A HIGHLY EFFICIENT PIPELINE ARCHITECTURE OF RDO-BASED MODE DECISION DESIGN FOR AVS HD VIDEO ENCODER

Chuang Zhu[1], Yuan Li[1], Hui-zhu Jia[1], Xiao-dong Xie[1], Hai-bing Yin[2]

[1]National Engineering Laboratory for Video Technology, Peking university, Bei Jing, China
[2]School of Electrical Engineering, China Ji Liang University, Hang Zhou, China
[12]{czhu, yuanli, hzjia, xdxie, hbyin}@jdl.ac.cn

ABSTRACT

Like H.264, AVS video coding standard also uses macroblock (MB) based motion compensation (MC) and mode decision (MD). Rate distortion optimization (RDO) is the best known mode decision method, but with a high computational complexity that limits its applications. In our paper, firstly an MD algorithm based on RDO is given, which makes more mode candidates enter into RDO mode decision with little hardware resource increment. We further analyze the pipeline structure in detail, and implement a block-level 5-stage hardware pipeline. It can support the real time RDO mode decision processing of 1080P@30fps, and the coding efficiency is about 0.5db higher than the traditional SAD method. Our design is described in high-level Verilog/VHDL hardware description language and implemented under SMIC 0.18-μm CMOS technology with 215K logic gates and 80 KB SRAMs.

Index Terms— mode decision, RDO, AVS, pipeline

1. INTRODUCTION

AVS video coding standard, which is established by China Audio Video Coding Standard (AVS) Working Group, has been accepted as an option by ITU-TFGIPTV for IPTV applications. The AVS part 2 (AVS-P2) is high resolution friendly profile, which is also known as the Jizhun Profile of AVS.

Compared with the other coding standards, such as MPEG4, H.264/AVC main profile and above can achieve higher coding efficiency by adopting several kinds of complex techniques. But the corresponding substantial increase of computation becomes unbearable. The key reason of the unbearable computational burden is that H.264 coding standard provides an abundant set of intra-inter modes to choose from. While AVS-P2 offers less modes when compared with H.264. Further more, for high-resolution applications, AVS-P2 Jizhun Profile shows comparable performance with H.264/AVC for most progressive sequences [1]. So, RDO based mode selection becomes possible for AVS-P2 systems with the reduced number of modes to be considered.

Generally, there are two different categories of the algorithms to reduce the complexity of RDO method. Some algorithms are proposed in the first category, like those in [2-4], to simplify the calculating process of distortion (D) and rate (R). These algorithms try to do a rough estimate on D and R instead of using the real distortion between the original picture and the reconstructed picture, and simplify the entropy coding process. Although these proposed algorithms can reduce the computational complexity dramatically, they are not hardware friendly to some extent, such as [2]. Some techniques, in the second category, are proposed to address the problem from another angle. Because there are so many modes to choose from, and this is the direct cause of computational complexity, these methods target at reducing the candidate modes [5-6]. But the less candidate modes result in significant degradation in encoding performance.

The rest of this paper is organized as follows. In section 2, we first give our proposed RDO-based MD method, and then analyze the complexity of the proposed method. To solve the computational complexity problem, in section 3, we analyze the pipeline throughput of our mode decision in detail and then give our proposed 5-stage pipeline structure. At last, the experimental results and the conclusion of our paper will be presented.

2. OUR PROPOSED MD ALGORITHM

2.1. Proposed MD method

Different from the previous two category mode decision methods, the basic starting point of our mode decision is using the RDO method as much as possible to get high encoding performance.

In AVS-P2, for every intra-luma block, there are 5 different modes; and for every intra-chroma block, 4 different modes in total. The inter modes for P-frame will be

This work was supported by 2009CB320903 and NSFC 60802025

978-1-61284-350-6/11/$26.00 ©2011 IEEE

{Pskip, P16x16, P16x8, P8x16, P8x8, Intra8x8}, while {Bdirect, B16x16, B16x8, B8x16, B8x8, Intra8x8} for B-frame.

