FAST SAO ESTIMATION ALGORITHM AND ITS VLSI ARCHITECTURE

Jiayi Zhu, Dajiang Zhou, Shinji Kimura, Satoshi Goto
The Graduate School of Information, Production and Systems, Waseda University

ABSTRACT

SAO estimation is the process of determining SAO parameters in video encoding. There are two difficulties for VLSI implementation of SAO estimation. The first is that there are huge amount of samples to deal with in statistic collection phase. The other is that the complexity of RDO in parameters determination phase is very high.

In this article, a fast SAO estimation algorithm and its corresponding VLSI architecture are proposed. For the first difficulty, we use bitmaps to collect statistic of all the 16 samples in one 4x4 block simultaneously. For the second difficulty, we simplify a series of complicated procedures in HM to balance the complexity and BD-rate performance.

Experimental results show that the proposed algorithm maintains the picture quality improvement. The VLSI design based on this algorithm can be implemented by 156.32K gates, 8832 bits SPRAM, 400MHz @ 65nm technology and is capable of 8Kx4K @ 120fps encoding.

Index Terms— HEVC, SAO, estimation, VLSI

1. INTRODUCTION

HEVC [1] is the new generation video coding standard which reduces 50% bit rates in encoding video sequences with same picture quality compared to its predecessor H.264/AVC. SAO [2] is a new in-loop filtering technique that reduces the distortion between original samples and reconstructed samples in HEVC. SAO estimation is the process of determining SAO parameters in video encoding, which has obvious BD-rate impact on HEVC [2].

Since SAO is a new type of encoding tool in video coding standard, the related research works on SAO estimation are limited. Zhu[3], Park[4] and Mihir[5] did researches on SAO decoding VLSI design. Praveen[6] did research on SAO encoding algorithm but not hardware architecture. No publications on hardware implementations on SAO encoding are found so far and only one software implementation instance is well known. That is HEVC reference model HM (x265 uses the same SAO estimation algorithm).

The SAO estimation algorithm in HM (we use version 12.0) has good BD-rate performance. But it is not easy for VLSI implementation. The SAO estimation algorithm in HM is divided into two phases. The first is statistic collection and the second is parameters determination. In the first phase, the difficulty for VLSI design is that there are so many samples to deal with for statistic collection. The algorithm in HM deals with each sample one by one without considering the throughput performance, which is obviously unacceptable for VLSI implementation. In the second phase, the difficulty for VLSI design is that the RDO frequently used in various SAO parameters determination in HM algorithm is unsuitable for VLSI implementation.

In this article, we propose fast encoding algorithm based on HM algorithm and its VLSI architecture. For the first difficulty, bitmaps are used to collect statistic of 16 samples in one 4x4 block simultaneously and thus the throughput is raised. For the second difficulty, a series of complicated procedures in HM algorithm are simplified to achieve a better balance between BD-rate and complexity.

Experimental results show that the proposed algorithm maintains the picture quality improvement. The VLSI design based on this algorithm can be implemented by 156.3K gates, 8832 bits SPRAM, 400MHz @ 65nm technology and is capable of 8Kx4K @ 120fps encoding.

The rest of this article is organized as follows. Section 2 and section 3 introduces the details of SAO estimation algorithm in HM and our improvements proposals respectively. Section 4 describes the VLSI architecture in details. The experiment results and implementation results are illustrated in Section 5. Finally, section6 concludes this article.

2. ALGORITHMS

The SAO estimation algorithm in HM 12.0 is divided into static collection phase and parameters determination phase.

2.1 Statistic collection

As shown in Figure 1, there are 4 classes for EO (edge offset, EO_0: horizontal; EO_1: vertical; EO_2: diagonal 135; EO_3: diagonal 45) and 4 categories for each of EO class. There are 32 bands for BO (band offset). Refer [2] for details.

16 EO categories and 32 BO bands are collectively called 48 classifications. For each classification, information count (C) and sum (S) shall be collected. C means the number of samples which belong to the specified classification within one CTB. S is the sum of difference between original samples and reconstructed samples which belong to the specified classification within one CTB.

2.2 Parameters determination

There are 4 procedures in parameters determination phases. Among them, procedure 1 is based on classification layer and other 3 procedures are based on CTB layer.

2.2.1. Offset, distortion and cost determination

For each classification, given information C and S, the first procedure of parameters determination is to obtain three parameters offset (O), distortion (D) and cost (CO).

