A Combined Deblocking Filter and SAO Hardware Architecture for HEVC

Weiwei Shen, Yibo Fan, Yufeng Bai, Leilei Huang, Qing Shang, Cong Liu, and Xiaoyang Zeng, *Member, IEEE*

***Abstract*—The latest video coding standard high-efficiency video coding (HEVC) provides 50% improvement in coding efficiency compared to H.264/AVC to meet the rising demands for video streaming, better video quality, and higher resolution. The deblocking filter (DF) and sample adaptive offset (SAO) play an important role in the HEVC encoder, and the SAO is newly adopted in HEVC. Due to the high throughput requirement in the video encoder, design challenges such as data dependence, external memory traffic, and on-chip memory area become even more critical. To solve these problems, we first propose an interlacing memory organization on the basis of quarter-LCU to resolve the data dependence between vertical and horizontal filtering of DF. The on-chip SRAM area is also reduced to about 25% on the basis of quarter-LCU scheme without throughput loss. We also propose a simplified bitrate estimation method of rate-distortion cost calculation to reduce the computational complexity in the mode decision of SAO. Our proposed hardware architecture of combined DF and SAO is designed for the HEVC intraencoder, and the proposed simplified bitrate estimation method of SAO can be applied to both intra- and intercoding. As a result, our design can support ultrahigh definition 7680 × 4320 at 40 f/s applications at merely 182 MHz working frequency. Total logic gate count is 103.3 K in 65 nm CMOS process.**

***Index Terms*—Deblocking filter (DF), hardware implementation, high-efficiency video coding (HEVC), sample adaptive offset (SAO), UHD.**

I. Introduction

ITU-T video Coding Experts Group and ISO/IEC Moving Picture Experts Group formed a Joint Collaborative Team on Video Coding in 2010, and the next generation coding standard, High Efficiency Video Coding (HEVC), is now being developed. HEVC aims to reduce 50% bit rate in comparison with the existing H.264/AVC high profile, under the same visual quality [1].

As the previous video coding standard H.264/AVC, HEVC adopts the block based hybrid coding framework [2], [3]. A quadtree based coding structure is an important feature of the HEVC standard.

Manuscript received February 11, 2014; revised May 20, 2014, October 30, 2014, February 01, 2015, May 08, 2015, and August 24, 2015; accepted February 06, 2016. Date of publication February 19, 2016; date of current version May 13, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61306023. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Shahram Shirani.

The authors are with the State Key Laboratory of ASIC and System, Fudan University, Shanghai 201203, China (e-mail: 10110720024@fudan.edu.cn; fanyibo@fudan.edu.cn; 12212020001@fudan.edu.cn; 10300720005@fudan.edu.cn; 11212020039@fudan.edu.cn; 11212020033@fudan.edu.cn; xyzeng@fudan.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2016.2532606

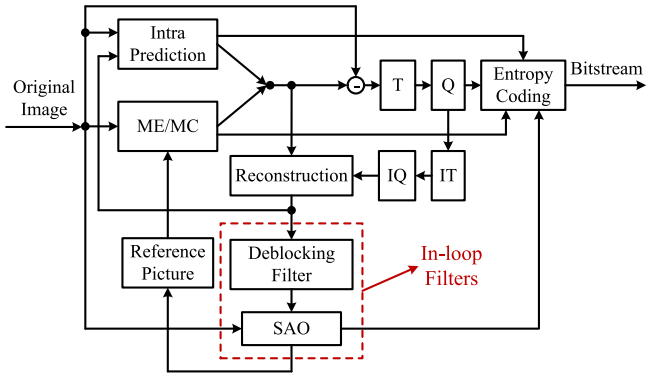


Fig. 1. Block diagram of the HM9.0 encoder.

Several new coding structures have been introduced in HEVC: coding unit (CU), prediction unit (PU) and transform unit (TU). CU is the basic unit of region splitting used for intra/inter coding [4]. It can be split from the largest coding unit (LCU, can be as large as 64 × 64 pixels) into the smallest coding unit (SCU, 8 × 8 pixels). Coupled with CU, PU carries the information related to the prediction processes. TU is used for transform and quantization, and it depends on PU partitioning modes.

Blocking effect is known as one of the most visual and objectionable artifacts of block-based compression methods [5]. Artifacts that are commonly seen in prior video coding standards at medium and low bitrates, such as blocking artifacts, ringing artifacts, color biases and blurring artifacts [6], may still exist in HEVC. Also HEVC adopts in-loop filters in order to reduce these artifacts. The HEVC defines two in-loop filters, shown in Fig. 1. In addition to the DF similar to the one in H.264/AVC, HEVC further introduces a completely new tool: sample adaptive offset (SAO). Meanwhile, the DF leads to 1.3-3.3% BD-rate reduction [7] on average, and the SAO achieves 3.5% BD-rate reduction [8] at the same quality. And the BD-rate calculation is introduced in [15].

The coding efficiency of HEVC comes with a cost. The computational complexity of HEVC is very high. From an encoder perspective, an encoder fully exploiting the capabilities of HEVC is expected to be several times more complex than an H.264/AVC encoder. To understand the computational complexity of the HEVC, a study mapped the HEVC codec into existing systems. In [9], authors mapped the HM (HEVC Test Model) encoder into a cluster containing Xeon-based servers (E5670 clocked at 2.93 GHz) and using gcc 4.4.5. Even for intra-only case, the encoding time at least exceeds 1000 times real-time (sequences' resolution is 832 × 480 at 30 f/s).

Due to an increasing diversity of services, the growing popularity of HD video, and the emergence of beyond-HD formats (e.g., 4 K × 2 K or 8 K × 4 K resolution) are creating even stronger needs for high throughput in hardware implementation superior to H.264/AVC. Hence, hardware realization of the HEVC standard for real-time applications is an essential and challenging task.

There are some previous works on the topic of in-loop filters. In [18], a five-stage pipelined and hybrid edge filtering sequence are applied; in [19], a five-stage pipelined and re-source-shared dual-edge filter to generate two filtering results every cycle is proposed; in [23], a parallelized scheme of processing the luminance and chrominance samples simultaneously is proposed. But all these works do not support HEVC in-loop filters. In [10], an HEVC in-loop filters architecture composed of fully utilized DF and SAO is proposed. But it does not support HEVC encoder. In [24], a hybrid pipeline with two processing levels is proposed for HEVC DF, which uses one 1-D filter and single-port on-chip SRAM. In [25], two parallel datapaths are used for the design of HEVC DF in order to increase its performance. Here, a combined DF and SAO hardware architecture for HEVC encoder is proposed.

A. Motivation

Among many algorithms adopted by HEVC, two in-loop filters' algorithm (DF and SAO) requires significant CPU times. According to Bossen *et al.* [9], the DF and SAO cost about one-fifth CPU times of the whole HEVC codec, which are two complicated modules in HEVC.

Moreover, beyond-HD format's applications become more and more popular, which are at the expense of large external traffic. For example, the DF and SAO require the demand of 0.93, 0.58 Gb/s I/O bandwidth respectively at an application of 2 K × 1 K at 30 f/s (2560 × 1600), without any data-reuse scheme. Thus a high throughput hardware architecture for DF and SAO becomes a critical issue in HEVC codec design.

B. Design Approach

Due to an increasing demand of high resolution applications, the high data access between on-chip memory and external memory becomes even more critical. Considering the trade-offs between on-chip memory area and external memory traffic, we present an interlaced pipeline to combine the DF with SAO on a quarter-LCU basis; the quarter-LCU is defined as a 32 × 32 pixels' block. In the process of DF, a novel filter is suggested in order to keep the same result on a picture basis based on the quarter-LCU structure. Meanwhile, we also propose an interlacing memory scheme to arrange the data in on-chip memory, and access the data in the process of both vertical and horizontal filtering efficiently in the DF phase.

In the process of statistics collection in SAO, the overall number of comparators is reduced by 83% with our proposed configurable comparator array. We also present a fragmentation adder scheme to balance the computational burden between pipeline stages of SAO. Meanwhile, a simplified bitrate estimation method of rate-distortion cost calculation is adopted to reduce the computational complexity in the mode decision of SAO.

