A 48 Cycles/MB H.264/AVC Deblocking Filter Architecture for Ultra High Definition Applications

Dajiang ZHOU†‡, Jinjia ZHOU†, Jiayi ZHU††, Nonmembers, and Satoshi GOTO†, Fellow

1. Introduction

The recent years have witnessed tremendous advances in digital video technology. While 1080HD has already become a standard of mainstream video applications such as TV broadcasting and home entertainment, even higher resolutions (e.g. QFHD and 8 K) and frame rates (e.g. 60 fps and 120 fps) are now targeted by high-end and next-generation solutions (e.g. QFHD and 8 K) and frame rates (e.g. 60 fps and 120 fps) are now targeted by high-end and next-generation solutions. As a result, only one 24×64 two-port SRAM as internal buffer is required in this design. When synthesized with SMIC 130 nm process, the architecture costs a gate count of 30.2 k, which is competitive considering its high performance.

**key words:** H.264/AVC, parallel, deblocking, ultra high resolution, QFHD

Fig. 1 Objective quality enhancement by adopting deblocking filter on the 3840×2160 “DucksTakingOff” sequence.
time budget for processing the QFHD (3840×2160) 60 fps sequences should be 52 cycles for each macroblock, which is to be even more critical for higher specifications.

Many of the previous works on deblocking filter architectures have been targeted to standard HD applications, for which efficient utilization of one single edge filter should be fast enough [6]–[9]. However, it is difficult to apply these architectures to higher specifications, due to their throughput limitation of 192 cycles/MB. Some of the other designs have paid attention to parallelize 2 edge filters [10]–[13]. H.-Y. Lin et al.’s architecture [12] makes full use of its edge filters to process one MB in 96 clock cycles, at the cost of large on-chip SRAMs. Dang [11] proposed a low cost solution of application specific processor with no SRAMs, while its throughput is also lower, which is 192 cycles/MB that can be reached by some single filter solutions. Among these designs, Y.-C. Lin et al.’s architecture [10] achieves the highest average throughput by applying skip patterns to avoid some unnecessary filter operations. Its processing cycles for one MB vary from 48 to 100 according to the edge features. However, strictly speaking, the variable throughput approach is incapable of improving the worst-case performance. Meanwhile, to give play to this approach’s advantage on average performance, large interconnection buffer is required between the deblocking filter and its neighboring components.

In this paper, 4 edge filters organized in 2 groups for simultaneously processing vertical and horizontal edges are applied in an H.264/AVC deblocking filter architecture to enhance its throughput. While parallelism increases, pipeline hazards arise owing to the latency of edge filters and data dependency of deblocking algorithm. To solve this problem, a zig-zag processing schedule is proposed for this architecture to eliminate the pipeline bubbles. Data path of the architecture is then derived according to the processing schedule and optimized through data flow merging, so as to minimize the cost of logic and internal buffer. Meanwhile, the architecture’s data input rate is designed to be identical to its throughput, while the transmission order of input data can also match the zig-zag processing schedule. As a result, no intercommunication buffer is required between the deblocking filter and its previous component for speed matching or data reordering. Finally, the proposed architecture can process one macroblock with 48 cycles in a pipeline, so that real-time processing of QFHD@60 fps sequences can be supported with less than 100 MHz.
2. Proposed Architecture

In this section, the proposed architecture is presented as follows. Firstly, Sect. 2.1 explains the pipeline structure of edge filters as well as their organizations in the deblocking filter architecture. Then, the zig-zag processing schedule proposed for eliminating pipeline hazards and enabling horizontal-vertical concurrency, is presented in Sect. 2.2. After that, derivation of a data path graph from the processing schedule by connecting some basic function elements, is discussed in Sect. 2.3. Further optimization of this graph by using data flow merging is given in Sect. 2.4. Finally Sect. 2.5 discusses the system-level interconnection between the proposed deblocking filter architecture and neighboring components.