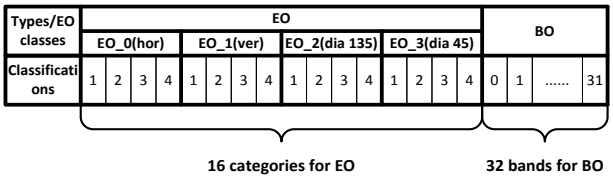

Figure 1 categories for EO and bands for BO

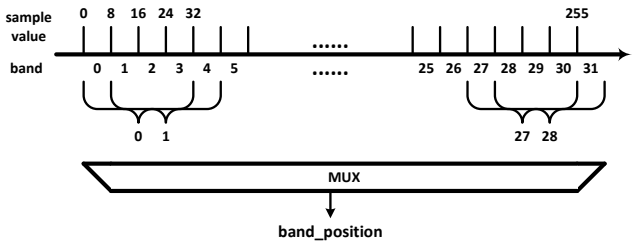

Figure 2 band position determination

RDO iterations are applied in the process to obtain O. All the values between S/C and 0 are iterated as O candidates. For each iterated item, the following formula is used to calculate D and CO. Rate is estimated according to value of iterated item and lambda is constant. The O whose corresponding cost is minimum is selected as the final O for the classification.

$$D=O*O*C-O*S*2$$

$$CO=D+rate*lambda$$

2.2.2. Band_position determination

There are 32 bands for BO. Continuous 4 bands form a band group. The CO of one band group is the sum of COs of 4 bands within that band group. Compare the COs of the 29 band groups, as shown in Figure 2. The band group with minimum CO wins and its first band is labeled as band_position.

2.2.3. Types and EO class determination

As shown in Figure 3, CO of four EO classes, BO and SAO not applied are compared and the minimum one is selected as the type of current CTB. D for each EO class and BO is the sum of Ds of four classifications. D for SAO not applied is 0. Rates of four EO classes, BO and SAO not applied are obtained through CABAC.

2.2.4. Modes (left, upper and no merge) determination

As shown in Figure 4, upper CTB merge, left CTB merge and current CTB parameters are compared and the best one is selected as the mode of current CTB. One very important thing is that the criteria in the comparison of this procedure is a transform of CO, we name it as COT(cost_transformed).

$$COT = COT\_y + COT\_c$$

$$COT\_y = D\_y / L\_y + R\_y$$

$$COT\_c = (D\_cb+D\_cr) / L\_c + R\_c$$

D_y, D_cb and D_cr are distortions of luma, cb and cr component of specified CTB respectively. L_y and L_c are

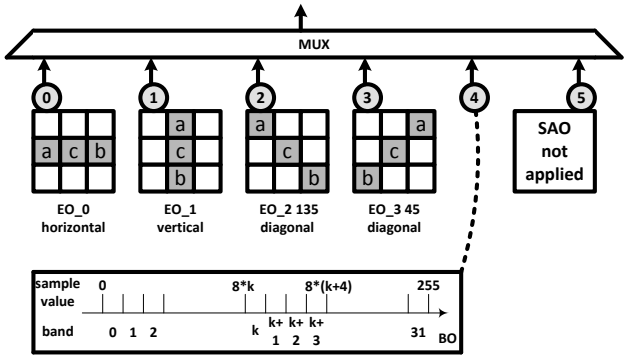

Figure 3 types determination

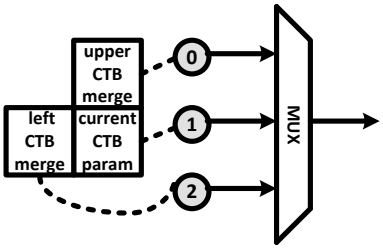

Figure 4 merge or not determination

lambda of luma and chroma. R_y and R_c are rates for luma and chroma, which is obtained through CABAC.

3. PROPOSALS

In statistic collection phase, we propose to use bitmaps to collect statistic of 16 samples in one 4x4 block simultaneously. This is efficient and suitable for hardware implementation. In parameters determination phase, a series of modification are adopted to balance the complexity and compression rate.

3.1 Statistic collection

In our proposal, statistic of one 4x4 block (16 samples) is collected in one round (cycle). So for 64 x 64 CTB, 256 rounds are needed to finish luma statistic collection and 64 rounds are needed to finish cb and cr statistic collection respectively. There are 48 4x4 bitmaps which match 48 classifications mentioned in 2.1. Each bit in the bitmap represents whether the corresponding sample in the 4x4 block belong the particular classification. C and S mentioned in 2.1 are easily collected by means of bitmaps.

