IEICE Electronics Express, Vol.16, No.22, 1–6

LETTER

Five-stage pipelined dual-edge deblocking filter architecture for H.265 video codec

Prayline Rajabai Christopher[1] **and Sivanantham Sathasivam**[1a)]

Abstract High-Efficiency Video Coding (HEVC/H.265) is the latest video coding standard used in various applications. In HEVC, the quality of the reconstructed video is enhanced by two in-loop filters (Deblocking filter and Sample Adaptive Offset filter). In this paper, we propose an efficient resource sharing hardware architecture for deblocking filter. This architecture utilizes four edge filters to filter two edges of the 8×8 block in parallel. The edges are filtered following the sequential filter ordering. The proposed architecture is implemented in Verilog HDL and synthesized using Synopsys DC. The simulation results show that this architecture can process a 16×16 block in 45 clock cycles and hence an LCU in 720 clock cycles. It utilizes an area of 93.5 K in 32 nm technology supporting UHD video which is suitable for real-time applications.

Keywords: deblocking filter, resource sharing, HEVC/H.265, hardware architecture, video codec

Classification: Integrated circuits

1. Introduction

Tremendous advancements in the electronics industry increased the production of data in the form of images and videos. 2.4 MB data is required to represent a single frame of a color image of resolution 1028×768 (0.8 Megapixel), and 25 to 30 frames per second has to be streamed to view a video/moving image without flicker which approximately requires a minimum of 60 MB of data per second. To stream a video for one minute, we would require 3.6 GB of video data to be stored, which is impossible. Hence various compression standards were introduced to compress the image/video data, which will either be stored in a storage device or transmitted through the transmission system. H.264 and H.265 are the compression standards employed to compress the video data in many applications in recent years [1]. Both the compression standards utilize adaptive deblocking filter in both the encoder and decoder to remove the blocking artifacts and thus to improve the subjective quality of the video [2, 3]. Many hardware architectures of deblocking filters are implemented for both H.264 and H.265 coding standards, which contributes to the optimization of the video codec. Deblocking filtering for H.264 is performed on 4×4 block edges, and the deblocking filter for H.265 is performed on 8×8 blocks [4]. Novel processing order of the block edges, memory access schemes, and the number of filter units contribute to the optimization of the architecture. Most existing architectures are based on a single filter unit for both horizontal and vertical filtering, and the use of different fast memory accessing techniques to satisfy the real-time constraints. Few architectures use parallel filter units for simultaneous filtering for horizontal and/or vertical edges. Dual standard deblocking filter architecture supporting both H.264 and H.265 coding standard is also implemented in [2] and [5]. In the design of dual standard deblocking filter architecture, common hardware components of H.264 and H.265 is reused to reduce the power consumption.

2. Related works

In HEVC coding standard, two in-loop filters, de-blocking filter and Sample Adaptive Offset (SAO) filter are applied in sequence to the reconstructed frames in order to increase the perceptual quality of the video frames. Deblocking filter is used to remove the blocking artifacts and the SAO filter adds offset values, either edge offset or band offset, to the deblocked pixel samples to improve the visual quality [6]. Hardware architectures of the in-loop filters in HEVC are implemented either as a combined deblocking and SAO filters or individually as a deblocking filter and SAO filter. The deblocking filter architectures are implemented using parallel and pipelining fashion in order to optimize the area and throughput. In few architectures, the novel filter ordering is proposed to improve the performance. Different architectures are implemented in [5, 7, 8, 9, 10, 11, 12, 13, 14] to realize the deblocking filter of H.265 coding standard in hardware. It is seen that the complexity of H.265 deblocking filter architecture is less compared to the H.264 deblocking filter architecture [15]. Combined deblocking and SAO filter are implemented in [8, 14, 16, 17, 18]. In [13, 19] the in-loop filtering is implemented in parallel fashion on graphics processing unit (GPU). The HEVC in-loop filtering is implemented in multicore co-processor in [10]. A Convolutional Neural Network (CNN) based in-loop filter with coding unit classification is implemented in [20, 21]. Among the two in-loop filters in HEVC, SAO filter alone is implemented in [22] and deblocking filter alone is implemented in [5, 23, 24, 25, 26, 27, 28, 29]. The architecture implemented in [24] is a multi-parallel architecture built with four parallel filtering cores along with boundary judgment. This architecture can filter an LCU in 288 clock cycles with parallel vertical and horizontal edge filtering order. However, it