2.1 4-Stage Pipelined Edge Filters

Design of edge filter has been well discussed in previous contributions. As illustrated in Fig. 3, 4-stage pipelined filters are employed in this work to insure high working frequency. For each of them, input data are directly buffered in stage 0 to preserve enough timing margin, since inputs may come from SRAMs that are located physically far away from the filter logic. Stage 1 derives the edge threshold, according to which decision is made on whether further filter operation is needed or not. If not, subsequent pipeline stages are filled with NOP for power saving. Otherwise the major part of edge filter algorithm is performed in stages 2 and 3, where logics are designed to distribute averagely in the two stages by applying a similar structure as proposed in [6].

For throughput enhancement, a total of 4 edge filters are applied in this architecture and organized in 2 groups, for filtering vertical (V) and horizontal (H) edges respectively (Fig. 3). Moreover, each group contains two edge filters that concurrently process two lines of samples across the edge. Accordingly, one edge between two 4x4 blocks can be filtered within 2 clock cycles, while one vertical edge and one horizontal edge are processed simultaneously. As a result, the optimum speed of this architecture is one edge per clock cycle, or one macroblock in 48 cycles.

2.2 Processing Schedule

With parallelism increasing, data dependency of the deblocking algorithm becomes a critical problem. Note that the edge filter leads to a latency of 4 cycles while it takes only 2 cycles to send the data for filtering one edge into the pipeline, as explained in Sect. 2.1. Therefore, two horizontally neighboring vertical edges (e.g. the edge between L0 and 00, and the edge between 00 and 01, in Fig. 4) can not be processed sequentially, otherwise pipeline hazard will arise and lead to a 2-cycle pipeline bubble. The same problem comes with vertically neighboring horizontal edges. Fig.

![Fig. 3 4-stage pipelined edge filters for the proposed deblocking filter architecture.](image)

![Fig. 4 The proposed horizontal-vertical parallel and pipeline hazard-free processing schedule (zig-zag) for the luma and chroma components of a macroblock. Deeper and lighter grey circles stands for vertical (V) and horizontal (H) edges, respectively.](image)
For use as ingredients of the architecture’s data path, several pipeline bubbles across macroblock boundaries. Then, after V1 has been sent into the edge filter, 4 clock cycles have passed since the start point so that the results of V0 are now coming out of the pipeline and filtering of V2 can start. The rest vertical edges of both luma and chroma components follow the same processing schedule, and the horizontal edges follow a transposed one based on the same consideration. Another benefit of the proposed schedule is that its input order for 4×4 blocks (00-10-01-11-02-...) is very similar to the block scan sequence defined in the standard, which may require less data reordering effort in implementation.

For concurrently filtering the vertical and horizontal edges without data conflict, the horizontal edges of a macroblock are processed with a 6-unit delay to the vertical ones, where a time unit corresponds to the length of one edge in pipeline, or 2 clock cycles. As shown in Fig. 4, the data for filtering edge V4 require a 2-unit latency in the edge filter and then results of block 01 come out to take another 1 time unit for transpose before it can be processed by horizontal filter. Therefore, the earliest time point for filtering of the upper edge of block 00 is to be together with V5. And that of the upper edge of block 01 should be together with V7. Since this V to H latency is the shortest with block 01, we mark its upper edge to be H7 and the processing of other horizontal edges are also delayed to comply to H7, so that the first of them to be processed in a macroblock is H6. Chroma (Cb and Cr) components of a macroblock are processed subsequently to its luma component, while parallel filtering of chroma vertical edges and luma horizontal edges are allowed (e.g. V16 and H16). Note that the chroma edges are actually processed as between 2×2 blocks. However the 2×2 edges inside aligned 4×4 blocks are neglected by the standard, so in this paper chroma filtering are explained in an equivalent 4×4 form.

Furthermore, since no restriction is set for this deblocking filter architecture to work with only one macroblock at one time, processing of the last 6 horizontal edges (H0 to H5) of a macroblock can be parallelized with the first 6 vertical edges (V0 to V5) of the next macroblock, so as to avoid pipeline bubbles across macroblock boundaries.