An example of how bitmaps are generated is shown in Figure 5. One 4x4 samples block together with its surrounding samples is input as one 6x6 block. 16 EO 4x4 bitmaps are generated through comparing the value of each sample in the 4x4 block with its surrounding samples. Four 4 non-zero BO bitmaps (for this example) are generated by analyzing the value of samples.

An example of how to use bitmap to generate S and C of one 4x4 block is shown in Figure 6.

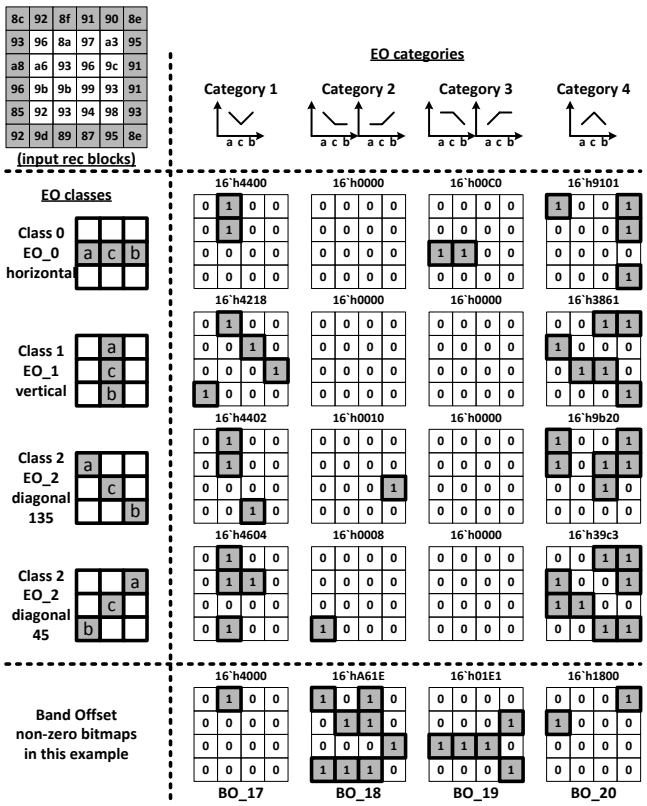

Figure 5 example of bitmap generation in statistic collection

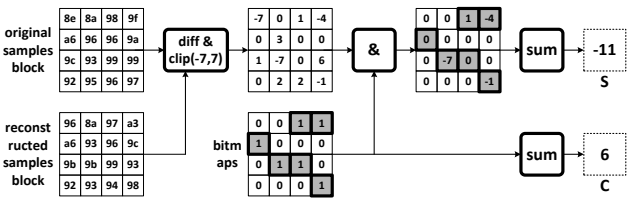

Figure 6 example of generating S and C through bitmaps

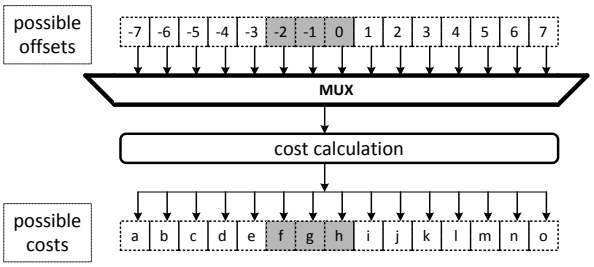

Figure 7 offset iteration

Table 1 Rate Approximation Value

rates name	rates value
sao_not_applied_rate	3
chroma_param_rate	16
luma_param_rate	10
left_merge_rate	1
upper_merge_rate	1

4x4 original samples, 4x4 reconstructed samples and 4x4 bitmaps are input and finally S and C of one 4x4 samples block are generated. S and C of one CTB is sum of Ss and Cs of all 4x4 blocks in that CTB.

3.2 Parameters determination

A series of modifications are proposed to reach a better balance between complexity and compression rate.

3.2.1. Offset, distortion and cost determination

In 2.2.1, there is an iteration process in finding offset. This process is removed. The offset is obtained directly by round(S/C). As shown in Figure 7, suppose S is -11, C is 6 and round(S/C) is -2. In original algorithm, -2, -1, 0 are iterated and RDO is used to evaluate the best offset. In the proposed algorithm, -1 and 0 are not iterated. Result of round(S/C) , which is -2, is directly selected as the offset.

