*International Journal of Engineering & Technology*, 7 (3.24) (2018) 176-184

International Journal of Engineering & Technology

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***

# Efficient Hardware Design of In-Loop Filter for High-Performance HEVC Encoder

**SeungyongPark[1], Kwangki Ryoo[*2]**

*Department of Information & Communication Engineering, Hanbat National University, Daejeon, Korea*
**Corresponding author E-mail: kkryoo@gmail.com[*2]*

## Abstract

**Background/Objectives**: An HEVC encoder consists of an in-loop and deblocking filter. In this paper, we propose an efficient in-loop filter hardware architecture for high performance HEVC encoder.
**Methods/Statistical analysis**: The proposed in-loop filter hardware structure is divided into deblocking filter, SAO, and in-loop filter scheduler. The proposed deblocking filter uses an internal line buffer to reduce the overhead of external memory access in order to minimize hardware footprint. SAO has a high throughput with a 4x4 block size unit operation. It uses minimal hardware area by simplifying the equation used to determine the offset value.
**Findings**: The in-loop filter hardware structure of the HEVC encoder proposed in this paper is synthesized at 260MHz with 90nm cell library, and it is possible to process 262.78K gates and 8K@120Fps in real time.
**Improvements/Applications**: The proposed in-loop filter hardware architecture can be applied to high-performance HEVC video codec hardware, and it can be used as an individual filter because of in-loop filter characteristics.

***Keywords:*** *HEVC, In-Loop filter, Deblocking filter, SAO, Hardware design*

## 1. Introduction

In recent times, user demand for high-definition video service is on the rise due to advancements in multimedia technology, communication and various multimedia contents services. As a result, the services for HD video are now provided not only by TV but also by smart phones and various portable terminal devices. Recently, 4K UHD (Ultra High Definition) broadcasting stations with resolution of 4 times or more of HD (High Definition) UHD broadcasting service has started in earnest. In addition, with the inception of self-broadcasting via internet platforms, 4K UHD broadcasting is possible and readily accessible by viewers on the internet. Recently, content demand for Virtual Reality (VR) and Augmented Reality (AR) is increasing, and demand for Mixed Reality (MR), which combines virtual reality and augmented reality, is also increasing. Consequently, various games and services to which virtual reality and augmented reality are applied are on the market, and a lot of research has been conducted on data processing methods. However, in the case of video service, the amount of information generated is higher than that of voice. As the number of users of the high-definition video service increases, the video service generates high traffic for the Internet and communication lines for transmitting the data and the data generated. According to Cisco, which offers networking hardware and security services, traffic to video among all internet traffic is expected to increase from 67% in 2016 to 82% in 2021 [1]. Image compression technology for image data processing is improving as HD video service evolves into UHD image service. In 2001, Video Coding Experts Group (VCEG) of ITU-T and Moving Picture Experts Group (MPEG) of ISO/IEC jointly organized Joint Video Team (JVT) and standardized H.264/AVC. Compared with the previous compression technologies MPEG-2 and MPEG-4, the compression technology has excellent performance, but there is a limitation in providing ultra-high-resolution image service of UHD level or higher [2]. In 2010, ITU-T VCEG and ISO/IEC MPEG formed a Joint Collaborative Team on Video Coding (JCT-VC) for UHD video service and standardization of HEVC (High Efficiency Video Coding) [3]. HEVC shows a bit reduction of about 50% in the subjective image quality criterion and about 40% in the objective image quality criterion compared to the previous compression standard H.264/AVC. The HEVC standard has a hybrid block-based encoding/decoding structure similar to the existing H.264/AVC, and adopts various techniques for high-efficiency compression in detail. The HEVC is composed of hierarchical encoding block unit and structure, intra prediction technique with 36 prediction directions, effective inter picture prediction technique through AMVP (Advanced Motion Vector Prediction) and Merge mode, DCT (Discrete Cosine Transform), a two-step in-loop filter technique using deblocking filter and SAO (Sample Adaptive Offset), and entropy technology using CABAC (Context-based Adaptive Binary Arithmetic Coding) [4]. In particular, the in-loop filter technology includes deblocking filters and the newly added SAO technology from the HEVC. The deblocking filter is a technique for eliminating the blocking phenomenon occurring in the block-based image compression. The SAO is an offset filter for improving the deterioration in the block. By adding a constant offset to the restored pixel, SAO is reduced. Since the filtered pictures are used as reference pictures in the inter-picture prediction technique through the in-loop filter technique, not only the subjective picture quality by the filtering is improved but also the encoding efficiency is improved in the inter picture prediction technique. This is because more accurate inter-picture prediction and compensation can be performed when a filtered picture is used as a reference picture rather than a restored picture having blocking deterioration or ringing phenomenon is used as a reference picture. However, the in-loop filter has a

disadvantage of requiring an additional amount of computation in a portion where filtering is performed on the restored pictures of the encoder and the decoder, although the filtering method has an advantage of improving the subjective image quality and improving the encoding efficiency. In particular, the deblocking filter causes frequent memory access by loading restored pixels stored in the memory when performing filtering, performing filtering, and storing the filtered pixels in the memory again. In addition, the deblocking filter itself is complicated in its filtering operation itself, and takes up 20% to 30% of the complexity in the decoder due to this computational complexity and overhead for memory access. Because SAO performs pixel-by-pixel operation, it needs high computation time and a large memory in order to select optimal mode and offset. Due to the complexity of such an in-loop filter, video codecs prior to H.264/AVC used mostly post-processing filters, and relatively recently standardized H.264/AVC and HEVC use in-loop filtering. In this paper, we propose an efficient in-loop filter hardware with low memory access count, computation time, and memory by internal buffer structure with routing function and 4x4 block-based operation for designing high performance HEVC encoder.

## 2. Overview of HEVC In-Loop Filter

An in-loop filter is a technique used to improve subjective image quality and encoding efficiency. The in-loop filter of HEVC consists of deblocking filter and SAO. It is used to remove the blocking phenomenon occurring in the block-based image compression standard and the ringing phenomenon inside the block. The reconstructed image reconstructed through the in-loop filter is used as a reference image when displaying an image or performing inter-view prediction. The deblocking filter and SAO of the HEVC in-loop filter have encoding efficiencies of up to 6% and 23.5%, respectively [5]. The in-loop filter of HEVC improves subjective image quality and encoding efficiency, but requires additional computation, resulting in overhead for computational complexity and memory access. The in-loop filter time of HEVC decoding accounts for 19% to 21% of the total decoding time [6]. Research has been actively carried out to reduce the computational complexity and sophistication of the in-loop filter of HEVC. Both the software algorithm and hardware structure design of efficient in-loop filter have been studied in these referenced researches [7-15].

