A COMBINED SAO AND DE-BLOCKING FILTER ARCHITECTURE FOR HEVC VIDEO DECODER

Jiayi Zhu, Dajiang Zhou, Gang He, Satoshi Goto

Waseda University, Kitakyushu Japan
Email: jzhu@aoni.waseda.jp

ABSTRACT

The up-coming video compression standard, high efficiency video coding (HEVC), reduces 50% bit rates in encoding video sequences with same picture quality compared to H.264/AVC. In the in-loop filter (LF) part of HEVC, sample adaptive offset (SAO) is newly added and de-blocking filter (DBF) has been changed a lot. Thus how to construct a high speed and low cost VLSI architecture for HEVC SAO and de-blocking filter is a challenge.

In this article, we propose a HEVC LF architecture composed of fully utilized de-blocking filter and SAO. Block based SAO and DBF are employed in this architecture to achieve seamless pipeline between them. The implementation results show that it can be synthesized to 240MHz with 65nm technology. Thus this solution can process 3.84G pixels/s and support 4320p(7680x4320)@120fps decoding.

Index Terms—DBF, SAO, HEVC, block-layer pipelined, fully utilized

1. INTRODUCTION

HEVC is the next generation video coding standard. Compared to its predecessor H.264/AVC, it has already reduced 50% bit rates in encoding video sequences with same picture quality. In-loop filter is an important part of video coding standard. In the main profile of HEVC, it is composed of 2 parts, de-blocking filter and SAO.

SAO is a new in-loop filtering technique that reduces the distortion between original samples and reconstructed samples. It has been observed that SAO can improve video compression in both objective and subjective measures with reasonable complexity [3]. SAO locates after the de-blocking filter. Since it refers neighboring reconstructed pixels after de-blocking filtering, SAO of current coding unit (CU) can be processed only after the right and bottom CUs have been de-blocking filtered. In a consequence, although de-blocking filter of HEVC is simplified by reducing the number of edges to be filtered, it is still a challenge for engineers to find a high speed and low cost solution for SAO and de-blocking filter.

There are some previous works on the topic of loop-filter. In [5], 4 edges filters organized in 2 groups for simultaneously processing vertical and horizontal edges are applied; in [6], a 5-stage pipelined and resource-shared dual-edge filter to generate two filtering results every cycle is proposed; in [8], a zero stall cycle in normal pipeline flow architecture is proposed. But all these works do not support HEVC LF. In [4], a HEVC and H.264 dual mode de-blocking filter is proposed. But it does not support SAO.

In this article, we propose a HEVC LF architecture composed of fully utilized de-blocking filter and SAO. Block based SAO and DBF are employed in this architecture to achieve seamless pipeline between them. Our implementation results show that it can be synthesized to 240MHz with 65nm technology. Thus the solution can process 3.84G pixels/s. It can support 4320p@120fps decoding.

The rest of this paper is organized as follows. Section 2 and 3 introduces the algorithms and data flow of de-blocking filter and SAO in HEVC. Section 4 describes the proposed VLSI architecture in details. The implementation results are illustrated in Section 5. Finally, Section 6 concludes this paper.

2. HEVC LF ALGORITHMS

2.1. De-blocking filter algorithms

In HEVC, edges of 8x8 blocks are needed to be de-blocking filtered. 4 pixels on both sides of one edge are referred and some of them are modified. There are 3 modes of de-blocking filter when filtering one line of pixels in the 2 sides of the edge: luma strong, luma weak and chroma. Different operations are employed in different filtering modes. The decisions for whether current edge is luma strong or luma weak is derived from parameter beta, tc and values of the 1st and 3rd lines of pixels in the 2 sides of the filtered edge, as shown in Fig 1.

2.2. Sample adaptive offset algorithms

SAO classifies reconstructed samples into different categories, obtains an offset for each category and then add the offset to each sample of the category. There are 2 types of SAO, edge offset (EO) and band offset (BO). BO implies
one offset is added to all samples of the same band. The offset is derived from the SAO parameters of current CTB and value of current pixel itself. There exists no neighboring pixels reference issue for BO.

EO uses 4 1-D directional patterns for sample classification: horizontal, vertical, 135-diagonal and 45-diagonal, as shown in Fig 2.

![Fig 2. 4 1-D directional patterns for EO sample classification](image)

3. DATA FLOW FOR PROPOSED ARCHITECTURE

3.1. De-blocking filter algorithms

In HEVC, de-blocking filter resides after the reconstructor in video decoders. Hardware reconstructor is usually designed to output data one 8x8 block after another. Shown as Fig 3(a), the 4 edges of one 8x8 block (white) to be de-blocking filtered are the boundaries of the 8x8 block. To filter these edges, upper and left blocks must be referred and it becomes complex. Suppose that input data can be re-ordered as the shifted 8x8 block ABCD, shown as Fig 3(b), thus the de-blocking filter is much easier since it is an independent unit for de-blocking filter without needing referring neighboring data block. 4 cycles are needed to process 4 edges to finish the de-blocking filter of shifted 8x8 block ABCD.

