PROGRAMMABLE LOW POWER IMPLEMENTATION OF THE HEVC ADAPTIVE LOOP FILTER

Ilkka Hautala, Jani Boutellier and Jari Hannuksela

Center for Machine Vision Research
Department of Computer Science and Engineering
FI-90014 University of Oulu

ABSTRACT

The Adaptive Loop Filter (ALF) is a subjective and objective image quality improving filter in the High Efficiency Video Coding standard (HEVC). The ALF has shown to be computationally complex and its complexity has been reduced during the HEVC development process. In the HEVC Test Model HM-7.0 ALF is a 9×7 cross + 3×3 square shaped filter.

This paper presents a programmable application specific instruction processor for the ALF. The proposed processor processes 1920×1080p luminance frames at 30 frames per second, when operated at a clock frequency of 311 MHz. Low power consumption and a low gate count make the proposed processor suitable for embedded devices. The processor program code is written in pure C-language, which allows versatile use of the circuit and updates to the filter functionality without modifying the processor design. To the authors' best knowledge this is the first programmable solution for ALF on embedded devices.

Index Terms— Coprocessors, Adaptive filters, Video signal processing

1. INTRODUCTION

The High Efficiency Video Coding standard (HEVC, H.265) is the latest video project of the ITU-T VCEG and ISO/IEC MPEG standardization organizations [1]. The first edition of the standard was released in January 2013. HEVC is the successor of H.264/MPEG-4 AVC and therefore HEVC has been designed to support the same applications as H.264/MPEG-4 AVC. However, the coding efficiency of HEVC can be even 50% better than that of its predecessor [2]. Better coding efficiency reduces network bandwidth requirements and allows the use of higher resolutions. HEVC also offers better tools for parallel processing [3].

In HEVC Working Draft (WD) 7, the Adaptive Loop Filter (ALF) is a filter which improves subjective and objective video quality. Although ALF improves coding gain significantly, especially at high resolutions [4], it has been left out of the HEVC WD 10 Final Draft International Standard (FDIS). In HEVC WD 7 ALF is the last stage of in-loop filtering followed by the de-blocking filter (DF) and Sample Adaptive Offset filter (SAO). ALF is very computation-intensive; according to [5] ALF alone consumes 28% of the HEVC decoder computation time on a single-core processor. Thus, speeding up the ALF computation has a great effect on the total HEVC performance.

During the last years the popularity of application-specific integrated circuit (ASIC) designs has decreased significantly. ASICs offer high performance and low power consumption, and have thus been suitable for embedded video processing devices. However, the increasing complexity of applications makes ASIC design expensive, complex, resource-intensive and a time-consuming job. In rapidly developing fields, a freshly released product can already be outdated due to the long design and fabrication process.

On the other hand, embedded processors offer the benefits of fast software based design and allow placing a wide range of applications in the same circuit. The performance of embedded processors has increased and they have started to look to be an increasingly attractive alternative to ASICs. The disadvantage of embedded processors is that they can not compete with ASICs in power efficiency and performance.

In this paper, we propose a programmable application specific instruction processor (ASIP) for HEVC ALF. The proposed ASIP is able to process 30 luminance frames a second for 1920×1080p resolution at the reasonable clock frequency of 311 MHz. The processor program code is written using pure C-language, which allows versatile use of the circuit and enables software updates. Based on the literature, the proposed design is the first embedded, programmable solution for HEVC ALF.

This paper is organized as follows: Section 2 explains the theory of Adaptive Loop Filtering, reviews related work and describes the processor architecture used. After that the proposed processor architecture is presented in Section 3. Section 4 presents the results of experiments. Finally, we discuss our achievements in Sections 5 and 6.

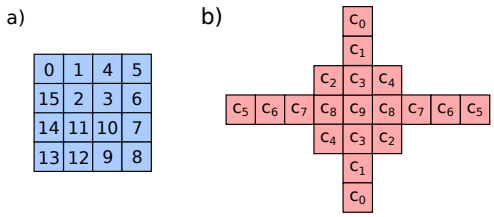

Fig. 1. **a)** A predefined RA pixel classification table and **b)** the 9×7 cross, 3×3 square filter shape in the HEVC test model version HM-7.0.

