] discussed MD pipeline scheduling and developed an optimized 5-stage block level pipeline which achieves the highest throughput compared with [17] and [18]. However, that work selected the pipeline architecture only according to the throughput analysis of I frame. Through thorough analysis, we find that using the same pipeline architecture (5-stage) for both I frame and PB frame cannot achieve the highest throughput for the whole pipeline. Therefore, in this paper we further analyze the pipeline throughput and design a highly efficient adaptive MD pipeline architecture which can achieve the highest throughput to alleviate the RDO burden. In the pipeline design, to obtain a general design rule, we focus on modeling the throughput of the pipeline in RDO-based MD. Based on the throughput model, we optimize the pipeline architecture of MD for AVS HD video encoder.

The main contributions are summarized as follows:

1) Establish the pipeline throughput model of RDO-based MD. The throughput model is deduced based on the analysis of the pipeline space-time diagram. For the convenience of the pipeline analysis, we divide the assignments (prediction blocks to be processed in MD) into two classes after detailed study of the restrictions in RDO-based MD: 1) assignments with data dependency (ADDs) and 2) assignments with no data dependency (ANDDs). We then use the ANDDs to fill the "holes" generated by the ADDs and finally formulize the throughput model of RDO-based MD pipeline according to the quantitative relationship of ANDDs and the "holes". The established throughput model can be used to guide the pipelining architecture design for different video encoding standards (like H.264 and AVS).
2) In the pipeline design, we analyze in detail the data dependency of intra luma blocks for I, P and B frames in AVS and depict the corresponding ADDs space-time diagram. Based on the analysis, we deduce the throughputs of MD for I, P and B frames based on the established pipeline throughput model. To design the optimal MD pipeline architecture, basic processing unit set (BPU) is defined and the corresponding time consumption of each basic processing unit is given. The highly efficient adaptive pipeline architecture (APA) is proposed through the analysis of different "combination" manners of the basic processing units.
3) The proposed MD method, which does not skip any candidate mode for RDO-based MD, achieves the best coding performance when compared with the fast MD methods in recent literature and far outperforms the LCMD method. The proposed adaptive pipelining architecture obtains higher throughput compared with previous RDO-based MD architecture.

The remainder of this paper is organized as follows. In Section II, we first review AVS video coding framework and then present our adopted RDO-based MD method. In Section III, we show the limitations in RDO-based MD pipeline design and model the pipeline throughput under these limitations. The pipeline design process is presented in detail based on the throughput model and a highly efficient adaptive MD pipeline architecture is developed for AVS HD video encoder in Section IV. At last, the performance evaluation, implementation results, and the conclusion of our paper will be elaborated.

II. AVS Framework Review and Adopted MD Algorithm

AVS video coding standard, established by China Audio Video Coding Standard (AVS) Working Group, has been accepted by ITU-TFGIPTV as an option for IPTV applications. The AVS part 2 (AVS-P2) is high resolution friendly profile, which is also known as the Jizhun Profile of AVS [23]. For high-resolution applications, AVS-P2 shows comparable performance with H.264/AVC for most progressive sequences [24].

A. AVS Video Coding Framework Review

As shown in Fig. 1 the AVS video coding standard, which is similar to H.264, is based on the traditional transform/prediction hybrid framework [24]–[26]. A picture to be encoded is partitioned into fixed-size macroblocks (MBs); each MB is composed of one 16×16 samples of luma (Y) component and two 8×8 samples of the two chroma components (Cr and Cb). Each input MB is both spatially and temporally predicted to remove the spatial and temporal redundancy of the input picture. After prediction, the resulted prediction residual is transformed and quantized to remove the components of the transformed data which are relatively unimportant to the visual perception of the picture. The quantized coefficients are encoded using entropy

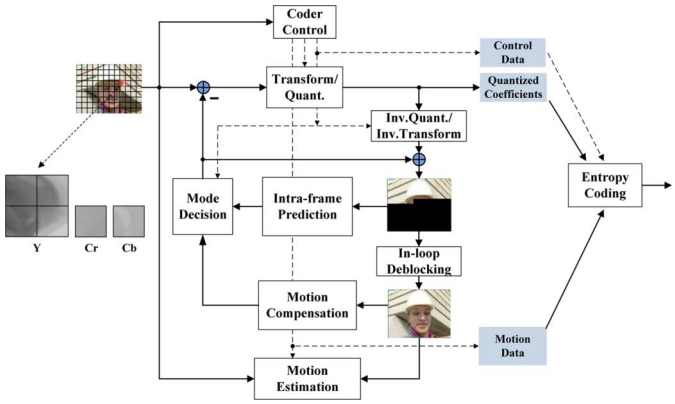

Fig. 1. AVS video coding framework.

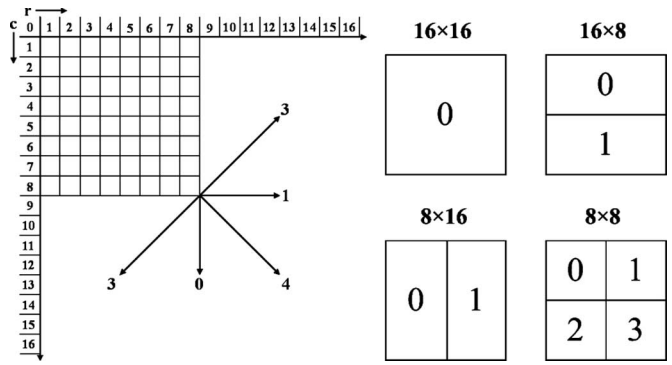

Fig. 2. Intra luma prediction directions and macroblock partitions in AVS-P2.

coding to remove statistical redundancy. As depicted in Fig. 1, the quantized coefficients are also processed with inverse quantization and inverse transform to produce reconstructed prediction error which is added to the prediction pixels to yield the reconstructed picture. The reconstructed picture is then filtered by the 6-tap in-loop deblocking unit to generate reference frame for the succeeding frames.

Specifically, intra-frame prediction produces spatial prediction of the current MB based on the neighboring samples. The intra prediction in AVS-P2 is 8×8 block-based because AVS-P2 mainly aims at high resolution videos where the smaller block prediction is less efficient than the bigger block prediction for the improvement of coding performance [24]. The AVS luma prediction directions are shown in Fig. 2 and the 5 prediction modes for intra luma block are Vertical (V), Horizontal (H), DC, Diagonal down right (DR) and Diagonal down left (DL). For each chroma block, 4 modes are supported in AVS-P2 and they are Vertical, Horizontal, DC, and Plane (P). Motion estimation (ME) and motion compensation (MC) make temporal prediction for the current MB. AVS standard uses variable block size (VBS) ME and MC technique which results in different inter modes to be chosen. In AVS-P2, MB partitions, as shown in Fig. 2, vary from 16×16 to 8×8. The minimum MB partition is 8×8 block because the block size less than 8×8 does not gain much improvement for high resolution video [24].

In AVS, RDO uses Lagrange Multiplier method to get the optimal solution. The cost function is as follows

$$RDcost = D + \lambda R \tag{1}$$

where D denotes the sum of squared differences (SSD) between the original pixels and the reconstructed pixels and R is the real entropy coded bits of a coding unit with a certain candidate mode. λ is Lagrangian multiplier. Like H.264, in AVS reference code, $\mathrm{RDO}_{\mathrm{off}}$ option (LCMD) is also supported and the cost function is shown as

$$RDcost = SA(T)D + \lambda R^1 \tag{2}$$

where *SAD* is the sum of absolute difference between the original block and the predicted block, and *SATD* is the sum of absolute difference of Hadamard transformed coefficients of the residues. R^1 stands for the bits for coding motion vector and other related information [27]. When in $\mathrm{RDO}_{\mathrm{off}}$ mode, DPCM loop (DCT/Q/IQ/IDCT) and the entropy coding are not needed, so the computation complexity is much lower than RDO. However, the corresponding coding performance is much worse than RDO.

This paper selects the best mode based on (1) and aims at achieving high quality coding performance, while the problem of high computational burden is addressed by high-efficiency pipeline design.

B. Adopted MD Method and the Complexity Analysis

Different from the fast mode decision methods in the literature, this work does not reduce any candidate mode to guarantee the RDO quality.

For each intra luma block of I frame, we choose the best mode from all 5 modes: {V, H, DC, DR and DL} and for each intra chroma block of I frame we choose the best mode from all four modes: {V, H, DC, P}. Therefore, for I frame, we choose the best mode using Full-RDO method without any simplification. We choose the best mode from {Pskip, P16 × 16, P16 × 8, P8 × 16, P8 × 8, Intra8 × 8} for P frame and the best mode from {Bdirect, Bskip, B16 × 16, B16 × 8, B8 × 16, B8 × 8, Intra8 × 8} for B frame. Based on the fact that the percentage of intra mode chosen as the best mode is much lower than that of inter modes for P and B frames [17], we just use SAD to decide the best block-level modes of Intra8 × 8 in P and B frames and this can alleviate RDO computational load.

Noticing that for Pskip and Bskip modes, the corresponding *RDcost* computation complexity can be neglected due to the following fact: on the one hand, the reconstructed pixels can be obtained directly without the processing of DPCM loop because they are identical to the prediction pixels, thus the SSD value can be derived directly; on the other hand, the coding bits R equals to 0. Therefore, the computational complexity of our adopted MD method can be obtained as Table I (Pskip and Bskip are excluded for P and B frames, respectively). According to Table I, MD needs to perform 28, 30 and 36 times of *RDcost* calculation to obtain the best mode in one MB for I, P and B frames, respectively.

