COST FUNCTION OPTIMIZATION AND ITS HARDWARE DESIGN FOR THE SAMPLE ADAPTIVE OFFSET OF HEVC STANDARD

Fabiane Rediess, Ruhan Conceição, Bruno Zatt, Marcelo Porto, Luciano Agostini

Federal University of Pelotas, Group of Architectures and Integrated Circuits, Pelotas, Brazil
{fkrediess, radconceicao, zatt, porto, agostini}@inf.ufpel.edu.br

ABSTRACT

This work presents a cost function optimization for the internal decision of the HEVC Sample Adaptive Offset (SAO) filter. The optimization approach is focused on an efficient hardware design implementation, and explores two critical points. The first one focus in the use of fixed-point data instead of float-point data, and the second focus on reduce the number of full multipliers and divisors. The simulations results show that those proposals do not present significant impact on BD-rate measurements. Based on both these two hardware-friendly optimizations, we propose a hardware design for this cost function module. The FPGA synthesis results show that the proposed architecture achieved 521 MHz, and are able to process UHD 8K@120 fps operating at 47 MHz.

Index Terms— Sample Adaptive Offset, HEVC, Video Coding, Hardware Design

1. INTRODUCTION

During the last years, the study and the improvement of efficient video encoder/decoders becomes more relevant, since the devices that process digital videos must be able to process high-resolution videos in real time, while maintaining a low energy consumption and high compression rate.

The coding process aims to reduce different types of redundancy presented in digital videos. During this process, the subjective quality can be deteriorated, especially through the quantization stage, which may insert artifacts in the video as a collateral effect to the increase in compression rate. Recently, new filters have been proposed intending to increase the subjective visual quality of the encoded videos.

The High Efficiency Video Coding [1] standard (HEVC), proposed in 2013 by the JCT-VC [2] (Joint Collaborative Team – Video Coding), brings a set of two filters called In-Loop Filters, the Deblocking Filter (DF) and the novel Sample Adaptive Offset (SAO). The SAO aims to reduce visible artifacts, like ringing artifacts, considering the local properties of the image such as the edge of objects. It shall be conditionally applied after the completion of the deblocking filter if it reduces the distortion between the original and reconstructed samples.

The SAO is not always applied, it uses a way to decide if it will or will not be applied to each region. This decision is based on cost function. The cost function proposed by the HEVC for SAO considers heuristics to estimate the distortion generated by SAO application and the number of bits to signal the selected SAO to the bitstream. The HEVC reference software uses full multipliers and float point data to estimate the cost. It is important to notice that two critical points are combined, i.e., the multipliers are applied to floating point data, which makes it even more complex to a hardware design.

This work proposes optimizations on the cost function used internally in the SAO filter, as well as the evaluations of the optimization impacts in the HEVC reference software (HM). The focus of the optimizations handles with two critical points for a hardware design: the use of floating point data and the use of full multiplier and divisors.

During the HEVC development multiple solutions for SAO algorithms were proposed [3]-[7]. These proposals, however, did not consider issues related to real-time implementation or hardware design, which is mandatory for high-definition real-time encoding, especially for embedded systems. Since the HEVC was recently released, there are only a few works focusing on hardware solutions for the HEVC SAO. The works [8] and [9] present hardware solutions for HEVC decoder. However, these solutions are not applicable to the encoder side once the decoder does not need to define the classification method and its offsets (see Section 2).

Up to the best of our knowledge, this is the first SAO hardware architecture targeting the internal cost function of the SAO filter in the literature.

2. SAMPLE ADAPTIVE OFFSET

The SAO filter is applied inside the coding loop, right after the DF, and aims to reduce undesirable visible artifacts, specially the ringing artifacts, which become more relevant when using larger transforms and interpolation filters [3].

This work was partially financed by the National Council for Scientific and Technological Development (CNPq) and Research Support Foundation of Rio Grande do Sul (FAPERGS).

The SAO also attempts to reduce the mean distortion between original and reconstructed samples [3].