2.3 Data Path Derivation

For use as ingredients of the architecture’s data path, several processing elements and buffers are pre-defined including t, \( \varepsilon^{-1} \), L and U. As a transpose element, t takes in a 4×4 block and then outputs it in a transposed manner. Since the minimum data width used by an edge filter group as well as the whole data path is 64 bits, which contains a luma or chroma block of 4×2 samples, each routine of t takes 2 clock cycles (or 1 time unit as defined in Sect. 2.2). While accomplishing the transpose operation, t also produces a 1-unit delay for the processed data. A delay element, \( \varepsilon^{-1} \) is used only for producing a 1-unit delay. Since \( \varepsilon^{-1} \) does not contain multiplexors for changing data order, it is smaller in area than t. L is a two-port SRAM (one port read only and another port write only) for buffering the left 8 4×4 neighboring blocks L0-L7 (4 for the luma component and 4 for both chroma components) to the current macroblock. U is an external buffer for storing the last 4 lines of luma samples and the last 2 lines of chroma samples of each macroblock.

Data path for the proposed architecture as shown in Fig. 5, is derived according to the proposed processing schedule, based on analysis of the inputs and outputs of horizontal and vertical edge filters. The outputs of vertical filtering are considered first, then the outputs of horizontal filtering, and the filter inputs.

The right block outputs of vertical filtering V0-V5, V8-V13, V16, V17, V20 and V21 (Fig. 4) can be directly used as the left block inputs of subsequent vertical filtering with a fixed edge number increment of 2. Note that the edge number can be used as a time reference for inputs and outputs, however 2 time units should be added to the edge number for an output when it is compared to the input of another edge, because of the 2-time-unit edge filter latency. The right block outputs of the last vertical filtering (V6, V7, V14, V15, V18, V19, V22 and V23) are subsequently used as the lower block inputs of a horizontal filtering. For example, right block output 03 of V6 is used as input of H11, which requires to be first transposed and then delayed by 2 additional time units, where 2 derived by 5 (V-to-H latency) minus 2 (filter latency) minus 1 (transpose latency). The 2-unit delay is also applicable to right outputs of V14, V18 and V22, while 3 time units are needed by those of V7, V15, V19 and V23 with a V-to-H latency equal to 6. Left block outputs of vertical filtering V0, V1, V8, V16 and V20 are final results as outputs of the deblocking filter. Those for V9, V17 and V21 should be transposed and stored into U for use by the next line of MBs, except when the current MB is at the last line. Left block outputs of vertical filtering not at MB boundaries should be transposed and then directed into
the horizontal filter after a certain delay, which varies from 0 to 2 according to the edge position. For example, besides the 1 \( t \) for transposed, 1 \( z^{-1} \) is needed from V2 to H6 and from V5 to H9, 2 from V3 to H8, and 0 from V4 to H7.

The lower block outputs of horizontal filtering H6-H15, H18, H19, H22, H23, H2 and H3 should be used as the upper block inputs of subsequent horizontal filtering. The results can be directly transferred except for H8, H9, H12 and H14 which require a 4-unit delay. The lower block outputs of H16, H17, H20, H30 and H4 are sent to \( U \). And those of H21, H1 and H5 should be first transposed and then sent to \( L \), for use by the next macroblock. The upper block outputs of H6-H12, H14-H18, H20, H22, H23, H0 and H2-H4 are final results that should be transposed before sent as outputs of the deblocking filter. Those of H13, H19, H21, H1 and H5 should be transposed and sent to \( L \). Note that for H21, H1 and H5, both lower and upper block outputs needed to be sent to \( L \) after transpose, which results in a conflict on the SRAM’s write port. To avoid this problem, an additional \( z^{-1} \) is inserted into the path for lower outputs to interlace the SRAM access.

The right block inputs of vertical filtering is always directed from the input of the deblending filter, with a fixed transmission rate of 64 bits per clock cycle. The left block input for vertical filtering may come from the right block output, or the SRAM of \( L \) for the vertical edges at macroblock boundaries. The lower block inputs of horizontal filtering can be the transposed and delayed results of either the left or right block outputs of vertical filtering. The upper block inputs of horizontal filtering can be either from the results or delayed results of a previous horizontal filtering of the same macroblock, or from buffer \( U \).

