An Efficient Zigzag Scanning and Entropy Coding Architecture Design*

Tongbing Cui, Chuang Zhu, Yangang Cai, Meng Li,
Huizhu Jia, Don Xie, and Wen Gao

National Engineering Laboratory for Video Technology, Peking University Beijing, China
{tbcui,czhu,ygcai,limeng,hzjia,xdxie,wgao}@jdl.ac.cn

Abstract. Rate distortion optimization (RDO) technique is the best known mode decision method in recent video coding standard, such as H.264 and AVS. However, the unbearable computational burden limits its application. According to the proposed block-level pipeline architecture of RDO-based MD, we find that zigzag scanning and entropy coding are the bottlenecks. In our paper, we firstly analyze the time consumption of the bottlenecks, and then we propose our efficient zigzag scanning and entropy coding architecture. Finally, our enhanced architecture is implemented in AVS encoder. The experimental results show that 20% throughput can be increased compared with the 4-way parallel scanning and entropy coding. With the proposed architecture, the real time RDO-based MD processing of 1080P@30fps can be supported. And our design is realized in high-level Verilog/VHDL hardware description language and implemented under SMIC 0.18μm CMOS technology with 50K logic gates and 6 KB SRAMs at 237MHZ operation frequency.

Keywords: VLSI architecture, RDO, zigzag scan, entropy, AVS.

1 Introduction

The RDO-based MD is adopted in many video coding standards, such as H.261, MPEG-1,2,4, H.264/AVC and AVS, etc. However, dramatic data processing throughput and computation complexity in genuine RDO-based mode decision is a big challenge [1].

To solve the problem of the unbearable computational burden in RDO-based MD, some algorithms, like [2], try to do a rough estimate on distortion (D) and rate (R) instead of using the real ones. The others, like [3], cut down the number of the candidate modes to alleviate the computation burden. The work [4-5], which belong to the second category of the algorithms above, adopted highly efficient pipeline structure to increase the throughput of RDO-based MD, as shown in Fig.1. To get the rate

* This work is partially supported by grants from the Chinese National Natural Science Foundation under contract No.61171139 and No. 61035001, and National High Technology Research and Development Program of China (863 Program) under contract No.2012AA011703.

B. Huet et al. (Eds.): PCM 2013, LNCS 8294, pp. 350–358, 2013.

distortion cost (RDcost) of every candidate mode, we need to perform discrete cosine transform (DCT), quantization (Q), inverse quantization (IQ), inverse discrete cosine transform (IDCT), reconstruction (REC) functions, zigzag scanning (ZIGZAG), and entropy coding (ENTROPY) for every block, in series. Among the processing units above, ZIGZAG and ENTROPY are always the bottlenecks.

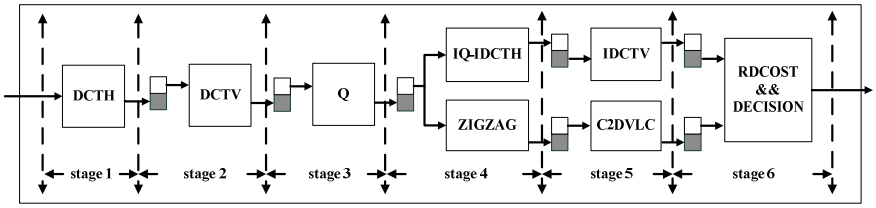

Fig. 1. Block-level pipeline structure

In [6], the 64 coefficients of one 8x8 block are segmented into eight groups along the scanning trace. The scanning process of the proposed parallel scan strategy is divided into 2 pipeline stages, and the total scanning cycles are different with different number of nonzero coefficients. Work [7] integrated the scanning unit into the entropy coding unit, the total scanning time consumption is equal to the total number of coefficients. But with the increase of coefficients, the total processing time of zigzag scanning and entropy coding will be very long. So, how to reduce the processing time when the coefficients are many will be a key issue.

In this paper, we propose an efficient zigzag scanning and entropy coding architecture, which is suitable for the genuine RDO-based MD. And our architecture is verified through AVS high definition video encoder. The rest of this paper is organized as follows. In section 2, we analyze the time consumption of ZIGZAG and ENTROPY in detail. Then the universal parallel ZIGZAG and ENTROPY architecture will be given in section 3. And we will realize our proposed architecture in AVS encoder in section 4. At last, the experiment result and conclusion of our paper will be presented.