2. BACKGROUND

2.1. HEVC video structure

In HEVC, a Coding Tree Unit (CTU) corresponds to the macroblock structure of H.264. The CTU contains a luma coding tree block (CTB), two or more chroma CTBs and syntax elements. For luma, the size of CTBs is 16×16, 32×32 or 64×64 and can be selected by the encoder. Better compression is usually achieved by using larger CTB sizes. Using quadtree syntax, a CTU can be divided into smaller blocks called Coding Units (CU). A CU may have the same size as a CTB if there is only one CU in the CTU. Each CU is split into prediction units (PU) of size 64×64 down to 4×4.

To offer better support for parallel processing architectures HEVC defines tiles and wavefront parallel processing (WPP) in addition to traditional slices. When slices consist of a variable number of sequential CTUs in the raster scan order, tiles are rectangular regions of the picture formed by CTUs. The tile is an independently-decodable and self-contained unit. In WPP, a frame is divided into slices which contain a CTU row. Thus CTU rows can be processed in parallel.

2.2. Adaptive loop filtering

In Test Model HM-7.0 of the HEVC standard, the Adaptive Loop Filter encoding algorithm includes three main stages [6]. The first stage is filter coefficient derivation. To solve the filter coefficients, the encoder classifies reconstructed pixels into different classes and uses Wiener-Hopf equations with minimized MSE between the original frame and the coded frame [7]. As a result of filter coefficient derivation, specific Wiener coefficients are generated for pixels in different classes.

The encoder can use two methods for pixel classification, block-based (BA) and region-based (RA) methods. In BA, local direction and textural characteristics are calculated on 4×4 blocks. In RA, pixel classification is based on the location of the pixel in the picture. Figure 1a) shows how a picture is divided in the RA mode. At its maximum, 16 different filter sets can be assigned to the luminance component of the picture and only one to the chrominance components. The RA mode is only supported in HM-7.0 due to its simplicity compared to the BA mode.

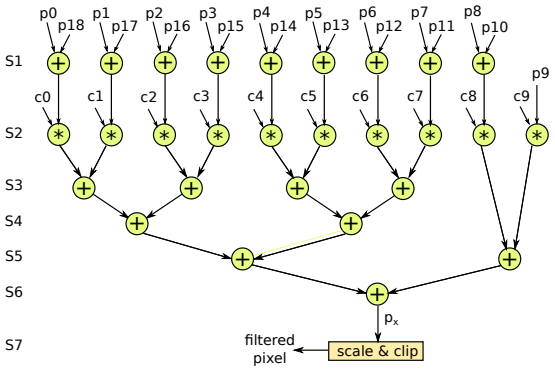

Fig. 2. The pixel filtering flow.

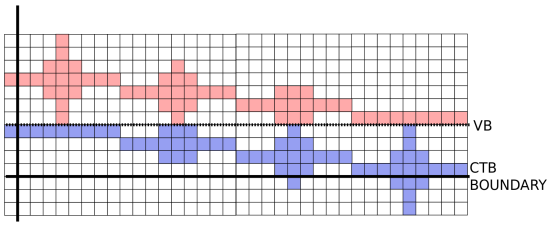

Fig. 3. Virtual boundary processing.

In the second stage, the encoder makes a decision on whether filtering is performed for the current frame or not. If filtering is done for the frame, a third stage is performed, where the encoder decides an on/off filter flag for every coding tree block (CTB) in the frame.

There is only the 9×7 cross + 3×3 square filter shape in HM-7.0; it is presented in Figure 1b). To filter one pixel, 19 pixel values and 10 coefficients are needed. Pixel filtering is performed as in Figure 2. The *Scale & Clip* function scales pixel values to the range [0,255] and is equivalent to the following equation:

$$sc(p_x) = \begin{cases} 0 & \text{if } (p_x + 2^7) * 2^{-8} \leq 0 \\ 255 & \text{if } (p_x + 2^7) * 2^{-8} \geq 255 \\ (p_x + 2^7) * 2^{-8} & \text{when others} \end{cases} \quad (1)$$

ALF filtering can be performed CTB by CTB in a raster scan order. To reduce memory requirements, virtual boundary (VB) processing is introduced in [8]. The VB is a horizontal boundary located above each CTB's bottom boundary. In VB processing, the filter shape is vertically reduced to avoid crossing the virtual boundary. Figure 3 shows how filtering is done for pixels above and below the VB.