For every I-frame intra luma block, we choose the best modes according to RDcost from all 5 modes (vertical, horizontal, DC, diagonal down right and diagonal down left mode); for every intra chroma blocks of I frame we choose the best modes from all four modes (vertical, horizontal, DC, and plane mode), also based on RDcost. So, for I frame, we choose the best modes using Full-RDO method, not doing any simplification just for the purpose of obtaining high coding efficiency.

Different from [6], for P-frame and B-frame we do not reduce any candidate modes to further enhance the coding efficiency. We choose the best MB modes from all modes (Pskip, P16x16, P16x8, P8x16, P8x8, Intra8x8) coming from fractional motion estimation (FME) for P-frame based on RDcost; and for B-frame we choose the best MB modes from Bdirect, B16x16, B16x8, B8x16, Bs8x8 and Intra8x8. For MB-level intra mode of P or B frame we do not use RDO method choose the best block-level modes of all blocks (six blocks in total) in one MB because of the intolerable computational complexity. Instead, we just use sum of absolute difference (SAD) method to choose the best block-level modes of MB-level Intra8x8 mode in P or B frame.

2.2. Complexity analysis of our proposed MD method

Considering the Function (1) bellow,

$$RD\cos t = D + \lambda R \qquad (0)$$

D, which is described as sum of squared differences (SSD) for AVS RDO based mode decision, stands for the distortion between the original picture and the reconstructed picture. λ is a weight parameter. R is the real coding bits for every Block. To get RDcost, we need to obtain the reconstructed pixels and the coding bits, R.

Notice that the Pskip mode of P-frame, we can get the RDcost directly, because on one hand, the reconstructed pixels are identical to the predicted pixels transmitted from FME, and on the other hand, the R equals 0. Besides, based on the analysis before, the computational complexity of our proposed MD method, as shown in Table 1, can be obtained.

Table 1. Computational complexity for I P B frame

PICTURE-TYPE	RDcosts for one MB(6 blocks)
I	4 x 5 + 2 x 4 = 28 (RDcosts)
P	5x6 = 30 (RDcosts)
B	6x6 = 36 (RDcosts)

Actually, in order to get SSD, we need to perform discrete cosine transform (DCT), quantization (Q), inverse quantization (IQ), inverse discrete cosine transform (IDCT), and reconstruction (REC) functions for every block. As the same, we need to perform DCT, Q, IQ, zigzag scan (ZIGZAG), context-based 2D-VLC (C2DVLC) entropy coding for every block to generate R. If all these processing units are connected in the serial order, the whole processing time will be unendurable. In the next section, we will analyze the structure of our MD method based on the pipelining theory and give our proposed efficient pipeline structure.

3. PIPELINE STRUCTURE ANALYSIS

3.1. Throughput analysis of ideal pipeline

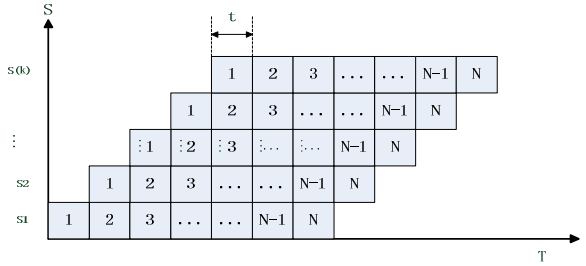


Fig. 1. Generally k-stage pipeline space time-diagram

Take a k-stage pipeline, space time-diagram Fig.1, for example. Assuming the condition is ideal, the k stages are perfectly balanced. The processing time of every stage is the same and assuming the value is "t". If there are N tasks in total to process in the pipeline, we can get the throughput, which is defined as the number of completed tasks per unit time, of the k-stage pipeline. And the throughput can be described as a function in the following.

$$throughput_k = \frac{N}{T_{total}} = \frac{N}{(k+N-1)\times t} \qquad (2)$$

Also, assuming the ideal condition, if the whole processing time of one task through the k-stage processing units is c_0, then we know that

$$t = \frac{c_0}{k} \qquad (3)$$

So, throughput can be described as in the Function (4),

$$throughput_k = \frac{N}{T_{total}} = \frac{N}{(k+N-1)\times t} = \frac{N}{\left(1+\frac{N-1}{k}\right)c_0} \qquad (4)$$

We can see from Function (4) that, if the pipeline design is ideal, the more pipeline stages, the more throughput.