Finally, a data path graph is concluded as Fig. 5, which is composed of 5 transpose elements \( t \), 10 delay elements \( z^{-1} \), and several multiplexors, besides the filters, \( L \) and \( U \).

### 2.4 Data Path Optimization

In order to reduce the hardware cost, the data path in Fig. 5 is optimized by merging some of the elements \( t \) and \( z^{-1} \). Firstly, it can be observed that the path from V-out right to H-in lower is only enabled for filtering V6, V7, V14, V15, V18, V19, V22 and V23. Meanwhile, the path from V-out left to H-in lower does not work at V8, V9, V16, V17, V20, V21, V0 and V1, which have exactly 2 time units of latency after the working time points of the former path. By delaying the outputs of V-out right by 2 units and then directing them to the path of V-out left, the two paths can be merged together while preserving the original function. This contributes to a saving of one \( t \) and one \( z^{-1} \), as the updated data path illustrated in Fig. 6. Then, the two elements \( t \) located on the paths from H-out lower to \( L \), and from V-out left to \( U \), are shared by avoiding the conflicted use of them. Finally, it is discovered that the read and write ports of \( L \) are standing by at many time, which may be utilized as temporal storage for some other data. The 4 \( z^{-1} \) located on the path from H-out lower to H-in upper are only activated at H8, H9, H12 and H13, when nothing is being written into \( L \). There is also no read access on \( L \) when H14, H15, H18 and H19 get these data as inputs. Therefore the 4 \( z^{-1} \) are removed by replacing this path with a new one across \( L \). Although the SRAM of \( L \) is thereby enlarged from 16×64 to 24×64 to preserve space for the 4 4×4 blocks, this increase can be compensated by the saved area of \( z^{-1} \) registers. At last, the optimized data path graph as shown in Fig. 6, includes only 3 \( t \) and 5 \( z^{-1} \), and achieves a saving of 2 \( t \) and 5 \( z^{-1} \) in comparison to the initial one (Fig. 5).

### 2.5 System Level Interconnection

In the video decoding system, the deblending filter is connected to 3 components, including the reconstruction component that provides compensated samples, the MV/BS component that provides parameters (BS, C0, \( \alpha \), \( \beta \),...), and an interface component for writing the output data of deblending filter into DRAM. Additionally, the buffer \( U \) is located external to the deblending filter, either as a large on-chip SRAM, or another interface component used to store to and fetch from DRAM the last 4 lines of luma and 2 lines of chroma samples.

I/O schedule of the proposed architecture is shown in Fig. 7, which shows that every 2 clock cycles the deblending filter accepts one block from the reconstruction component, with a data width of 64 bits. The results of vertical filters come out 4 clock cycles later than the input data, which corresponds to the 2-time-unit filter latency. The results of horizontal filters are with another 12-cycle (6-time-unit) delay, as explained in 2.2. Two ports (output 1 and output 0) are used for outputting the filtering results of left neighboring blocks and the rest blocks, respectively. The output data are from the horizontal or vertical edge filters, either directly or with a 2-cycle delay. Although there is a maximum 16-cycle delay from the input data to the output, this time overhead can be concealed by overlapping the processing of consecutive MBs, except for the last MB of one slice.
The data input order is the same as that used by the motion compensation and reconstruction components, which contributes to a faster MC interpolation for partitions larger than 4×8. As a result, no intercommunication buffer is required between the deblocking filter and reconstruction component for speed matching or data reordering. Even if the standard scan order of H.264/AVC has to be adopted, only one additional \(z^{-1}\) for reordering will be enough to eliminate the need of a large buffer, owing to the similarity between the two scan orders.