[1]School of Electronics Engineering, Vellore Institute of Technology, Vellore 632014, Tamil Nadu, India
a) ssivanantham@vit.ac.in

DOI: 10.1587/elex.16.20190500
Received August 3, 2019
Accepted September 17, 2019
Publicized November 7, 2019
Copyedited November 25, 2019

uses nine SRAM modules, and the area utilization of one filter core with the boundary strength calculation without boundary judgment module is 31.58 K. Since four such filter cores are used, the area utilization is extremely high. The area utilization in this architecture can be greatly reduced by the resource sharing of common units by exploiting the filtering equations. This architecture achieves a target frequency of 278 MHz, which is suitable for real-time applications. Deblocking filter implemented in [23], filters an LCU in 768 clock cycles with the area complexity of 466.5 K. This architecture uses a novel memory restructuring and data access schemes based on the proposed edge filter order. Though it is claimed that it achieves high throughput, more efficient architectures are studied in the literature. Also, the design complexity is very high in this architecture. Deblocking filter architectures supporting dual-standard (both H.264 and H.265) are implemented in [2] and [5]. Even though H.265 is the successor of H.264, these architectures are implemented to provide backward compatibility and to support applications using H.264 coding standard.

3. Five-stage pipelined dual parallel edge deblocking filter (V-DPEDBF)

The proposed V-DPEDBF architecture is shown in Fig. 1 has three main units, viz i) Control unit, ii) Boundary Strength (BS) Calculation unit and iii) Filter unit. The control unit is responsible for controlling all the operations of the deblocking filter. The control unit is enabled when it receives the deblocking filter enable signal from the external world. All the operations, such as to read data from the external memory, to read data from the internal memory, write data to the internal memory and write data to the external memory, enabling the BS calculation unit and enabling the filter unit are monitored and administered by the control unit. The BS calculation unit is responsible for calculating the boundary strength parameter values, which ranges from 0 to 2. Based on the BS value, the filter unit performs the filtering operation.

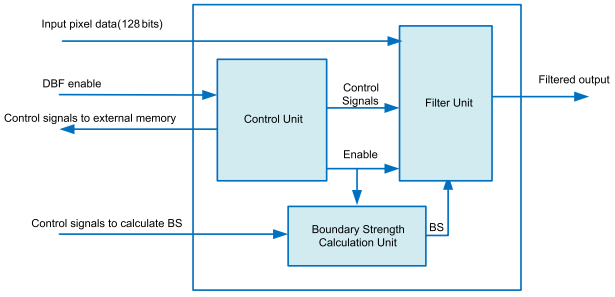

Fig. 1. Top-level architectural diagram of V-DPEDBF for H.265.

3.1 Control unit

The control unit of the H.265 deblocking filter has a state machine to control all the deblocking filtering operation. The control unit is enabled by an enable signal from the external unit. When this signal is low, all the filtering process is turned off, and thus the deblocking filter is in power save mode. When this signal goes high, the control unit enables the deblocking filter operation by triggering the state machine. Deblocking filter operation is performed in five stages, such as memory read, parameter calculation, filter decision, filter, and memory write. Initially, the state machine generates a control signal to read the pixel data from the external memory. The data from the external memory is read as 4×4 blocks (128 bits) per clock cycle. The data read ordering from the external memory for an LCU is shown in Fig. 2(a) and the data read ordering from the external memory for a 16×16 block is shown in Fig. 2(b) and Fig. 2(c). Once four 4×4 blocks of data are ready, the state machine generates control signals to enable the BS calculation unit and the filter unit.

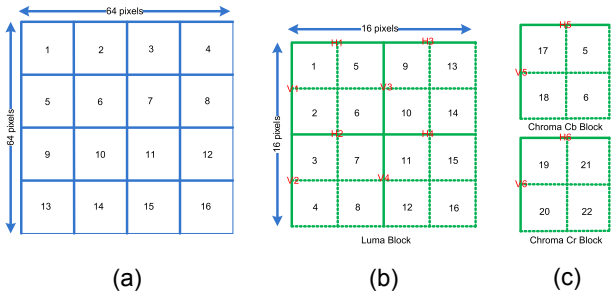

Fig. 2. (a) Read ordering of an LCU from external memory (b), (c) Read ordering from external memory for a 16×16 block.