3.2. Restrictions of RDO-based MD pipeline structure

Of course, we can't achieve the ideal pipeline condition in a RDO-based MD. For our mode decision, there are 3 restrictions that must be considered:

- Data dependency
- Bottleneck of the pipeline
- The increase of hardware usage with the increase of number of stages

The first limitation, is that the pipeline will be interrupted once every five modes, as shown in Fig.3, for luma blocks, due to data dependency, as shown in Fig.2. Because b01 needs the reconstructed pixels of b00, the reconstructed pixels at the rightmost column, as shown in Fix.2, to perform the intra prediction (IP) of b01. The same situation also happens to b10, which needs the bottom line of reconstructed b01, and b11, that needs the rightmost column of reconstructed b10.

Another limitation comes from the bottleneck of the pipeline. We can't reduce the processing time of every stage without a limit and we can't balance all the stages perfectly due to the inherent processing dependency and the unbearable control complexity with the increase of number of stages. So, generally, different stage has different time consumption in the pipeline. What's more, because the basic data processing unit is one 8x8 block, the memory usage will increase significantly due to buffer the output of the additional stage along with the increase of number of stages. In fact, we need additional 2x8x8 bytes of memory for every one increased stage at least.

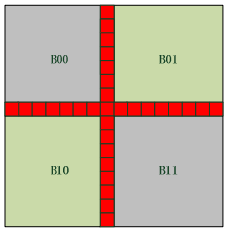


Fig. 2. Data dependency

3.3. Throughput analysis of RDO-based MD pipeline and our proposed RDO-based MD pipeline structure

In Haibing Yin's work [6], it assumes each stage of the block level mode decision pipeline has the same time consumption T, and the whole mode decision pipeline is divided into 6 stages. In fact, different stage has different time consumption due to the restrictions mentioned above. Here, the more detailed analysis will be presented and then our more efficient pipeline strategy will be given.

Different from [6][7], which adopt 4-way Q and IQ, we used 8-way Q and IQ. In order to address the bottleneck of the pipeline, we also use the 4-way zigzag scan [7], and the zigzag scan with the process of determining the C2DVLC table number will cost 22 cycles. The processing time of C2DVLC is not unique since it is related to the number of run-level pairs. Through a large scale statistical analysis, we observed that the processing time consumption is about 24 cycles in our 4-way C2DVLC in average. Due to this, we can get the time consumption of all the processing units, in Table 2, all include 8 cycles of data access time due to caching one 8x8 block pixels.

Table 2. Time consumption of every processing unit

Processing Unit		Time Consumption(cycles)	
DCT-H		18	
DCT-V		18	
Q(8-way)		16	
IQ(8-way)	ZIGZAG-SCAN(4-way)	14	22
IDCTH	C2DVLC(4-way)	18	24
IDCTV	-	18	-
REC && RDCOST && MD		17	

Based on Function (2) and the data dependency described before, also noticing that there are no data dependency for chroma blocks, we divide N into 2 parts, one part is for luma blocks which suffer from the data dependency, the other part is for the chroma blocks which don't. So, N = 4x5(modes) + 2x4(modes) = 28. For the luma part, based on Fig.3, the time consumption can be described as Function (5).

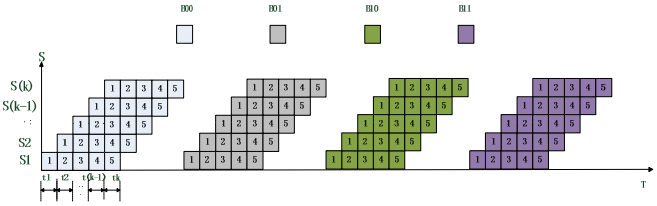


Fig. 3. Pipeline structure of luma blocks

$$T_{luma} = 4\times\left\{\sum_{i=1}^{k} t_i + (5-1)\times\max\{t_1, t_2, \ldots, t_{k-1}, t_k\}\right\} \quad (5)$$
$$= 4\times\left\{\sum_{i=1}^{k} t_i + 4\times\max\{t_1, t_2, \ldots, t_{k-1}, t_k\}\right\}$$