Actually, in order to get SSD, the mode decision unit has to perform discrete cosine transform (DCT), quantization (Q), inverse quantization (IQ), inverse discrete cosine transform (IDCT), and reconstruction (REC) functions for each prediction block; to obtain R, the mode decision unit has to perform DCT, Q, IQ, zigzag scan (ZIGZAG) and context-based 2D-VLC (VLC) entropy coding for each block. If all these processing

TABLE I
COMPUTATIONAL COMPLEXITY FOR I, P AND B FRAMES

PICTURE-TYPE	*RDcost* calculation frequency for one MB (6 blocks)
I	4×5 + 2×4 = 28 (*RDcost*s)
P	5×6 = 30 (*RDcost*s)
B	6×6 = 36 (*RDcost*s)

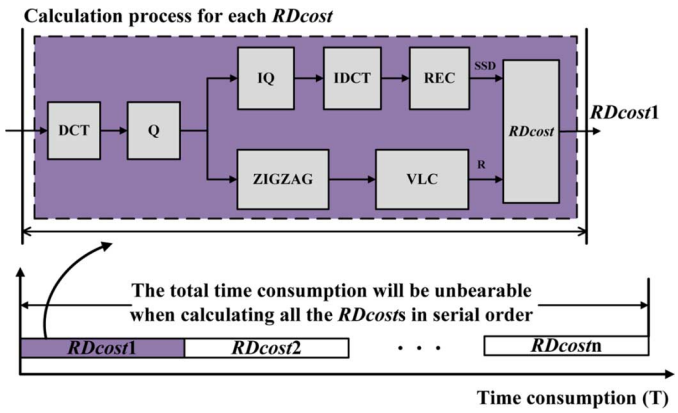

Fig. 3. Calculation process for each *RDcost* and calculating all the *RDcosts* in serial order.

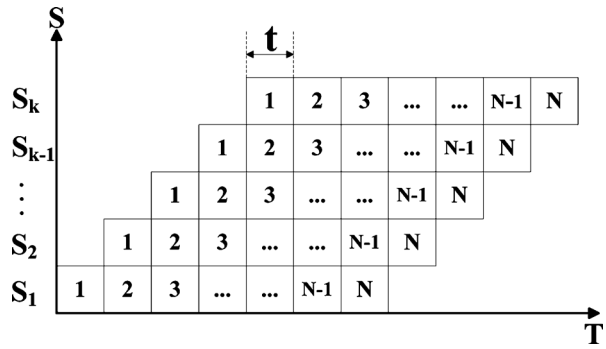

Fig. 4. General k-stage pipeline space-time diagram.

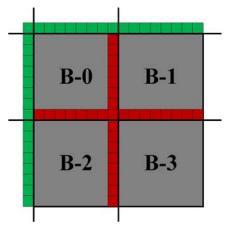

Fig. 5. Data dependency of intra luma blocks in AVS.

units are connected in sequential manner and then all the *RDcost*s calculations will be performed in serial order, the total processing time will be unendurable, as shown in Fig. 3. In the next sections, we will analyze and model the throughput of RDO-based MD based on the pipeline theory and design the optimal pipeline structure. Finally, our proposed high efficiency pipeline architecture will be presented.

III. THROUGHPUT MODELING OF RDO-BASED MD

A. Throughput Analysis of Ideal Pipeline

Take a k-stage (k denotes the number of stages) pipeline as an example, with its space-time diagram shown in Fig. 4, where $\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_{k-1}$, S_k stand for the k functional processing stages. Assume the ideal condition: the k stages are perfectly balanced and the processing time of each stage (t) is the same. If there are N assignments in total to be processed in the pipeline, the throughput, which is defined as the number of completed assignments per unit time, can be described as the following function

$$throughput_k = N/T_{total} = N/((k+N-1)\cdot t) \quad (3)$$

where $throughput_k$ denotes throughput of the k-stage pipeline and T_{total} denotes the total time consumption of the N assignments. Also, assume the ideal condition: there is no latency between two stages. If the whole processing time of the k stages is constant C_0, we can get that

$$t = \mathrm{C}_0/k \quad (4)$$

Substituting (4) into (3) will yield

$$throughput_k = N/T_{\mathrm{total}} = N/((k+N-1)\cdot t) = N/((1+(N-1)/k)\cdot \mathrm{C}_0) \quad (5)$$

According to (5), for an ideal pipeline, the throughput improves with the increase of k (number of stages), but unfortunately, we cannot achieve the ideal pipeline condition in RDO-based MD. In the following, we will point out the restrictions in RDO-based MD pipeline design and then deduce the throughput model under these restrictions.

B. Restrictions of RDO-Based MD Pipeline Structure

There are three restrictions that must be considered:
1) Data dependency
2) Bottleneck time of the processing units in the pipeline
3) Limited hardware resource

The first limitation is data dependency which can be classified into two types: intra data dependency and inter data dependency. Intra data dependency stems from the reconstruction loop which is caused by the intra prediction [28]. Taking the AVS standard for example, the data dependency comes from the block-level intra luma blocks, as shown in Fig. 5, and the pipeline will be interrupted once every five modes to wait for the reconstruction of the proceeding block to finish. In detail, to perform the intra prediction (IP), block "B-1" needs the rightmost column pixels of reconstructed "B-0", block "B-2" needs the bottommost line pixels of reconstructed "B-1" (except for "B-0") and block "B-3" needs the rightmost column pixels of reconstructed "B-2". The second type, inter data dependency, mainly refers to that the predicted motion vector (PMV) of current MB or block can be generated only after motion vectors (MVs) of neighboring blocks are determined. Different techniques like using modified PMVs to replace the exact PMVs [28] or adopting zigzag coding order [29] are proposed to address the inter-prediction data dependency problem. This work uses multi-stage motion vector prediction (MVP) schedule strategy [30] to break the inter-prediction data dependency. Thus, we just need to take the first type of data dependency, intra data dependency, into account in our throughput modeling of RDO-based MD.

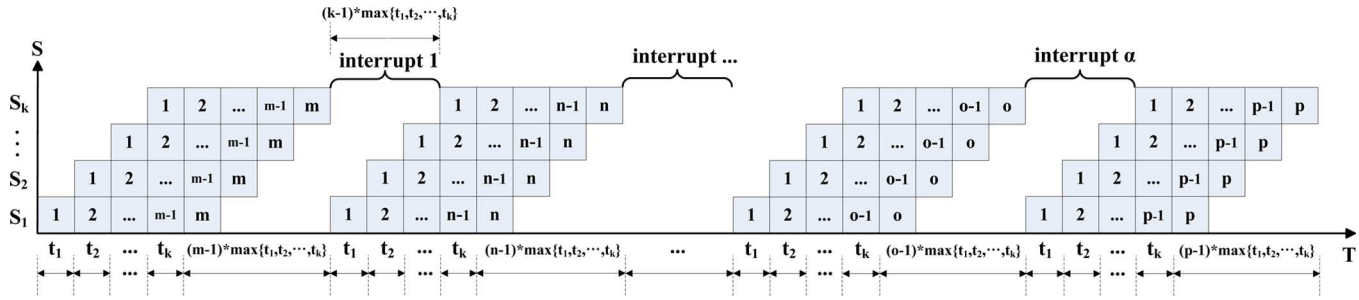

Fig. 6. k-stage pipeline space-time diagram of q ADDs.

Another limitation is the bottleneck time of the pipeline. Different from the ideal pipeline, we cannot reduce the processing time of each stage unlimitedly and we cannot balance all the stages perfectly due to the inherent processing dependency and the unbearable control complexity. Therefore, in practice different stages have different time consumptions and the stage with the maximum time cost becomes the bottleneck of the pipeline.

The third limitation comes from the hardware limitation in the pipeline design. Although high efficiency pipeline is targeted, the tradeoff between pipeline throughput and hardware cost still needs to be made considering the limitation of the hardware resource. What's more, the memory usage will increase significantly to buffer the output of the additional stages when the number of stages increases.

We next deduce the throughput of pipeline considering the first limitation. Limitation 2 and 3 will be taken into account during the specific AVS pipeline design in Section IV.

C. Pipeline Throughput Modeling of RDO-Based MD

Assuming that the whole MD process is divided into k functional processing stages, we now model the pipeline throughput. Let N be the total assignments of the k-stage pipeline. $t_1, t_2, \ldots, t_k$ denote the processing time of each stage and the whole pipeline is interrupted by α times. Assuming q assignments in N belong to ADDs, therefore the $N-q$ assignments belong to ANDDs. The space-time diagram of the q ADDs is depicted as Fig. 6.

Let $m, n, \ldots o, p$ be the number of each successive ADD assignments group in Fig. 6 and then we have $q = m + n + \ldots + o + p$. According to Fig. 6, the total time of the q ADDs can be written as

$$\begin{aligned} T_q = &\sum_{i=1}^{k} t_i + (m-1)\cdot\max\{t_1,t_2,\ldots,t_k\} + \\ &\sum_{i=1}^{k} t_i + (n-1)\cdot\max\{t_1,t_2,\ldots,t_k\} + \\ &\qquad \cdots \ + \ \cdots \ + \\ &\sum_{i=1}^{k} t_i + (o-1)\cdot\max\{t_1,t_2,\ldots,t_k\} + \\ &\sum_{i=1}^{k} t_i + (p-1)\cdot\max\{t_1,t_2,\ldots,t_k\} \end{aligned} \tag{6}$$

Considering the pipeline is interrupted by α times and $q = m + n + \ldots + o + p$, so (7) holds.

$$\begin{aligned} &(m-1) + (n-1) + \ldots + (o-1) + (p-1) \\ &= m + n + \ldots + o + p - 1 - 1 - \ldots - 1 - 1 \\ &= q - (\alpha + 1) \end{aligned} \tag{7}$$

Substituting (7) into (6), we obtain

$$T_q = (\alpha+1)\cdot\sum_{i=1}^{k} t_i + (q-\alpha-1)\cdot\max\{t_1,t_2,\ldots,t_k\} \tag{8}$$

From Fig. 6, we notice that there are many "holes" in the space-time diagram due to the interruptions and we can use the $N-q$ ANDDs to fill the "holes". In the pipelined space-time diagram, the $N-q$ ANDDs will cost

$$T_{N-q} = (N-q)\cdot\max\{t_1,t_2,\ldots,t_k\} \tag{9}$$

The wasted time of the α "holes" can be written as

$$T_{hole} = \alpha\cdot(k-1)\cdot\max\{t_1,t_2,\ldots,t_k\} \tag{10}$$

Then the time consumption of the total N assignments (T_{total}) can be summarized to have two situations: $T_{N-q} \le T_{hole}$ and $T_{N-q} > T_{hole}$ which are depicted in Figs. 7 and 8, respectively.

Through detailed observation, T_{total} can be developed as

$$T_{total} = \begin{cases} T_q & T_{N-q} \le T_{hole} \\ T_q + T_{N-q} - T_{hole} & T_{N-q} > T_{hole} \end{cases} \tag{11}$$

Combining (8)–(10) for $T_{N-q} > T_{hole}$ case to produce (12), shown at the bottom of the next page.

Substituting (8) and (12) into (11) yields (13), shown at the bottom of the next page.