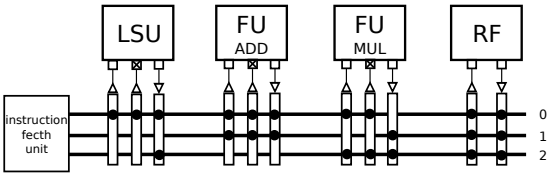

Fig. 4. An example TTA processor.

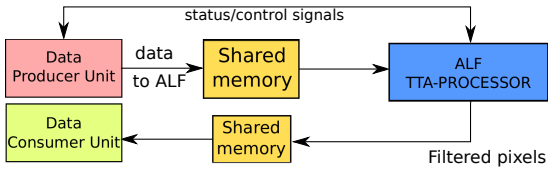

Fig. 5. ALF TTA-processor interface.

2.3. Related work

Du et al. [9] propose combined deblocking and ALF hardware architecture for H.264/AVC. The architecture uses 9×9 tap ALF and can process a macroblock in 843 clock cycles. The maximum clock frequency of the architecture is 211 MHz.

Radiess et al. [10] have designed a hardware implementation for ALF filtering based on the Test Model HM-4.0 of the HEVC standard. Since HM-4.0 includes three different filter shapes (diamond 5×5, 7×7 and 9×9), three hardware architectures are proposed. Their implementation (diamond 7×7) is able to process 102 1080p frames at an operating frequency of 212 MHz. The authors have not indicated gate counts or power consumption.

Alvarez-Mesa et al. [5] decode multiple LCU rows in parallel. Their proposal achieved a resolution of 1920×1080p at 50 fps using 12 Intel Xeon cores operating at 3.3 GHz. With 12 threads, the ALF part of the total decoding time was 18%.

The works of Du et al. and Radiess et al. present higher processing rates than the proposed solution, which is natural since their proposals are a fixed hardware design. Furthermore, the proposal of Radiess et al. does not support frame/tile boundary padding or virtual boundary processing at all, unlike our proposal. The work of Du et al. is directed at H.264/AVC. The solution of Alvarez-Mesa, on the other hand, is not suitable for embedded devices.

2.4. Transport triggered architecture

Transport triggered architecture processors (TTAs) resemble Very Long Instruction Word (VLIW) architectures [11]. TTAs exploit instruction level parallelism similarly to VLIW processors; multiple instructions are executed in parallel every clock cycle. TTAs have only one, *move*, instruction which is used to transport data between function units (FU) and register files (RF). Computational operations are triggered by data transports to the *trigger* ports of FUs. The results of operations can be moved from FU outputs directly to inputs of other FU input ports, which reduces the use of register files. The visibility of all data transports allows the compiler to control and optimize data moves.

Figure 4 presents an example TTA processor. The processor consists of three *transport buses* (black horizontal lines), FUs (load store unit, adder, multiplier) and an RF. FUs and the RF are connected to transport buses over *sockets* (rectangular vertical bars with black dots). The arrows above the sockets tell us whether the socket is an input or an output socket. Because of the three transports buses, three data transports can be done every clock cycle.

The TTA development environment is called the TTA-based Co-design Environment (TCE) [12]. The TCE compiler generates machine code for the target processor from high level language source codes. The TCE simulator can analyze cycle by cycle how a program runs on a target TTA processor. A synthesizable RTL description of the given processor design can be created using a Processor Generator.

3. PROPOSED SOLUTION

The interfaces of the proposed ALF processor are presented in Figure 5. A unit which produces data to the ALF is assumed to store the following information in the shared memory: pixels to be filtered, ALF filter coefficients, and information about which filter set is in use at any given time (defined for each 4x4 block). When all necessary data is stored in the shared memory, a request to perform filtering is transmitted to the ALF processor using a control signal. The ALF processor reports its progress via status signals.