3.2.2. Band_position determination

In 2.2.2, CO is used to determine the band_position. We use D instead of CO as the criteria.

3.2.3. Types and EO class determination

In 2.2.3, CO is used to determine the types and EO classes. We use D to compare BO and 4 EO classes and select the best one. When we compare the selected best one with SAO not applied. We use constants instead of CABAC results to estimate the rate. The constants are shown in Table 1.

3.2.4. Modes (left, upper and no merge) determination

Firstly, we use constants instead of CABAC results to estimate the rates. The constants are shown in Table 1. Secondly, we use the following formula in calculating COT:

$$COT = D\_y + D\_cb + D\_cr + (R\_y + R\_c) * (L\_y + L\_c) / 2$$

4. VLSI ARCHITECTURE

The whole SAO estimation architecture is divided to two modules as shown in Figure 8. Info means S and C for 48 classifications. pRec and pOrg means samples block from reconstructed and original pictures. The statistic collection module costs 256 cycles for luma and 64 cycles for cb and cr respectively. The parameters determination module costs 64 cycles to process each component of each CTB. The pipeline between static collection module and parameters determination module is shown in Figure 9.

4.1 Statistic collection module

The block diagram of statistic collection module is shown in Figure 10. On each cycle, one 4x4 block with its surrounding samples are input to eo_classification and bo_classification and then 16 EO bitmaps and 32 BO bitmaps are generated. The boundary samples of one CTB are not under statistic, which avoids reference samples of neighboring CTB in EO. This is achieved by the mask module. B2n module and diff module implement the function shown in Figure 6, which generate S and C of each 4x4 block. 48 accumulators store the S and C of the 64x64 CTB.

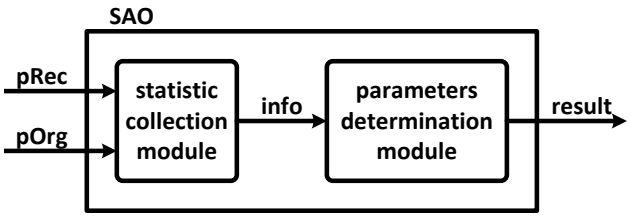

Figure 8 SAO estimation block diagram

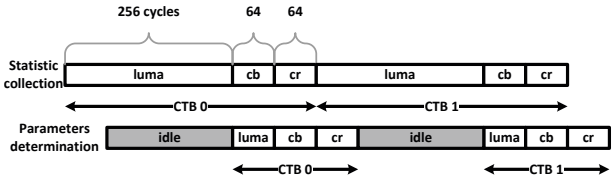

Figure 9 pipeline of SAO estimation

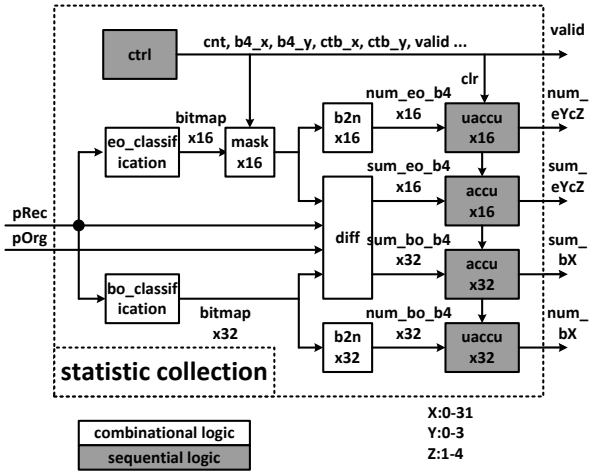

Figure 10 block diagram of statistic module

4.2 Parameters determination module

The block diagram of parameters determination module is as shown in Figure 11. The info in Figure 8 is registered into the dist & offset generation sub-module. On each cycle, one S and C pair is processed and corresponding distortion and offset are generated. Current CTB has 48 pairs to process, the upper and left CTB have 4 pairs to process respectively. Thus it takes around 56 cycles to process the S and C pairs. The cost generation & decision module stores all the distortion of various types and calculates its corresponding cost if necessary. Finally the choice of minimum cost is selected as the result. The result will be stored to left CTB SAO parameters registers and upper CTB SAO parameters SRAM for future usage.

5. EXPERIMENTAL RESULT

The modified SAO estimation algorithm is proved to keep good BD-rate performance by experiments. We set the experimental condition to be HM12.0, low delay P main because SAO effect is most obvious under this condition [2].