Based on the calculated BS value, the vertical edges V1 and V2 of the 16×16 Luma block are filtered in parallel, and the filtered data is transposed and written to the internal memory (blocks 1–8). Once the twelfth block of data is available, the filter unit is triggered again to filter the edges V3 and V4 and the vertically filtered data is transposed and written to the internal memory (blocks 9–16). The control unit then generates control signals to read the data from the internal memory for horizontal filtering. The horizontal edges H1 and H2 are filtered in parallel and transposed before being written into the external frame memory. Then the edges H3 and H4 are filtered in parallel, transposed and is written to the external frame buffer. The same approach is followed for the Chroma Cb and Cr blocks to perform the vertical filtering followed by the horizontal filtering of the edges V5, V6, H5, and H6. The filtered data is finally written to the external memory as 4×4 blocks, i.e., 128 bits per clock cycle.

3.2 BS calculation unit

The BS calculation unit reckons the boundary strength value based on the received control signals to compute the BS. Based on the control signals, the BS value is computed as in Fig. 3. The evaluated BS value ranges from 0 to 2 where the value 0 indicates no filtering; value 1 stipulates the use of weak/normal filter and the value 2 specifies the use of strong filter.

- If the pixel block of data read from the external memory is from the edge of the frame, either the left edge or the top edge, then the BS value is 0.
- If the data is not from the frame edge and if the two adjacent 8×8 blocks are not intra-coded and the two

adjacent blocks do not have non-zero transform coefficients and if the difference of the motion vector is less than 4, then the BS value is 0.

- If the data is not from the frame edge and if the two adjacent 8×8 blocks are not intra-coded and the two adjacent blocks do not have non-zero transform coefficients and if the difference of the motion vector is greater than or equal to 4, then the BS value is 1.
- If the data is not from the frame edge and if the two adjacent 8×8 blocks are not intra-coded and the two adjacent blocks do have non-zero transform co-efficient, then the BS value is 1.
- If the data is not from the frame edge and if the two adjacent 8×8 blocks are intra-coded, then the BS value is 2.

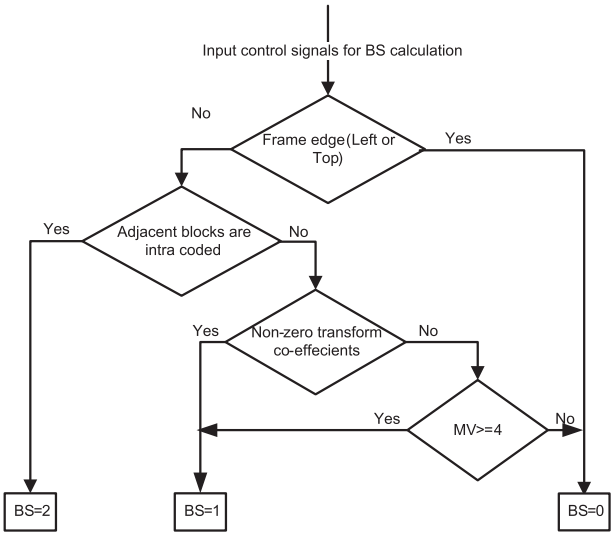

Fig. 3. Boundary strength computation.

3.3 Filter unit

The filter unit is the sophisticated computational unit which is enabled by the control unit once the data is ready for the filtering operation. The architectural diagram of the filter unit is shown in Fig. 4. The filter unit encompasses i) Parameter Calculation unit, ii) Buffers, iii) Filter decision unit, iv) Internal Memory unit and v) Filter modules viz., strong filter and weak filter.

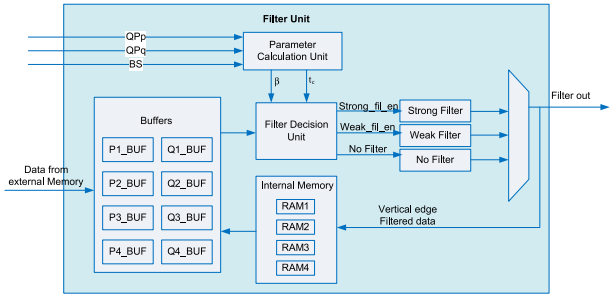

Fig. 4. Filter unit of V-DPEDBF architecture.