### 2.1. Deblocking Filter

The deblocking filter of HEVC is used to remove the blocking deterioration that occurs in block-based image compression. The deblocking filter of H.264/AVC, which is the conventional image compression standard, performs filtering in units of 4x4 blocks. However, since the 4x4 block boundary unit filtering has a high computational complexity in the UHD class image, the deblocking filter of the HEVC is changed to the 8x8 block boundary unit. Also, for high filtering efficiency, the deblocking filter strength can be varied in units of 4 pixels on the 8x8 block boundary. Figure 1 shows the process of deblocking filter execution of HEVC. The deblocking filter is divided into Boundary Judgment, Bs Calculation, Filter Decision, and Filtering [16].

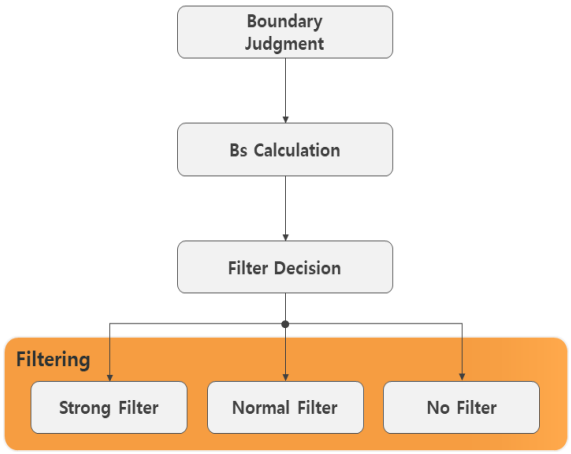


**Figure 1**: The process of deblocking filter

Blocking degradation occurs only at the boundary of the block size for which prediction and conversion are performed, so there is no need to perform a deblocking filter at all 8x8 block boundaries. The Boundary Judgment process divides the boundaries of the blocks used in intra prediction, inter prediction, and conversion. The block types used in prediction and conversion are CU, PU, and TU. Block information of CU, PU, and TU can be obtained through Depth, puSize, and TransformIdx information. Bs Calculation determines the strength of the filtering. The Bs Calculation of the deblocking filter supports three filtering strengths at the block boundary determined by Boundary Judgment. The filter decision finally determines the presence or absence of the filter, and uses the threshold values β and tc obtained by using the pixel and QP average values of the P and Q blocks. In addition, we use the information determined in Boundary Judgment and Bs Calculation to decide whether to apply filtering. Filtering is classified into strong filtering, normal filtering, and no filtering. Information about each filtering is determined and input by the filter decision.

### 2.2. Sao

The HEVC's SAO technology compensates for the degradation caused by image compression using the original image and the reconstructed image through offset. In particular, when the QP value is large, the ringing phenomenon may occur in the edge region of the input image. The ringing phenomenon is a main cause of deterioration of the subjective image quality as well as the blocking phenomenon. The HEVC uses SAO to improve the encoding performance by improving the main picture quality by removing the ringing phenomenon and minimizing the error between the original image and the restored image. The offset used in SAO is divided into edge offset and band offset, and is performed based on CTU. SAO is independent of both the luminance component and the chrominance component, and is performed in the same manner. The edge offsets are divided into classes according to the angle, into four categories according to each class. The band offset is divided into a maximum value from 0 to represent a pixel, and an offset value is obtained according to the position of the input pixel value. Since SAO performs operations on a pixel-by-pixel basis, it takes a long time to compute, demanding a large memory space as well [17]. The edge offset of SAO is used to effectively correct the error of the reconstructed pixel, considering the edge direction within the block to be coded. The edge direction is classified into classes, and the edge offset class is shown in Figure 2. The classes of edge offsets have four orientations, horizontal and vertical, 135° and 45°.

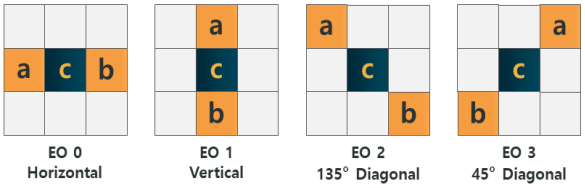


**Figure 2:** Classes of edge offsets

Table 1 shows the categories of edge offsets and the determination conditions. There are five categories in each of the four classes, and each of the categories 1 to 4 has one offset. Thus, the edge offsets take a total of 16 offsets, and the RDO selects an optimal class.

**Table 1:** Categories of edge offsets and determination criteria

| Category | Condition |
|---|---|
| 1 | c<a && c<b |
| 2 | (c<a && c==b) \|\| (c<b && c==a) |
| 3 | (c>a && c==b) \|\| (c>b && c==a) |
| 4 | c>a && c>b |
| 0 | None of the above |

The band offset of SAO classifies the pixels in the block to be coded into bands having similar brightness values, and the band offset is composed of 32 bands. If one pixel is represented by 8 bits, the value of the pixel has a value between 0 and 255. In this case, the values of 0 to 255 pixels are divided into 32-band bands, and one band width has a value of 8. Figure 3 shows an example of band offset and offset determination.

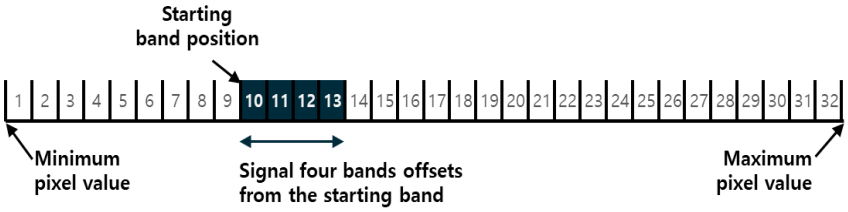


**Figure 3:** Example of band offset interval and offset determination

The band offset produces 32 offsets corresponding to 32 bands, and the optimal band position is determined by RDO at all positions. RDO determines the optimum band position, and sends offsets of four consecutive bands at the determined optimum band position. In addition, since the band offset cannot distinguish the positive edge offset from the negative edge offset differently from the edge offset, the code information must be transmitted to the decoder. The offset of the SAO is generated in pixel units, and has a value from -7 to 7. In addition, since RDO determines the optimal offset, consideration must be given not only to the filtering efficiency but also to the bit amount generated. For example, offset 1 and 7 have different bit sizes. The offset 1 is represented by 2 bits, and the offset 7 is represented by 7 bits. The band offset is required up to the sign information bit for each offset, and the sign information bit is represented by one bit. The optimal offset is selected in consideration of the number of bits generated at the offset and the image quality improved when the offset is applied. The distortion of the original pixel and the reconstructed pixel before SAO is applied can be obtained by equation (1). s(k) and x(k) denote the kth original pixel and the reconstructed pixel in the block to be coded.

$$D_{pre} = \sum_{k \in C}(s(k) - x(k))^2 \quad (1)$$

The distortion of the original pixel and the reconstructed pixel after applying SAO can be obtained by equation (2). h denotes an offset.