### 3. Implementation Results

The proposed architecture is implemented in Verilog HDL on RTL level, and then synthesized with Synopsys DesignCompiler by using SMIC 0.13\(\mu\)m G standard cell library. Under a timing constraint of 200MHz, synthesis result shows a logic gate count of 30.2k. Additionally, a 24×64 two-port storage element is generated as a 2P register file for use in this architecture. This design is verified both independently in a test environment with inputs given as software generated data, and in a whole video decoding system we previously designed [14], by substituting its original deblocking filter component.

A comparison between this architecture and state-of-the-art works is shown in Table 1. The worst-case number of clock cycles required for processing one macroblock is reduced by at least 50% from the previous works, at the cost of increased parallelism which also adds to the logic gate count. However, owing to the approaches of data path derivation and optimization adopted in the design stage, size of buffers used in this architecture is very small, in terms of both internal buffer and the interconnection buffer for data reordering and speed matching. The 24×64 two-port SRAM (register file) applied in this work is the least in size when compared to those in the other ASIC based implementations, except that [11] makes use of up to 32 4×4 register arrays to substitute the function of SRAMs. Another merit

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Cycles/MB</th>
<th>Gate count</th>
<th>Process</th>
<th>No. of edge filters</th>
<th>No. of 4×4 arrays</th>
<th>Internal SRAMs</th>
<th>Line buffer</th>
<th>Interconnect. buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>[13] 2004</td>
<td>136</td>
<td>n/a</td>
<td>FPGA</td>
<td>2</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>[9] 2005</td>
<td>192</td>
<td>9.6</td>
<td>.18</td>
<td>1</td>
<td>0</td>
<td>DP 20x32</td>
<td>External</td>
<td>n/a</td>
</tr>
<tr>
<td>[8] 2006</td>
<td>214-246</td>
<td>20.9</td>
<td>.18</td>
<td>1</td>
<td>2</td>
<td>1P 96x32</td>
<td>On-chip</td>
<td>n/a</td>
</tr>
<tr>
<td>[12] 2006</td>
<td>96</td>
<td>13.9</td>
<td>.18</td>
<td>2</td>
<td>0</td>
<td>2P 40x32</td>
<td>External</td>
<td>n/a</td>
</tr>
<tr>
<td>[7] 2007</td>
<td>243</td>
<td>21.1(^1)</td>
<td>.18</td>
<td>1</td>
<td>4</td>
<td>1P 96x32</td>
<td>On-chip</td>
<td>1 MB(^1)</td>
</tr>
<tr>
<td>[6] 2008</td>
<td>204</td>
<td>21.5</td>
<td>.18</td>
<td>1</td>
<td>7</td>
<td>1P 96x32</td>
<td>On-chip</td>
<td>1 MB(^1)</td>
</tr>
<tr>
<td>[11] 2008</td>
<td>192</td>
<td>24.4</td>
<td>.065</td>
<td>2</td>
<td>32</td>
<td>0</td>
<td>External</td>
<td>n/a</td>
</tr>
<tr>
<td>[10]</td>
<td>48-100</td>
<td>17.1</td>
<td>.13</td>
<td>2(^2)</td>
<td>2</td>
<td>2P 28x32</td>
<td>External</td>
<td>4 MBs</td>
</tr>
<tr>
<td>This work</td>
<td>48</td>
<td>30.2</td>
<td>.13</td>
<td>4</td>
<td>8(^3)</td>
<td>2P 24x64</td>
<td>External</td>
<td>0</td>
</tr>
</tbody>
</table>

\(^{1}\) The cost for edge filters and 4×4 arrays has been included in “gate count”.
\(^{2}\) Including a post-loop deblocking filter.
\(^{3}\) Included in internal SRAMs.
\(^{4}\) Implemented with a dual-edge-filter architecture.
\(^{5}\) Consisting of 3 \(\tau\) and 5 \(z^{-1}\).

Fig. 7 I/O schedule of the deblocking filter, and output schedule of edge filters. White blocks stand for the data of current macroblock, with grey ones for previous or next macroblocks.
of this architecture is that, in our video decoding system, it requires no interconnection buffer to match the transmission speed of or reorder the data from the previous reconstruction component, which is needed in almost all the other designs, although not discussed explicitly in some of these papers.