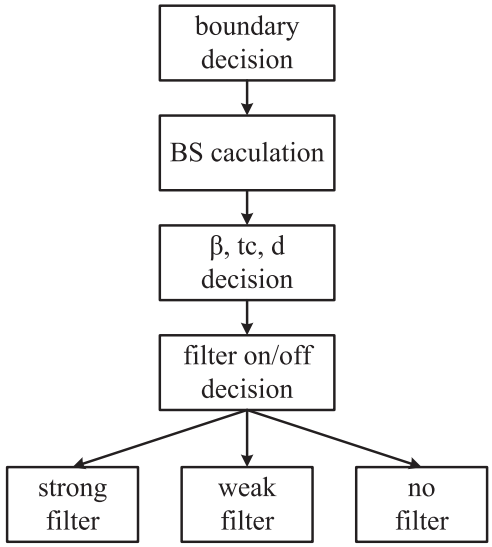


Fig. 2. Overall processing flow of deblocking filter.

With these design approaches, a LCU can be filtered in 558 cycles, and our design can support UHD (7680 × 4320) at 40 f/s applications at merely 182 MHz working frequency.

C. Organization of the Paper

The rest of this paper is organized as follows. Section II introduces the DF and SAO algorithm in HEVC. In Section III, our proposed hardware architecture is described. Section IV formulates the simplified bitrate estimation method of rate-distortion cost calculation in the mode decision of SAO. The hardware implementation result and comparison are provided in Section V, followed by a conclusion in Section VI.

II. Deblocking Filter and SAO Algorithm in HEVC

In HEVC, two in-loop filters, the DF followed by SAO are applied to the reconstructed samples. The DF is intended to reduce the artifacts caused by block-based coding. While the DF is only applied to the samples located at PU or TU boundaries, the SAO is applied adaptively to all samples satisfying certain conditions, (e.g., based on gradient) [11], in order to reduce the mean sample distortion of a region.

A. Deblocking Filter Algorithm

The DF is applied to all samples adjacent to a PU or TU boundaries except the picture boundary. Unlike H.264/AVC, where the DF is applied on a 4 × 4 sample grid, HEVC only applies the DF to the PU or TU boundaries which are also aligned on an 8 × 8 sample grid.

Fig. 2 illustrates the overall processing flow of the DF [3]. Firstly, the decision is made that whether the current boundary is a boundary of PU or TU. If not, the filter isn't applied to the current boundary. Given that P block and Q block are two adjacent 4 × 4 blocks with the boundary involved in filtering,

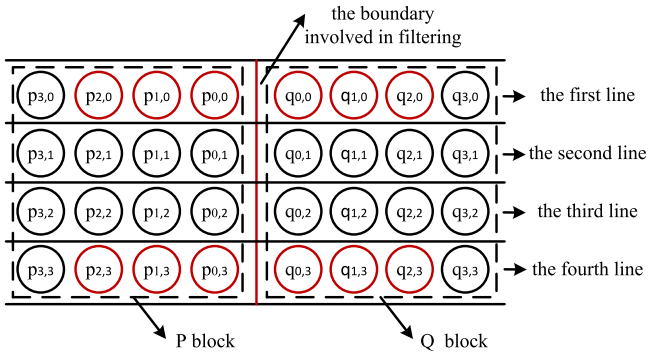


Fig. 3. Two adjacent 4 × 4 blocks.

TABLE I
DEFINITION OF BS VALUES

Conditions	BS value
At least one of the P or Q blocks is intra	2
At least one of the P or Q blocks has non-zero coded residual coefficient and the boundary is a TU boundary	1
Absolute differences between MVs between the P and Q blocks are >= 1 in units of integer pixels	1
Motion-compensated prediction for the P and Q blocks refers to different pictures or the number of MVs is different for the two blocks	1
Otherwise	0

shown in Fig. 3. The boundary strength (BS) reflects how strong the filter is needed for the boundary, which is controlled by several syntax elements. The BS value can take one of three possible values: 0, 1 and 2. The definition of the BS value is clearly shown in Table I. If the BS value is greater than zero, additional conditions are checked in order to determine whether the DF should be applied to the block boundary, and also make a selection on strong/weak filter.

According to the BS value and the quantization parameter (QP) of P and Q blocks, two thresholds, β and tc, are determined from pre-defined tables. Together with β and tc, the value d is used for filter on/off decision and strong/weak filter selection, which is derived from the value of twelve pixels in the first and the fourth line. These 12 pixels are labeled as red circles shown in Fig. 3. Note that this decision is shared across four lines (four rows for vertical boundary filtering, while four columns for horizontal boundary filtering). Meanwhile, each line has its respective values, such as *dE*, *dEp* and *dEq*, which are used for the decision of filter on/off, strong and weak filter for its own line

$$dE \neq 0,\ \text{BS} \neq 0 \tag{1}$$

$$\text{BS} > 1. \tag{2}$$

For example, the values of eight pixels across the boundary are denoted as $p_{3,0}$, $p_{2,0}$, $p_{1,0}$, $p_{0,0}$, $q_{0,0}$, $q_{1,0}$, $q_{2,0}$, $q_{3,0}$, which are labeled as the first line in Fig. 3. For the luminance component, the first line is filtered only when the condition of (1) is satisfied. For the chrominance component, the first line is filtered only when the condition of (2) is satisfied. The decisions of strong and weak filter are detailed in [3].

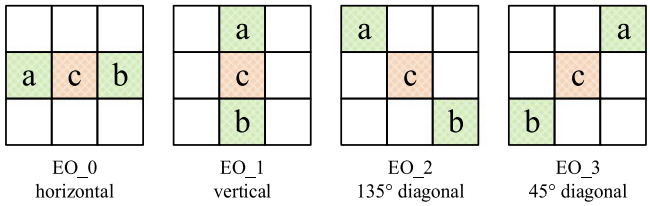


Fig. 4. Four 1-D directional patterns for EO classification.

TABLE II
SAMPLE CATEGORIES IN EO

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

B. SAO Algorithm

SAO is located after the DF and also belongs to in-loop filters, which modifies the samples after the DF. The concept of SAO is to classify reconstructed pixels into different categories, obtaining an offset for each category, and the adding to each sample of the category [7]. It is performed on LCU basis in HM 9.0 (reference software for HEVC).

Two SAO types are adopted in HEVC: Edge offset (EO) and band offset (BO). For EO, the sample classification is based on comparison between current samples and neighboring pixels. For BO, the sample classification is based on sample values [7]. Note that, each color component (Y, Cb and Cr) has its own SAO parameters. To reduce the information to be coded by entropy coding, the current LCU can reuse the SAO parameters from left LCU or upper LCU by SAO merging mode.

EO: EO uses four 1-D direction patterns for pixel classification including EO_0 class (horizontal), EO_1 class (vertical), EO_2 class (135° diagonal) and EO_3 class (45° diagonal), shown in Fig. 4. Sample labeled as "c" indicates a current sample to be considered. Two samples labeled as "a" and "b" specify two neighboring pixels. According to these patterns, four EO classes are specified, each class corresponds to one pattern. Please note that, only one EO class would be selected for each LCU that enables EO.

For a given EO class, each sample is classified into one of the five categories by comparing its own value with the two neighboring samples' value as shown in Table II. If the current sample does not belong to any of the category 1–4, then SAO would be not applied to it.

BO: BO performs the sample classification based on the sample's own value. The BO classifies all pixels of a LCU into multiple bands where each band contains the pixels in the same intensity interval in HM 9.0. The concept of band in BO is similar to the category in EO. The pixel intensity range is equally divided into 32 uniform bands from zero to the maximum value (e.g., 255 for 8-bit pixels), and each band has its offset, shown in Fig. 5.

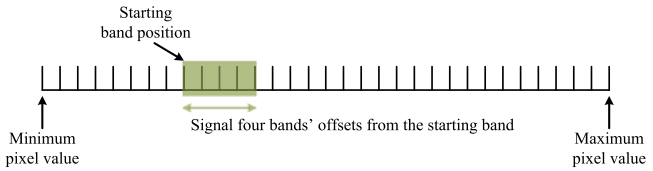


Fig. 5. Four bands are grouped together and signal from the starting band.