$$D_{post} = \sum_{k \in C}(s(k) - (x(k) + h))^2 \quad (2)$$

When SAO is applied, the bit rate increases because additional bits are generated as compared with the case where the SAO is not applied. Therefore, in order for SAO to be selected after the RDO process, the distortion ($D_{post}$) after applying the SAO should be less than the distortion before the SAO application ($D_{pre}$). If the difference between $D_{post}$ and $D_{pre}$ is defined as ΔD, the equation (3) is obtained. N is the number of pixels in the block to be coded, and E is equation (4).

$$\Delta D = D_{post} - D_{pre} = \sum_{k \in C}\left(h^2 - 2h\big(s(k) - x(k)\big)\right) = Nh^2 - 2hE \quad (3)$$

$$E = \sum_{k \in C}(s(k) - x(k)) \quad (4)$$

In equation (3), N is the number of pixels, so it has a constant value. E can be obtained from the original pixel and the restored pixel. The HEVC reference software HM encoder uses ΔD to define the delta RD cost as in equation (5) and to determine the offset h with the smallest value. The HM encoder uses the initial h value E/N to compute the h value at high speed.

$$\Delta j = \Delta D + \lambda R \quad (5)$$

## 3. Proposed Hardware Structure

The proposed in-loop filter hardware structure is shown in Figure 4. Figure 4 shows only the functional parts of the in-loop filter hardware architecture. The proposed in-loop filter hardware structure consists of DF(Deblocking Filter) module, SAO module and In-Loop Filter Scheduler module.

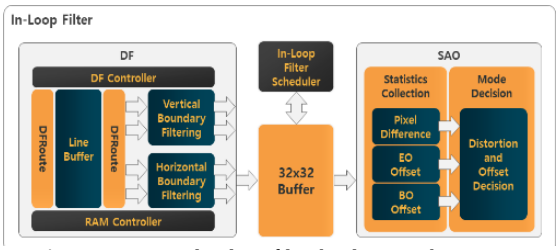


**Figure 4**: Proposed in-loop filter hardware architecture

The DF module performs a deblocking filter function and includes a vertical and horizontal filter module for deblocking filtering of the 8x8 block boundary. Each vertical and horizontal filter module performs two 4-pixel filtering. The SAO module consists of a Statistics Collection (SC) part for collecting information on EO (Edge Offset) and BO (Band Offset) and a Mode Decision (MD) part for determining SAO mode. The SC portion includes a Pixel Difference module for obtaining the difference between the original pixel and the restored pixel, and an EO Offset and Band Offset module for generating the offset information of the EO and the BO. The MC part carries out the distortion calculation and the optimal offset determination with the offset information and the pixel difference generated in the SC. The SAO module is also based on a 4x4 block. The In-Loop Filter Scheduler module uses the 32x32 Buffer to transfer the deblocking filtering value to the SAO input.

### 3.1. Deblocking Filter

The proposed deblocking filter hardware structure is shown in Figure 5. It includes a DF controller module for controlling the deblocking filter, a data controller module for controlling pixel data in the external memory, and information for determining whether pixel data and filtering are stored in a buffer module for performing filtering, and a filtering module for performing filtering. The Buffer module consists of a Route module and a Line Buffer module. Because the deblocking filter reuses the filtering pixels, it performs the step of storing in the memory. This memory access degrades the performance of the hardware module. Therefore, the line buffer module reduces the memory access by storing the filtered pixel data directly in the CB without storing it in the memory through the common buffer (CB) which can simultaneously read and write. In addition, the number of line buffers used through the route module is minimized. The filtering section is divided into a vertical filter module and a horizontal filter module, and each filter module performs filtering on eight lines. In addition, the 8-line filter module is composed of 4-line filters in consideration of 4x4 block size. In order to design high performance and efficient deblocking filter hardware, the conditional expressions used in each module are designed as parallel structure and resource sharing structure. Also, the design has a low power hardware structure by applying clock gating through an information signal indicating the presence or absence of filtering.

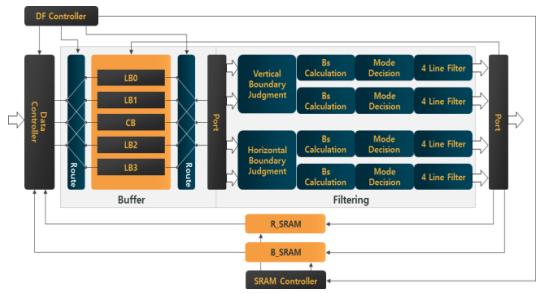


**Figure 5**: Proposed deblocking filter hardware architecture

### 2.1.1. Deblocking Filtering and Routing Order

The proposed deblocking filtering order and routing order are shown in Figure 6. The total block size applying the deblocking filter is 32x32 blocks and the internal block size is 4x4 blocks. The deblocking filter divides an 8x8 block line into 12x32 size units, and uses 5-line buffers 4x32 in size. A 32x32 block size has a size of 36x32 block including pixel data for a 32x32 block boundary, and a 36x32 block has 9x32 line buffers. Table 2 shows the routing procedure and operation of the proposed line buffer structure.

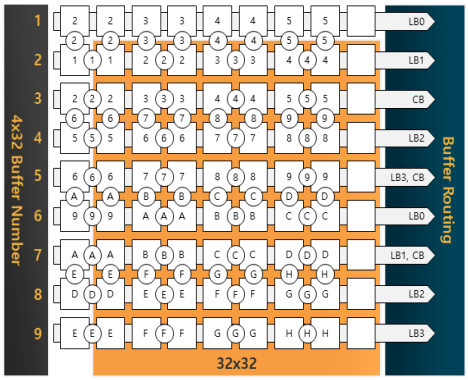


**Figure 6:** Deblocking filtering and routing order

**Table 2**: Routing order and behavior of line buffers

| Route Number | 4x32 Buffer Number | Line Buffer | Filtered Pixel |
|---|---|---|---|
| 1 | 1 | LB0 | Output |
| | 2 | LB1 | Output |
| | 3 | CB | CB Stored |
| 2 | 3 | CB | Output |
| | 4 | LB2 | Output |
| | 5 | LB3 | CB Stored |
| 3 | 5 | CB | Output |
| | 6 | LB0 | Output |
| | 7 | LB1 | CB Stored |
| 4 | 7 | CB | Output |
| | 8 | LB2 | Output |
| | 9 | LB3 | Output |

The routing order of the proposed deblocking filter is divided into four stages. In the case of Route 1, the third 4x32 line buffer is stored in CB because it is used in Route 2 after vertical filtering. The second 4x32 line buffer in the second route is filtered and stored in the CB. CB reduces the internal 4x32 line buffer in the deblocking filter hardware and reduces the memory access latency because it does not store the filtered 4x32 pixel data in external memory. Figure 7 shows the data flow of the proposed routing structure. It reads the data from the external memory and stores it in the line buffer. The filter module for deblocking filtering reads the data from the line buffer. When reading data from external memory, it processes 4x4 block size. When reading line buffer or storing CB, it processes 4x8 block size.