In order to achieve these objectives, the SAO classifies the samples of a region using one of two SAO types, the Band Offset (BO) or Edge Offset (EO). The regions are aligned to the Coding Tree Unit (CTU) and each CTU can be classified as BO, EO, or neither of them, based on the rate-distortion optimization [4]. If the CTU is not classified as BO or EO, the SAO is not applied to that CTU.

The SAO algorithm consists basically on three steps:

1. Classify the reconstructed samples into different categories from each one of the two SAO types (BO or EO);
2. Obtain the offset related to each category; and
3. Add the obtained offset to each sample of the corresponding category [3].

2.1. Edge Offset (EO)

The EO classification is based on comparisons between current and neighboring samples. Firstly, EO uses four 1-dimensional (1-D) patterns to classify the samples into one out of four EO classes. Figure1 shows the four patterns: horizontal (EO class 0), vertical (EO class 1), 135º diagonal (EO class 2) and 45º diagonal (EO class 3) [6]. Only one EO class can be selected for each CTU that enables EO.

For each EO class, i.e. each pattern in Figure 1, the current sample 'c' is compared with two neighbors (shady samples) in order to classify the samples in one of the five categories according to the rules presented in the Table 1. For example the condition on category 1 means that the sample 'c' is smaller than its two neighbors. Similarly, in category 4, 'c' is greater than its two neighbors.

The categories 1 and 4 are associated with local valleys and peaks, respectively. Categories 2 and 3 are associated with concave and convex corners, respectively [3]. If the sample belongs to the category 0, the SAO is not applied.

The EO with unrestricted offset values can introduce some visual artifacts, which usually looks like salt and pepper noise, or it can appears as a line of a false contour [7]. In order to avoid this kind of artifacts, some restrictions on offset values, proposed on [7], were adopted by the HEVC standard. The offset values for categories 1 and 2 were restricted to positive values, while for categories 3 and 4 the values were restricted to a negative range. This restriction also reduces the number of bits to encode the offsets, since there is no need to encode the sign of the offset, it can be inferred by the category. Note, the chosen classification method, the best EO class and the offset are signaled to the coded video bitstream.

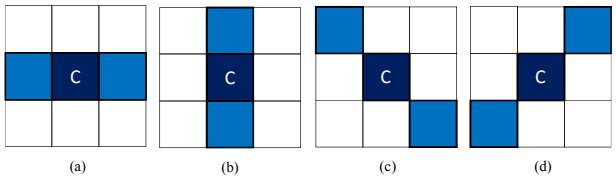

Fig. 1. Edge offset classification patterns.

Category	Condition	Meaning
1	c < 2 neighbors	local min
2	c < 1 neighbor && c == 1 neighbor	edge
3	c > 1 neighbor && c == 1 neighbor	edge
4	c > 2 neighbors	local max
0	None of above	flat area

Table 1. EO category classification conditions.

For each EO class (pattern) and category, the initial offset is calculated as the average difference between original e and reconstructed samples in the category.

2.2. Band Offset (BO)

The Band Offset classification is based on the samples intensity. In this method, all the samples in a region are classified into 32 bands, according to the sample range. The sample range is equally distributed between the 32 bands, so, for 8-bit samples (range 0..255), the samples are equally distributed among the 32 bands of 8 values. The first band contains the samples with values between 0 and 7, the second contains values from 8 to 15, and so on. The most significant bits of the sample can be used to proceed this classification, which represent low computational cost, especially for hardware design [3].

For each band, the initial offset is calculated as the average difference between original and reconstructed samples. Only the offsets of four consecutive bands will be signaled to the bitstream, and, unlike the EO, there is no restriction to the offset values or signals. Figure 2 represents the 32 bands of the BO classification method, indicating four selected bands. The main reason for using only four, and consecutive bands, lies in the fact that flat areas where banding artifacts could appear, most sample amplitudes in a region tend to be concentrated in only few bands [6]. Another reason is to reduce the number of offsets to be transmitted from 16 to 4. The four consecutive bands with the best result in rate-distortion will be chosen. Furthermore, besides the chosen classification method, the initial band position also needs to be signaled to the bitstream.