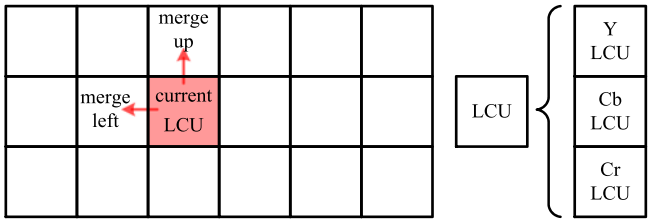


Fig. 6. SAO merging mode.

Only four offsets of four consecutive bands and the starting band position would be coded by entropy coding to be transmitted to the decoder. The possible choice number of starting band is 29. The reason of selecting only four bands is that the sample range in a region can be quite limited after the regions are reduced from picture quadtree partitions to CTBs [9].

SAO merging: The current LCU can reuse the SAO parameters (SAO type and four offsets) from left LCU (merge-left) or upper LCU (merge-up) as shown in Fig. 6. Please note that the SAO information (SAO type and offsets) are shared by Y, Cb and Cr components. If current LCU selects the merge-left or merge-up, all SAO information of current LCU are reused from the left or upper LCU. The SAO merging mode can reduce side information to be coded by entropy coding effectively.

Fast distortion estimation: The selection of SAO type and offsets is performed on fast distortion estimation method based on a region of LCU in HM 9.0. The fast distortion estimation method [8] implementation for SAO would need to add offsets to pre-SAO samples (post-DF samples) to generate post-SAO samples and then calculate distortion between original samples and post-SAO samples, which can be implemented as follows.

Let *k*, *p(k)* and *d(k)* be the pixel positions, original pixels and pre-SAO pixels, where *k* belongs to *C* and *C* is a set of pixels which belong to specified SAO type (EO, BO), a specified category of an EO class or a starting band in BO. The distortion between original pixels and pre-SAO pixels can be calculated in the following equation:

$$D_{\text{pre}} = \sum\nolimits_{k\in C} \left(p\left(k\right) - d(k)\right)^2. \tag{3}$$

The distortion between the original pixels and the post-SAO pixels can be described in the following equation. In (4), *O* is the offset of a given pixel set

$$D_{\text{post}} = \sum\nolimits_{k\in C} \left(p\left(k\right) - \left(d\left(k\right) + O\right)\right)^2. \tag{4}$$

The delta distortion is defined in the following equation:

$$\Delta D = D_{\text{post}} - D_{\text{pre}} = NO^2 - 2OE \tag{5}$$

$$E = \sum\nolimits_{k\in C} \left(p\left(k\right) - d(k)\right). \tag{6}$$

In (5), *N* is the number of pixels of a given set, and *E* is the sum of differences between original pixels and pre-SAO pixels as defined in (6). Next, the delta rate-distortion cost is defined in the following equation:

$$\Delta J = \Delta D + \lambda R. \tag{7}$$

In (7), λ is the Lagrange multiplier, and *R* represents the estimated bits of side information for a specified SAO type. The calculation of λ is performed by entropy coding which is a time-consuming process in HM 9.0. A simplified bitrate estimation method is proposed to reduce the computation complexity, which is elaborated in Section IV.

III. Proposed Hardware Architecture

Fig. 7 shows an overall proposed hardware architecture of combined DF and SAO. We adopt an interlaced pipeline architecture to speed up the combined DF and SAO. The whole process is partitioned into three phases: DF, SAO statistics collection and SAO mode decision.

The whole design is based on quarter-LCU (32 × 32 pixels for Y, 16 × 16 pixels for Cb and Cr) scheme which has the benefit of on-chip SRAM (which stores the reconstructed and original pixels) area reduction by about 75% compared to the LCU basis, without throughput loss. Owing to that the SAO is performed on a LCU basis in HM 9.0, a z-scan order (shown in Fig. 8) is proposed to keep the same results in picture level (from quarter-LCU_0 to quarter-LCU_3 in an LCU).

In DF phase, a four-stage pipeline is adopted to enhance the throughput on the basis of a four-line unit, while a two-stage pipeline is also adopted in SAO statistics collection phase. During the DF process, the statistics information for deriving offset and fast distortion estimation of SAO is collected after horizontal edges filtering. Consequently, an interlaced pipeline scheme for DF and SAO is proposed to avoid fetching any post-DF pixels from SRAM and improve the throughput significantly.

A. Analysis of Data-Reuse for the Combined DF and SAO

In order to clarify the advantage of the combined DF and SAO hardware architecture, an external memory access factor *Em* is defined to evaluate the efficiency of the external memory access for DF and SAO, the *Em* is expressed as (8) as shown at the bottom of the next page. where Em_r and Em_w represent the average number of memory accesses from and to the bus per 4 × 4 block, respectively. *Em* stands for the average number of external memory accesses per 4 × 4 block for DF and SAO. It is well known that the smaller *Em* has the greater reduction of the external memory bandwidth.

In the general hardware design, the DF and SAO are based on an LCU level, so the block size is set as a LCU size (64 × 64 pixels) according to (8). Because of the sixteen vertical boundaries and sixteen horizontal boundaries should be filtered by DF, the number of 4 × 4 block memory reading accesses is 867 (=16 × 17 + 2 × 8 × 9 + 17 × 17 + 2 × 9 × 9), and the number of 4 × 4 block memory writing accesses is 867 without data-reuse scheme. Moreover, the SAO requires both the post-DF and

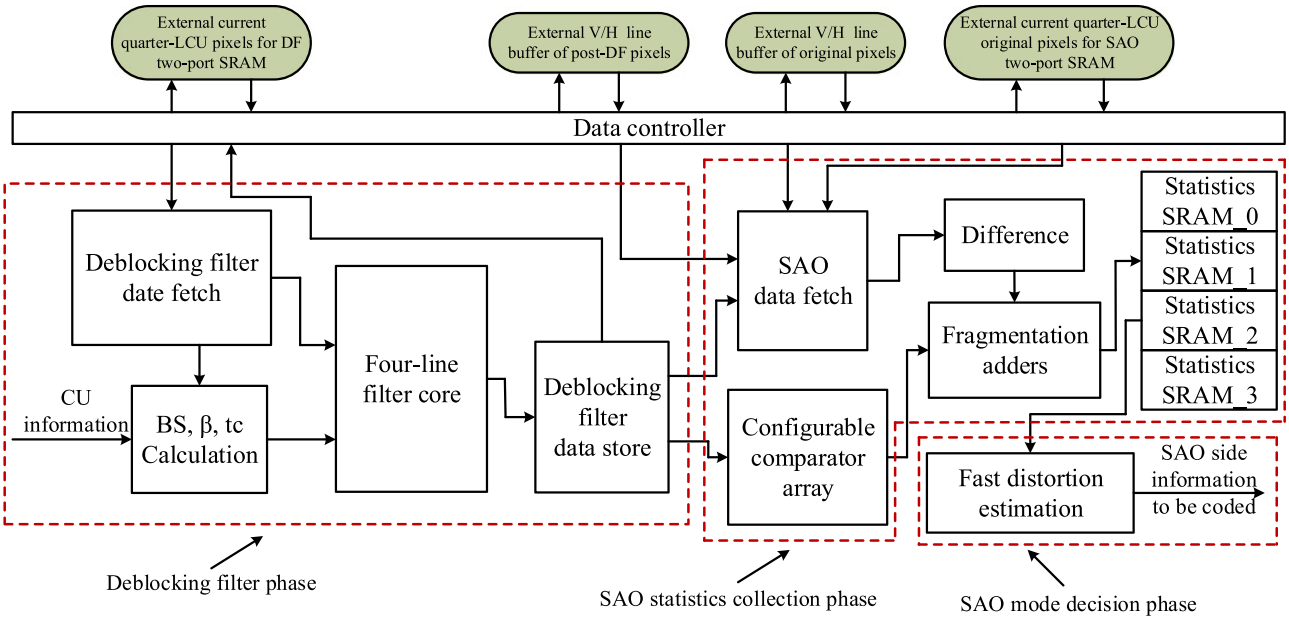


Fig. 7. Proposed hardware architecture of the combined deblocking filter and SAO.