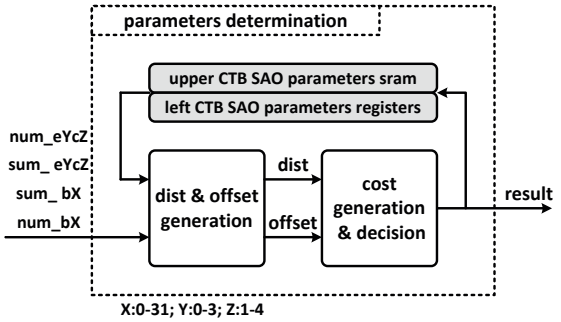

Figure 11 block diagram of estimation module

Table 2 BD rate reduction comparison between HM 12.0 SAO estimation and modified SAO estimation (based on SAO off)

	HM 12.0 SAO estimation			Modified SAO estimation		
	Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)
Cactus	-6.1	-9.1	-8.0	-5.2	-11.5	-11.1
BQMall	-2.5	-2.9	-4.1	-1.8	-4.4	-6.3
RaceHorses	-3.5	-2.9	-3.9	-2.8	-4.7	-5.8
FourPeople	-2.4	-3.1	-2.7	-1.2	-5.1	-5.8
ChinaSpeed	-6.4	-5.8	-7.9	-5.6	-7.1	-11.0

Table 3 Synthesis results

Types	Data
Technology	65nm
Synthesizable Freq	400MHz
Cost	156.3Kgates
SPRAM	8832 bits (for 8Kx4K)
cycles to finish one 64x64 CTB	384cycles
8Kx4K@120 fps frequency	378MHz

As shown in Table 2, the luma BD-rate has some degradation. The chroma BD-rate has been even better than original ones. This means that the original HM12.0 algorithm is not perfect. Firstly, the formulas in 2.2.4 take the sum of luma and chroma cost as the cost of each mode. This may underestimate the importance of chroma. In addition, these formulas use division while our proposed ones use multiplication. This also makes differences. Secondly, the rates obtained from CABAC in the parameters determination phase are not accurate because only partial rather than a complete set of SAO parameters is through CABAC in that phase. The experimental results show that there exists space to raise the BD-rate by improving SAO estimation algorithm. The synthesis results are shown in Table 3. Reasonable hardware cost achieves high performance by the modified algorithm.

6. CONCLUSION

In this article, we propose fast SAO estimation algorithms and its corresponding VLSI architecture. Our proposals effectively solve the huge amount samples and complex RDO difficulties. The modified algorithm still keeps good video BD-rate performance. And it is suitable for high performance VLSI implementation.

7. ACKNOWLEDGEMENT

This research was supported by the regional innovation strategy support program of MEXT.

8. REFERENCES

[1] B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, "High Efficiency Video Coding (HEVC) Text Specification Draft 8"*, document JCTVC-K1003*, Sep., 2012.

[2] C.M., Fu, E., Alshina, A., Alshin, Y.W., Huang, C.Y., Chen, and C.Y. Tsai, C.W., H, S.M. L, J.H., P, and W.J., Han "Sample Adaptive Offset in the HEVC Standard", *IEEE Transactions on Circuits and System for Video Technology, vol., 22,* no. 12, December 2012. Digital Object Identifier: 10.1109/TCSVT.2012.2221529

[3] Zhu Jiayi; Zhou Dajiang; He Gang; Goto Satoshi. "A combined SAO and de-blocking filter architecture for HEVC video decoder" Image Processing (ICIP), 2013 20th IEEE International Conference on

Digital Object Identifier: 10.1109/ICIP.2013.6738405

[4] Seungyong Park; Kwangki Ryoo "*The hardware design of effective SAO for HEVC decoder*" Consumer Electronics (GCCE), 2013 IEEE 2nd Global Conference on

Digital Object Identifier: 10.1109/GCCE.2013.6664837

[5] M. Mihir, N. Niraj, and H. Tamama, "*High throughput VLSI architecture supporting HEVC loop filter for Ultra HDTV*", IEEE Third International conference on ICCEBerlin 2013, Berlin, 2013 Digital Object Identifier: 10.1109/ICCE-Berlin.2013.6698026

[6] Praveen, G.B. ; Adireddy, Ramakrishna "*Analysis and approximation of SAO estimation for CTU-level HEVC encoder*" Visual Communications and Image Processing (VCIP), 2013

Digital Object Identifier: 10.1109/VCIP.2013.6706414