According to (3) and (13), we then get the throughput of the pipeline with data dependency as (14), shown at the bottom of the next page.

Equation (14) indicates: for $k \ge (N-q)/\alpha + 1$, the throughput of a pipeline with data dependency depends on the total assignments (N), the total interrupting times (α), the number of ADDs (q), the bottleneck $(\max\{t_1,t_2,\ldots,t_k\})$ and $\sum_{i=1}^{k} t_i$ which denotes the total processing time of k stages; for $k < (N-q)/\alpha + 1$, the throughput also depends on k in addition to the other variables talked above. In the next section, we will show the specific pipeline design in AVS based on (14).

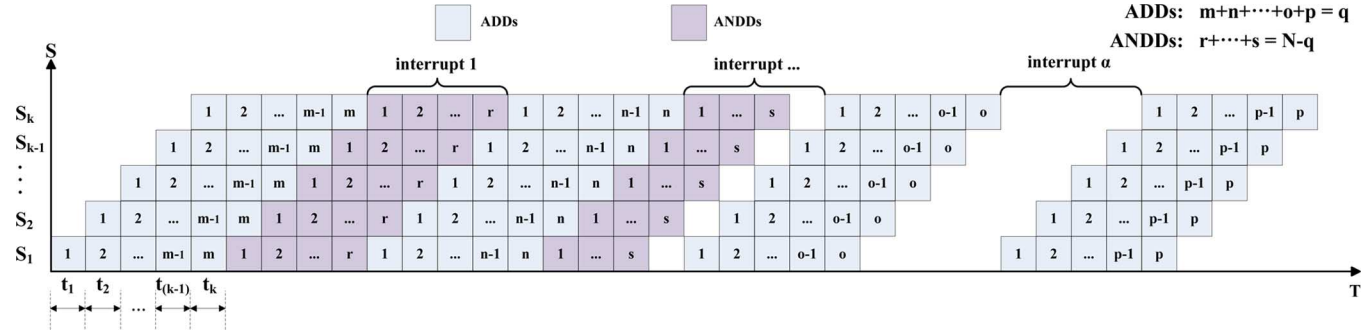

Fig. 7. k-stage pipeline space-time diagram of N assignments when $T_{N-q} \leq T_{hole}$.

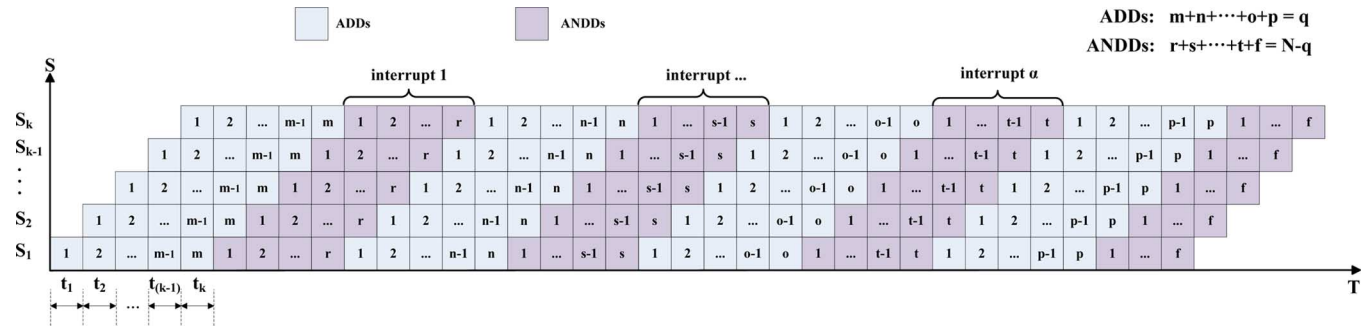

Fig. 8. k-stage pipeline space-time diagram of N assignments when $T_{N-q} > T_{hole}$.

IV. Pipeline Design of RDO-Based MD in AVS and the Proposed Architecture

A. Throughput Analysis for I, P and B Frames in AVS

Based on the analysis in Section III, we know that the data dependency comes from 4 intra luma blocks in AVS, and the corresponding ADDs space-time diagram of I frame is depicted in Fig. 9.

At first glance of Fig. 9, all the 20 prediction blocks (4×5) of 4 intra luma blocks belong to ADDs, but through detailed analysis we find that some intra luma modes can be converted to ANDDs by smart pipeline scheduling [31] which is shown in the following.

$$\begin{aligned} & T_q + T_{N-q} - T_{hole} \\ &= (\alpha+1) \cdot \sum_{i=1}^{k} t_i + (q-\alpha-1) \cdot \max\{t_1, t_2, \ldots, t_k\} \\ &\quad + (N-q) \cdot \max\{t_1, t_2, \ldots, t_k\} - \alpha \cdot (k-1) \cdot \max\{t_1, t_2, \ldots, t_k\} \\ &= (\alpha+1) \cdot \sum_{i=1}^{k} t_i + (N-\alpha \cdot k-1) \cdot \max\{t_1, t_2, \ldots, t_k\} \end{aligned} \tag{12}$$

$$T_{total} = \begin{cases} (\alpha+1) \cdot \sum_{i=1}^{k} t_i + (q-\alpha-1) \cdot \max\{t_1, t_2, \ldots, t_k\} & k \geq \frac{N-q}{\alpha} + 1 \\ (\alpha+1) \cdot \sum_{i=1}^{k} t_i + (N-\alpha \cdot k-1) \cdot \max\{t_1, t_2, \ldots, t_k\} & k < \frac{N-q}{\alpha} + 1 \end{cases} \tag{13}$$

$$throuthput_k = \begin{cases} \dfrac{N}{(\alpha+1) \cdot \sum_{i=1}^{k} t_i + (q-\alpha-1) \cdot \max\{t_1, t_2, \ldots, t_k\}} & k \geq \frac{N-q}{\alpha} + 1 \\ \dfrac{N}{(\alpha+1) \cdot \sum_{i=1}^{k} t_i + (N-\alpha \cdot k-1) \cdot \max\{t_1, t_2, \ldots, t_k\}} & k < \frac{N-q}{\alpha} + 1 \end{cases} \tag{14}$$

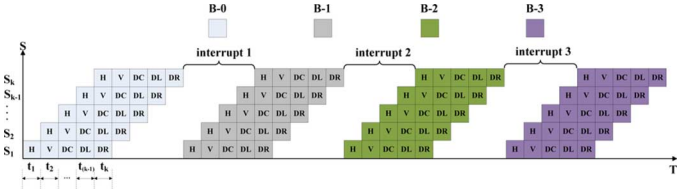

Fig. 9. Pipeline space-time diagram of intra luma blocks for I frame.

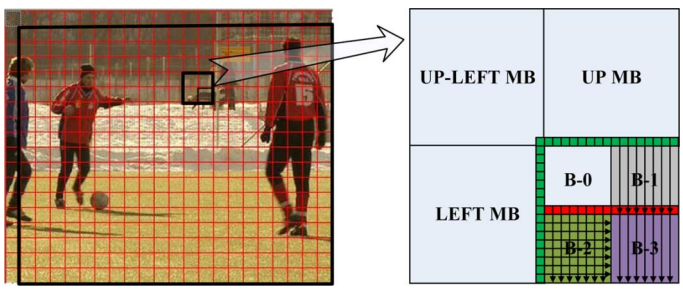

Fig. 10. Intra prediction of current MB by using neighboring reconstructed MBs in advance.

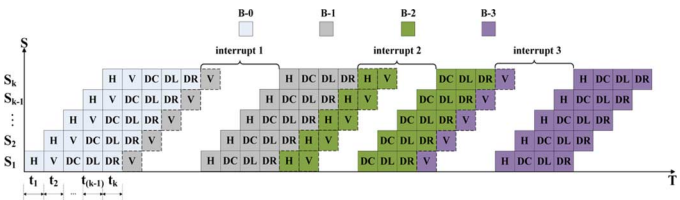

Fig. 11. Convert four "special" ADD prediction modes to ANDDs through intelligent schedule arrangement.

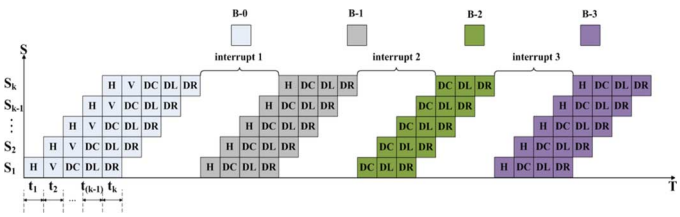

Fig. 12. Pipeline space-time diagram of all intra ADDs for I frame.

For an I frame which is to be encoded as shown in Fig. 10, all the MBs except for the rightmost column and topmost row of MBs, will have UP-MB, LEFT-MB and UP-LEFT-MB neighbors which have already been encoded. Therefore, when encoding a MB in the black pane in Fig. 10, we can make use of the reconstructed pixels of the neighboring MBs in advance.

Further observation shows that not all the modes of block "B-1" need reconstructed pixels of "B-0" to perform the intra prediction. The Vertical mode intra prediction of "B-1" can be performed directly which does not need to wait for the reconstruction of "B-0" because the needed reconstructed pixels of the UP-MB is already available. Similarly, the Horizontal intra prediction of "B-2" can be performed immediately too. In addition, we also notice that the Vertical intra prediction of "B-2" can be performed right after the reconstruction of "B-0". Similarly, the Vertical intra prediction of "B-3" does not need to wait for the reconstruction of "B-2" to complete. Therefore, these 4 "special" prediction modes can be treated as ANDDs to "fill" the "holes" in Fig. 9 by intelligent scheduling arrangement, as shown in Fig. 11.

Now, the space-time diagram of all intra ADDs can be re-drawn as Fig. 12.

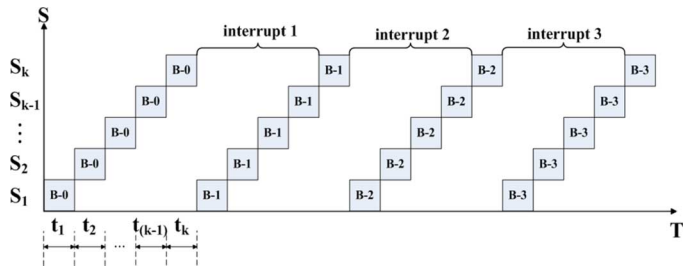

Fig. 13. Pipeline space-time diagram of all ADDs for P and B frames.