![Fig 3. Current 8x8 block, corresponding shifted 8x8 blocks for DBF and corresponding pixels for SAO](image)

3.2. Sample adaptive offset algorithms

EO refers neighboring pixels to obtain the offset. If the shifted 8x8 block ABCD is to be through SAO, then the pixels of right and bottom blocks are referred, as shown in Fig 3(c), which might have not been de-blocking filtered. In order to avoid this issue, we shifted the SAO pixels to be the pattern in Fig 3(d). 10x10 upper left shifted pixels block will be input and the center 8x8 pixels are modified. By this means, right and bottom blocks of the shifted 8x8 block ABCD are not needed to refer. Only its upper and left 4x4 blocks (i.e. E, F, G, H and I in Fig 4(a)) are needed to refer.

To match the throughput of de-blocking filter, 4 cycles are needed to do SAO. 4x4 blocks A, B and D go into SAO one by one. When A is input to SAO, partial pixels of E, F and H are also needed to input to SAO. And similar cases exist for B, C and D, as shown in Fig 4 (b). After the de-blocking filter and SAO process of ABCD, the pixels of the right column and bottom row of B, C, D have not completed SAO. But all the pixels in E, F, H and A have completed SAO and will be output.

![Fig 4. Data block involved in DBF and SAO](image)

In SAO process, I and C are stored to line buffer. When processing the most right column shifted 8x8 blocks of one CTB, G, B and D are stored to the left buffer. For maximum 64x64 CTB, there are 8 GBD to be stored.

4. PROPOSED VLSI ARCHITECTURE

As shown in Fig 5, the proposed architecture is divided to DBF part and SAO part. Reconstructed pixels go into the de-blocking filter firstly and then go into SAO. SAO referred the left and upper pixels to do the offset and output completed pixels. In our actual architecture, luma and chroma pixels are processed in parallel. But here we only introduce the luma logic to keep this article concise.

![Fig 5. DBF, SAO and data buffers](image)
DBF and SAO. So on each cycle, different data sources are input to the cores or registers groups. The gray rectangles labeled as “m” in these diagrams mean multiplexes. The red number on the arrow in these diagrams selects the source of multiplex output when the counter counts to that value.

![Fig 6. 4 DBF filters, its input data multiplexes and data input in the whole period to filter one 8x8 block](image)

### 4.1. De-blocking filter VLSI architecture

The 4 DBF cores are combinational logic. Each DBF core filters one line in one cycle. Thus 2 4x4 blocks in the 2 sides of one edge can be filtered in one cycle. 4 edges of one shifted 8x8 block (shown in Fig 4 (b)) are filtered in 4 cycles. In Fig 6, in_1_h and in_r_h means the rec pixels in Fig 5. *_h means 4x4 data block are input horizontally to filter the vertical edges. *_l_* means left 4x4 block and *_r_* means right 4x4 block of the filtered edge. Dr_*_v means 4x4 data block in DBF registers groups a, b, c or d is input vertically to filter the horizontal edges.

![Fig 7. 4 DBF registers groups and its input multiplex](image)

The 4 DBF 4x4 registers groups are named as dr_a, dr_b, dr_c, dr_d, as shown in Fig 7. P’ and q’ are output of DBF cores, shown in Fig 6. *_h means the 2 4x4 data blocks come from 4 DBF cores horizontally for vertical edge filtering, and vice versa for _v_.

### 4.2. SAO VLSI architecture

SAO flow is shown in Fig 4 (b), 4x4 blocks A, C, B and D together with partial pixels of their neighboring blocks (EFHA, HAIC, FGAB and ABCD) are input to the SAO core one by one. So theoretically, 9 4x4 registers groups are needed to store 9 4x4 blocks (A-I). But by dynamically allocating them, 6 registers groups are enough to finish the work. For example, block E can be output after EFHA finish SAO; blocks H and I can be output or line buffered after HAIC finish SAO; block F can be output after FGAB finish SAO; blocks F and G can be fetched from line buffer when EFHA and FGAB are ready to SAO; blocks A, B, C and D can be fetched from DBF registers group when they are ready to SAO. The overall pipeline of the process is illustrated in section 3.3.