Among the previous works shown in Table 1, [10] gave the closest throughput to this work. In worst case, it processes one MB in 100 cycles. Meanwhile, skip patterns are adopted to improve its speed when only a few or no edges of one MB have non-zero boundary strength (BS). However, these skip patterns can not be so effective in enhancing the real-time performance of the whole decoder system, because of the following reasons. Firstly, the number of edges that require filtering fluctuates to a large extent among video sequences with different features, to some of which skip patterns can hardly be applied. Secondly, a variable-speed component usually can not work efficiently in a pipelined video decoder structure, unless large intercommunication buffers are inserted between it and its adjacent components. Thirdly, the use of skip patterns requires a wider I/O data path, which may bring overhead to its neighboring components and buffers (e.g. the area of SRAMs increase with its port width).

4. Conclusion

In this paper, we apply 4 edge filters organized in 2 groups that simultaneously process vertical and horizontal edges, so as to enhance the throughput of a H.264/AVC deblocking filter architecture. While parallelism increases, pipeline hazard problems arise owing to the latency of edge filters and data dependency of deblocking algorithm. To solve this problem, a zig-zag processing schedule is adopted and pipeline bubbles are successfully eliminated. Data path of the architecture is then derived according to the processing schedule and optimized through data flow merging, so as to minimize the cost of logic and internal buffer. Meanwhile, the architecture’s data input rate is designed to be identical to its throughput, while the transmission order of input data can also match the zig-zag processing schedule, so that no intercommunication buffer is required between the deblocking filter and its previous component for speed matching or data reordering. Finally, the proposed architecture can process one macroblock with 48 cycles in a pipeline, so that QFHD@60 fps sequences can be supported with a clock frequency of less than 100 MHz. As internal buffer, only a 24×64 two-port SRAM is needed in this design. When synthesized with SMIC 130 nm process, the architecture costs a gate count of 30.2 k, which is competitive considering its high performance. We also verified this design by simulating it in a previously designed HD video decoder.

Acknowledgment

This research was supported by Waseda University Ambi-ent SoC Global COE Program of MEXT, Japan, by Knowledge Cluster Initiative (Second Stage) of MEXT, Japan, and by the CREST project of Japan Science and Technology Agency.

References

Dajiang Zhou received B.E. and M.E. degrees from the Department of Electronic Engineering of Shanghai Jiao Tong University, China, in 2005 and 2008, respectively. Since Apr. 2008, he has been working toward a Ph.D. degree at the Graduate School of Information, Production and Systems of Waseda University, Japan. Since Apr. 2009, he has been a research fellow of Japan Society of the Promotion of Science (JSPS). Currently, his main interest is in algorithms and VLSI architectures for multimedia and communication technologies.

Jinjia Zhou received B.E. degree from the Department of Electronic Engineering of Shanghai Jiao Tong University, China, in 2007. She is currently pursuing an M.E. degree at the Graduate School of Information, Production and Systems of Waseda University. Her interest is in VLSI implementations for video coding technologies.

Jiayi Zhu received B.E. and M.E. degrees from Shanghai Jiao Tong University, at the Department of Electronic Engineering in 2006, and the School of Microelectronics in 2009, respectively. Currently he is working toward a Ph.D. degree at the Department of Electronic Engineering of Shanghai Jiao Tong University. His interest is in VLSI implementations for video coding technologies.

Satoshi Goto received B.E. and M.E. degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970, respectively. He received the Doctor of Engineering degree from the same university in 1981. He joined NEC Laboratories in 1970 where he worked on LSI design, multimedia systems and software as GM and VP. Since 2003, he has been a professor at the Graduate School of Information, Production and Systems of Waseda University. Currently, his main interest is in VLSI design methodologies for multimedia and mobile applications. He has published 7 books, 38 journal papers and 67 international conference papers. He served as GC of ICCAD and ASPDAC and was a board member of IEEE CAS society. He is IEEE Fellow and Member of Academy Engineering Society of Japan.