2.3. SAO Internal Decision

In both SAO types, EO and BO, and for all categories, the initial offset is calculated as an average of the difference between original and reconstructed samples from that category. This calculation implies the use of a full divisor to

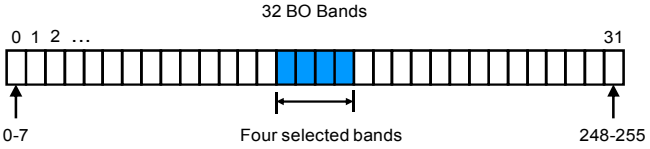

Fig. 2. Band offset classification

divide the accumulated difference by the number of occurrences in the category.

In the Figure 3 there is the pseudo-algorithm that represents the SAO internal cost function. After the classification method and the initial offset generation for each SAO type and category, the filter clips the initial offset between -7 and 7. Next, the clipped offset will be incremented or decremented until it reaches zero value. The cost of all these iterated offsets will be evaluated and the best cost result will be chosen as the offset of the SAO type and category.

The cost calculation used internally in SAO filter applies heuristics to estimate the distortion and the number of bits required for the target offset and category.

The heuristics involve two steps. The first one is estimating the generated distortion for each SAO type and category, as shown in (1), where '*n*' corresponds to the number of occurrences in the category, '*a*' to the accumulated difference and '*b*' to the average difference between original and reconstructed samples, i.e., the initial offset.

$$d = (nb^2) - (2ab) \qquad (1)$$

The second step of the heuristics is the estimation of the number of bits required to write the information for the target SAO type and category. This estimation considers the actual offset and if it is BO SAO type, the estimation increased by 1. This is due, as explained in Section 2, the BO need to transmit the signal of the offset unlike the EO.

This step can be represented by (2), where '*c*' is resulting cost, '*d*' is the distortion calculated in (1), λ is the Lagrange multiplier and '*r*' is the estimated number of bits.

$$\begin{aligned} c &= d + r.\lambda, && \text{if SAO\_type = EO} \\ c &= d + (r+1).\lambda, && \text{if SAO\_type = BO} \end{aligned} \qquad (2)$$

3. PROPOSED OPTIMIZATIONS

The focus of the optimizations proposed on this work are in two critical points for a hardware design, which are the use of floating point data and the use of full multiplier and divisors. It is important to notice that, in this context, the two critical points are combined, i.e., the multiplier are applied to floating point data, which makes it even more complex to a hardware design.

Focusing a hardware implementation of the equations (1) and (2) with high throughput, all the loop iterations must be generated in one clock cycle. Alternatively, a pipeline structure where one iteration of the offset loop is represented by a single or by a group of stages could be used. However, in order to maintain a high throughput the pipeline structure would be multiplied by 7, i.e. the maximum number of offset iterations. Otherwise, if the structure is not multiplied, the performance will be divided by 7, because the cost of only one offset would be generated at a time.

```
For each SAO type
  For each SAO category
    If there is samples in the type-category
      iniOffset = accumDif / Count
      Clip (ini_offset, -7, 7)
      While iniOffset != 0
        //Cost calculation of iniOffset
        //Store best cost and best offset
        //Define next iteration
```

Fig. 3. Pseudo-algorithm for the internal SAO cost function.

The SAO algorithm in the HEVC reference software considers floating point data. The use of floating point data in hardware leads to high hardware consumption and decreased performance.

Thus, this work proposed two types of optimizations in the HEVC SAO algorithm for the internal cost function, which are:

1. The first one aims to use integers or fixed point numbers instead of floating point. In order to evaluate the impact of this proposal, some experiments in the reference software were done. The experiments considered integer data and fixed point data with precision varying from 1 to 8 bits.
2. The second one consists on process all the 7 offset iterations at only one step. With this, each offset was applied to the cost equation. This definition allows to turn the full multiplications into sums and displacements, which is much more hardware-friendly.

This last optimization also allows the elimination of the full divider that was necessary at the end of each statistical collection step for each SAO type and category. This division offers the initial offset, which will be iterated until 0. Once the cost of all possible offsets will be calculated, there is no need to know the first one and so, there is no need of a full divider.