We notice that there are many "holes" in the pipeline, so for the chroma part, we can make use of these "holes". Then, the total processing time can be drawn as Function (6) (To simplify the analysis process, we assume a constant T_{luma} because the pipeline filling of chroma samples will just slightly affect T_{luma}).

$$T_{total} = \begin{cases} T_{luma} & (\text{if } T_{vacancy} \geq T_{chroma}) \\ T_{luma} + T_{chroma} - T_{holes} = 4\times\left\{\sum_{i=1}^{k} t_i + 4\times\max\{t_1, t_2, \ldots, t_{k-1}, t_k\}\right\} + & \\ 8\times\max\{t_1, t_2, \ldots, t_{k-1}, t_k\} - 3\times(k-1)\times\max\{t_1, t_2, \ldots, t_{k-1}, t_k\} & \\ = 4\times\sum_{i=1}^{k} t_i + (27-3k)\times\max\{t_1, t_2, \ldots, t_{k-1}, t_k\} & (other) \end{cases} \quad (6)$$

Besides, $T_{holes} \geq T_{chroma}$ is equivalent to $3(k-1) \geq 8$, so Function (6) can be re-written as Function (7).

$$T_{total}=\begin{cases} 4\times\left\{\sum_{i=1}^{k}t_i+4\times\max\{t_1,t_2,\cdots,t_{k-1},t_k\}\right\}_{(k=4,5,6.....)} \\ 4\times\sum_{i=1}^{k}t_i+(27-3k)\times\max\{t_1,t_2,\cdots,t_{k-1},t_k\}_{(k=1,2,3)} \end{cases} \quad (7)$$

According to Function (2), Function (7) and the basic processing units in Table.2, we can get the throughputs of different pipeline structure with different number of stages through combining or splitting the basic processing units.

Table 3. Throughput of different stages

K	$\sum_{i=1}^{k} t_i$	$\max\{t_1,t_2,\cdots,t_{k-1},t_k\}$	T_{total}	*throughput*
1	91	91	2548	0.0110
2	91	51	1435	0.0195
3	107	37	1094	0.0256
4	105	34	946	0.0296
5	105	24	804	0.0348
6	115	24	844	0.0332
7	133	24	916	0.0306

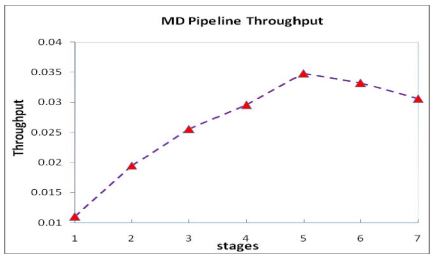


Fig. 4. MD Pipeline Throughput

From Table.3 and Fig.4, one can clear see that, 5-stage MD pipeline structure will obtain the highest throughput when considering the restrictions discussed before. Another observation we can get from Table.3 is that $\Sigma_{i=1}^{k}t_i$ is increasing with the increase of number of stages. It is because of the additional data memory access time when splitting a big processing unit into 2 smaller processing units without inherent data dependency. For example, if we split (DCTV+Q) into DCTV and Q, we need additional 8 data preparation cycles of data access time which is redundant in fact and that leads to the increase of $\Sigma_{i=1}^{k}t_i$, as shown in Fig.5. But this will not happen in the case when there are some inherent data dependency, such as splitting DCT into DCTH and DCTV, as shown in Fig.6, will not make $\Sigma_{i=1}^{k}t_i$ increase significantly. DCTV can't be started unless all the output of DCTH are generated and the output pixels of DCTH are transformed from the rows into columns (H-V transform). So, in this situation the 8 cycles data access time, with which to complete the H-V transform, is not redundant due to the inherent dependency between DCTH and DCTV. Further more, when k is more than 5, due to the bottleneck of processing one 8x8 block, C2DVLC, the throughput begin to decrease, as shown in Fig.4. So, we choose 5-stage pipeline structure for mode decision, and the whole structure of mode decision is given by Fig.7. The corresponding space time-diagram of I, P, B frame are shown in Fig.8, Fig.9, Fig.10 respectively.