2 Time Consumption Analysis

2.1 Time Consumption Analysis of ZIGZAG

For an M×M block in the traditional serial scan strategy, we need one cycle to judge whether each coefficient is zero or nonzero, thus M^2 cycles are totally required for an M×M block. To reduce the time consumption of ZIGZAG, some parallel scan algorithms [5][6] have been proposed. Generally, these algorithms used multiple scan engines to accelerate the scanning process. But how many engines should be used to get the minimum time consumption will be a crucial problem.

To scan the coefficients in parallel manner, we firstly divided M^2 coefficients into several groups. In each group, we use one scan engine to detect the (run, level) pairs in reverse zigzag order. After the detection, the (run, level) pairs are generated from each group. We know that the run of (run, level) pair is the number of zeros preceding

each non-zero coefficient and the continuous zeros may be interrupted at the end of every group, so the first and last (run, level) pair of each group may be wrong because of the interruption, as the red points shown in Fig.2. So, we need to fix those wrong (run, level) pairs of each group. It is clear to see that the more groups we divide, the more (run, level) pairs are to be fixed.

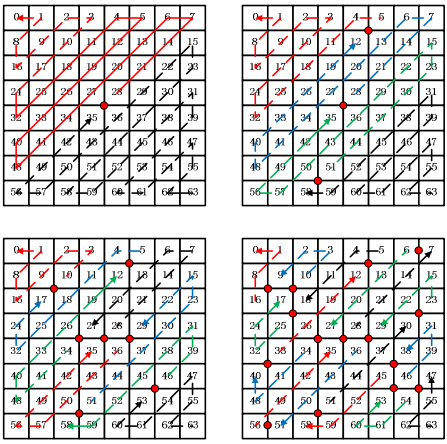

Fig. 2. Scanning diagrams of different segmentation

Assuming the number of groups is N and the block size M×M equals 8×8. Fig.2 shows different scanning diagrams when N equals 2, 4, 8, and 16. The number of the red points represents the number of (run, level) pairs need to be fixed. For N = 2, 4, 8, and 16, the corresponding number of (run, level) pairs need to fixed are 1, 3, 7 and 15.

Let T_{ZIGZAG} the total time consumption of ZIGZAG, $T_{DETECTION}$ the detection time of each group, T_{FIX} the total time of fixing all the possible wrong (run, level) pairs. Then, we can get function (1).

$$T_{ZIGZAG} = T_{DETECTION} + T_{FIX.} \tag{1}$$

In Function (1), $T_{DETECTION}$ is directly proportional to the total number of coefficients in each group. In fact, we need one cycle to detect every coefficient, and the number of groups is N, so, $T_{DETECTION}$ can be described as function (2).

$$T_{DETECTION} = \text{ceil}(M^2 / N). \tag{2}$$

Here, ceil(M^2 / N) returns the smallest integer greater than M^2 / N. In addition, T_{FIX} is proportional to the number of (run, level) pairs which need to be fixed. It is clear that the number of (run, level) pairs need to be fixed is N - 1. Due to the parallel detection, the run of (run, level) pair in the interface between every 2 groups may be interrupted. And, we can not fix all the (run, level) pairs at the same time but fix them one by one in the zigzag order. For every (run, level) pair, we need α cycles to fix it. Then, we can get the following function (3).

$$T_{FIX} = \alpha \times (N - 1). \quad (3)$$

So, function (1) can be re-written as function (4),

$$T_{ZIGZAG} = T_{DETECTION} + T_{FIX} = \text{ceil}(M^2 / N) + \alpha \times (N - 1). \quad (4)$$

2.2 Time Consumption Analysis of Entropy

In order to generate the total bits of one block, we need to look up the codenums of all the (run, level) pairs according to the appropriate variable length coding tables (such as C2DVLC in AVS, CAVLC in H.264) and then generate bits through Exp-Golomb coding. We use pipeline structure to generate the total bits of one coding unit. Therefore, for ENTROPY, the total time consumption is composed of three parts. The first part, T_{SETUP} is the pipeline setup time. The second part, T_{PRO} is the processing time of the (run, level) pairs. Consider the worst case, $T_{PRO} = \text{ceil}(M^2 / N)$. The last part of the time consumption, T_{BIT} comes from the total bits accumulating, and the time consumption will increase with the increase of N. It is because that the number of addend increases with the increase of N (for every group, there is one number of bits, called an addend).