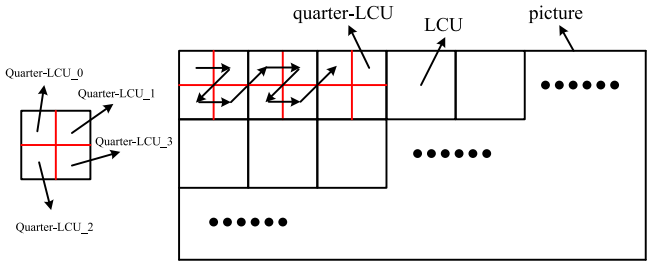


Fig. 8. Proposed z-scan order of quarter-LCU in a picture level.

original pixels, the number of 4 × 4 block memory reading accesses is 1048 (=2 × (17 × 17 + 2 × 9 × 9 + 73)), while the number of 4 × 4 block memory writing accesses is 0 (the SAO just send the selected type and offsets to the system) without data-reuse scheme. *Em* can be calculated as the following equation:

$$Em = \frac{(867 + 1048) + (867 + 0)}{16 \times 16 + 8 \times 8 + 8 \times 8} = 7.2. \tag{9}$$

It is observed that each pixel would be access about eight times to complete the DF and SAO, which causes a huge overhead of the external memory. In order to reduce the external memory bandwidth, the on-chip memory scheme is a well-known method to store the reuse data in chip. Here, a novel filter order scheme is proposed to reuse the temporary data after vertical boundary filtering for further horizontal boundary filtering in DF phase, cooperating with the proposed on-chip memory organization. Moreover, the combined DF and SAO hardware architecture can reuse the post-DF pixels for SAO statistics collection phase.

With the above methods, the block size is set as a quarter-LCU size (32 × 32 pixels) according to (8), at the bottom of the page. The number of 4 × 4 block memory reading accesses is 112 (=8 × 9 + 2 × 4 × 5), and the number of 4 × 4 block memory writing accesses is 16 (=8 + 2 × 4) of DF. Meanwhile, the SAO just reads the original pixels from the external memory, and the number of 4 × 4 block memory reading accesses is 112 (= 8 × 9 + 2 × 4 × 5). And *Em* for the above data-reuse methods can be expressed as

$$Em = \frac{(112 + 112) + (16 + 0)}{8 \times 8 + 4 \times 4 + 4 \times 4} = 2.5. \tag{10}$$

It means that the total number of 4 × 4 block external memory accesses dramatically reduces to 35%, which greatly eases the external memory traffic. Moreover, the combined DF and SAO hardware architecture can eliminate the SRAMs of storing

$$\begin{aligned} Em &= Em_r + Em_w \\ &= \frac{\text{the number of memory reading accesses from the bus of a block}}{\text{the number of } 4 \times 4 \text{ blocks of a block}} \\ &\quad + \frac{\text{the number of memory writing accesses to the bus of a block}}{\text{the number of } 4 \times 4 \text{ blocks of a block}} \\ &= \frac{\text{total number of memory accesses of a block}}{\text{the number of } 4 \times 4 \text{ blocks of a block}} \end{aligned} \tag{8}$$

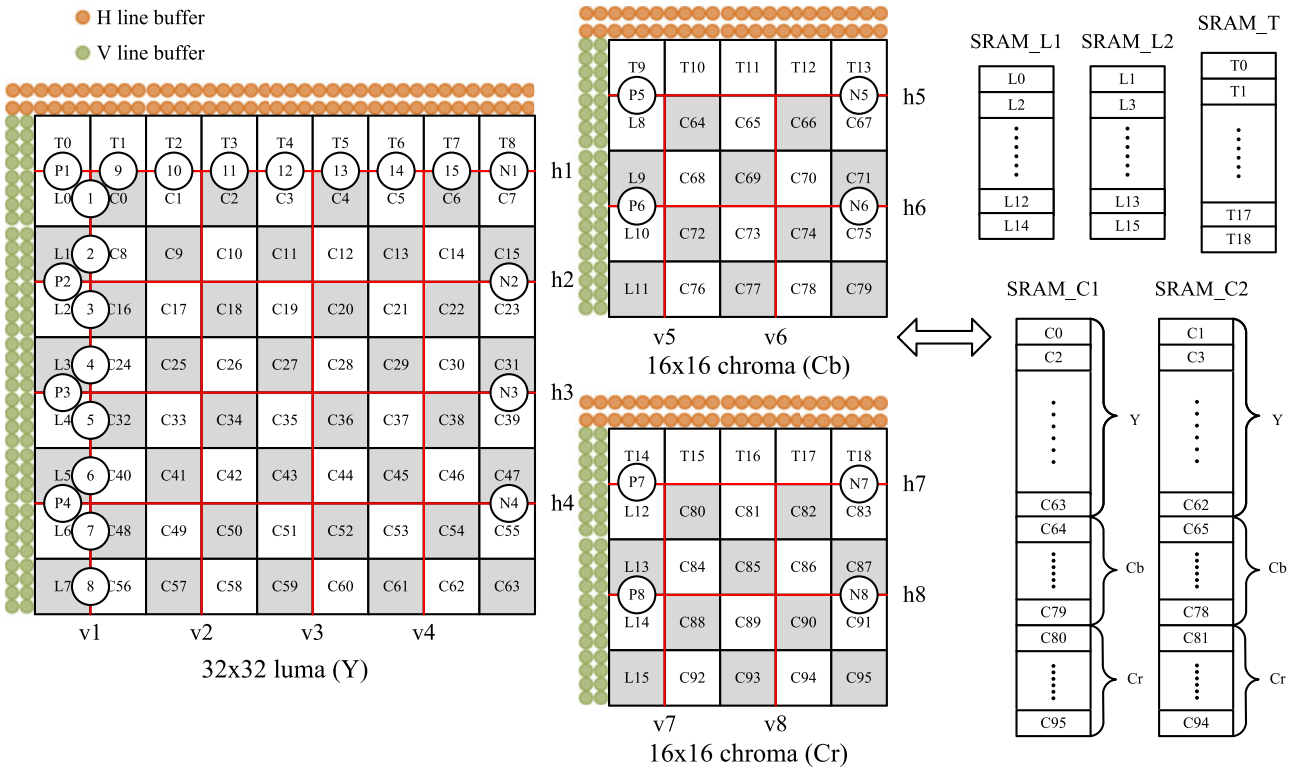


Fig. 9. Memory organization of reconstructed pixels.

post-DF pixels for SAO. The SAO can utilize the post-DF pixels directly without using SRAMs for storing the post-DF pixels.

B. Bandwidth Utilization Discussion

The external memory access factor *Em* is a theoretical number to evaluate the efficiency of the external memory access. Furthermore, the bandwidth utilization of external memory is an extremely important issue in a real video encoder design, especially for DRAM.

DRAM, such as SDRAM or DDR1/2/3, is widely used as external memory, which prefers block-based access. In other words, if the desired data is tiny and fractional distributed in external memory, then the pre-fetched data mechanism of DDR would be wasted, resulting in a very low utilization of external memory bandwidth. The improvements of the DRAM bandwidth utilization should be a serious concern in a real video encoder design. Thus, great efforts should be put on how to join fractional data into continuous data chain stored in external memory.

In our external memory system, we use DDR2 (data width is 32 bits, burst length is 4 equal to a 4×4 block one access time), and the pixels in DDR2 are ordered as a quarter-LCU as a big chain, in which the pixels are ordered as 4×4 blocks. With our proposed data-reuse scheme, we only access the whole current quarter-LCU and a row of 4×4 blocks from the neighboring top quarter-LCU. And the data in on-chip SRAMs are stored with 4×4 blocks as a basic unit. So we always access consecutive integral multiple of 4×4 blocks' data, and there is no overhead in external memory access.

C. Novel Filter Order Scheme

In this section, three filter order levels, named picture level, boundary level and four-line unit level, are defined to describe our proposed filter order.

1) Picture level filter order: The vertical edges in a reconstructed picture are filtered first, and then the horizontal edges are filtered with samples modified by DF of vertical edges as input. The vertical edges are filtered starting with the edge on the left-hand through the edges towards the right-hand in their geometrical order. The horizontal edges are filtered starting with the edge on the top-hand through the edges towards the bottom-hand in their geometrical order. This is the basic filter order principle of DF in HEVC [3].

2) Boundary level filter order: The boundaries involved in filtering on a quarter-LCU basis are labeled as red lines in Fig. 9. The vertical boundaries including {v1, v2, v3, v4} are filtered from left to right (if any), and the horizontal boundaries including {h1, h2, h3, h4} are filtered from top to bottom (if any) for component Y. And the filter order of component Cb and Cr are similar to the one of component Y.