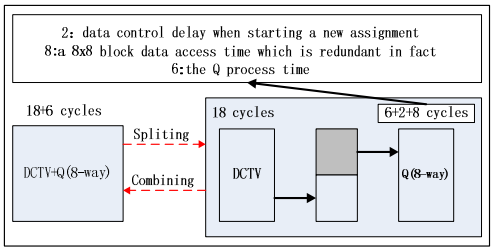


Fig. 5. DCTV-Q Splitting/Combining

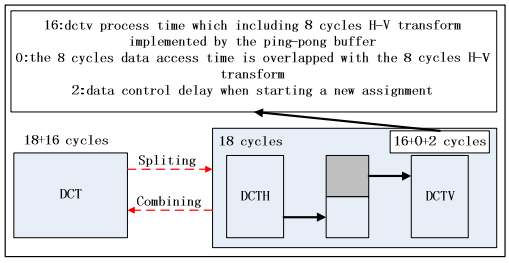


Fig. 6. DCT Splitting/Combining

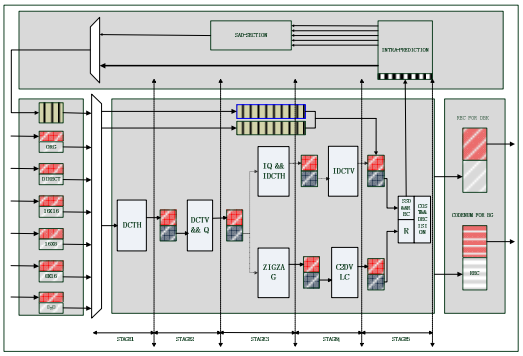


Fig. 7. MD pipeline structure

As Fig.7 shows, the RDO core (middle section) is divided into 5 stages, the first stage is DCTH and the second stage is DCTV-Q. After quantization, the pipeline is divided into 2 branches. IQ-IDCTH and ZIGZAG are both belong to the third stage, while IDCTH and C2DVLC are at the same stage, the fourth stage. At the last stage, the 2 branches of the pipeline again merging into one part, for one part we can calculate SSD with the data from IDCTV and the ORG-PRED data from the outside buffer; for the other part we can get the real coding bits R which have already been generated in the C2DVLC processing unit. Then the RDcost, from which to choose the best mode, can be generated based on the SSD and R. All the buffers between 2 stages are ping-pong buffered inside the pipeline structure.