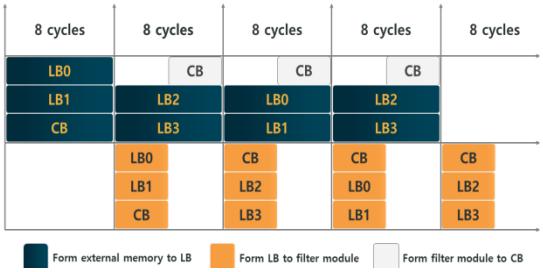


**Figure 7:** Routing data flow

### 2.1.2. Low-Power Hardware Structure

The deblocking filter determines whether to filter by Boundary Judgment, Bs Calculation, and Filter Decision. For example, in a Boundary Judgment, a non-boundary part of a PU or TU does not perform a filtering operation thereafter. In the case of software, filtering is performed on the next boundary. However, since the same time is given to all the filtering operations in the hardware, it is possible to reduce the operating power by determining the presence or absence of filtering and then turning off the clock when filtering is not performed. Figure 8 shows the clock gating structure of the proposed deblocking filter for a low-power hardware architecture.

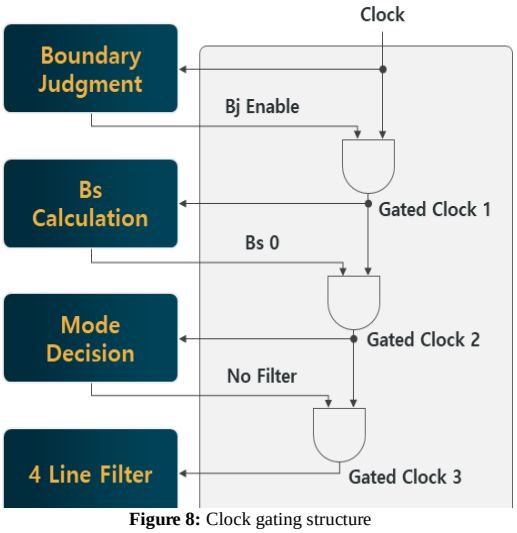


**Figure 8:** Clock gating structure

### 2.1.3. 4-Stage Pipeline Structure of Filtering Module

The filtering module has a unit vertical filtering and a horizontal filtering structure, and each filtering is performed in units of 8 pixels. In addition, to perform the 4-pixel block boundary strength, vertical filtering and horizontal filtering are composed of two 4-pixel filtering structures. Filtering module consists of Boundary Judgment module, Bs Calculation module, Mode Decision module and the 4-Line Filter module, and it is designed with a 4-stage pipeline structure for high throughput. Figure 9 shows the submodules and pipeline structure inside the Filtering module.
The proposed deblocking filter stores the data to be filtered from the external memory into the line buffer for 8 cycles, and the stored line buffer data is input to the filtering module for 4 cycles. The Filtering module filters for 7 cycles except one cycle that takes the first data to be filtered. Therefore, the Filtering module adds up to the cycle of reading data from the line buffer to complete the filtering in 8 total cycles.

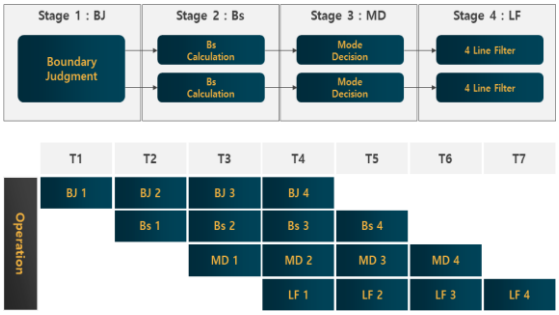


**Figure 9:** 4-Stage Pipeline Structure of Filtering Module

### 2.1.4. Parallel Structure of Bs Calculation Module

Bs Calculation functions to select different filtering intensities in 4-pixel units. Figure 10 shows the Bs Calculation method used in the HEVC standard. Bs Calculation of HEVC standard is determined after confirming conditions of maximum 5 steps. These conditions introduce delays in the hardware, which degrade the performance of the entire system. Therefore, it is possible to determine Bs in a way that all conditions are checked in a parallel structure and priority is given to the hardware, and a conditional expression requiring a maximum of 5 steps in designing in parallel structure can be implemented in one step. Figure 11 shows the parallel structure of Bs Calculation. The priority of Bs is 2 to 0. In the parallel structure of Bs Calculation, all conditions are used to determine Bs by generating 1-bit information after comparison in the first step.

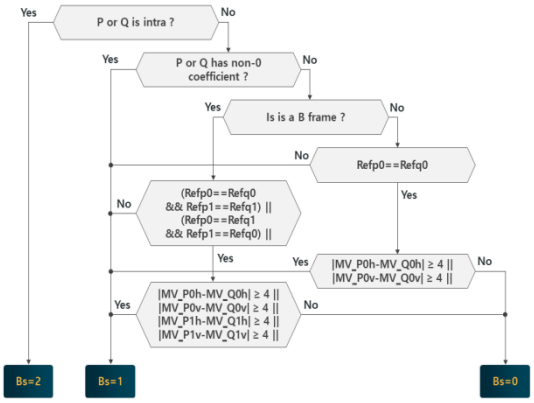


**Figure 10:** Bs calculation structure of HEVC

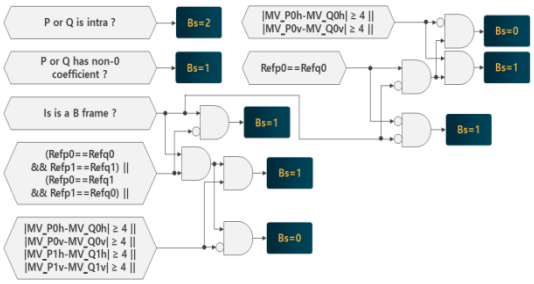


**Figure 11:** Bs calculation parallel structure

## 3.2. Sao

The proposed SAO hardware architecture is shown in Figure12. It consists of the Statistics Collection part (which collects information to determine the offset of the edge offset (EO) and the band offset (BO), the optimal offset determination of EO and BO) and a mode decision part. Since SAO performs EO and BO on all pixels within a 64x64 block, the computation time is high. Therefore, the proposed SAO hardware architecture computes 4x4 block size rather than one pixel, and supports a maximum 64x64 block size. The Pixel Difference (PD) module performs the operation of obtaining the difference between the original pixel and the restored pixel. The EO module receives 6x6 block size restoration pixels for edge offset computation of 4x4 block size, generates class and category information of each pixel, accumulates pixel difference input from PD module using class and category information. The BO module receives the restored pixels of 4x4 block size, generates band position information of each pixel, and accumulates and counts the pixel difference input from the PD module using the band position information. The DIST module receives the accumulated pixel differences and counters from the EO and BO modules, generates the offset, and selects the optimal offset type and offset information considering the RDO.