So, we can get the total time consumption of ENTROPY, which is described as function (5).

$$T_{ENTROPY} = T_{SETUP} + \text{ceil}(M^2 / N) + T_{BIT}. \quad (5)$$

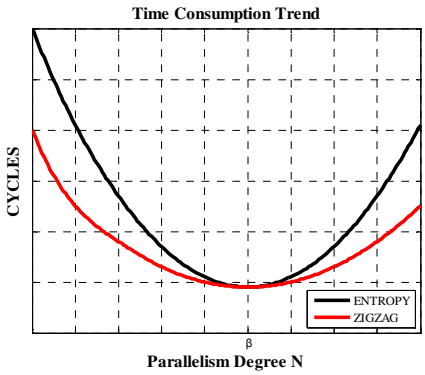

Fig. 3. Time consumption trend of ENTROPY and ZIGZAG

3 The Proposed Universal Architecture

Based on the analysis above, the method of solving the bottleneck of ZIGZAG and ENTROPY is to find the best solution of function(6). In other words, we should find the best degree of parallelism in ZIGZAG and ENTROPY.

$$T_{BEST} = \min\{\max\{ T_{ZIGZAG}, T_{ENTROPY} \}\}$$
$$= \min\{\max\{ \text{ceil}(M^2 / N) + \alpha \times (N - 1), T_{SETUP} + \text{ceil}(M^2 / N) + T_{BIT} \}\}. \quad (6)$$

Assuming the best degree of parallelism is β. Fig.3 shows the time consumption trend of ZIGZAG and ENTROPY. For the ZIGZAG, when N is less than β, due to $T_{DETECTION}$ is the major part of T_{ZIGZAG}, total scan time dropped significantly with the growth of N. But with the increase of N, T_{FIX} gradually becomes the major part of T_{ZIGZAG}, when N is more than β, the scan time begin to increase. It is similar for the ENTROPY, when N is less than β, T_{PRO} is the major part of $T_{ENTROPY}$, and T_{BIT} will be the major part after N is more than β.

We can get the best degree of parallelism β by optimizing (6). So the enhanced parallel ZIGZAG and ENTROPY architecture is proposed, as shown in Fig.4. β detection engines detect the coefficients belonging to the different groups of one block in parallel. After detection, β groups of (run, level) pairs generated by the β detection engines are stored in the (run, level) buffer array. Because the (run, level) pairs from the β detection engines may be wrong, they will be fixed by the fix engine. To fix them, the fix engine examines the boundary (run, level) pairs between two groups. If the level of the last (run, level) pair (in reverse zigzag order) in previous group equals 0, then we modify the run of the first (run, level) pair in current group to run_previous + run_current.

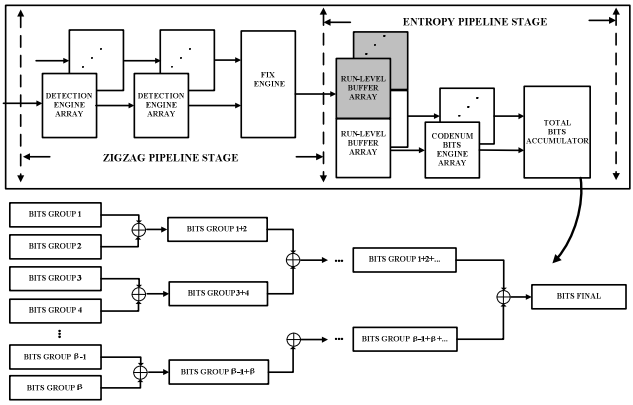

Fig. 4. Parallel ZIGZAG and ENTROPY coding architecture

While at the same time, the (run, level) pairs, which have been fixed in the previous block-level pipeline period, residing in the other buffer array are transferred to codenum engine array of VLC to generate the codenums of the (run, level) pairs. After that, the codenums generated are used to produce bits of every group. At last, all the bits from different groups are added together to form the final bits (R) for an M×M block.

4 Implementation

4.1 MD Pipeline

Our MD module is integrated to an AVS encoder system, as shown in Fig.5, in which MB_CTRL is a central control unit to synchronize all the processing modules in concert and to configure all the MB-level pipelining modules. We adopted a 5-stage MB-level pipelining structure, and they are: fetch unit (FETCH), integer motion estimation (IME), fractional motion estimation (FME), MD, bit stream generating unit (BG) and de-blocking unit (DBK). Besides, we used Multi-stage Motion Vector Prediction (MVP) Schedule Strategy for the AVS HD Encoder, to solve the data dependency problem [8].

The MD module, which belongs to the fourth stage of the MB-level pipelining, gets the original and predicted pixels from FME and IP, and then chooses the best mode based on RDcost. After that, MD transmits the codenums to BG, which are the the values of encoded coefficients generated by C2DVLC, and reconstructed pixels of the best mode to DBK, which filters the reconstructed pixels to filter out the block effect, respectively.