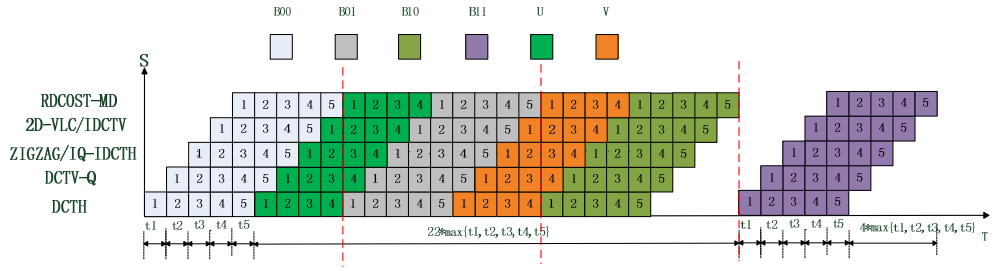


Fig. 8. Pipeline space time-diagram for I-frame

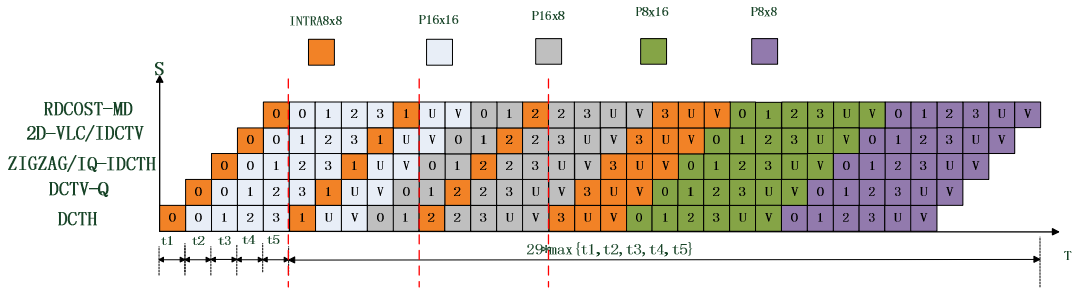


Fig. 9. Pipeline space time-diagram for P-frame

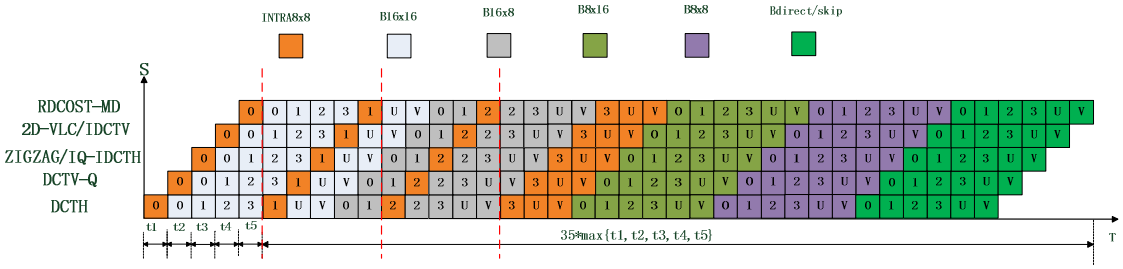
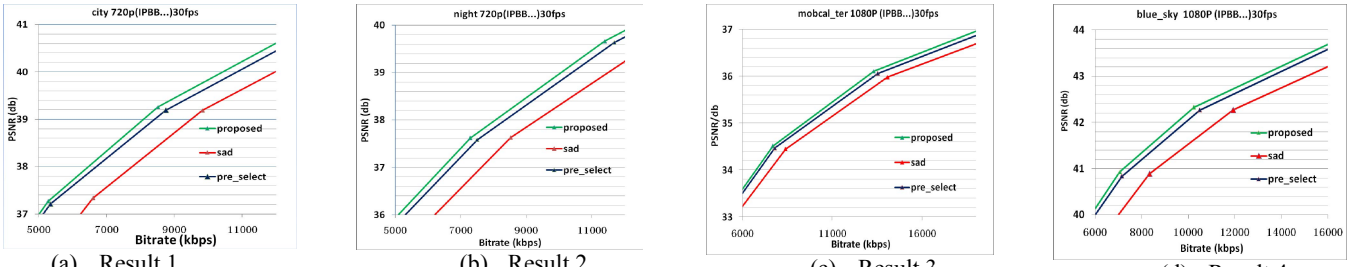


Fig. 10. Pipeline space time-diagram for B-frame

4. EXPERIMENTAL RESULTS

4.1. Encoding performance comparison

To implement our architecture using hardware, we first verified the encoding performance of our encoder. We compare the coding performance between our proposed mode decision algorithm and traditional SAD method based on AVS reference code RM52J. The test results are shown in Fig.11 (a ~d), all the sequences tested are high definition sequences. From the figure we can see that the encoding performance of our proposed method is far higher, about 0.5db rise in PSNR, than the traditional SAD method. We also compared our proposed method with the MD method by reducing the candidate modes, pre-select method like [6], also in Fig.11. From the figure we can clearly see that our proposed RDO-based method outperform the pre-select method, and the increase of PSNR is 0.15~0.2(db) in average.



(a) Result 1 (b) Result 2 (c) Result 3 (d) Result 4

Fig. 11. Encoding performance of different MD methods

4.2. Implementation

The actually implementation result shows that we can accomplish one MB-level mode decision in 864 cycles, 831 cycles, and 975 cycles for I, P, B frame respectively. We implemented our design based on SMIC 0.18-µm CMOS technology, the on-chip memory is 77824 bits and the gate count is 215088 as shown in Table 4. And the final frequency is 234HZ, with which the system can support real time 1080p@30fps .