TABLE II
ASSIGNMENTS, ADDS AND INTERRUPTIONS FOR I, P AND B FRAMES

PICTURE TYPE	N	q	α
I	28	16	3
P	30	4	3
B	36	4	3

From Section II we know that for P and B frames we use SAD to choose the best block-level intra modes. Therefore, the only 4 ADDs are the 4 best intra luma prediction modes and the pipeline space-time diagram of all ADDs is shown in Fig. 13.

Together with Table I and Figs. 12–13, we summarize N, q and α for I, P and B frames, shown in Table II.

Substituting N, q and α into (14), we obtain throughput for I, P and B frames, respectively, as shown in (15) to (17) at the bottom of the next page.

We now see from (15) to (17) that the throughput depends on 3 factors: the stage number (k), the total time consumption of all stages $(\sum_{i=1}^{k} t_i)$, which just slightly changes in different pipeline structures, and the bottleneck $(\max\{t_1, t_2, \ldots, t_k\})$ which lies on the selection of k. Then the most important concern will be how to select the optimal pipeline stage number k to produce maximum throughput. In the following, we first define our basic processing unit set and give the corresponding time consumption of each unit. Then based on the basic processing unit set we select the optimal pipelining stages for I, P and B frames by different "combination" manners of the basic units.

B. Basic Processing Unit Set

In this work, we split the *RDcost* calculation into small processing units and define the basic processing unit set $\{\mathrm{BPU}\}$ as {DCTH, DCTV, Q, IQ, ZIGZAG, IDCTH, VLC, IDCTV, RECMD}, among which (I)DCTH/V denotes the (inverse) horizontal/vertical DCT and RECMD (reconstruction and mode decision) unit is responsible for reconstruction process, *RDcost* value generation and mode decision.

All the processing units in $\{\mathrm{BPU}\}$ process one 8×8 block at one time, and we adopt 8-pixel parallelism (processing 8 pixels in each cycle) to make the tradeoff between the hardware resource cost and the bottleneck time (maximum time consumption) of basic processing units. Besides, although this work aims at high throughput pipeline design, higher parallelism for basic processing units is not adopted based on the fact that more parallel degree does not guarantee higher processing speed for some special basic processing units like ZIGZAG, which will be explained in the following. For one 8×8 block, ZIGZAG needs at least 64 cycles to judge the 64 quantized coefficients

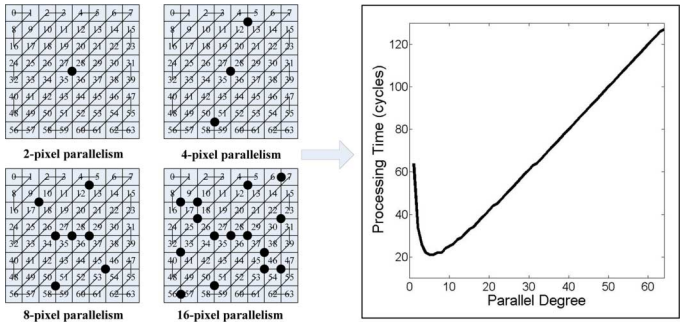

Fig. 14. Relationship between processing time and parallel degree of ZIGZAG.

TABLE III
TIME CONSUMPTION OF BASIC PROCESSING UNITS

Processing Unit (8-pixel parallelism)		Time Consumption (cycles)	
DCT-H		18	
DCT-V		18	
Q		16	
IQ	ZIGZAG	14	22
IDCTH	VLC	18	21
IDCTV	-	18	-
RECMD		17	

in serial scan strategy. Dividing the 64 coefficients into several groups and scanning all groups in parallel manner can accelerate the generation of (*run*, *level*) pairs, where *run* indicates the number of zeros before a non-zero value and *level* indicates the sign and magnitude of the non-zero coefficient. However, due to the segmentation, as shown in Fig. 14, the *run* of the last (*run*, *level*) pair generated in each group may be wrong and additional cycles are required to fix those wrong pairs one by one. With the increase of parallel degree (more groups), more (*run*, *level*) pairs are needed to be fixed and therefore the processing time of ZIGZAG rises sharply, as depicted in Fig. 14 (right part), thus more parallel degree does not produce higher processing speed.

We now present the time consumption of all basic processing units under 8-pixel parallelism. DCTH and DCTV both need 18 cycles to process an 8×8 block. Different from [17] and [18], which adopt 4-pixel parallel Q and IQ, the processing time of our 8-pixel parallel Q and IQ are 16 and 14 cycles, respectively. The time consumption of IDCTH and IDCTV are both 18 cycles, which is equal to the time consumption of DCTH and DCTV. Generally, ZIGZAG and VLC are the two most time-consuming processing units. Our previous work [22] used a 4-pixel parallel zigzag scan to address these two bottlenecks of the basic processing units. The processing time of VLC is not fixed since it is related to the number of (*run*, *level*) pairs. In [22], through a large scale statistical analysis, the processing time of the 4-pixel parallel VLC is about 24 cycles on average. However, in the situation of extremely complex scenes or when the quantization step is very small, the time consumption of VLC will reach 28 cycles. In this paper, the time consumption of the adopted 8-pixel parallel ZIGZAG and VLC, even in the worst situation, are no more than 22 cycles and 21 cycles, respectively. We summarize the time consumption of all the basic processing units in $\{\mathrm{BPU}\}$ as Table III.

Based on the defined set $\{\mathrm{BPU}\}$ and Table III, we next present the selection of the optimal pipelining stage number k by combining the basic processing units of $\{\mathrm{BPU}\}$ in different ways.

C. Pipeline Stage Number k Selection

Before selecting the optimal pipeline stage number k, we first point out that k should be no more than 7 based on the fact that $\{\mathrm{BPU}\}$ can just supports up to 7-stage pipeline, and more stages do not lead to higher throughput.

$$throughput_{k\_I} = \begin{cases} \dfrac{28}{4 \cdot \sum_{i=1}^{k} t_i + 12 \cdot \max\{t_1, t_2, \ldots, t_k\}} & k \geq 5 \\ \dfrac{28}{4 \cdot \sum_{i=1}^{k} t_i + (27 - 3 \cdot k) \cdot \max\{t_1, t_2, \ldots, t_k\}} & k < 5 \end{cases} \tag{15}$$

$$throughput_{k\_P} = \begin{cases} \dfrac{30}{4 \cdot \sum_{i=1}^{k} t_i} & k \geq 10 \\ \dfrac{30}{4 \cdot \sum_{i=1}^{k} t_i + (29 - 3 \cdot k) \cdot \max\{t_1, t_2, \ldots, t_k\}} & k < 10 \end{cases} \tag{16}$$

$$throughput_{k\_B} = \begin{cases} \dfrac{36}{4 \cdot \sum_{i=1}^{k} t_i} & k \geq 12 \\ \dfrac{36}{4 \cdot \sum_{i=1}^{k} t_i + (35 - 3 \cdot k) \cdot \max\left\{t_1, t_2, \ldots, t_{(k)}\right\}} & k < 12 \end{cases} \tag{17}$$

TABLE IV
PIPELINE THROUGHPUTS WITH DIFFERENT STAGE NUMBER K FOR I, P AND B FRAMES

k	$\sum_{i=1}^{k} t_i$	$\max\{t_1, t_2, ..., t_k\}$	T_{total_I}	$throughput_I$	T_{total_P}	$throughput_P$	T_{total_B}	$throughput_B$	Pipelining architecture
1	88	88	2464	0.0114	2640	0.0114	3168	0.0114	DCTH-DCTV-Q | IQ-IDCTH-IDCTV ZIGZAG-VLC | RECMD
2	88	48	1360	0.0206	1456	0.0206	1744	0.0206	DCTH-DCTV-Q → IQ-IDCTH-IDCTV ZIGZAG-VLC | RECMD
3	106	40	1144	0.0245	1224	0.0245	1464	0.0246	DCTH-DCTV-Q → IQ-IDCTH ZIGZAG-VLC → IDCTV-RECMD
4	108	33	927	0.0302	993	0.0302	1191	0.0302	DCTH → DCTV-Q → IQ-IDCTH ZIGZAG-VLC → IDCTV-RECMD
5	102	24	696	0.0402	744	0.0403	888	0.0405	DCTH → DCTV-Q → IQ-IDCTH ZIGZAG → IDCTV VLC → RECMD
6	112	22	712	0.0393	690	0.0435	822	0.0438	DCTH → DCTV → Q → IQ-IDCTH ZIGZAG → IDCTV VLC → RECMD
7	130	22	784	0.0357	696	0.0431	828	0.0435	DCTH → DCTV → Q → IQ ZIGZAG → IDCTH VLC → IDCTV FIFO → RECMD

TABLE V
TIME CONSUMPTION OF EACH STAGE IN DIFFERENT PIPELINE ARCHITECTURES

k	t_1	t_2	t_3	t_4	t_5	t_6	t_7
1	88	-	-	-	-	-	-
2	40	48	-	-	-	-	-
3	40	33	33	-	-	-	-
4	18	24	33	33	-	-	-
5	18	24	22	21	17	-	-
6	18	18	16	22	21	17	-
7	18	18	16	22	21	18	17

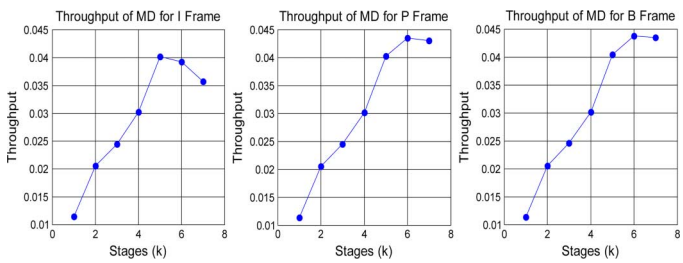

Fig. 15. MD Pipeline Throughput For I, P and B Frames.

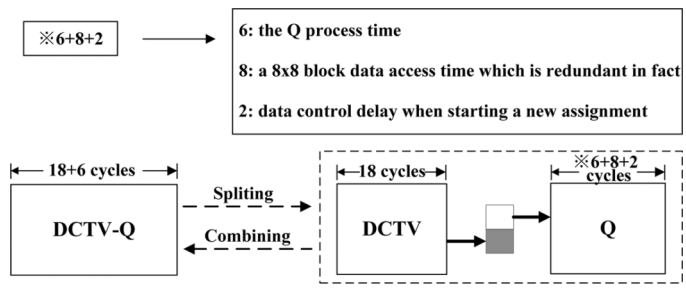

Fig. 16. DCTV-Q splitting/combining.