3) Four-line unit level filter order: Given that the four-line unit is composed of two adjacent 4×4 blocks in Fig. 3. Each boundary consists of several four-line units. The vertical boundaries are filtered from top four-line unit to the bottom four-line unit (e.g., from four-line unit 1 to four-line unit 8 in vertical boundary v1). The horizontal boundaries are filtered from left four-line unit to the right four-line unit (e.g., from four-line unit 9 to four-line unit 15 in horizontal boundary h1).

For our quarter-LCU structure, we adopt the filter order scheme combining the boundary level filter order with the four

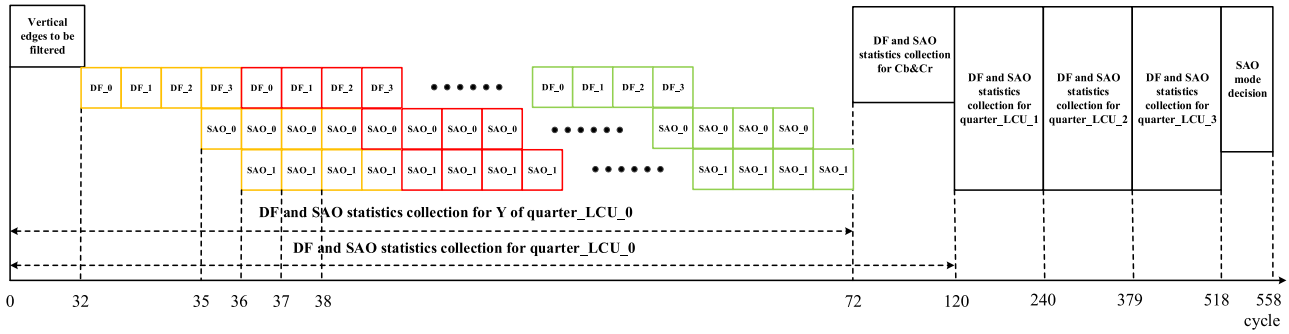


Fig. 10. Proposed timing diagram of an LCU.

line level filter order in a quarter-LCU. Firstly, vertical boundaries are filtered from v1 to v4, then from h1 to h4 for component Y (the process for Cb, Cr is similar to Y). And each boundary obeys the four-line unit level filter order. Note that, four-line unit {N1, N2, N3, N4, N5, N6, N7, N8} would not be filtered in current quarter-LCU by explicitly setting the BS value to 0, it is expected that the current quarter-LCU is on the right boundary of the entire picture. These lines would be filtered in right quarter-LCU which is on the right of current quarter-LCU, corresponding to {P1, P2, P3, P4, P5, P6, P7, P8} in the right quarter-LCU to keep the same result in HM 9.0.

Owing to that the SAO is performed on LCU basis in HM 9.0, a z-scan order (shown in Fig. 8) is proposed to keep the same results in picture level (from quarter-LCU_0 to quarter-LCU_3 in a LCU). Once the DF and SAO statistics collection phase is completed of quarter-LCU_0, quarter-LCU_1, quarter-LCU_2 and quarter-LCU_3 (shown in Fig. 10), the SAO mode decision phase for a LCU would be invoked.

D. Proposed Memory Organization

An on-chip memory is used to reduce the I/O bandwidth between the chip and the system. The main challenge is to properly arrange the data in memory modules in order to access the data smoothly from the memory modules to the in-loop filter core, on both vertical and horizontal boundaries. We present an interlacing memory organization to access the data effectively in the processing of both vertical and horizontal filtering for the DF.

Our approach is to divide the on-chip memory of reconstructed pixels (used for the DF) into five modules (SRAM_C1, SRAM_C2, SRAM_L1, SRAM_L2, and SRAM_T), shown in Fig. 9. SRAM_C1 and SRAM_C2 store the pixels of the current quarter-LCU; those in SRAM_L1 and SRAM_L2 are from the left neighboring quarter-LCU. Meanwhile, the pixels in SRAM_T come from the top neighboring quarter-LCU and left-top neighboring quarter-LCU. Each square in Fig. 9 stands for a 4×4 pixels of luminance or chrominance component. For example, the grey squares from C0 to C95 belong to SRAM_C1, while the white squares from C1 to C94 belong to SRAM_C2. With our proposed interlacing memory organization, two 4×4 pixel blocks on both sides of every boundary always come from different memory modules. As a result, all pixels can be easily accessed from the SRAMs on the vertical and horizontal filtering operations.

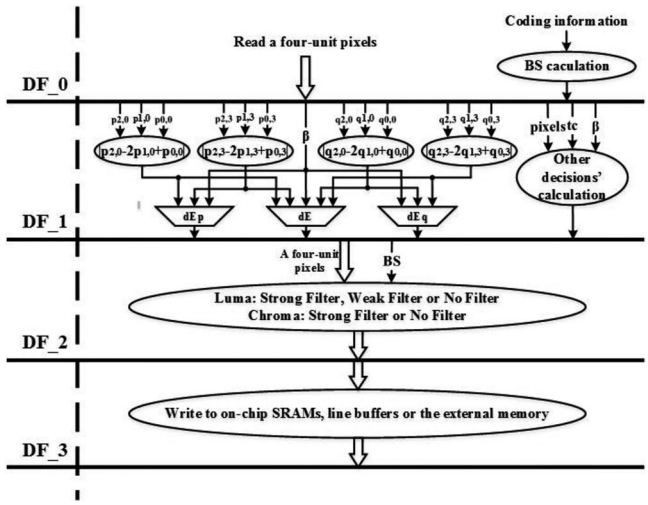


Fig. 11. Four-stage pipeline for DF.

A two-port SRAM can support one read and one write at the same clock cycle. In our proposed design, we utilize five two-port SRAMs for reconstructed pixels. These SRAMs can also be used to store the immediate pixels for horizontal edges filtering after vertical edges filtering. Therefore, no more on-chip memories are needed for the DF.

Due to the directional patterns of EO in SAO statistics collection phase, another V and H line buffers are needed to store some post-DF pixels for the category selection (shown in Fig. 9, each yellow and green circle stand for a pixel). Another two SRAMs (SRAM_O1, SRAM_O2) and V, H line buffers are also adopted to store the original pixels for SAO statistics collection. The organization of these memories is similar to the memory scheme of the reconstructed pixels in DF.

With the proposed memory organization based on quarter-LCU structure, the on-chip memory area would be reduced to about 25% compared to the basis of LCU.

E. Pipeline Schedule

Fig. 10 shows the timing diagram of our design. We employ a four-stage pipeline which can filter four lines simultaneously in DF phase (shown in Fig. 7). The function of each stage is illustrated as follows, depicted in Fig. 11.

Stage 0 (DF_0): Calculate the BS value according to the information such as the prediction mode, motion vector

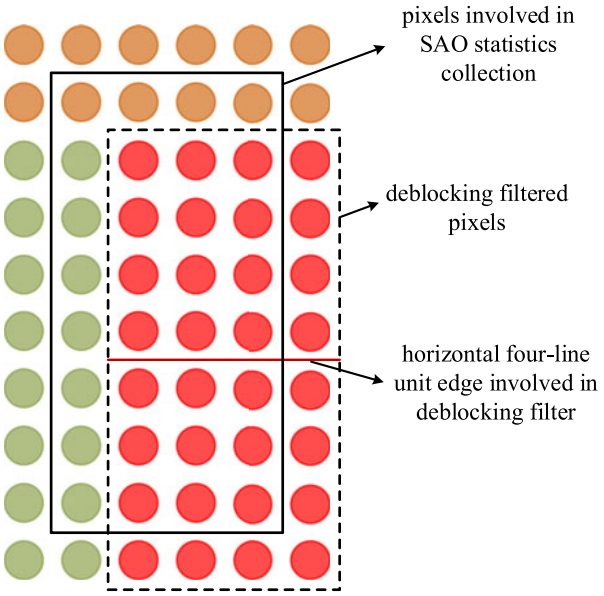


Fig. 12. Proposed pixels' selection of the combined deblocking filter and SAO.

and so on. Reading the pixels to be filtered from on-chip SRAM (SRAM_C1, SRAM_C2, SRAM_L1, SRAM_L2, and SRAM_T) is also accomplished in this stage.