Replacing the fixed offsets, for the full loop optimization, in (1) we have as result the fourteen equations in (3). These equations in (3) can be factored aiming to transform the multiplications by constants in sums and displacements, as shown in equations in (4). This manipulation extremely reduce the complexity of a hardware implementation since displacement can be done with only wire connections and the sum is much more simpler than full multiplier.

$$\begin{aligned} d_1 &= n - 2a & d_{-1} &= n + 2a \\ d_2 &= 4n - 4a & d_{-2} &= 4n + 4a \\ d_3 &= 9n - 6a & d_{-3} &= 9n + 6a \\ d_4 &= 16n - 8a & d_{-4} &= 16n + 8a \\ d_5 &= 25n - 10a & d_{-5} &= 25n + 10a \\ d_6 &= 36n - 12a & d_{-6} &= 36n + 12a \\ d_7 &= 49n - 14a & d_{-7} &= 49n + 14a \end{aligned} \qquad (3)$$

$$\begin{aligned} d_1 &= n - 2a & d_{-1} &= n + 2a \\ d_2 &= 4\,(n - a) & d_{-2} &= 4\,(n + a) \\ d_3 &= 2\,[2\,(2n - a) - a] + n & d_{-3} &= 2\,[2\,(2n - a) + a] + n \\ d_4 &= 8\,(2n - a) & d_{-4} &= 8\,(2n + a) \\ d_5 &= 8\,(2n + n - a) + n - 2a & d_{-5} &= 8\,(2n + n + a) + n + 2a \end{aligned}$$

$d_6 = 8\,(4n - a) + 4\,(n - a)$ $\quad d_{-6} = 8\,(4n + a) + 4\,(n + a)$
$d_7 = 16\,(2n + n - a) + n + 2a$ $\quad d_{-7} = 16\,(2n + n - a) + n - 2a$ (4)

For the second step of the cost calculation, the cost itself, represented by (2), the replacing of the initial offset '*b*' by the fixed offsets of the full loop proposal and proceeding the factoring, we have the equations in (5).

$$
\begin{aligned}
c_{1\,or\,-1} &= d_{1\,or\,-1} + 2\lambda, && \text{if EO}\\
&\quad d_{1\,or\,-1} + 2 + \lambda, && \text{if BO}\\
c_{2\,or\,-2} &= d_{2\,or\,-2} + 2 + \lambda, && \text{if EO}\\
&\quad d_{2\,or\,-2} + 4\lambda, && \text{if BO}\\
c_{3\,or\,-3} &= d_{3\,or\,-3} + 4\lambda, && \text{if EO}\\
&\quad d_{3\,or\,-3} + 4\lambda + \lambda, && \text{if BO}\\
c_{4\,or\,-4} &= d_{4\,or\,-4} + 4\lambda + \lambda, && \text{if EO}\\
&\quad d_{4\,or\,-4} + 4\lambda + 2\lambda, && \text{if BO}\\
c_{5\,or\,-5} &= d_{5\,or\,-5} + 4\lambda + 2\lambda, && \text{if EO}\\
&\quad d_{5\,or\,-5} + 8\lambda - 2\lambda, && \text{if BO}\\
c_{6\,or\,-6} &= d_{6\,or\,-6} + 8\lambda - 2\lambda, && \text{if EO}\\
&\quad d_{6\,or\,-6} + 8\lambda, && \text{if BO}\\
c_{7\,or\,-7} &= d_{7\,or\,-7} + 8\lambda - 2\lambda, && \text{if EO}\\
&\quad d_{7\,or\,-7} + 8\lambda, && \text{if BO}
\end{aligned}
\qquad (5)
$$

4. SOFTWARE EVALUATION RESULTS

The Table 2 shows the average results in BD-rate of the evaluation of using integer data and fixed-point data varying the bit precision from 1 to 8 bits. The experiment was executed using the reference software HM 10 [10] for the first 16 frames of 11 video sequences from the Common Test Conditions (CTCs) [11]. For each sequence, the experiment considered the configurations (Intra only, random access and low delay) and the QP (Quantization Parameter) of 22, 27, 32 and 37.