3.3.1 Parameter calculation unit

This block is used to compute the filtering parameters like β and t_c based on table 8–12 of [30]. These parameters are dependent on the BS value and the quantization parameter values of the adjacent P and Q blocks named as QP_p and QP_q respectively. Parameter Calculation unit is a LUT that has the outputs β and t_c which are dependent on the inputs BS, QP_p and QP_q.

3.3.2 Buffers

There are eight buffers used in the filter unit, and each buffer can hold a 4×4 pixel block of data (128 bits). Initially, all the buffers are initialized with zeros. When the control unit starts to read the data from the external memory, the data blocks 1–4 shown in Fig. 2(b) are loaded to the buffers Q1_BUF, Q2_BUF, Q3_BUF, and Q4_BUF respectively and filtering is performed along the vertical edges V1 and V2. Meanwhile, the pixel blocks 5–8 are loaded to the buffers P1_BUF, P2_BUF, P3_BUF, and P4_BUF respectively. Fig. 5 shows the mapping of each 4×4 pixel block to the corresponding buffer for both luma and chroma blocks. On completion of the vertical edge filtering of the edges V1 and V2, the filtered data is written into the internal memory. The same strategy is followed for filtering the edges V3 and V4. The data from the internal memory is again loaded to these buffers for horizontal filtering following the same technique as in vertical filtering.

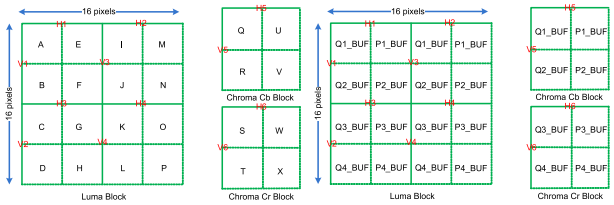

Fig. 5. Pixel block to buffer mapping.

3.3.3 Filter decision unit

The filter decision unit decides the type of filtering that has to be applied for a 4×4 block based on the parameters β, t_c and the pixel threshold values of the two adjacent blocks.

3.3.4 Internal memory unit

The internal memory has four dual-port RAM, which is used to hold the vertically filtered data. The size of each RAM is 64 bytes, which is divided into four segments, where each segment can hold a 4×4 block of data (128 bits or 16 bytes). Each 4×4 vertically filtered pixel data blocks is stored in these four RAMs. Fig. 6 shows the mapping of the data from the internal memory to the internal buffers for both luma and the chroma Cb, Cr blocks. This novel data storing technique reduces the access cycles of the external memory and circumvents the use of transpose buffers. For chroma blocks, only the first two memory locations are used, and the remaining two locations are unused.

3.3.5 Filter modules

The filter modules have strong filter and weak/normal filter. Based on the filtering decisions from the filter decision unit, either the strong filter or the weak filter is enabled to perform the filtering operation. In case if the filtering