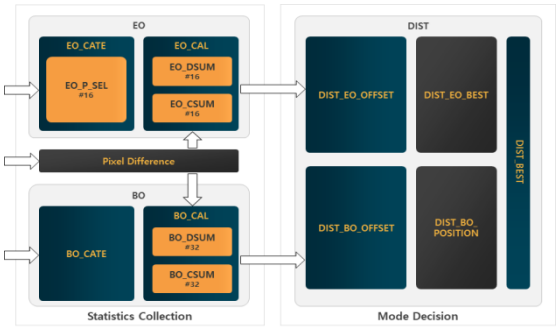


**Figure 12:** Proposed SAO hardware architecture

### 3.2.1. Parallel Structure of EO_CATE Module

The EO_CATE module consists of 16 EO_P_SEL submodules, and receives pixels corresponding to a 6x6 block size. The EO_P_SEL module receives pixels corresponding to a 3x3 block size and generates category information for each class. Figure 13 shows the surrounding pixels needed to perform EO in 4x4 block size.

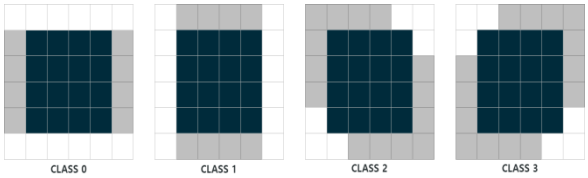


**Figure 13:** Schematic pixel required to perform EO in 4x4 block size

Figure 14 (a) shows the method of determining the existing EO category, and (b) shows the simplified hardware category decision

structure. In hardware design, hardware resources can be optimized by designing redundant operations as resource sharing structure.

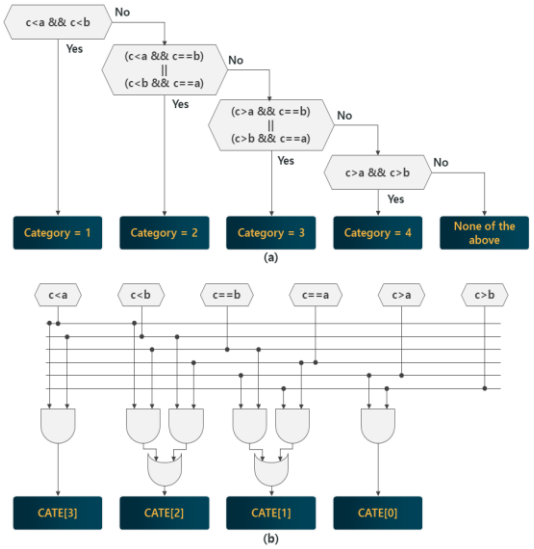


**Figure 14:**EO category decision method and hardware structure optimization

### 3.2.5. Proposed Offset Decision Hardware Algorithm

The DIST_EO_OFFSET module performs the function of determining the optimal offset of the category corresponding to each class. In the HEVC standard, the initial offset is obtained through E/N, but in the proposed hardware, it is performed in parallel with the offset from -7 to 7. In equation (3), $Nh^2$-2hE is applied to obtain ΔD. N and E are input through the EO_CAL module, and h is a constant value that does not change since it is an offset value from -7 to 7. Table 3 shows the values of $h^2$ and 2h according to the offset value.

**Table 3**: Values of h2 and 2h according to the offset value

| Offset | $h^2$ | 2h | Offset | $h^2$ | 2h |
|---|---|---|---|---|---|
| -7 | 49 | -14 | 1 | 1 | 2 |
| -6 | 36 | -12 | 2 | 4 | 4 |
| -5 | 25 | -10 | 3 | 9 | 6 |
| -4 | 16 | -8 | 4 | 16 | 8 |
| -3 | 9 | -6 | 5 | 25 | 10 |
| -2 | 4 | -4 | 6 | 36 | 12 |
| -1 | 1 | -2 | 7 | 49 | 14 |
| 0 | 0 | 0 | - | - | - |

As shown in Table 4, the $h^2$ and 2h values for all offsets are always fixed, only N and E are variable. Therefore, it is possible to implement hardware structure by using shift and adder for $h^2$ and 2h. In hardware, multipliers have high computational complexity and computation time, which causes performance degradation of the whole system. The proposed hardware architecture minimizes the multiplier for high performance in-loop filters. Equation (6) shows the relationship between left shift and multiplier. Table 5 shows the values of $h^2$ and 2h as a shift and adder structure using equation (6). For example, when the offset value h is 7, $h^2$ has a value of 49, and 49 can be divided into N×32 and N×16, +N. Therefore, it can be calculated as N << 5 corresponding to N×32 and N<<4,+N corresponding to N×16.

$$\begin{aligned} x \ll 1 &= x \times 2 \\ x \ll 2 &= x \times 4 \\ x \ll 3 &= x \times 8 \\ x \ll 4 &= x \times 16 \\ x \ll 5 &= x \times 32 \end{aligned} \tag{6}$$

**Table 4:** $h^2$ and 2h shift and adder structures

| \|h\| | $h^2$ | Shift and Adder | 2h | Shift and Adder |
|---|---|---|---|---|
| 7 | 49 | N≪5 + N≪4 + N | 14 | E≪3 + E≪2 + E≪1 |
| 6 | 36 | N≪5 + N≪2 | 12 | E≪3 + E≪2 |
| 5 | 25 | N≪4 + N≪3 + N | 10 | E≪3 + E≪1 |
| 4 | 16 | N≪4 | 8 | E≪3 |
| 3 | 9 | N≪3 + N | 6 | E≪2 + E≪1 |
| 2 | 4 | N≪2 | 4 | E≪2 |
| 1 | 1 | N | 2 | E≪1 |

The DIST_EO_OFFSET module computes the Δj value using the equation (5) for the number of bits of each offset after the ΔD operation, and determines an offset value having the minimum Δj value. λ is a Lagrangian function with a value of 49.2221, and R is the number of bits that occur when encoding the offset. Table 5 shows the number of generated bits according to the EO offset and the λ value in the proposed hardware structure. Finally, the DIST_EO_OFFSET module outputs the optimal offset and Δj value for each class.