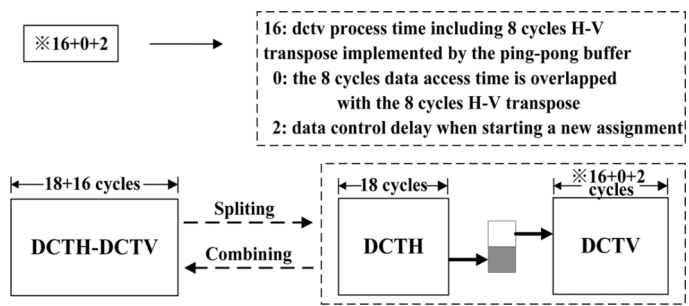

Fig. 17. DCT splitting/combining.

According to (15)–(17) and Table III, we can get the throughputs of different pipeline structures with different k through combining and splitting technique. Here, "combining" means merging the small basic processing units to form big processing unit and "splitting" refers to splitting big processing unit apart into smaller processing units (the minimum unit is basic processing units in $\{\mathrm{BPU}\}$).

The results are summarized into Table IV. Table V shows the time consumption of each stage in different pipeline structures. Note that the results in Table IV are specific cases for our work which directly depend on Table III; if the $\{\mathrm{BPU}\}$ and time consumption of the basic processing units in $\{\mathrm{BPU}\}$ change, the results will be different. In the following, we select the optimal pipeline stages for I, P and B frames based on Table IV and further analyze the results in the table.

From Table IV and Fig. 15, we can clearly find that 5-stage MD pipeline structure will produce the highest throughput for I frame and 6-stage pipelining will yield the highest throughput for P and B frames. Another observation from Table IV is that $\sum_{i=1}^{k} t_i$ is increasing with the increase of k. It is because splitting a big processing unit into 2 smaller processing units without inherent data dependency brings additional data access time. For example, if

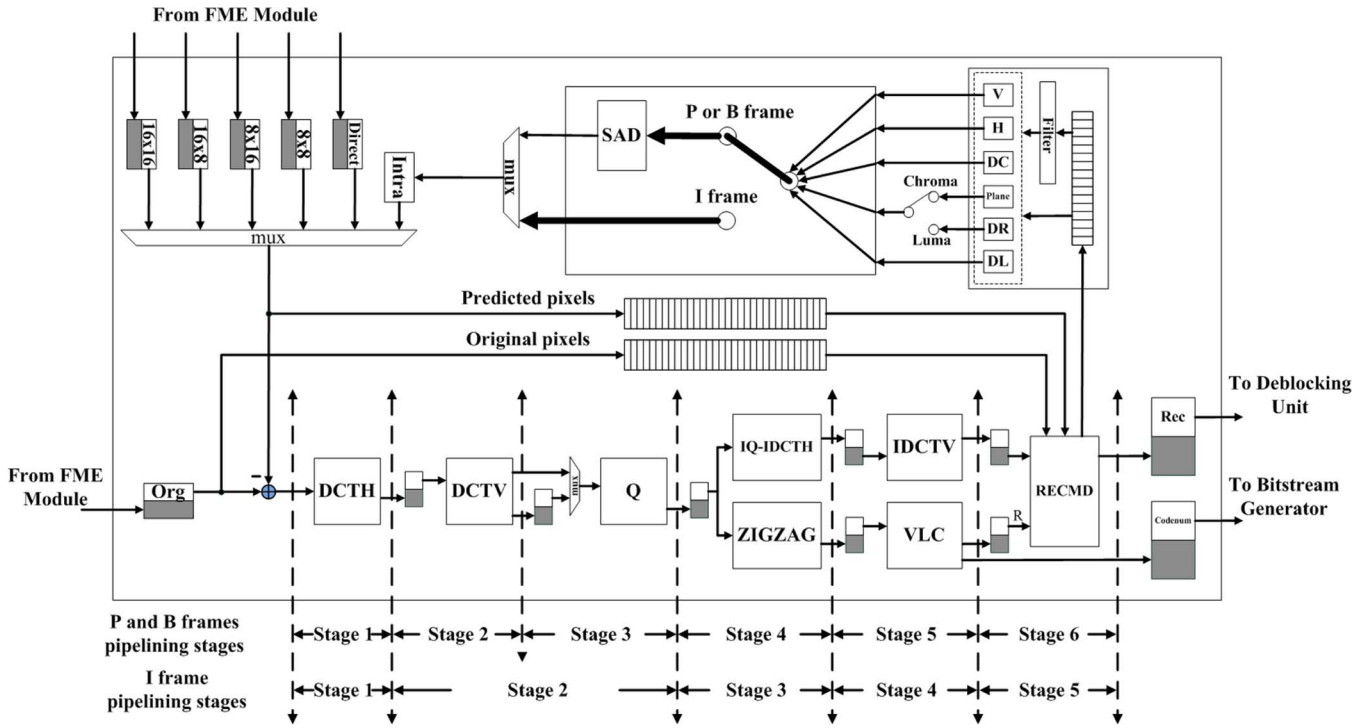

Fig. 18. Our proposed MD pipeline architecture.

we split DCTV-Q into DCTV and Q, we need additional eight cycles of data access time which is redundant in fact and that leads to the increase of $\sum_{i=1}^{k} t_i$, as shown in Fig. 16.

However, $\sum_{i=1}^{k} t_i$ will not increase when splitting an unit with inherent data dependency, such as splitting DCT into DCTH and DCTV, as shown in Fig. 17. DCTV cannot be started unless all the outputs of DCTH are transposed from the rows into columns (H-V transpose). Therefore, in this situation the 8 cycles data access time, with which to complete the H-V transpose, is not redundant. Furthermore, for I frame when k is more than 5, as $\sum_{i=1}^{k} t_i$ increases and bottleneck time of the pipeline cannot be reduced efficiently, the throughput begin to decrease; for P and B frames, when k is more than 6, also due to the increase of $\sum_{i=1}^{k} t_i$ and the bottleneck of pipeline, the throughput drops slightly, as shown in Fig. 15. In Table IV, the bottlenecks of different pipeline architectures are marked as purple. Therefore, to obtain the highest throughput we choose 5-stage pipeline for I frame and 6-stage for P and B frames. We propose an adaptive pipeline architecture which can support both 5-stage and 6-stage pipeline structures and the whole structure of mode decision will be discussed in detail in the next.

D. Proposed Adaptive Pipeline Architecture

As Fig. 18 shows, for P and B frames, the RDO core (lower part in Fig. 18) is divided into 6 stages and the first to third stages are DCTH, DCTV and Q. After quantization, the pipeline is divided into 2 branches. IQ-IDCTH and ZIGZAG both belong to the fourth stage, while IDCTV and VLC are at the same stage, the fifth stage. At the last stage (the sixth) the 2 branches of the pipeline again merge into one part: RECMD. RECMD calculates SSD with the output data from IDCTV, the original and predicted data from the on-chip buffer. Then the *RDcost*, with which to choose the best mode, can be produced based on SSD and the real coding bits R which have already been generated in VLC. All the buffers between 2 consecutive stages are ping-pong mode buffers. For I frame, the only differences are that the ping-pong buffer between DCTV and Q is bypassed and the output port of DCTV is directly connected with the input port of Q. In this situation, DCTV and Q are combined to form a big processing unit DCTV-Q and the pipeline switch to 5-stage from 6-stage adaptively. The corresponding space-time diagrams of I, P and B frames are shown in Figs. 19, 20 and 21, respectively.

The time consumption of intra prediction has not yet been taken into account in the above pipeline design and the further analysis will be made in the following.

As shown in Fig. 18, parallel architecture is adopted for intra prediction and 5/4 modes of one luma/chroma block can be performed in every 17 cycles. For I frame, the pipeline needs to wait 4 times for intra prediction, marked by blue arrows in Fig. 19, so additional $17 \times 4 = 68$ cycles are needed for MD. By careful observation from Figs. 20–21, we can find that "B-0" of intra mode is not the first assignment entering into the pipeline in P and B frames. Besides, when "B-0" of intra mode is finished, "B-1" is not put into pipeline at once and "Cr" is inserted instead. The same happens to "B-2" and "B-3" of intra mode. In fact, for P and B frames, we insert one ANDD block before each ADD block ("B-0", "B-1", "B-2" and "B-3" of intra mode) to avoid the wait. Therefore, we do not need any additional cycles to wait intra prediction in processing P and B frames.

V. Performance Evaluation and Implementation Results

A. Encoding Performance Comparisons

Before the implementation of our architecture in hardware, we first verify the encoding performance of our proposed MD

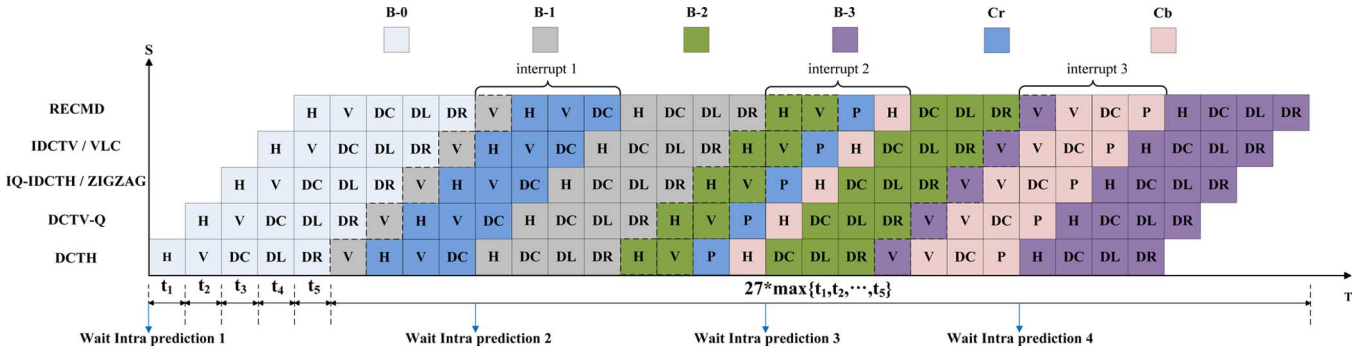

Fig. 19. Pipeline space-time diagram for I-frame.