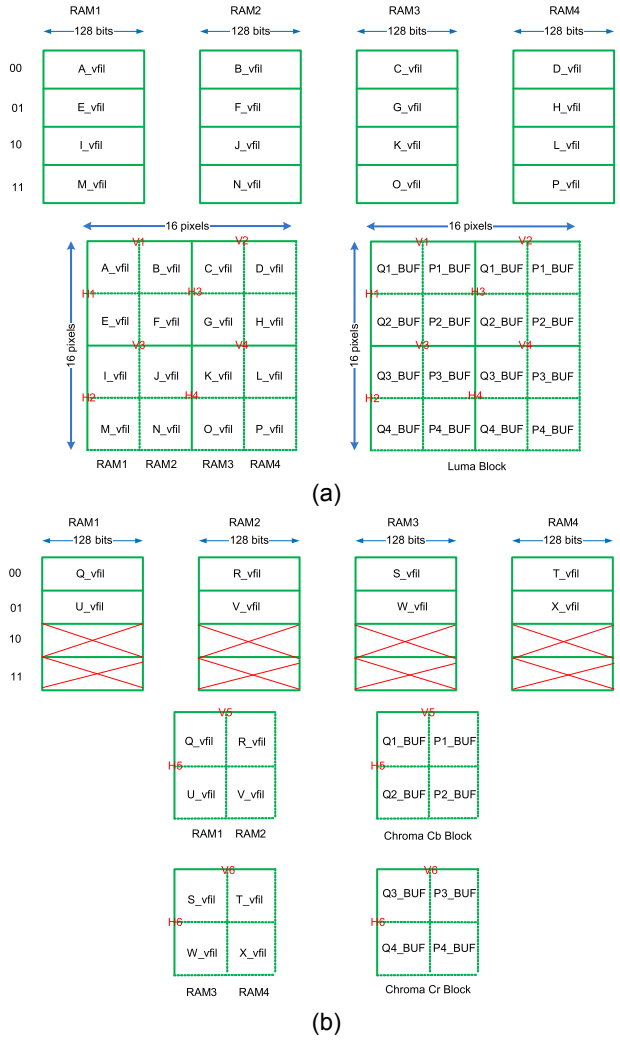

Fig. 6. Internal memory to buffer mapping before horizontal filtering.

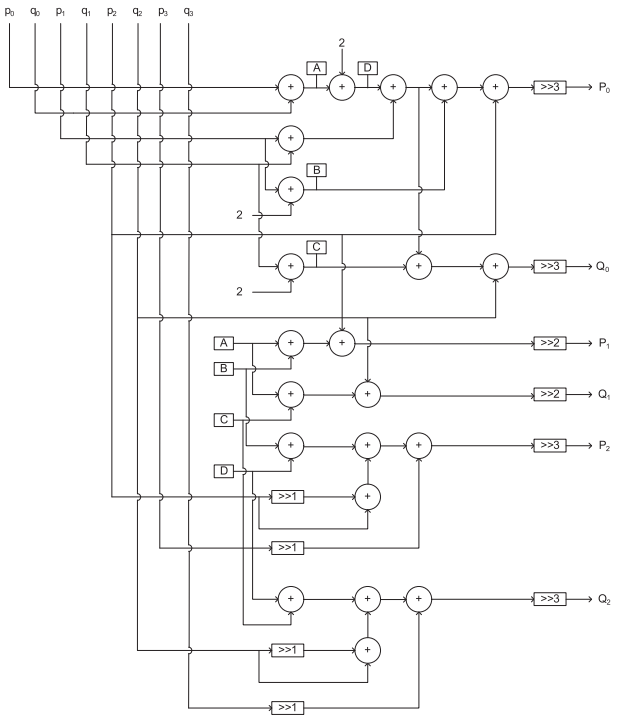

Fig. 7. The partial architecture of strong filter using resource sharing.

decision unit decides no filtering, then the filtering process is bypassed. The strong or weak filtering is performed based on the equations specified in [30]. By exploiting the similarities in the equations, a resource sharing architecture of the filter module is designed to optimize the area. The filtered pixel data block is then written to the internal memory after vertical filtering and to the external memory after the horizontal filtering.

3.4 Resource sharing architecture

The strong and the weak filter are designed based on the deblocking filtering algorithm given in [23] complies with the HEVC standard. Based on the similarities in the filtering equations, the filter architecture is designed to share the common resources which subsequently reduces the area and hence the power. The clip3 function is to modify the pixels of the two adjacent blocks for the strong filter. The third argument of the clip3 function is an equation which involves two or more of the pixel values of the adjacent P and Q blocks. These equations are implemented in the hardware using adders and shifters. It is noted that $p_0 + q_0$ is used in all the equations and $p_0 + q_0 + 2$ is used in most of the equations. Also, $p_1 + 2$ and $q_1 + 2$ are used twice. Hence to add $p_0 + q_0$ one 8 bit adder is used and the output is shared to all the equations, and the output of this adder is provided as input to another adder whose other input is 2 which gives the output $p_0 + q_0 + 2$ and this value is used in the respective equations. Similarly, $p_1 + 2$ and $q_1 + 2$ are implemented using two more adders and the output of these adders are used wherever required. Thus by sharing the common resources, an area-efficient filter is implemented. The resource sharing architecture of the third argument of the clip3 function for the strong filter is shown in Fig. 7.