These results show that the use of integer data instead of floating point has an increase of 0.028 in BD-rate, which means that with this proposal in order to maintain the same quality, it will result in an increase of only 0.028% in bitrate. Considering the evaluated fixed point precision, specially from 3-bit precision, the large majority of sequences and configurations present a difference less than 0.01. However, the average gain of 0.004 is due to the introduced precision error resulted in a little bitrate reduction in few sequences/configurations.

	Y BD-rate			
	Intra Only	**Random Access**	**Low Delay**	**Average**
Integer	-0.005	0.103	-0.014	0.028
1 bit	0.000	0.000	-0.006	-0.002
2 bits	0.000	-0.020	-0.008	-0.009
3 bits	0.000	-0.005	-0.008	-0.004
4 bits	0.000	-0.005	-0.007	-0.004
8 bits	0.000	0.000	-0.006	-0.002

Table 2. Simulation results for integer and fixed point data.

	Y BD-rate			
	Intra Only	**Random Access**	**Low Delay**	**Average**
HM	0.0000	0.0404	4.1215	1.3873
Integer	-0,0421	-0.0578	0.0542	-0.0153
1 bit	0.0000	0.0000	0.0000	0.0000
2 bits	0.0000	0.0000	0.0000	0.0000
3 bits	0.0000	0.0000	0.0000	0.0000

Table 3. Simulation results for the full loop proposal.

The second proposed optimization consists on process the 7 possible offsets instead of starting from the calculated one. The use of this full loop results, as showed in equations (4) and (5) in the elimination of all full multipliers. This optimization was also evaluated with modifications in the reference software. The Table 3 shows the simulations results comparing the version with full loop with the corresponding version starting with the initial offset calculated. That comparison is between the regular HM execution and a HM which tests all the possible offsets. In a similar way, the integer experiment represents the comparison between the software version that uses integer data with a version which uses integer data and tests all the possible offsets.

This simulation was done to less video sequences because its behavior is much more linear. Considering the HM comparison, all the sequences achieved the same or better results. This occurs because the proposed adaptation test more offset than regular HM. So, the proposed full loop is equivalent or better than the regular HM behavioral.

For the integer comparison, the results for the full loop present variations. This may be because of the truncation in the division to get the initial offset and in the cost calculation. For the next results, from the 1-bit precision, the behavioral of the full loop version and the version that starts at the initial offset are the same.

With these results, the decision for the hardware design was to use the full loop and a fixed point precision of 3-bits, where the average difference was low, presenting better results than regular HM. However, from this point, the most of video sequences and its configurations presents differences lower than 0.01 comparing to regular HM.

5. DESIGNED ARCHITECTURE

The high-level vision of the proposed architecture is shown in Figure 4, where are represented that with the accumulated differences and the count of occurrences in the category, the

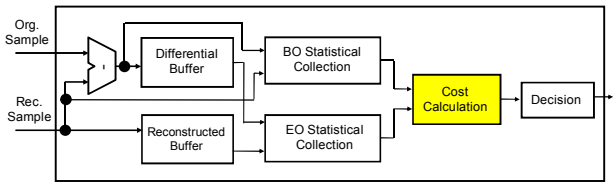

Fig. 4. Top-level SAO architecture

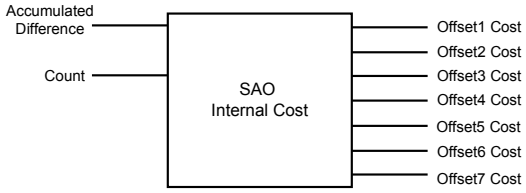

Fig. 5. Block diagram of proposed architecture.

architecture will give the cost for all possible offsets.

The Figure 4 shows a top-level view of the entire SAO architecture, where there are the buffers, the statistical collection step, the cost function and finally the decision mode. The decision mode, in the context of this work, consists on a comparator which selects the lower value.

The Figure 5 shows the block diagram for the proposed architecture. The outputs represent the cost for all possible offsets, from 'c_1' to 'c_7' or from 'c_{-1}' to 'c_{-7}'. Each output implements the corresponding equation in (5). For example, the 'Offset1 Cost' corresponds to the 'c_1' or 'c_{-1}' in equations (5). In total, there are 14 equations, two for each output, selected based on the signal of the 'Accumulated Difference'.