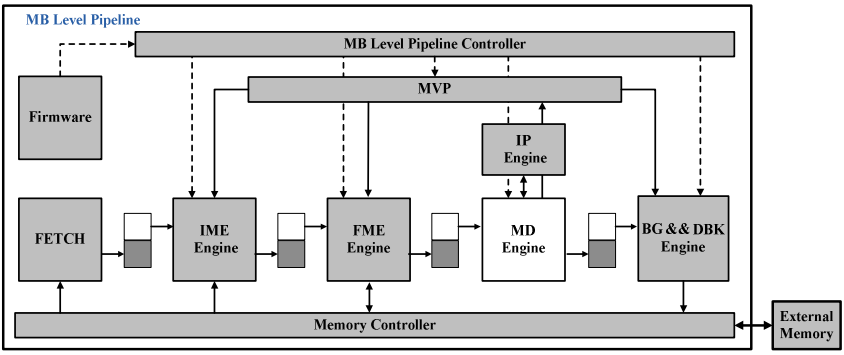

Fig. 5. MB-level pipeline structure

Just like the work [5], we selected block-level 6-stage pipeline to get the RDcost for every mode. As Fig.1 shows, the first stage is DCTH and the second stage is DCTV. The third stage is Q and we adopted a 8-way quantization. After quantization, the pipeline is divided into 2 branches. IQ-IDCTH and ZIGZAG both belong to the fourth stage, while IDCTV and C2DVLC are at the same stage, the fifth stage. At the last stage, the 2 branches of the pipeline again merge into one part, and the RDcost can be then calculated.

4.2 Architecture Realization

In section 3, we have given the universal method to solve the bottleneck of ZIGZAG and ENTROPY. Now, we need to determine related parameters for the special realization in AVS.

For T_{ZIGZAG}, the size of base block is 8 × 8, so M = 8, and we need 2 cycles to fix every (run, level) pair (one cycle to read out the (run, level) pair from buffer, the other to fix it), so α = 2. According to Function (4), we can get function (7).

$$T_{ZIGZAG} = \text{ceil}(64 / N) + 2 \times (N - 1). \tag{7}$$

So, we can get the time consumption of ZIGZAG in different degree of parallelism, as Fig.6 shows.

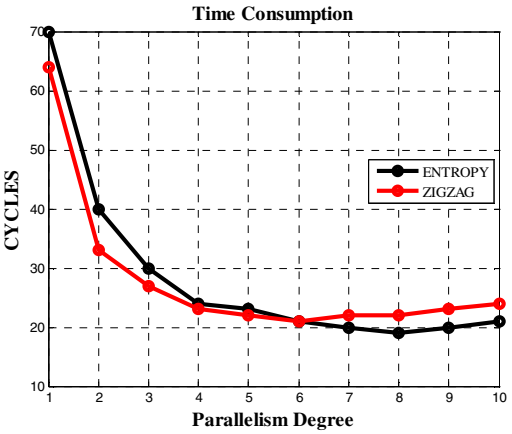

Fig. 6. Time consumption of C2DVLC and ZIGZAG in different degree of parallelism

From Fig.6, we can see that when N equals 5 or 6, the time consumption is the least.

For $T_{ENTROPY}$, we need 6 cycles to setup the pipeline in our implementation, so $T_{SETUP} = 6$. And Table 1 shows the T_{BIT} with different N.

Table 1. Time consumption T_{BIT} with different N

N	1	2	3	4	5	6	7	8	9
T_{BIT}	1	2	3	3	4	4	4	4	5

According to Table 1 and Function (5), we can get function (8).

$$T_{ENTROPY} = 6 + \text{ceil}(M^2 / N) + T_{BIT} \tag{8}$$

So, we can get C2DVLC part in Fig.6, which depicts the time consumption of ENTROPY coding.

From the figure above, it is obvious that the time consumption of ENTROPY coding declines with the increase of N. When N equals 7, the time consumption of ENTROPY will be 20 cycles, which is less than the time consumption, when N is 6. But if we choose N = 7, ZIGZAG will be the bottleneck of the MD pipeline, and the number of cycles is 22. However, we should make good balance of ZIGZAG and ENTROPY. Through careful observation of Fig.6, when N equals 6, the bottleneck of MD pipeline will be the minimum, and the corresponding processing time is 21 cycles. So we adopted 6-way ZIGZAG and ENTROPY coding architecture to get the best performance.