**Table 5**: Number of occurrences and λ value according to EO offset

| Offset | R | λ |
|---|---|---|
| ±7 | 7 | 345 |
| ±6 | 7 | 345 |
| ±5 | 6 | 295 |
| ±4 | 5 | 246 |
| ±3 | 4 | 197 |
| ±2 | 3 | 148 |
| ±1 | 2 | 98 |
| 0 | 1 | 49 |

The DIST_BO_OFFSET module performs the function of determining the offset of each band, and its structure is similar to the DIST_EO_OFFSET module. Unlike the DIST_EO_OFFSET module, however, the DIST_BO_OFFSET module performs offset operations on 32 bands. In addition, the BO must transmit the band offset to the decoder and to the code bit and the band position. The additional bits generated at this time are 4 bits for the sign bit and 5 bits for the band position, and the DIST_BO_OFFSET module calculates the lambda value in consideration of the sign bit. Table 6 shows the number of generated bits according to BO offset and the value of λ in the proposed hardware structure.

**Table 6:** Number of occurrences and λ value according to BO offset

| Offset | R | Sign Bit | λ |
|---|---|---|---|
| ±7 | 7 | 1 | 394 |
| ±6 | 7 | 1 | 394 |
| ±5 | 6 | 1 | 345 |
| ±4 | 5 | 1 | 295 |
| ±3 | 4 | 1 | 246 |
| ±2 | 3 | 1 | 197 |
| ±1 | 2 | 1 | 148 |
| 0 | 1 | 0 | 49 |

### 3.3. In-Loop Filter Scheduler

The proposed in - loop filter scheduler hardware uses a 32x32 block size buffer considering the processing unit and operation cycle of deblocking filter and SAO. Figure 15 shows the

deblocking filter and SAO's scheduler.

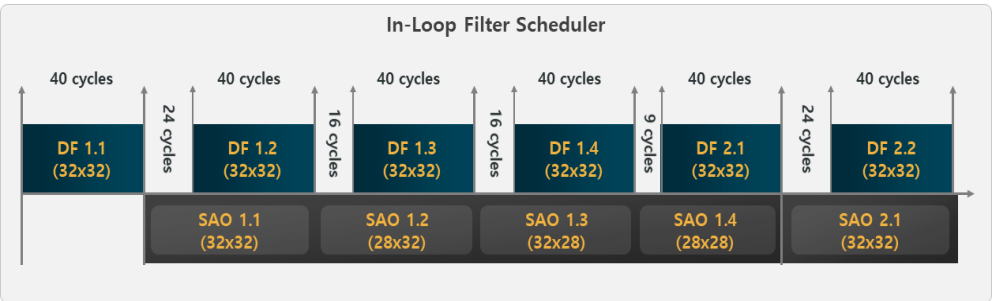


**Figure 15:** Scheduling of in-loop filters

The deblocking filter performs filtering in units of 32x32 blocks, and takes a total of 40 cycles. On the other hand, SAO performs filtering in units of 64x64 block size, and takes a total of 226 cycles. SAO operates after 32x32 blocks are filtered by the deblocking filter. The deblocking filter filters the next 32x32 block after 24th, 16th, and 9th cycles as shown in Figure 15. Because SAO processes in pixels, more cycles are needed than deblocking filters. The in-loop filter scheduler hardware applies the deblocking filter order as shown in Figure 16 for the deblocking filter and SAO processing unit. When the deblocking filter operates in the order of the pictures' refresh rate, SAO can be performed only after the first line ends and the deblocking filtering of the first 32x32 block of the second line is completed. Therefore, the deblocking filter performs filtering by the zig-zag scan method in consideration of the processing unit of the SAO.

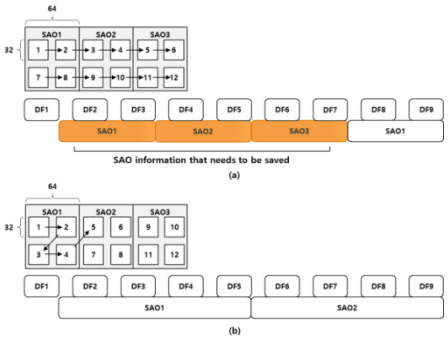


**Figure 16:** SAO processing sequence according to deblocking filtering order

## 4. Implementation Result

The in-loop filter hardware architecture proposed in this paper was designed as Verilog HDL and input data and output data were generated using hardware standard HM-16.9 for hardware verification.

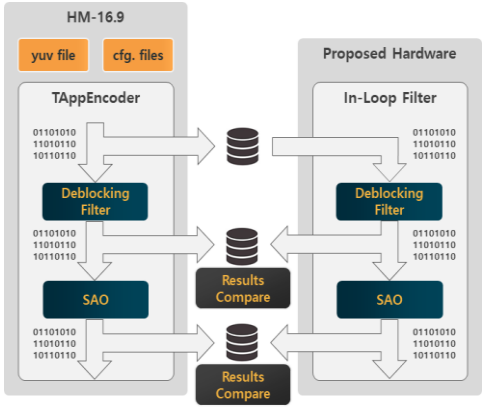


**Figure 17:** Verification method of proposed hardware structure

Category Figure 17 shows HM-16.9 and hardware verification method. The input data generated by HM-16.9 is used as in-loop filter hardware input, and the output data generated by HM-16.9 is compared with the data output through hardware simulation. Table 7 shows the verification environment and the tools used.

**Table 7:** Verification environment and usage tool

| Environment/Tool | | Specification |
|---|---|---|
| PC | CPU | Intel Core i7-4770 |
| | RAM | 8 GB |
| SW Debugger | Microsoft | Visual Studio 2017 |
| HW Simulation | Mentor Graphics | ModelSim SE-64 10.1c |
| ASIC | Design Compiler | syn_vL-2006.03-SP5-5 |
| | PrimeTime | pts_vM-2017.06 |

Due to the nature of the in-loop filter of the HEVC encoder, the deblocking filter and the SAO are separated into independent modules. Therefore, we compare the performance of the proposed In-loop filter hardware design by separating the deblocking filter and SAO. Also, DE (Design Efficiency) is applied for fair performance evaluation, and the formula of DE is equation (7). Format indicates the size of a supported image, and Fps indicates the number of frames that can be processed per second. The gate count represents the total number of gates of the hardware module.

$$DE = \frac{Format \times Fps}{Gate\ Count} \tag{7}$$

The proposed In-loop filter hardware architecture is based on ASIC synthesis using 90-nm cell library. In addition, the hardware structure of the proposed deblocking filter and SAO is synthesized at 260MHz with maximum operating frequency for smooth performance comparison. Proposed1 and Proposed2 represent the hardware synthesis results of the deblocking filter, Proposed3 and Proposed4 represent the hardware synthesis results of the SAO. Proposed1 and Proposed3 are synthesized with maximum operating frequency, Proposed2 and Proposed4 are synthesized with 260MHz operating frequency. The number of gates was calculated by dividing the hardware area results by the Synopsys Design Compiler tool using a 90nm cell library divided by the 2NAND gate area of the 90nm cell library. Previous paper [18] and [19] proposed a deblocking filter, and [21] and [22] proposed SAO. [20] also suggests both a deblocking filter and SAO

.