Stage 1(DF_1): Calculate the threshold value β, tc, the condition dE, dEp, dEq used for the decision of filter on/off, strong and weak filter.

Stage 2(DF_2): According to the BS, tc, dE, dEp, dEq obtained from the above stage, different taps are applied to filter the pixels across the current boundary. And the detailed filtering process of both the strong filter and the weak filter can be found in HEVC [3].

Stage 3(DF_3): Write the filtered data back to on-chip SRAM to be further filtered, or to the external memory to be referenced for the other modules of the video encoder.

The pre-DF pixels would be accessed and modified both in filtering on vertical edges and horizontal edges in DF phase. These pixels after filtering on horizontal edges would be fetched by SAO statistics collection phase.

As shown in Fig. 12, because of that EO should refer its neighboring pixels to obtain category selection; only 32 pixels' information is collected inside the region marked with solid-line. These pixels inside the dash-line region are filtered by DF. The pixels of yellow circles come from H line buffer of post-DF, while the pixels of green circles come from V line buffer.

Owing to that all these 32 pixels' information would be collected by EO_0, EO_1, EO_2, EO_3 and BO, an interlaced pipeline scheme, coupled with a configurable two-stage SAO pipeline, is proposed to reduce the chip area of SAO statistics collection phase to about 25%. As shown in Fig. 10, each pipeline stage of DF and SAO takes one cycle. The pipeline of DF would be invoked at the interval of four cycles in order to wait until all 32 pixels' information statistics are collected by

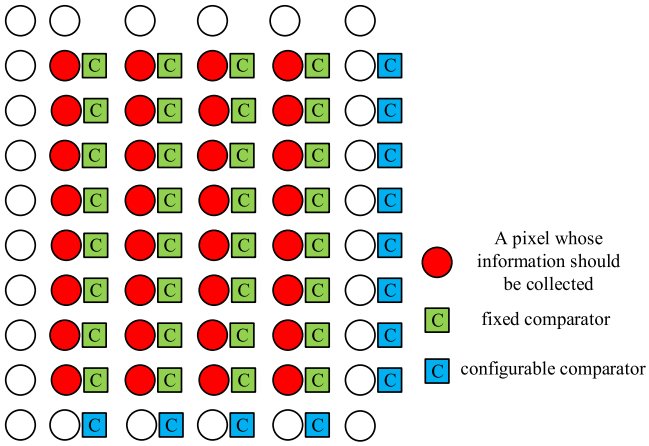


Fig. 13. Proposed configurable comparator array.

SAO. The function of each stage of SAO statistics collection phase is illustrated as follows (using the notations in Fig. 10).

Stage 0 (SAO_0): Obtain 32 pixels' category and band selection information of EO_0, EO_1, EO_2, EO_3 and BO, and perform some adder calculation of (6) at the same time.

Stage 1 (SAO_1): Complete the rest adder calculation of (6).

SAO takes four cycles to collect statistics information for a four-line unit. For example, EO_0 and BO' 0–7 bands are processed in cycle 35; EO_1 and BO' 8–15 bands are processed in cycle 36; EO_2 and BO' 16–23 bands are processed in cycle 37; EO_3 and BO' 24–31 bands are processed in cycle 38 shown in Fig. 10. The interlaced pipeline scheme achieves a good tradeoff between high-throughput and chip area.

Configurable comparator array and fragmentation adder scheme are proposed to cooperate with the SAO pipeline. Fig. 13 shows the proposed configurable comparator array employed to category classification in EO. In Fig. 4, "c" is the current pixel, and "a" and "b" are the two neighboring pixels. As shown in Table II, the current pixel "c" would be compared to "a" and "b". Data reuse between pixels can be further applied for the next samples' category classification. For example, assuming the SAO type is EO_0, the current pixel's comparison value of "c" and "b" can be derived from the its right pixel's comparison value of "c" and "a." In Fig. 13, besides 32 fixed comparators, 12 configurable comparators are also applied when the pixels could not reuse the data from its neighboring pixel at the boundary of these 32 pixels. With the proposed configurable comparator array and data reuse, the number of comparator is reduced to 17% (32 fixed comparators + 12 configurable comparators with proposed method, versus 32 comparators $\times$ 2 (one pixel needs two comparators) $\times$ 4 (four EO classes) without proposed method).

Five right column lines and four bottom lines for Y component, meanwhile three right column lines and two bottom lines for Cb&Cr component should not be collected the statistics information in an LCU, detailed in [13]. Owing to the irregularity in a 32 $\times$ 32 pixels' region statistics collection, we summarize eight structures in Fig. 14. The statistics information of

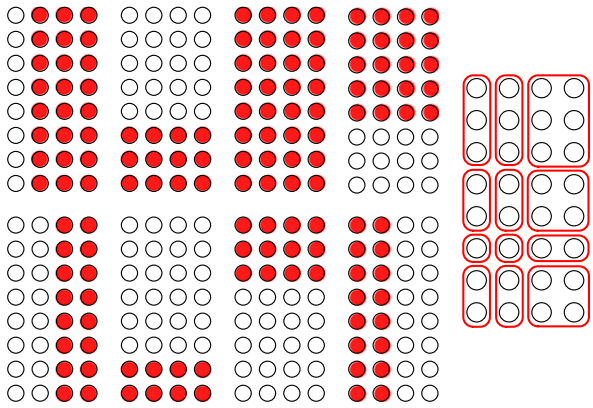


Fig. 14. Proposed fragmentation adder scheme.

the pixels in red should be collected in a specified category or band in Fig. 14, according to (6). Meanwhile, 32 pixels' difference between its original value and post-DF value will be added to further fast distortion estimation. If the addition process of these 32 pixels' difference is done in one cycle, this cycle should be very time-consuming, which restricts the highest working frequency capability of the chip. The proposed fragmentation adder scheme divides these 32 pixels into 12 regions, as shown in Fig. 14. The pixels' difference of each region would be added in stage 0 of SAO pipeline. Then these 12 regions are added selectively in stage 1 of SAO. The proposed fragmentation adder scheme accomplishes a good tradeoff between high-throughput and maximum working frequency capability of the chip.

With the proposed scheme, the DF phase and SAO statistics collection phase take 518 cycles in an LCU, shown in Fig. 10. The rest of the fast distortion estimation would be described in Section IV.

IV. Simplified Fast Rate Estimation

In HM9.0, λ for Y and Cb&Cr are two floating-point numbers according to (7), which are derived from two exponential function of the QP. And floating-point number multiplication is a both time and area consuming in hardware design. In order to simplify the multiplication, two look-up tables are utilized to generate the λ for Y and Cb&Cr respectively where QP is the integer index ranging from 0 to 51, and the values of λ are all integers in the tables (floating to integer method, FTI).

Besides, we also propose a simplified bitrate (*R*) estimation method (SBE) of rate-distortion cost calculation, according to (7). The value of *R* is derived from entropy coding (coding a specified SAO type and the four offsets of this type) in HM9.0. The calculation of *R* takes at least eight cycles according to the hardware architecture of CABAC proposed in [12] and [13], which is a time-consuming process. Here, we propose a simplified bitrate estimation method, which employs a linear model to derive the value of *R* in a cycle.