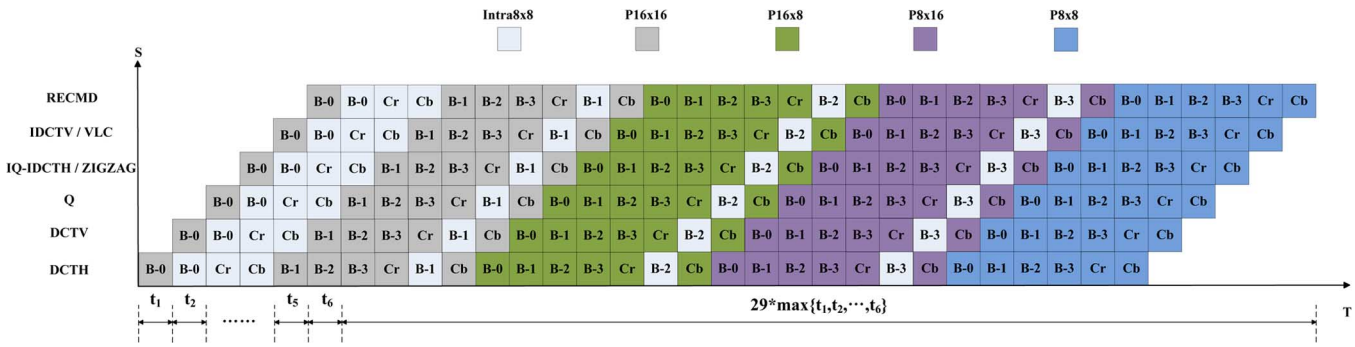

Fig. 20. Pipeline space-time diagram for P-frame.

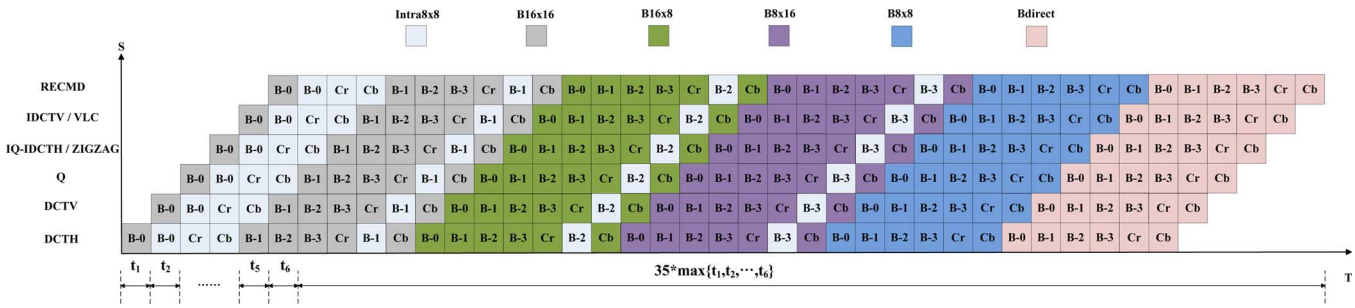

Fig. 21. Pipeline space-time diagram for B-frame.

method. Fast hardware MD method in the first category (as illustrated in Section I) like Yin *et al.*'s work [17] and the traditional LCMD method of the second category like work [19] are tested to make comparisons with our proposed MD method. High coding performance is reported in Zhao *et al.*'s work [13] and we include it to further validate our method, although we have not found the hardware version of [13] in the literature. To ensure the validity of the comparisons, we implement all the above algorithms in the same encoder platform-RM52J (AVS reference code). Different sequences under different quantization parameters (ranging from 28 to 40) are tested and Figs. 22–23 show the RD curves of the selected 12 high definition sequences. More test results are tabulated in Table VI and the results show that our proposed method achieves the best coding performance among all 4 MD methods: it far outperforms the traditional LCMD method (0.57 dB PSNR gain in average) and exceeds the MD methods in work [13] (0.04 dB PSNR gain in average) and [17] (0.15 dB PSNR gain in average). LCMD has bigger performance penalty against the other 3 methods due to turning off the RDO. The inaccuracy of the R-D estimating model and the less candidate modes introduce the PSNR loss in [13] and [17] compared with our method, respectively. PSNR and bit-rate differences in Table VI are calculated according to the numerical averages [32] between the RD-curves of our proposed method and the other 3 algorithms. Note that PSNR and bit-rate differences should be regarded as equivalent, i.e., there is either the increase in PSNR or the decrease in bit-rate—not both at the same time.

B. Implementation and Comparisons

The internal implementation architectures of key modules (DCT/IDCT, Q/IQ) in MD are depicted in Fig. 24. All the internal architectures are pipelined designs which guarantee high implementation frequency.

The needed operation frequency of an encoder system being real-time is given by

$$frequency = MBCycle \times MBnum \times fr \qquad (18)$$

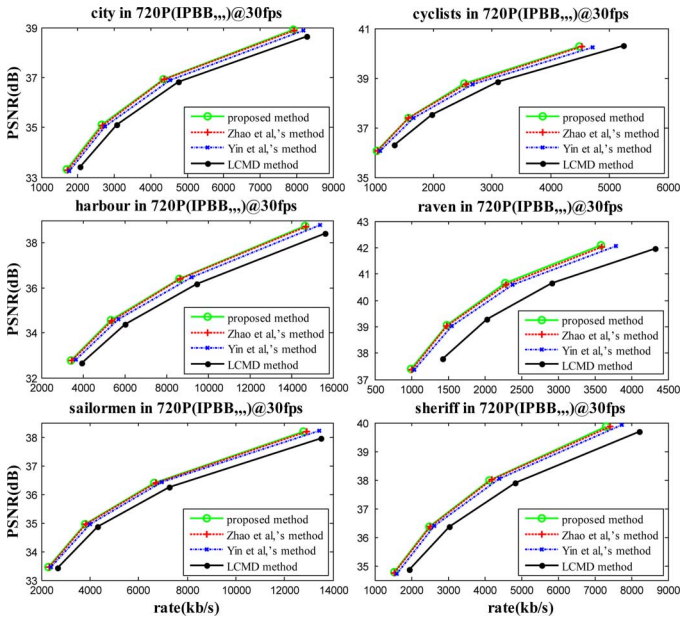

Fig. 22. Encoding performance of different MD methods (720P).

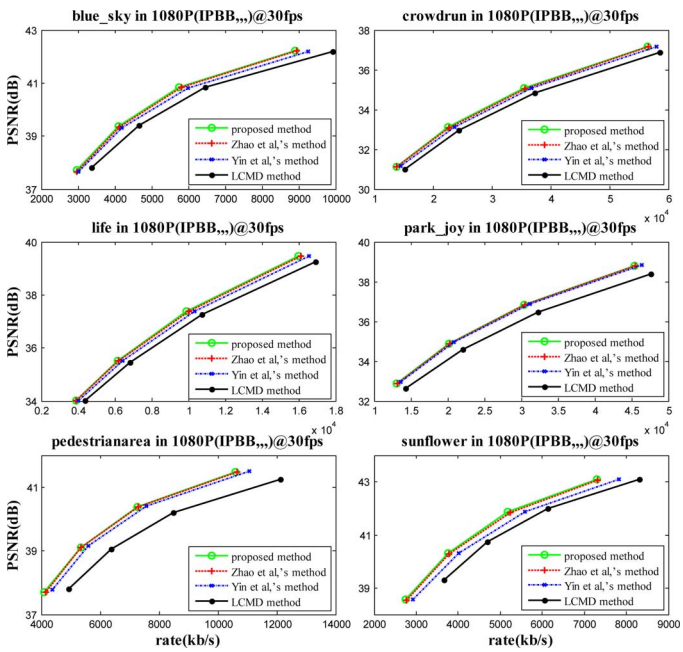

Fig. 23. Encoding performance of different MD methods (1080P).

where *MBCycle* and *MBnum* represent the cycles to process one MB and the number of the total MBs in one frame, respectively. *fr* is encoding frame rate.

The actual implementation results show that our proposed architecture can accomplish MD in 776 cycles, 698 cycles, and 832 cycles for I, P and B frame, respectively. We implemented our design based on SMIC 0.18-μm CMOS technology; the on-chip memory is 84760 bits and the gate count (2-input NAND gate) is 232054 as shown in Table VII. The final implementation frequency of our MD module is 237 MHZ. In case of 1080 HD sequence at 30 fps, *MBnum* will be 8160. To roughly estimate the needed operation frequency, let *MBCycle* be 832, and then the required frequency will be $832 \times 8160 \times 30$ HZ (204 MHZ). In other words, with 237 MHZ our system can sufficiently supports real time processing of 1080P@30 fps.

The comparisons with the previous hardware encoder designs are illustrated in Table VIII. In this paper, we reuse DPCM loop and VLC for our RDO-based MD and the only additional hardware cost for MD is RECMD unit. The gate count in Table VIII includes DPCM loop, VLC and MD process (not including deblocking). Many previous works like [21] place intra MD in IP module and inter MD in FME module. We add the gate count of IP module and EC (entropy coding) module together to approximate the total gate count of work [21]. Compared with the previous work [22], the processing capacity of our design is increased by 11.3%, 19% and 17% for I, P and B frames, respectively. Although work [21] achieves higher processing capacity, it just used SATD-based inter MD and edge based intra MD, which produces lower coding performance compared with our RDO-based MD method.

C. Encoder System Integration With MD Module

Finally, our MD module is integrated to an encoder chip developed by National Engineering Laboratory for Video Technology (NELVT) of Peking University, as shown in Fig. 25. Firmware is used for frame-level control, responsible for parameter configuration. Capture module captures original video data and stores it into external memory. Efficient four-stage MB-pipelining architecture is applied for our encoder core in the purple block and the five major components are integer motion estimation (IME), fractional motion estimation (FME), MD, deblocking filter (DB) and bit stream generating (BG). Besides, MB level pipelining controller dispenses MB-level runtime parameters for each module and multi-stage motion vector prediction (MVP) schedule strategy [30] is adopted to produce predicted motion vectors for IME, FME and BG.