**Table 8:** Comparison of proposed deblocking filter hardware structure

| Deblocking Filter | | Proposed1 | Proposed2 | [18] | [19] | [20] |
|---|---|---|---|---|---|---|
| Process (nm) | | 90 | 90 | 180 | 90 | 65 |
| LCU Size | | 32x32 | 32x32 | 32x32 | 64x64 | 32x32 |
| Frequency (MHz) | | 1000 | 260 | 322 | 278 | 200 |
| Cycles/LCU | | 40 (64x64@160) | 40 (64x64@160) | 31 | 288 | 558 |
| Resolution | | 7680x4320 | 7680x4320 | 7680x4320 | 7680x4320 | 7680x4320 |
| Fps | | 771 | 200 | 320 | 123 | 40 |
| Gate Count (K) | Boundary judgment | 0.11 | 0.08 | No | 0.04 | No |
| | Bs calculation | 2.01 | 1.45 | - | 7.70 | - |
| | Filter | 61.78 | 44.62 | - | 23.88 | - |
| | Total | 114.49 | 46.15 | 865.00 | 31.62 | 30.30 |
| Memory (Kbyte) | | 0.60 | 0.60 | - | 8.50 | 4.80 |
| DE ($x10^3$) | | 400.37 | 143.78 | 12.17 | 129.06 | 43.80 |

Table 8 shows the comparison of the proposed deblocking filter hardware structure. [18] performs a 36x36 block-based filter for a 32x32 block boundary filter, and performs parallel vertical and horizontal filtering by dividing a 36x36 block into 4x36 block sizes. It also did not include a boundary judgment module to identify the boundary between PU and TU. The architecture proposed in [18] has high throughput because it performs 36x36 block-based parallel filtering, and the number of gates of the proposed hardware structure has high results due to parallel filtering. The proposed Proposed2 result and [18] are compared with DE, and Proposed2 has a high efficiency of 1071.4%. [19] minimizes the number of external memory accesses using internal memory, and treats conditional statements used for boundary prediction, filtering strength, and filtering as a parallel structure. It also has high throughput using four filtering modules. The deblocking filter proposed in this paper has a structure that uses less memory than [19], and has a 11.4% higher efficiency compared to the Proposed2 result and DE [19].The hardware architecture of the proposed deblocking filter proposed a 32x32 block-based memory configuration to solve the dependency between vertical and horizontal filtering, and has a high throughput with a 4-stage pipeline structure. The internal memory used in [20]'s hardware structure is 4.8 KB, and external memory is used to minimize the hardware area. However, the size of the external memory used was not included in the results. As a result of comparing the Proposed2 results [20] proposed in this paper with DE, Proposed2 has a high efficiency of 228.2%.

**Table 9:** Comparison of proposed SAO hardware structure

| Deblocking Filter | | Proposed3 | Proposed4 | [20] | [21] | [22] |
|---|---|---|---|---|---|---|
| Process (nm) | | 90 | 90 | 65 | 28 | 40 |
| LCU Size | | 64x64 | 64x64 | 32x32 | 64x64 | 64x64 |
| Frequency (MHz) | | 970 | 260 | 200 | 266 | 1300/217 |
| Cycles/LCU | | 266 | 266 | 558 | 1600 | 905/40 |
| Resolution | | 7680x4320 | 7680x4320 | 7680x4320 | 3840x2160 | 7680x4320 |
| Fps | | 450 | 120 | 40 | 82 | 120 |
| Gate Count (K) | Statistics collection | 98.43 | 83.71 | 55.90 | - | 30.00 |
| | Mode decision | 59.07 | 50.23 | 17.10 | - | 21.00 |
| | Total | 157.50 | 133.94 | 73.00 | 300.00 | 51.00 |
| Memory (Kbyte) | | - | - | 4.80 | - | 1.14 |
| DE ($x10^3$) | | 94.79 | 29.72 | 18.18 | 2.27 | 78.06 |

Table 9 shows the comparison of proposed SAO hardware structure. The proposed SAO hardware structure has a divided adder structure considering the deblocking filtering process step and is composed of a 2-stage pipeline. Proposed 4 results are compared with DE [20], and Proposed 2 shows 63.5% higher efficiency. [21] is designed as a 3-stage pipeline for high-performance SAO hardware architecture and performs 4x4 block-based processing. It also includes a parallel offset decision structure and a distortion calculation structure. The 4x4 block-based processing and parallel offset and distortion operation architecture of [21] requires high hardware area and does not include memory. Compared to Proposed 4 and DE [21], Proposed 4 has 1211.1% higher efficiency. The proposed hardware structure is divided into a Statistics Collection (SC) module and a Parameter Decision (PD) module. The SC module that requires pixel processing uses a 1.3 GHz operating frequency and the PD module uses a 217 MHz operating frequency It is a structure to use. SC modules require 905 cycles to process a single 64x64 LCU, and PD modules require 40 cycles. The BO of [22] minimizes the hardware area by using only 8 bands instead of 32 bands, and has an average BDrate reduction of -10.17% compared with HM-16.0 at 8k image size. Proposed 4 proposed in this paper is synthesized at operating frequency of 970MHz and the efficiency of DE is compared with that of [22].

# 5. Conclusion

In this paper, we describe efficient in-loop filter hardware design for high performance HEVC encoders. The in-loop filter of HEVC improves subjective image quality and encoding efficiency, but requires additional computation, resulting in an overhead for computational complexity and memory access. The in-loop filter time of the HEVC accounts for 3.5% to 23.5%.Therefore, this paper proposes an efficient in-loop filter hardware architecture for high performance HEVC encoders. The proposed in-loop filter hardware structure consists of a deblocking filter to remove block degradation, SAO to remove ringing inside the block, deblocking filter and in-loop filter scheduler for SAO pipeline structure. The proposed deblocking filter has a line buffer structure to reduce the number of external memory accesses, and it can be processed with a minimum line buffer by applying a routing technique. In addition, for the low power design, the internal block of the deblocking filter is designed as a clock gating structure, and the filtering module is designed with a 4-stage pipeline structure and has high throughput. In order to reduce the memory usage in the in-loop filter, the jig-jag scanning method is used for the deblocking filtering order. The hardware structure of the proposed SAO performs 4x4 block-based operation for deblocking filter and parallel processing, and minimizes the hardware area by implementing the multiplier required for offset and RDO