4. Results and discussion

The proposed architecture is implemented using Verilog HDL and is synthesized using Synopsys DC targeting for 90 nm and 32 nm technology library. Table I shows the area utilization of various modules, and Table II shows the summary of the hardware implementation results. The implemented architecture has an area utilization of 120.6 K with 90 nm technology library and 93.5 K with 32 nm technology library. It uses four dual-port RAM of size 64 bytes each. This architecture, with few internal buffers and a novel internal memory organization strategy, avoids the usage of transpose buffers and reduces the external memory access cycles and hence achieves optimized performance. The simulation results show that the implemented architecture can process a 16×16 block in 45 clock cycles. The filtering operation performed on every clock cycle is shown in Fig. 8. It takes 35 clock cycles to filter the Luma block and 20 clock cycles to filter the Chroma Cb and Cr blocks with an overlap of 10 clock cycles. The input data path is set to 128 bits to read a 4×4 block of data per cycle. In a 16×16 block, there are sixteen 4×4 blocks, and hence it requires 16 clock cycles to read a 16×16 Luma block from the external memory. As filtering is done

along the edges of 8 × 8 blocks, vertical filtering is done immediately on receiving the 4th block and then after receiving the 12th-pixel block. The filtered block of data is transposed and written into the internal memory, which is then fetched again for horizontal filtering. It takes four clock cycles to read the data from the internal memory and 16 clock cycles to write the filtered data to the external memory. Overall it takes 35 clock cycles to filter the luma block. Similarly, for the chroma Cb and Cr blocks, eight clock cycles are required to read eight 4 × 4 blocks of data from the external memory, and once the blocks are filtered, the data is transposed and written into the internal memory. It takes two clock cycles to read the data from the internal memory and eight clock cycles to filter the data and write the data to the external memory. Thus it takes 20 clock cycles to filter the chroma block. Chroma block filtering is performed only if the calculated BS value is equal to 2. If the BS value is less than 2, then the skip mode is selected, and the overall filtering of a 16 × 16 block is done within 35 clock cycles. Table III shows the comparison of this work with the previous architectures. Throughput for V-DPEDBF architecture is computed as in Eq. (1).

$$\textit{Throughput}\ (\text{kLCU/s}) = \frac{\textit{Frequency}\ (\text{kHz})}{\textit{Processing time}\ (\text{cycles/LCU})} \quad (1)$$

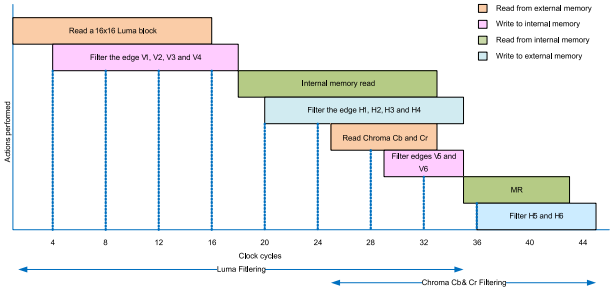

Fig. 8. Clock cycle distribution to filter a 16 × 16 block.

Table I. Area utilization of various modules

Module	Area (K) [90 nm]	Area (K) [32 nm]
Control unit	1.012	0.227
BS Calculation unit	0.067	0.0153
Filter Decision unit	34.9	6.458
Parameter Calculation unit	0.223	0.323
Strong Filter unit	144.965	22.924
Weak Filter unit	36.842	7.737
Total Area (Including buffers)	120.6	93.5

Table II. Hardware implementation results

Parameter	90 nm	32 nm
Area (K)	120.6	93.5
Frequency (MHz)	100	333.33
Power (mW)	12.25	12.01
Processing Time Cycles/LCU	45 × 16 = 720	45 × 16 = 720
Throughput (kLCU/s)	139	463

It is noted that the throughput of DPEDBF is 463 kLCU/s in 32 nm technology and 139 kLCU/s in 90 nm technology. The throughput achieved in 90 nm technology is marginally higher than the throughput of the architecture implemented in [23]. Though the architecture implemented in [18] achieves high throughput, the area and power consumption are very high. Our proposed V-DPEDBF achieves 79.67% lesser area and 96.2% lesser power with the compensation in the throughput, which is 93.3% lesser compared to [18]. The architecture implemented in 32 nm technology achieves a frequency of 333.33 MHz with the target area of 93.5 K, which shows the scope of this architecture for real-time applications in hand-held electronic gadgets.