As Fig. 25 depicts, the proposed MD is at the third stage of the MB-level pipeline. It receives necessary configuration parameters (QP, λ, picture type, etc.) from MB level pipelining controller through local bus. Original pixels and inter-prediction pixels of different MB modes (16×16, 16×8, 8×16, 8×8, Direct) are passed to MD from FME. The two-dimensional quantized coefficients of the best mode are organized into one sequence in the form of (*run*, *level*) pairs in MD and then the (*run*, *level*) pairs are converted into *CodeNum*s according to VLC tables [24]. All the generated *CodeNum*s and MB header information including the best mode and MVs are stored in the ping-pong buffer between MD and BG which maps these information into bit stream. Besides, MD provides reconstructed pixels of the best mode to DB engine to generate standard reference frame for the next video coding frame.

VI. Conclusion

This work adopts a high-quality RDO-based mode decision scheme and designs the corresponding highly efficient pipeline architecture. It makes possible a truly RDO-based hardware real-time encoder. The proposed method does not skip any candidate mode and the coding performance method far outperforms the traditional LCMD method (0.57 dB PSNR gain in average) and exceeds Zhao *et al.*' method (0.04 dB PSNR gain in average) and Yin *et al.*'s method (0.15 dB PSNR gain

TABLE VI
PERFORMANCE COMPARISONS

Format	Sequence	Proposed method VS Zhao *et al.*'s method		Proposed method VS Yin *et al.*'s method		Proposed method VS LCMD method	
		PSNR Gain (dB)	**Bit-rate change (%)**	**PSNR Gain (dB)**	**Bit-rate change (%)**	**PSNR Gain (dB)**	**Bit-rate change (%)**
720P	*City*	0.0527	-1.450	0.1699	-4.554	0.4881	-12.672
	Crew	0.0172	-0.547	0.0970	-3.115	0.3915	-12.267
	Cyclist	0.0481	-1.645	0.1873	-6.295	0.5057	-16.088
	Night	0.0465	-1.179	0.1532	-3.824	0.5465	-12.995
	Optis	0.0369	-1.198	0.1433	-4.559	0.6824	-19.912
	Raven	0.0581	-1.538	0.2067	-5.501	0.9111	-22.149
	Sheriff	0.0305	-0.932	0.1585	-4.723	0.6225	-17.331
	Harbour	0.0537	-1.298	0.2177	-5.139	0.6393	-14.397
	Sailorman	0.0251	-0.912	0.1115	-3.974	0.4036	-13.736
	Shuttlestart	0.0564	-1.758	0.1460	-4.912	0.4312	-13.959
	Spincalendar	0.0244	-0.893	0.0822	-3.143	0.2857	-10.544
1080P	*Life*	0.0423	-1.078	0.1708	-4.285	0.4557	-11.051
	Park_joy	0.0190	-0.396	0.0905	-1.884	0.6648	-13.007
	Blue_sky	0.0594	-1.388	0.1789	-4.181	0.4952	-11.375
	Riverbed	0.0483	-0.849	0.0304	-0.535	1.4407	-23.505
	Sunflower	0.0564	-1.143	0.3165	-6.527	0.6340	-13.109
	Crowdrun	0.0406	-0.933	0.1506	-3.424	0.4738	-10.398
	In_to_tree	0.0277	-2.230	0.0646	-5.579	0.1350	-9.166
	Mobcal_ter	0.0431	-1.813	0.0668	-2.813	0.3510	-14.032
	Pedestrianarea	0.0184	-0.463	0.1472	-3.687	0.7621	-17.694

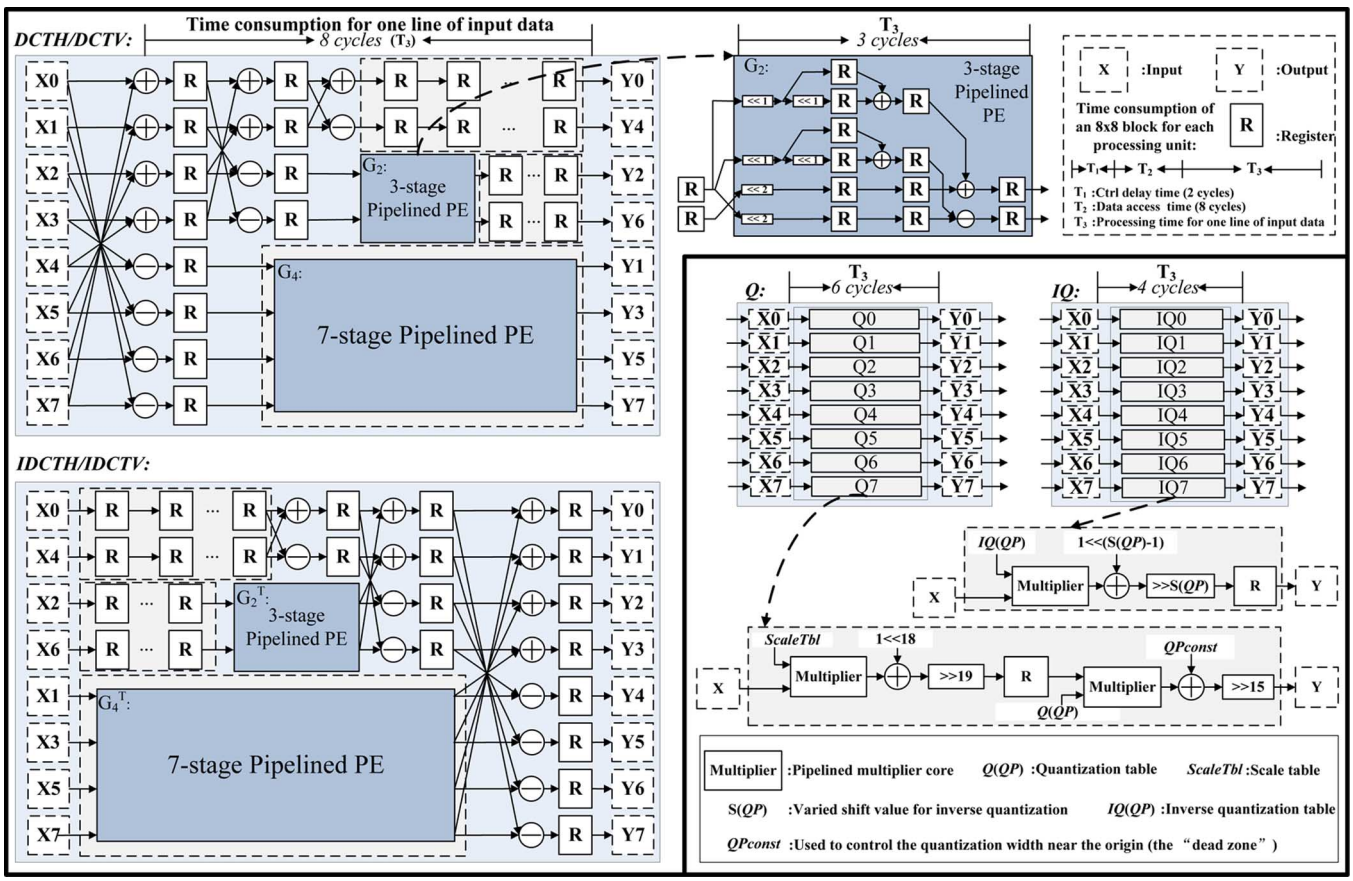

Fig. 24. Internal architectures of key modules in MD.

in average). The developed throughput model of RDO-based MD can be used to guide the pipeline design in different coding standards, like H.264 and AVS. The proposed architecture can support the real time processing of 1080P@30 fps and can be applied to high-quality and high-processing capacity HD video encoder systems. In the future, more studies will be conducted on tradeoffs between the processing speed and the hardware area of RDO-based MD in low power applications.

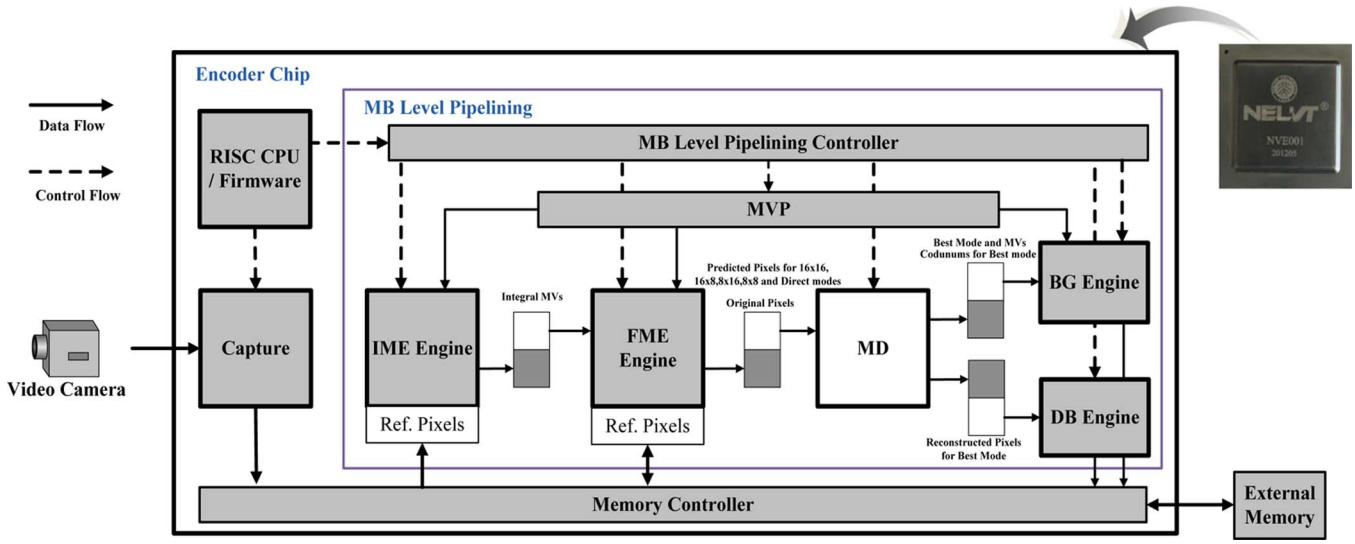

Fig. 25. MB-level pipeline structure and front view of the encoder chip.