![Fig 8. SAO core and data input by counter value](image)

On each cycle of SAO process, a 6x6 data block is input to SAO core and only center 4x4 data block are modified, as shown in Fig 8. The SAO core is combinational logic. The 6x6 block is divided into 4 parts, which are called sf_0 to sf_3. Sf_0 is composed 4 samples, sf_1 and sf_2 is composed of 8 samples, sf_3 is composed of 16 samples.

![Fig 9. SAO registers groups](image)

In the 4 cycles, different data blocks are input to the 4 parts. For sf_0 part, E, H, F and A are input to SAO core when counter is 0, 1, 2 and 3. The corresponding hardware holding the data is sr_a, sr_b, sr_c, and sr_d (SAO registers groups). Similar cases happen for sf_1 to sf_3. Among all
the hardware multiplexed to SAO core, sr_a to sr_f are 6 SAO registers groups which serve as the temporary storage. Up is the data from line buffer. Dr_a and dr_c are DBF registers groups shown in Fig 7.

In addition to the SAO core, there are 6 registers groups together with their multiplexes in the SAO. Each registers group is composed of 4x4 normal registers and another 7 shadow registers (the brown ones in Fig 9). The shadow registers are used to store the date from DBF. When the data of this block is input to SAO core, the value in the normal registers is modified while that in the shadow registers keeps un-changed. The value in the shadow registers are used to store the surrounded pixels for SAO, i.e. the red cycles in the boundary of the 6x6 pixels block, shown in center of Fig 8. In Fig 9, all the sources of each multiplex to the SAO registers groups are illustrated. Red letter in cycle means the data block has just been modified by SAO core. The modified pixels are from output of SAO core instead of the storage hardware mentioned above.

### 4.3. Overall Pipeline

![Pipeline Diagram](image)  
**Fig 10. Pipeline of DBF and SAO**

The pipeline of DBF and SAO is shown in Fig 10. The upper half is de-blocking filter pipeline and the lower half is SAO pipeline. sf_io means the input and output of SAO core. Up_ra, up_rd and up_wa/d are the read address, read data, write address and write data of line buffer. Lf_ra, Lf_rd and Lf_wa/d are the read address, read data, write address and write data of left buffer. GBD in left buffer are only read when processing the most right shifted 8x8 blocks of one CTB. They will be registered into sr_a, sr_b and sr_c. Single G from left buffer is only read when processing the most right upper shifted 8x8 block in one CTB. This is because that this 4x4 block is not stored in line buffer. Ou_wa/d is write address and write data of output data buffer. The green blocks mean the registered DBF core or SAO core output. The red blocks means DBF core or SAO core is working.

### 5. IMPLEMENTATION RESULTS

The implementation results are shown in Table 1. Our work is the only one to support HEVC SAO, which is the critical part of HEVC LF. Our work can be synthesized to 240MHz, which makes it capable of decoding 4320p@120fps. It is twice of [4]. For [5] and [6], even if their technology is scaled to 65nm and frequency reaches 240MHz, their performance is still one-third of or lower than the proposed work. The registers groups in DBF are shared as the input buffer to SAO, which remove any intermediate storage between DBF and SAO. Our work supports parallel luma and chroma hardware, which effectively increases the throughput. The blocked based SAO feature ensures that throughput of SAO matches that of DBF and guarantees the throughput.

<table>
<thead>
<tr>
<th></th>
<th>This Work</th>
<th>[4]</th>
<th>[5]</th>
<th>[6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gates</td>
<td>31.0K DBF</td>
<td>43.3K</td>
<td>30.2K</td>
<td>17.1K</td>
</tr>
<tr>
<td># of DBF cores</td>
<td>4(luma) 2(cb) 2(cr)</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Tech</td>
<td>65nm</td>
<td>65nm</td>
<td>130nm</td>
<td>130nm</td>
</tr>
<tr>
<td>Cycles/MB</td>
<td>16 (HEVC) 24 (H.264)</td>
<td>48</td>
<td>74 (average)</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>HEVC DBF HEVC SAO</td>
<td>HEVC DBF H.264 DBF</td>
<td>H.264 DBF</td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>240MHz</td>
<td>200MHz</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>Max Throughput</td>
<td>4320p@120fps</td>
<td>4320@60/30fps</td>
<td>na</td>
<td>na</td>
</tr>
</tbody>
</table>

### 6. CONCLUSION

In this article, we propose a VLSI architecture for HEVC in-loop filter. In this architecture, only 4 cycles are needed to finish the DBF and SAO of one 8x8 block. SAO and DBF are 8x8 block layer pipeline. The latency between DBF and SAO is only 3 cycles. The implementation results show that the proposed architecture can be synthesized to 240MHz. It can process data @ 3.84Gpixels/s, which is sufficient to support 4320p@120fps.

### 7. ACKNOWLEDGEMENT

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

---

1970
8. REFERENCES