5.1 Synthesis Results

The proposed architecture was described in VHDL and synthesized for the Stratix V 5SGXEA7H3F35C3 FPGA device from Altera using the Quartus II software.

The Table 4 presents the synthesis results. The proposed architecture used few hardware resources from the device, 796 ALUTs and 158 registers. And achieved a high operation frequency, 521MHz.

The proposed architecture need 48 clock cycles to calculate the cost of all SAO types and categories, 32 BO bands and 16 EO types/categories. Then, it was possible to estimate the processing rate of the proposed architecture. Considering the high frequency achieved and that only 48 clock are needed to generate all the cost of and CTU (4,096 samples), the proposed architecture achieved a high throughput, 1,330 UHD 8K frames per second.

Due to this high throughput, the proposed architecture may also be used in lower frequency, in order to reduce the power consumption. To process UHD 8K videos at 120 fps, the propose architecture can operate at 47 MHz.

Up to the best of our knowledge, there is only two works in the literature that propose hardware implementation of the HEVC SAO in [8] and [9]. However, these works focus on the decoder side and thus the do not lead with the decision mode or cost functions, because the selected offsets are signaled to the bitstream.

	Developed Architecture
ALUTs	796
Registers	158
Frequency (MHz)	521
QFHD (3840x2160) (fps)	5,321
UHD 8K (7680x4320) (fps)	1,330

Table 4. Synthesis results and estimated processing rates.

6. CONCLUSIONS

This work presented cost function optimizations of the internal decision for the Sample Adaptive Offset filter of the HEVC standard. This internal decision is responsible to define which offset will be set for each SAO type and category. The target of these optimizations is a hardware implementation of the SAO. The loop optimization resulted in the elimination of 1 full divider and 2 full multipliers, both hardware and time consuming. Another optimization was to use fixed-point data instead of float-point data, which is also hardware and time consuming. Those optimizations present almost insignificant impacts on BD-rate, lower than 0.01 for most of 11 video sequences from the CCTs. Furthermore, the average results present a gain of 0.004.

The optimizations allowed the development of an efficient hardware design with reduced hardware consumption and achieving high processing rate. The proposed architecture achieved operation frequency of 521 MHz and with this it is able to process 1,330 UHD 8K fps and it is able to process UHD 8K@120 fps at only 47MHz.

REFERENCES

[1] JCT-VC Editors, Recommendation ITU-T H.265 - High Efficiency Video Coding (ITU-T Rec.H.265), Apr 2013.

[2] G. Sullivan and T. Wiegand, "Draft Requirements for next-generation video coding project", document VCEG-AL96, Jul, 2009.

[3] C.-M. Fu et al., "Sample adaptive offset in the HEVC standard",*IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1755–1764, Dec. 2012.

[4] C.-M. Fu, C. Chen, Y. Huang and S. Lei. "Sample Adaptive Offset for HEVC". IEEE 13th International Workshop on Multimedia Signal Processing (MMSP), pp. 1-5, Oct, 2011.

[5] K. McCann, W.-J. Han, and I.-K. Kim, "Samsung's Response to the Call for Proposals on Video Compression Technology",document JCTVC-A124, Apr. 2010.

[6] G. Sullivan, J.-R. Ohm, W.-J. Han and T. Wiegand. "Overview of the High Efficiency Video Coding (HEVC) Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649-1668, Dec, 2012.

[7] W.-S. Kim and D. Kwon. "Improved Sample Adaptive Offset for HEVC", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May, 2013.

[8] J. Zhu, D. Zhou, G. He, S. Goto. "A Combined SAO and Deblocking Filter Architecture for HEVC Video Decoder", IEEE International Conference on Image Processing (ICIP), 2013.

[9] S. Park, K. Ryoo. "The hardware design of effective SAO for HEVC decoder", IEEE Global Conference on Consumer Electronics (GCCE), pp. 303-304, Oct, 2013.

[10] I. Kim et al. "High Efficiency Video Coding (HEVC) Test Model 10 (HM10) Encoder Description", document JCTVC-L1002, Jan. 2013.

[11] F. Bossen. "Common test conditions and software reference configurations", document JCTVC-L1100, Jan. 2013.