The ALF processor's software is designed to perform filtering only across slice boundaries, thus the HEVC *slice_loop_filter_across_slices_enabled_flag* must be equal to 1. The whole frame or tile is processed CTB row by CTB row, and the software assumes that the shared memory contains pixels of the four last lines from the previous CTB row and pixels from the current CTB row excluding the four last lines.

The CTB row filtering is performed using a raster scan order in a 4×4 block at a time using a predefined filterset. At a boundary of a tile or a frame, the filter's shape is changed so that pre-boundary padding is not necessary. A 32-bit word in the memory includes four 8-bit pixel values. At the beginning of the row, 16 32-bit memory accesses are needed to get the necessary data for filtering a 4×4 block. Already read pixels are maintained in register files as long as they are needed. Thus, for the next 4×4 block only ten 32-bit memory read operations are required. The proposed TTA processor has several registers since the values of 64 pixels are maintained to reduce memory operations. Reducing memory read operations is desirable due their power and time consuming nature (a memory read takes 3 clock cycles).

Table 1. Function units of the ALF Processor.

	FU name	*inputs*	*outputs*	*dataw*
1×	lsu_datamem	2	1	32
1×	lsu_sharedmem	2	1	32
6×	add	2	1	21
7×	mul	2	1	21
1×	add_shl_shr_shru_sub	2	1	32
1×	eq_gt_gtu	2	1	32
1×	and_ior_xor	2	1	32
2×	4INPUTADD	4	1	21
1×	SEPARATE	1	4	32/8
1×	JOIN	4	1	8/32
1×	SCALE_CLIP	1	1	21
1×	ALF_STATUS	1	1	8
1×	bool	1	1	1
5×	rf_16×21	2	2	21
1×	rf_16×32	2	2	32
1×	gcu	1	1	32

Fifteen buses has been observed to be the minimum bus requirement for five clock cycles per pixel with the current software. The average utilization rate of buses is over 90%. Increasing the number of transports buses could significantly reduce the cycle count; however, the circuit gate count would grow and the maximum clock frequency would be reduced.

Figure 2 shows that nine add (stage 1) and ten multiply (stage 2) units can be exploited at a time. In practice, six adder FUs and seven multiplier FUs are considered to be sufficient. The latency of the multiplier units is 2 clock cycles and the latency of the adders is one clock cycle. Two *four-input adders* are presented to merge stages 3-4 and stages 5-6. A four-input add is done in one clock cycle, as is *scale and clip* which is also implemented with a special function unit.

To handle 32-bit pixel packets, two special function units are used: *separate* and *join*. Decomposing of 32-bit pixel packets to four 8-bit pixels *separate* is used. *Join* works inversely and constructs 32-bit pixel packets from four 8-bit pixels. Both operations are performed in a one clock cycle. All FUs in the architecture are listed in Table 1.

4. EXPERIMENTS

The ALF processor was verified using the Altera Stratix III (EP3SE260F1152C2) FPGA. A luminance frame was decoded to ensure that the design works correctly. For the verification, a simple data producer unit is implemented. The data producer is connected to the ALF processor as in Figure 5. The shared memory used was a dual port memory (2 address ports, one data write port and one data read port), instantiated from the Altera IP core libraries. The FPGA synthesis results are listed in Table 2.

The ALF processor was synthesized with the UMC 90 nm standard cell library (fsd0k_generic_core_1d0vtc). Synopsys Design Compiler was used to estimate the gate count and the maximum clock frequency. Power consumption measurement was done using Synopsys PrimeTime with internal signal transitions recorded by Mentor Graphics ModelSim.

Table 2. FPGA synthesis results for the ALF-processor.

Resource	*unit*
Logical Elements	12983
Total Registers	5723
DSP 21-bit elements	16
Embedded Multiplier 21-bit	7

The ALF processor's area estimate was 142491 NAND g.e. and power consumption of the ALF processor was 43 mW. The processor ran at a maximum clock frequency of 290 MHz, which produces 27 1920×1080p luminance frames per second. That is very close to reaching the HDTV 1920×1080p at 30 fps which requires a clock frequency of 311 MHz. Synthesis using the 65 nm standard cell library would easily enable 311 MHz clock frequency or higher [13]. For supporting chroma channels and UHD resolutions (2160p and 4320p), multiple parallel instances of the proposed processor can be placed in one design.