N randomly generated data $x_1, x_2, \ldots x_N$ are used as the input to the simplified bitrate estimation function with the corresponding output $y_1, y_2 \ldots y_N$. In our problem, $x_i (i = 1, 2, \ldots, N)$ is the sum of absolute values of several offsets in a specified SAO type, and $y_i (i = 1, 2, \ldots, N)$ is the estimated bits of side information of the corresponding SAO type. We assume a linear model on this I/O relationship to reduce the cost of hardware design and improve the throughput significantly, where

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} a + b \tag{11}$$

TABLE III
Comparison With Previous Designs

		FTI + SBE method		
		BD-rate(%)		
Class	Sequence	AI	LD	RA
Class A [2560 × 1600]	Traffic	0.1	0.1	0.0
	PeopleOnStreet	0.2	0.2	0.2
	Nebuta	0.0	0.2	0.1
	SteamLocomotive	0.1	1.1	0.5
Class B [1920 × 1080]	Kimono	0.1	0.2	0.1
	ParkScene	0.1	0.2	0.2
	Cactus	0.0	0.2	0.1
	BasketballDrive	0.1	0.1	0.3
	BQTerrace	0.0	0.2	0.2
Class C [832 × 480]	BasketballDrill	0.1	0.5	0.0
	BQMall	0.0	0.1	0.0
	PartyScene	0.0	0.0	0.1
	RaceHorses	0.0	0.1	0.2
Class D [416 × 240]	BasketballPass	0.1	0.1	0.1
	BQSquare	0.1	0.3	0.4
	BlowingBubbles	0.0	0.1	0.0
	RaceHorses	0.0	0.2	0.0
Class E [1280 × 720]	FourPeople	0.1	0.3	0.0
	Johnny	0.1	0.0	0.1
	KristenAndSara	0.0	0.4	0.2
Average		0.1	0.2	0.1

TABLE IV
Hardware Implementation Result

Purpose	Gate count (NAND2)
Deblocking filter	30.3 K
SAO statistics collection	55.9 K
SAO mode decision	17.1 K
Total	103.3 K

TABLE V
Hardware Implementation Result

Technology	TSMC 65 nm CMOS
Gate count	103.3 K
On-chip SRAM	4.2 kB
On-chip buffer	0.6 kB
Working frequency	200 MHz

TABLE VI
COMPARISON WITH PREVIOUS DESIGNS

Design	Standard	Implementation Style	DF	SAO	Processing time (cycle counts/LCU)1 LCU=16 MB	Process	Gate count (NAND2)	On-chip SRAM	Supporting video format	Frequency (MHz)	DE ($\times 10^3$)
[17]	H.264	Gate-level	√	×	260*16	0.13 μm	36.9 K	672	16VGA at 30 f/s (2560 × 1920)	225	4.0
[18]	H.264	Real-chip	√	×	204*16	0.18 μm	21.4 K	512 + 64N	1080p at 30 f/s (1920 × 1088)	200	2.9
[19]	H.264	Real-chip	√	×	100*16	0.13 μm	22.9 K	416	QFHD at 30 f/s (3840 × 2160)	98	10.9
[20]	H.264	Gate-level	√	×	243*16	0.18 μm	21.1 K	864 + 8N	1080p at 30 f/s (1920 × 1088)	238	3.0
[21]	H.264	Gate-level	√	×	192*16	0.18 μm	26.0 K	512 + 8N	QFHD at 30 f/s (3840 × 2160)	187	9.6
[22]	H.264	Gate-level	√	×	112*16	0.18 μm	12.1 K	256	QFHD at 30 f/s (3840 × 2160)	109	20.6
[23]	H.264	Gate-level	√	×	136*16	90 nm	17.9 K	1056	QFHD at 30 f/s (3840 × 2160)	133	14.0
[24]	HEVC	Gate-level	√	×	2043	0.13 μm	17.6 K	384	QFHD at 60 f/s (3840 × 2160)	250	28.3
[25]	HEVC	FPGA	√	×	2560	40 nm	36.8 K	832	1080p at 86 f/s (1920 × 1088)	108	4.8
Proposed	HEVC	Gate-level	√	√	558	65 nm	DF: (30.3 + 0.148) K SAO: 73 K	4200	UHD at 40 f/s (7680 × 4320)	182	DF: 43.6 SAO: 18.2

*N represents the frame width in pixels.

which can be represented concisely as follows:

$$\boldsymbol{y} = [\boldsymbol{x} \;\; 1_{N\times 1}]\,\boldsymbol{\theta} \tag{12}$$

where $\boldsymbol{y} = [y_1 y_2 \ldots y_N]^T$, $\boldsymbol{x} = [x_1 x_2 \ldots x_N]^T$, $\mathbf{1}_{N\times 1}$ is an all one vector of size $N \times 1$, and $\boldsymbol{\theta} - (a\ b)^T$, which denotes the parameters of the linear model.

Least square technique [14] is applied to estimate $\boldsymbol{\theta}$

$$\boldsymbol{\theta} = \left([\boldsymbol{x} \;\; 1_{N\times 1}]^H\,[\boldsymbol{x} \;\; 1_{N\times 1}]\right)^{-1}[\boldsymbol{x} \;\; 1_{N\times 1}]^H \boldsymbol{y} \tag{13}$$

which could be solved easily by a linear equation in standard manner. With the above method, we obtain the following equation for *R* estimation:

$$R_{Y\_EO} = \sum_{i=0}^{3} |\text{offset}_i| + 13 \tag{14}$$

$$R_{Y\_BO} = \sum_{i=0}^{3} |\text{offset}_i| + 16 \tag{15}$$

$$R_{Cb\&Cr\_BO} = \sum_{i=0}^{7} |\text{offset}_i| + 25 \tag{16}$$

$$R_{Cb\&Cr\_BO} = \sum_{i=0}^{7} |\text{offset}_i| + 25 \tag{17}$$

$$R_{Y\&Cb\&Cr} = 3\sum_{i=0}^{11} |\text{offset}_i| + 1 \tag{18}$$

$$R_{\text{merging}} = 4. \tag{19}$$

The performance of the proposed FTI + SBE method is evaluated in terms of the change of average Bjontegaard Delta rate (BD-rate) [15]. The performance gain or loss is measured with the respect to the HEVC reference software platform (HM9.0). The experiments are carried for "All Intra-Main (AI)," "Low Delay-Main (LD)" and "Random Access-Main (RA)" settings as stipulated by the common condition proposed in [16]. The configuration files are provided in the common software package of HM9.0. QP values of 22, 27, 32 and 37 cover a broad range of qualities and bit rates.

Table III shows the experimental results of the proposed FTI + SBE scheme as compared to HM9.0 tested on 20 sequences. It is observed that BD-rate increment is less than 0.2% for all sequences of "AI" setting which is negligible, with best case of 0.0% increment and worst case of 0.2% increment. And the average BD-rate increment for all sequences of "LD" and "RA" settings is 0.2% and 0.1% respectively, which shows that the FTI + SBE method can also be applied to inter coding. Meanwhile, the cycles for SAO mode decision are reduced by 87.5% in hardware design. The SAO mode decision phase takes only 40 cycles for an LCU with the proposed FTI + SBE scheme according to Fig. 10.

V. IMPLEMENTATION RESULT AND COMPARISON

We have implemented the proposed architecture in Verilog HDL and synthesized it targeted towards a TSMC 65 nm CMOS cell library under a timing constraint of 200 MHz. Tables IV and V show the hardware implementation result. Our design requires 103.3 K gate count and 4.2 kB on-chip SRAM, and 0.6 kB on-chip buffer. It is observed that the SAO collection phase consumes more than 50% area of the whole design, owing to the large usage of addition, subtraction and comparator calculations according to (6).

The hardware performance comparison between our proposed design and other designs is presented in Table VI. Our design is the only one can support UHD applications, which also works at a lower working frequency, compared with other de-

signs. In order to store not only the reconstructed pixels but also the original pixels for SAO collection phase module, our design needs more on-chip SRAM sizes than other designs. Moreover, a quarter-LCU based hardware architecture is proposed, while the DF is based on a MB in H.264 standard design. The above two reasons causes to a huge consumption of on-chip SRAM sizes. Our design consumes a larger SRAM sizes than the other designs [24] and [25], because that the designs of HEVC DF of Ye *et al.* [24] and Ozcan *et al.* [25] are based on 16×16 CU. According to the processing time, the throughput is faster about 3–7 times than other designs [17]–[25]. Our proposed hardware architecture of combined DF and SAO is designed for HEVC intra encoder, neverthless the difference of DF between intra and inter coding is only the caclulation of BS value. The area of BS caculation supporting inter coding is 148 gate count larger than intra one. To give a fair comparsion, we add this part to the area cost of our design.

In order to perform a fair comparison, we introduce a normalized criterion called Design Efficiency (DE). It is defined as

$$\mathrm{DE} = \frac{\mathrm{Format} \times \mathrm{Fps}}{\mathrm{Gate\ Count}} \tag{20}$$

where Format is the multiplication of width and height of the supporting video format in Table VI. Fps is the fps of the supporting video format. Gate count is the number of equivalent NAND2 of the design. It is well known that the bigger DE has the greater DE.