TABLE VII
GATE COUNT

Functional module	Gate count (2-input NAND gate)
DCTH	8928
DCTV-Q	25660
ZIGZAG	17440
IQ-IDCTH	22188
VLC	50722
IDCTV	12660
RECMD	11304
OTHERS	83152
TOTAL	232054

TABLE VIII
DESIGN COMPARISONS

Design Feature		This work	Previous work [22]	Work [20]	Work [21]
Max operation freq.		237	234	125	200
Pixel parallelism		8	8	4	N/A
Standard		AVS-P2	AVS-P2	H.264 base line	H.264 base line [4.0]
CMOS technology		SMIC 0.18um	SMIC 0.18um	UMC 0.18um	TSMC 0.18um
Gate count		232K	215K	About 103K	Between 98K and 309K
Gate count only for MD		11K	11K	N/A	N/A
Max target size		HD 1920×1080	HD 1920×1080	HD 1280×720	HD 1920×1080
Cost function		Function (1)	Function (1)	Function (2) with Enhanced DCT-based SATD	SATD based inter MD and edge based intra MD
cycles/MB	I	776	864	1080	672
	P	698	831	N/A	672
	B	832	975	N/A	672

ACKNOWLEDGMENT

The authors would like to thank all the anonymous reviewers for their thoughtful comments and suggestions which help to improve the technical presentation of this paper.

REFERENCES

[1] *Video Codec Design: Developing Image and Video Compression Systems*. Chichester, U.K.: Wiley, 2002, pp. 1–2.
[2] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, ITU-T Rec. H.264 and ISO/IEC 14496-10, Mar. 2003, document JVT-G050.doc, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG.
[3] *Information Technology-Generic Coding of Audio-Visual Objects Part 2: Visual*, ISO/IEC 14496-2 MPEG-4, 1999.
[4] *AVS Video Expert Group, Information Technology—Advanced Audio Video Coding Standard Part 2: Video, in Audio Video Coding Standard Group of China (AVS)*, Doe. AVS-N1063, Dec. 2003.
[5] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 74–90, Nov. 1998.
[6] F. Pan, X. Lin, R. Susanto, K. P. Lim, Z. G. Li, G. N. Feng, D. J. Wu, and S. Wu, "Fast mode decision algorithm for intra prediction in H.264/AVC video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 813–822, Jul. 2005.
[7] J.-F. Wang, J.-C. Wang, J.-T. Chen, A.-C. Tsai, and A. Paul, "A novel fast algorithm for intra mode decision in H.264/AVG encoders," in *Proc. ISCAS2006*, Jul. 2006, pp. 3498–3501.
[8] M. Eom and Y. Choe, "Fast mode decision for intra prediction in H.264/AVC encoder," in *Proc. IEEE Int. Conf. Image Processing*, 2007.
[9] D. Wu, F. Pan, K. P. Lim, S. Wu, Z. G. Li, X. Lin, R. Susanto, and C. C. Kuo, "Fast intermode decision in H.264/AVC video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 953–958, Jul. 2005.
[10] C.-H. Tseng, H. M. Wang, and J.-F. Yang, "Enhanced intra 4×4 mode decision for H.264/AVC coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 8, pp. 1027–1032, Aug. 2006.
[11] M. G. Sarwer and L.-M. Po, "Fast bit rate estimation for mode decision of H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1402–1407, Oct. 2007.
[12] Y.-K. Tu, J.-F. Yang, and M.-T. Sun, "Efficient rate-distortion estimation for H264/AVC coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 5, pp. 600–611, May 2006.
[13] Z. Xin, S. Jun, S. Ma, and W. Gao, "Novel statistical modeling, analysis and implementation of rate-distortion estimation for H.264/AVC coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 5, pp. 647–660, May 2010.
[14] T.-C. Chen, S.-Y. Chien, Y.-W. Huang, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, and L.-G. Chen, "Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 6, pp. 673–688, Jun. 2006.
[15] J.-C. Wang, J.-F. Wang, J.-F. Yang, and J.-T. Chen, "A fast mode decision algorithm and its VLSI design for H.264/AVC intra prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1414–1422, Oct. 2007.

[16] R. Hashimoto, K. Kato, G. Fujita, and T. Onoye, "VLSI architecture of H.264 RD-based block size decision for 1080 HD," in *Proc. Picture Coding Symp., ThPM4.9.1–ThPM4.9.4*, Nov. 2007.
[17] H. Yin, X. H. Wang, X. K. Zhu, and H. Qi, "Hardware friendly mode decision algorithm for high definition AVS video encoder," in *Proc. 2nd Int. Congr. Image and Signal Processing, CISP'09*, 2009.
[18] X. Wang, C. Zhu, H. Yin, W. Gao, and X. Xie, "Fast mode decision based on RDO for AVS high definition video encoder," in *Lecture Notes in Computer Science, 2011 Advances in Multimedia Information Processing-PCM*, 2010, pp. 62–72.
[19] K. Babionitakis, G. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, I. Sifnaios, and N. Vlassopoulos, "A real-time H.264-AVC VLSI encoder architecture," *J. Real-Time Image Process.*, vol. 3, no. 1–2, pp. 43–59, 2008.
[20] C.-W. Ku, C.-C. Cheng, G.-S. Yu, M.-C. Tsai, and T.-S. Chang, "A high-definition H.264/AVC intra-frame codec IP for digital video and still camera applications," *Trans. Circuits Syst. Video Technol.*, vol. 16, no. 8, pp. 917–928, Aug. 2006.
[21] Z. Liu, Y. Song, M. Shao, S. Li, L. Li, S. Ishiwata, M. Nakagawa, S. Goto, and T. Ikenaga, "HDTV1080p H.264/AVC encoder chip design and performance analysis," *IEEE J. Solid-State Circuits*, vol. 44, no. 2, pp. 594–608, Feb. 2009.
[22] C. Zhu, Y. Li, H.-Z. Jia, X.-D. Xie, and H.-B. Yin, "A highly efficient pipeline architecture of RDO-based mode decision design for AVS HD video encoder," in *Proc. ICME*, Jul. 2011.
[23] W. Gao, C. Reader, F. Wu, Y. He, L. Yu, H. Lu, S. Yang, T. Huang, and X. Pan, "AVS—The Chinese next-generation video coding standard," in *National Association of Broadcasters*, Las Vegas, NV, USA, 2004.
[24] W. Gao, S. Ma, L. Zhang, L. Su, and D. Zhao, AVS Vide Coding Standard, vol. 280, 2010, pp. 125–166, Studies in Computational Intelligence, Intelligent Multimedia Communication: Techniques and Applications.
[25] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[26] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, document JVT-G050.doc, ITU-T Rec. H.264 and ISO/IEC 14496-10, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Mar. 2003.
[27] Z. Wei and K. N. Ngan, "A fast rate-distortion optimization algorithm for H.264/AVC," in *Proc. 32nd Int. Conf. Acoustics, Speech, and Signal Processing*, 2007.
[28] T.-C. Chen, C.-J. Lian, and L.-G. Chen, "Hardware architecture design of an H.264/AVC video codec," in *Proc. ACM Asia South Pacific Design Automation Conf.*, 2006.
[29] K. Wei, S. Zhang, H. Jia, D. Xie, and W. Gao, "A flexible and high-performance hardware video encoder architecture," in *Proc. IEEE Picture Coding Symposium (PCS)*, Krakow, Poland, May 2012, pp. 373–376.
[30] W. Yang, H. Yin, W. Gao, H. Qi, and X. Xie, "Multi-stage motion vector prediction schedule strategy for AVS HD encoder," in *2010 Digest of Technical Papers, Int. Conf. Consumer Electronics (ICCE)*, 2010.
[31] T. S. Kim, C. E. Rhee, and H.-J. Lee, "Prediction mode reordering and IDCT direction control for fast intra 8×8 prediction," in *Proc. 54th IEEE Int. Midwest Symp. Circuits and Systems*, Aug. 2011.
[32] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," presented at the 13th VCEG-M33 Meeting, Austin, TX, USA, Apr. 2001.

Chuang Zhu received the B.S. degree in computer science from The North University of China, Taiyuan, China, in 2008. He is currently pursuing the Ph.D. degree in the School of Electronics Engineering and Computer Science, Peking University, Beijing, China.

His research interests include video encoding, image processing and VLSI chip design.

Huizhu Jia received his Ph.D. degree in electrical engineering from the Chinese Academy of Sciences, Beijing, China, in 2007. He is currently with the National Engineering Laboratory for Video Technology, Peking University, Beijing. His research interests include: ASIC design, image processing, multimedia data compression and VR.

Shanghang Zhang received the B.S. degree from The Southeast University, Nanjing, China, in 2006. She is currently pursuing the Master degree in the School of Electronics Engineering and Computer Science, Peking University, Beijing, China.

Her research interests include video encoding, image processing and VLSI chip design.

Xiaofeng Huang received the B.S. degree in microelectronics from The Nanjing University of Posts and Telecommunications, Nanjing, China, in 2010. He is currently pursuing the Ph.D. degree in the School of Electronics Engineering and Computer Science, Peking University, Beijing, China.

His research interests include VLSI design, digital IC design and digital signal processing with focus on data transfer optimization and memory management for image and video applications.

Xiaodong Xie received his Ph.D. degree in electrical engineering, University of Rochester, USA. He was a Senior Scientist at Eastman Kodak Company, New York, USA, from 1994 to 1997; a Principal Scientist at Broadcom Corporation, California, USA, from 1997 to 2009.

He is currently a Professor with the Department of Electrical Engineering, Peking University, Beijing, China. His research interests include multimedia SoC design, embedded system. He holds 24 U.S. Patents.

Wen Gao (M'92–SM'05–F'09) received the Ph.D. degree in electronics engineering from the University of Tokyo, Japan, in 1991.

He is a professor in computer science at Peking University, China. Before joining Peking University, he was a professor at Harbin Institute of Technology from 1991 to 1995, and a professor at the Institute of Computing Technology of Chinese Academy of Sciences from 1996 to 2006. He has published extensively including five books and over 600 technical articles in refereed journals and conference proceedings in the areas of image processing, video coding and communication, pattern recognition, multimedia information retrieval, multimodal interface, and bioinformatics.

Prof. Gao served or serves on the editorial board for several journals, such as IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Multimedia, IEEE Transactions on Autonomous Mental Development, EURASIP Journal of Image Communications, Journal of Visual Communication and Image Representation. He chaired a number of prestigious international conferences on multimedia and video signal processing, such as IEEE ICME and ACM Multimedia, and also served on the advisory and technical committees of numerous professional organizations. He is a member of Chinese Academy of Engineering.