5. DISCUSSION

At the moment, ALF has been left out of the HEVC standard, due to the fact that it is computationally very expensive. Thus, the latest reference software version which includes ALF is WD 7. However, it has been proposed that ALF should be included in the upcoming HEVC profiles or in the main profile with some clean-ups and complexity reductions [14]. This work shows that efficient, software based ALF implementations even for embedded devices are possible. Thus, including ALF in the high-performance HEVC profiles is a realistic option for future HEVC development.

6. CONCLUSION

This paper presents an application-specific instruction processor for Adaptive Loop Filtering. The ALF is a subjective and objective image quality improving filter in the HEVC standard. The proposed processor is fully programmable through C language. The design was verified on FPGA and synthesized using the UMC 90 nm standard cell library. The processor synthesis reports a gate count of 142491 g.e. and a power consumption of 43 mW. For filtering one pixel, 5 clock cycles are consumed by the designed processor, and it is able to filter 30 1920x1080p luminance frames per second at a clock frequency of 311 MHz. To the authors' best knowledge this is the first programmable solution for ALF on embedded devices.

7. REFERENCES

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. Circuits and Systems for Video Tech*, 2012.

[2] J. Ohm, G.J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684.

[3] C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, "Parallel Scalability and Efficiency of HEVC Parallelization Approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[4] T. Ikai, Y. Yasugi, T. Yamamoto, T. Tsukaba, and T. Aono, "Inclusion of ALF in Main profile and additional test results," Tech. Rep., 2012, Document of Joint Collaborative Team on Video Coding (JCT-VC), JCTVC-J0330r3.

[5] M. Alvarez-Mesa, C. C. Chi, B. Juurlink, V. George, and T. Schierl, "Parallel video decoding in the emerging HEVC standard," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 1545–1548.

[6] C.-Y. Chen and C.-Y. Tsai, "AHG6: ALF with non-normative encoder-only improvements," Document of Joint Collaborative Team on Video Coding, July 2012, JCTVC-J0048.

[7] C.-Y. Tsai, C.-Y. Chen, C.-M. Fu, Y.-W. Huang, and S. Lei, "One-pass encoding algorithm for Adaptive Loop Filter in High Efficiency Video Coding," in *2011 IEEE Visual Communications and Image Processing (VCIP)*, 2011, pp. 1–4.

[8] C.-M. Chen, C.-Y. Fu, "Non-CE8.c.7: Single-source SAO and ALF virtual boundary processing with cross 9×9," Input Document to JCT-VC, November 2011, JCTVC-G212.

[9] J. Du and L. Yu, "A parallel and area-efficient architecture for deblocking filter and Adaptive Loop Filter," in *2011 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2011, pp. 945–948.

[10] F. Rediess, L. Agostini, C. Cristani, P. Dall'Oglio, and M. Porto, "High throughput hardware design for the Adaptive Loop Filter of the emerging HEVC video coding," in *2012 25th Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2012, pp. 1–5.

[11] H. Corporaal, *Microprocessor Architectures: From VLIW to TTA*, John Wiley & Sons, Inc., New York, NY, USA, 1997.

[12] O. Esko, P. Jääskeläinen, P. Huerta, C. S. de La Lama, J. Takala, and J. I. Martinez, "Customized exposed datapath soft-core design flow with compiler support," in *20th International Conference on Field Programmable Logic and Applications*, Milano, Italy, 2010, pp. 217–222.

[13] M. Muller, "Embedded Processing at the Heart of Life and Style," in *IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 2008, pp. 32–37.

[14] D.-K. Kwon and M. Budagavi, "Crosscheck of JCTVC-J0390: AHG6: Further cleanups and simplifications of the ALF in JCTVC-J0048," Input Document to JCT-VC, July 2012, JCTVC-J0440.