operations with shift and adder. In addition, the redundant arithmetic expressions are designed in such a structure that they can be processed in parallel to share computation results. Finally, the in-loop filter scheduler uses a 32x32 block buffer for the pipelining structure of deblocking filter and SAO. The proposed in-loop filter was designed with Verilog HDL and hardware verification was performed by storing the information required for deblocking filter operation and the filtered data in a file with HM-16.9, the HEVC standard software. SAO was also performed in the same manner. The software debugger was used to verify the hardware simulation waveforms. As a result, HM-16.9 and the proposed hardware show the same performance results. As a result of comparing the proposed in-loop filter hardware structure with the hardware structure proposed by the existing papers, the deblocking filter has a maximum efficiency of 1071.4% and a minimum efficiency of 11.4%, and SAO has a maximum of 1211.1% and a minimum of 21.4% Efficiency. The proposed in -loop filter hardware structure was synthesized by Synopsys' Design Compiler using a 90nm cell library at 260MHz and synthesized with 262.78K gates. It also supports real-time processing of 8K@120Fps at an operating frequency of 260MHz. As a result of synthesizing the proposed in-loop filter hardware structure at maximum operating frequency, deblocking filter supports real time processing of 8K@771Fps at 1GHz and SAO supports real time processing of 8K@450Fps at 970MHz. Therefore, real-time processing of deblocking filter 16K@192FPS, SAO 16K@112FPS is possible in the virtual reality and augmented reality contents service using the image size of 16K (15,360x8,640), and suggestion in the image size of 32K (30,720x17,280) Hardware architecture, the deblocking filter can support real-time processing of 32K@96Fps and SAO 32K@56Fps. Therefore, it is possible to use not only 8K image size but also 32K image size through the proposed in-loop filter hardware structure.

## Acknowledgment

This research was supported by the MSI (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (IITP-2017-0-01681) supervised by the IITP (Institute for Information and Communication Technology Promotion)

## References

[1] The Zettabyte Era: Trends and Analysis, Trend 4: Applications traffic growth [Internet]. CISCO; 2017 [updated 2017 Jun 7; cited 2018 Aug 27]. Available from: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html#_Toc484556821

[2] Wiegand T, Sullivan GJ, Bjontegaard G, Luthra A. Overview of the H.264/AVC Video Coding Standard. IEEE Transactions on Circuits and Systems for Video Technology. 2003 Jul;13(7):560-576.

[3] ITU- SG16WP3, ISO/IEC JCT1/SC29/WG11. High Efficiency Video Coding (HEVC) Text Specification Draft 10 (for FDIS & Content). JCTVC document L1003. Geneva; 2013. Available from: http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=7243

[4] Sullivan GJ, Ohm J, Han W. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology. 2012 Dec;22(12):1649-68.

[5] Souza DF, Ilic A, Roma N, Sousa L. HEVC In-Loop Filters GPU Parallelization in Embedded Systems. In: Proceedings of the 2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XV); 2015 Jul 19-23; Samos, Greece. IEEE; 2015 [cited 2018 Aug 27]; p. 123-30. Available from: IEEE Xplore.

[6] Bossen F, Bross B, Suhring K, Flynn D. HEVC Complexity and Implementation Analysis. IEEE Transactions on Circuits and Systems for Video Technology. 2012 Dec;22(12):1685-96.

[7] Ozcan E, Adibelli Y, Hamzaoglu I. A High Performance Deblocking Filter Hardware for High Efficiency Video Coding. IEEE Transactions on Consumer Electronics. 2013 Aug;59(3):714-20.

[8] Choi Y, Joo J. Exploration of Practical HEVC/H.265 Sample Adaptive Offset Encoding Policies. IEEE Signal Processing Letters. 2015 Apr;22(4):465-8.

[9] Hautala I, Boutellier J, Hannuksela J, Silven O. Programmable Low-Power Multicore Coprocessor Architecture for HEVC/H.265 In-Loop Filtering. IEEE Transactions on Circuits and Systems for Video Technology. 2015 Jul;25(7) 1217-30.

[10] Srinivasarao BKN, Chakrabarty I, Ahmad MN. High-Speed Low-Power Very-Large-Scale Integration Architecture for Dual-Standard Deblocking Filter. IET Circuits, Devices & Systems. 2015 Sept;9(5):377-83.

[11] Abeydeera M, Karunaratne M, Karunaratne G, Silva KD, Pasqual A. 4K Real-Time HEVC Decoder on an FPGA. IEEE Transactions on Circuits and Systems for Video Technology. 2016 Jan;26(1):236-49.

[12] Hsu P, Shen C. The VLSI Architecture of a Highly Efficient Deblocking Filter for HEVC Systems. IEEE Transactions on Circuits and Systems for Video Technology. 2017 May;27(5):1091-103.

[13] Baldev S, Shukla K, Gogoi S, Rathore PK, Peesapati R. Design and Implementation of Efficient Streaming Deblocking and SAO Filter for HEVC Decoder. IEEE Transactions on Consumer Electronics. 2018 Feb;64(1):127-135.

[14] Kim H, Ko J, Park S. An Efficient Architecture of In-Loop Filters for Multicore Scalable HEVC Hardware Decoders. IEEE Transactions on Multimedia. 2018 Apr;20(4):810-24.

[15] Zhang Y, Shen T, Ji X, Zhang Y, Xiong R. Residual Highway Convolutional Neural Networks for In-Loop Filtering in HEVC. IEEE Transactions on Image Processing. 2018 Aug;27(8):3827-41.

[16] Norkin A, Bjontegaard G, Fuldseth A, Narroschke M, Ikeda M, Andersson K, et al. HEVC Deblocking Filter," IEEE Transactions on Circuits and Systems for Video Technology. 2012 Dec;22(12):1746-54.

[17] Fu C, Alshina E, Alshin A, Huang Y, Chen C, Tsai C, et al. Sample Adaptive Offset in the HEVC Standard. EEE Transactions on Circuits and Systems for Video Technology. 2012 Dec;22(12):1755-64.

[18] Peesapati R, Das S, Baldev S, Ahamed SR. Design of Streaming Deblocking Filter for HEVC Decoder. IEEE Transactions on Consumer Electronics. 2017 Aug;63(3):1-9.

[19] Zhou W, Zhang J, Zhou X, Liu Z, Liu X. A High-Throughput and Multi-Parallel VLSI Architecture for HEVC Deblocking Filter. IEEE Transactions on Multimedia. 2016 Jun;18(6):1034-47.

[20] Shen W, Fan Y, Bai Y, Huang L, Shang Q, Liu C, et al. A Combined Deblocking Filter and SAO Hardware Architecture for HEVC. IEEE Transactions on Multimedia. 2016 Jun;18(6):1022-33.

[21] Mody M, Garud H, Nagori S, Mandal DK. High throughput VLSI Architecture for HEVC SAO Encoding for Ultra HDTV. In: 2014 IEEE International Symposium on Circuits and Systems (ISCAS); 2014 Jun 1-5; Melbourne VIC, Australia. IEEE; 2014 [cited 2018 Aug 27]; p. 2620-3. Available from: IEEE Xplore.

[22] Zhou J, Zhou D, Wang S, Zhang S, Yoshimura T, Goto S. A Dual-Clock VLSI Design of H.265 Sample Adaptive Offset Estimation for 8k Ultra-HD TV Encoding. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2017 Feb;25(2):714-24

.