The normalized result in Table VI shows that our design of DF is at least 54% more efficient than any previous works about DF. And our proposed SAO has a DE value of 18.2, which is designed for HEVC intra encoder. The difference of SAO between intra and inter coding is that the SAO of inter coding need one more operation. The SAO should add the offset to the output samples of DF according to the best SAO type.

VI. Conclusion

This paper proposes a hardware implementation on the combined DF and SAO for HEVC intra encoder, which features a high throughput. With the featured balanced pipeline scheme, the maximum speed can reach 200 MHz as shown in Table V. As a result, our design can support UHD at 40 f/s with merely 182 MHz working frequency, and it is also capable for higher resolution or low power applications. Meanwhile, such an implementation of DF and SAO benefits the capability of working frequency of the whole HEVC intra encoder.

References

[1] C.-M. Fu, C.-Y. Chen, Y.-W. Huang, and S. Lei, "Sample adaptive offset for HEVC," in *Proc. IEEE 13th Int. Workshop Multimedia Signal Process.*, Oct. 2011, pp.1–5.

[2] *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, ITU-T Rec. H.264/AVC/ISO/IEC 14496-10 AVC, JCT-G050, 2003.

[3] *High Efficiency Video Coding (HEVC) Text Specification Draft 9 (SoDIS)*, JCTVC-K1003, Oct. 2012.

[4] *High Efficiency Video Coding (HEVC) Test Model 7 (HM 7) Encoder Description*, JCTVC-I1002, May 2012.

[5] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: The new gold standard for video compression: How does HEVC compare with H.264/AVC?" *IEEE Consum. Electron. Mag.*, vol. 1, no. 3, pp. 36–46, Jul. 2012.

[6] M. Yuen and H. R. Wu, "A survey of hybrid MC/DPCM/DCT video coding distortions," *J. Signal Process.*, vol. 70, no. 3, pp. 247–278, Nov. 1998.

[7] A. Norkin *et al.*, "HEVC deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol.22, no. 12, pp. 1746–1754, Dec. 2012.

[8] C.-M. Fu *et al.*, "Sample adaptive offset in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1755–1764, Dec. 2012.

[9] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec. 2012.

[10] J. Zhu, D. Zhou, G. He, and S. Goto, "A combined SAO and de-blocking filter architecture for HEVC video decoder," in *Proc. IEEE 20th Int. Conf. Image Process.*, Sep. 2013, pp. 1967–1971.

[11] *Description of Core Experiment 1 (CE1): Sample Adaptive Offset Filtering*, JCTVC-H1101, Feb. 2012.

[12] R. Song, H. Cui, Y. Li, and X. Song, "A five-stage pipeline design of binary arithmetic encoder in H.264/AVC," in *Proc. Asia-Pacific Signal Inform. Process. Assoc. Annu. Summit Conf.*, Dec. 2012, pp. 1–4.

[13] V. Rosa, L. Max, and S. Bampi, "High performance architectures for the arithmetic encoder of the H.264/AVC CABAC entropy coder," in *Proc. IEEE 17th Int. Conf. Electron., Circuits, Syst.*, Dec. 2010, pp. 383–386.

[14] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*, 1st ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[15] *Calculation of Average PSNR Differences Between RD Curves*, ITU-T SC16/Q6, VCEG-M33, 2001.

[16] *HM Reference Software*. (2013). [Online]. Available: svn://hevc.kw.bbc.co.uk/svn/jctvc-hm/tags/ HM-9.0. Accessed on: Jan. 2014.

[17] C.-A. Chien, H.-C. Chang, and J.-I. Guo, "A high throughput in-loop deblocking filter supporting H.264/AVC BP/MP/HP video coding," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Nov.-Dec. 2008, pp. 312–315.

[18] K. Xu and C.-S. Choy, "A five-stage pipeline, 204 Cycles/MB, single-port SRAM-based deblocking filter for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, pp. 363–374, Mar. 2008.

[19] Y.-C. Lin and Y.-L. Lin, "A two-result-per-cycle deblocking filter architecture for QFHD H.264/AVC decoder," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 6, pp. 838–843, Jun. 2009.

[20] T.-M. Liu, W.-P. Lee, and C.-Y. Lee, "An in/post-loop deblocking filter with hybrid filtering schedule," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 7, pp. 937–943, Jul. 2007.

[21] N. T. Ta, J. S. Youn, H. G. Kim, J. R. Choi, and S.-S. Han, "Low-power high-throughput deblocking filter architecture for H.264/AVC," in *Proc. Int. Conf. Electron. Comput. Technol.*, Feb. 2009, pp. 627–631.

[22] M. Nadeem, S. Wong, G. Kuzmanov, and A. Shabbir, "A high-throughput, area-efficient hardware accelerator for adaptive deblocking filter in H.264/AVC," in *Proc. IEEE/ACM/IFIP 7th Workshop Embedded Syst. Real-Time Multimedia*, Oct. 2009, pp. 18–27.

[23] J. Zhou *et al.*, "A 136 cycles/MB, luma-chroma parallelized H.264/AVC deblocking filter for QFHD applications," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jun.-Jul., 2009, pp. 1134–1137.

[24] X. Ye, D. Ding, and L. Yu, "A cost-efficient hardware architecture of deblocking filter in HEVC," in *Proc. IEEE Vis. Commun. Image Process. Conf.*, Dec. 2014, pp. 209–212.

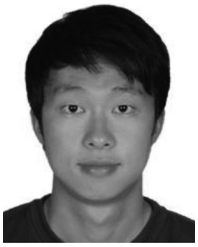
[25] E. Ozcan, Y. Adibelli, and I. Hamzaoglu, "A high performance deblocking filter hardware for high efficiency video coding," *IEEE Trans. Consum. Electron.*, vol. 59, no. 3, pp. 714–720, Aug. 2013.

Weiwei Shen received the B.S. and Ph.D. degrees in microelectronics and solid electronics from Fudan University, Shanghai, China, in 2010 and 2015, respectively.

His research interests include VLSI design, algorithms, and the VLSI architectures for multimedia signal processing.

Yibo Fan received the B.E. degree in electronics and engineering from Zhejiang University, Hangzhou, China, in 2003, the M.S. degree in microelectronics from Fudan University, Shanghai, China, in 2006, and the Ph.D. degree in engineering from Waseda University, Tokyo, Japan, in 2009.

He was an Assistant Professor with Shanghai Jiao Tong University, Shanghai, China, from 2009 to 2010, and is currently an Associate Professor with the College of Microelectronics, Fudan University. His research interests include image processing, video coding, and associated VLSI architecture.



Yufeng Bai received the B.S. degree in electronics engineering from Sun Yat-Sen University, Guanghzhou, China, in 2008, and is currently working toward the M.S. degree in microelectronics at Fudan University, Shanghai, China.

His research interests include video coding and its VLSI architecture design.

Leilei Huang received the B.S. degree in microelectronics and solid electronics from Fudan University, Shanghai, China, in 2014, where he is currently working toward the M.S. degree in microelectronics and solid electronics.

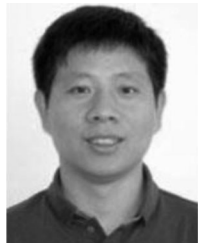
His research interests include VLSI design, algorithms, and corresponding VLSI architectures for multimedia signal processing.

Qing Shang received the B.S. and M.S. degrees in microelectronics and solid electronics from Fudan University, Shanghai, China, in 2011 and 2014, respectively.

His research interests include VLSI design, algorithms, and VLSI architectures for multimedia signal processing and design for test.

Cong Liu received the B.S. and M.S. degrees in microelectronics and solid electronics from Fudan University, Shanghai, China, in 2011 and 2014, respectively.

His research interests include VLSI design, algorithms, and the VLSI architectures for multimedia signal processing and design for test.



Xiaoyang Zeng (M'07) received the B.S. degree from Xiangtan University, Xiangtan, China, in 1992, and the Ph.D. degree from the Changchun Institute of Optics, Fine Mechanics, and Physics, Chinese Academy of Sciences, Changchun, China, in 2001.

From 2001 to 2003, he was a Postdoctoral Researcher with Fudan University, Shanghai, China. Then, he became an Associate Professor with the State Key Lab of ASIC and System, Fudan University, where he is currently a Full Professor and the Director. His research interests include information security chip design, system-on-chip platforms, and VLSI implementation of digital signal processing and communication systems.