5 Experimental Results

We implemented our proposed 6-way ZIGZAG scanning and ENTROPY coding design based on SMIC 0.18-μm CMOS technology, the on-chip memory is 6K bits and the gate count is 50K. And the working frequency is 237MHZ, with which the system can support real time 1080p@30fps.

Table 2. Time consumption of every processing unit

Processing Unit	Time Consumption(cycles)
DCT-H	18
DCT-V	18
Q(8-way)	20
IQ && IDCTH	21
ZIGZAG(4-way)	22
IDCTV	18
C2DVLC(4-way)	26(MAX)
RDCOST && DECISION	17

Compared with the previous work as shown in Table 2, which adopted 4-way ZIGZAG and ENTROPY coding. We reduced the bottleneck of MD pipeline from 26 cycles to 21 cycles. If we use RDO-based MD, for I frame, there are 28 block-level tasks entering into the MD pipeline (In AVS-P2, for every intra-luma block, there are 5 different modes; and for every intra-chroma block, 4 different modes in total). While for P frame, {P16x16, P16x8, P8x16, P8x8, Intra8x8} will enter into the MD pipeline. And for B frame, {Bdirect, B16x16, B16x8, B8x16, B8x8, Intra8x8}, 6 MB-level modes will enter into the pipeline. Then, we can get the number of cycles saved in I, P and B frame respectively, shown in Table 3.

Table 3. Cycles saved in proposed architecture

PICTURE-TYPE	Saved cycles for one MB(cycles)
I	28×(26-21)=140
P	30×(26-21)=150
B	36×(26-21)=180

6 Conclusion

To solve the bottleneck of RDO-based MD pipeline, we analyzed the time consumption of ZIGZAG and ENTROPY coding in detail respectively. Based on the analysis, we proposed an enhanced parallel ZIGZAG and ENTROPY coding architecture, which can be used to genuine RDO-based MD. The implementation result shows that the proposed architecture can support real time 1080p@30fps encoding when integrated to the encoder system. Compared with the previous work, we reduce the bottleneck of MD pipeline from 26 cycles to 21 cycles, and the number of cycles saved in one MB of I, P and B frame are 140, 150 and 180 respectively when using RDO-based MD.

References

1. Zhang, T., Li, S., Tian, G., Ikenaga, T., Goto, S.: High throughput VLSI architecture of a fast mode decision algorithm for H.264/AVC intra prediction. In: International Conference on Communications, Circuits and Systems, ICCCAS (May 2008)
2. Wang, Q., Zhao, D., Gao, W., Ma, S.: Low complexity RDO mode decision based on a fast coding-bits estimation model for H.264/AVC. In: IEEE International Symposium on Circuits and Systems, ISCAS 2005 (May 2005)
3. Pan, F., Lin, X., Susanto, R., Lim, K.P., Li, Z.G., Feng, G.N., Wu, D.J., Wu, S.: Fast mode decision algorithm for intraprediction in H.264/AVC video coding. IEEE Trans. Circuits Syst. Video Technol. 15(7), 813–822 (2005)
4. Yin, H., Wang, X., Zhu, X., Qi, H.: Hardware Friendly Mode Decision Algorithm for High Definition AVS Video Encoder. In: 2nd International Congress on Image and Signal Processing, CISP 2009 (October 2009)
5. Wang, X., Zhu, C., Yin, H., Gao, W., Xie, X., Jia, H.: Fast Mode Decision Based on RDO for AVS High Definition Video Encoder. In: Qiu, G., Lam, K.M., Kiya, H., Xue, X.-Y., Kuo, C.-C.J., Lew, M.S. (eds.) PCM 2010, Part II. LNCS, vol. 6298, pp. 62–72. Springer, Heidelberg (2010)
6. An, D., Tong, X., Zhu, B., He, Y.: A novel fast DCT coefficient scan architecture. In: Picture Coding Symposium, pp. 1–4 (2009)
7. Huang, Y.-W., Hsieh, B.-Y., Chen, T.-C., Chen, L.-G.: Hardware Architecture Design for H.264/AVC Intra Frame Coder. In: Proceedings of ISCAS 2004, May 23-26, vol. 2, pp. II-269–II-272 (2004)
8. Yang, W., Yin, H., Gao, W., Qi, H., Xie, X.: Multi-stage motion vector prediction schedule strategy for AVS HD encoder. In: 2010 Digest of Technical Papers International Conference on Consumer Electronics, ICCE (January 2010)