Table 4. Gate count

Functional module	Gate count
DCTH	8928
DCTV-Q	25560
ZIGZAG	16356
IQ-IDCTH	22188
C2DVLC	35940
IDCTV	12660
RDCOST-MD	11304
OTHERS	82152
TOTAL	215088

Finally, our MD module is integrated to the encoder system, as shown in Fig.12, which is composed of 2 parts: Frame-level pipelining and MB-level pipelining. As the figure shows, firmware takes on the central control of the Frame-level pipelining. The first stage of the Frame-level pipelining is capturing the original uncompressed data. The second stage of the Frame-level pipelining is the encoding core.

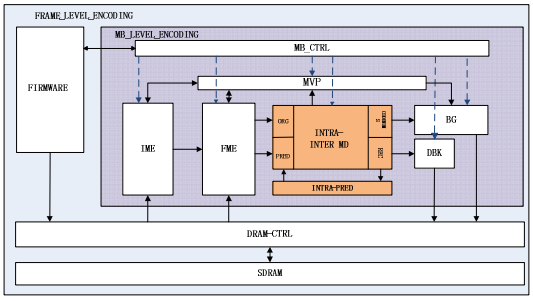


Fig. 12. MB-level pipeline structure

MB_CTRL is for the control of the hardware part, which configures all the MB-level pipelining modules. We adopt 4-stage MB-level pipelining structure, and they are integer motion estimation (IME), FME, MD, bit stream generating unit (BG) and de-blocking effect unit (DBK). Besides, we used Multi-stage Motion Vector Prediction (MVP) Schedule Strategy for AVS HD Encoder, to solve the data dependency problem [8]. Our MD module, which belongs to the fourth stage of the MB-level pipelining, get the original and predicted pixels from FME and IP, and then choose the best mode based on RDcost. After that, MD transmits the codenums, the values of encoding coefficients generated by C2DVLC, and reconstructed pixels of the best mode to BG, which generate bit stream, and DBK, filtering the reconstructed pixels to filter out the block effect, respectively.

5. CONCLUSION

In video system, RDO technique plays an important role in choosing the optimal prediction mode. However, its implementation complexity increases drastically due to the calculation of RDcosts. In this work, we adopted a RDO-based method, which can make more candidate modes to choose based on RDO. The performance of our MD method outperform both the pre-select method and the traditional SAD method. At the same time, we proposed an efficient 5-stage pipeline structure, with which to address the high computational complexity problem, based on the detailed analysis about the pipeline throughput. In the future, we will further our study in MD algorithms without data dependency and try to solve the bottleneck, C2DVLC, of the pipeline.

6. REFERENCES

[1].Wen Gao, et al. AVS Vide Coding Standard. *Studies in Computational Intelligence*, 2010, Volume 280, Intelligent Multimedia Communication: Techniques and Applications, Pages 125-166.

[2].Xin, Z., et al., Novel Statistical Modeling, Analysis and Implementation of Rate-Distortion Estimation for H.264/AVC Coders. *Circuits and Systems for Video Technology*, *IEEE Transactions on*, 2010. 20(5): p. 647-660.

[3].Qiang, W., et al. Low complexity RDO mode decision based on a fast coding-bits estimation model for H.264/AVC. in *Circuits and Systems*, 2005. ISCAS 2005. IEEE International Symposium on. 2005.

[4].Sarwer, M.G. et al. Fast Bit Rate Estimation for Mode Decision of H.264/AVC. *Circuits and Systems for Video Technology*, *IEEE Transactions on,* 2007: p. 1402-1407.

[5].Tianruo, Z., et al. High throughput VLSI architecture of a fast mode decision algorithm for H.264/AVC intra prediction. in Communications, *Circuits and Systems*, 2008. ICCCAS 2008. International Conference on. 2008.

[6].Hai, B.Y., et al. Hardware Friendly Mode Decision Algorithm for High Definition AVS Video Encoder. in *Image and Signal Processing*, 2009. CISP '09. 2nd International Congress on. 2009.

[7].Xiao, H.W., et al. Fast Mode Decision Based on RDO for AVS High Definition Video Encoder .*Lecture Notes in Computer Science*, 2011 Advances in Multimedia Information Processing-PCM 2010, Pages 62-72.

[8].Wei, Y., et al. Multi-stage motion vector prediction schedule strategy for AVS HD encoder. in *Consumer Electronics (ICCE)*, 2010 Digest of Technical Papers International Conference on. 2010.