Table III. Comparison with previous architectures

Parameter	[18]	[23]	Proposed	
Technology	90	90	90	32
Processing Order	Sequential	Hybrid	Sequential	
RAM size	32 × 32	393	256	256
Area (K)	593.32	466.5	120.6	93.5
Processing Time Cycles/LCU	172	768	720	720
Throughput (kLCU/s)	2133	130	139	463
Level of parallelism	4	2-line	4	4
Pipeline stages	5	6	5	5
Frequency (MHz)	366.96	100	100	333.33
Power (mW)	339.83	12.76	12.25	12.01
Transpose Buffers	Yes	Yes	No	No
Resolution	8k × 4k	4k × 2k	8k × 4k	8k × 4k

5. Conclusion

This work proposes a parallel deblocking filter architecture for the H.265 coding standard with five pipeline stages. This architecture can filter the edges of two 8 × 8 blocks in parallel and filters the luma block in 35 clock cycles and the chroma Cb and Cr blocks in 20 clock cycles with the overlap of 10 clock cycles. So the overall processing cycles for a 16 × 16 block is 45 clock cycles, and hence it requires 720 clock cycles to process an LCU. The implemented architecture can operate at a frequency of 200 MHz, and hence, it can support UHD applications.

References

[1] G. J. Sullivan, *et al.*: "Overview of the high efficiency video coding (hevc) standard," IEEE Trans. Circuits Syst. Video Technol. **22** (2012) 1649 (DOI: 10.1109/TCSVT.2012.2221191).
[2] B. K. N. Srinivasarao, *et al.*: "High-speed low-power very-large-scale integration architecture for dual-standard deblocking filter," IET Circuits Dev. Syst. **9** (2015) 377 (DOI: 10.1049/iet-cds.2014.0310).
[3] H. Lei, *et al.*: "Parallel deblocking filter based on modified order of accessing the coding tree units for hevc on multicore processor," KSII Trans. Internet Inf. Syst. **11** (2017) 1684 (DOI: 10.3837/tiis.2017.03.024).

[4] K. Maiti, *et al.*: "Efficient deblocking filter implementation on reconfigurable processor," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2016) 1050 (DOI: 10.1109/ICASSP.2016.7471836).
[5] M. Li, *et al.*: "De-blocking filter design for hevc and h.264/avc," Pacific Rim Conference on Multimedia (PCM) (2012) 273 (DOI: 10.1007/978-3-642-34778-8_25).
[6] Y. Wang, *et al.*: "Gpu-based optimization for sample adaptive offset in hevc," IEEE International Conference on Image Processing (ICIP) (2016) 829 (DOI: 10.1109/ICIP.2016.7532473).
[7] W. Shen, *et al.*: "A high-throughput vlsi architecture for deblocking filter in hevc," IEEE International Symposium on Circuits and Systems (ISCAS) (2013) 673 (DOI: 10.1109/ISCAS.2013.6571936).
[8] S. Shen, *et al.*: "A pipelined vlsi architecture for sample adaptive offset (SAO) filter and deblocking filter of hevc," IEICE Electron. Express **10** (2013) 20130272 (DOI: 10.1587/elex.10.20130272).
[9] E. Ozcan, *et al.*: "A high performance deblocking filter hardware for high efficiency video coding," IEEE Trans. Consum. Electron. **59** (2013) 714 (DOI: 10.1109/TCE.2013.6626260).
[10] I. Hautala, *et al.*: "Programmable low-power multicore coprocessor architecture for hevc/h.265 in-loop filtering," IEEE Trans. Circuits Syst. Video Technol. **25** (2014) 1217 (DOI: 10.1109/TCSVT.2014.2369744).
[11] C. Yan, *et al.*: "Parallel deblocking filter for hevc on many-core processor," Electron. Lett. **50** (2014) 367 (DOI: 10.1049/el.2013.3235).
[12] M. Mody, *et al.*: "High throughput vlsi architecture supporting hevc loop filter for ultra hdtv," IEEE Third International Conference on Consumer Electronics (ICCE-Berlin) (2013) 54 (DOI: 10.1109/ICCE-Berlin.2013.6698026).
[13] Y. Wang, *et al.*: "Parallel in-loop filtering in hevc encoder on gpu," IEEE Trans. Consum. Electron. **64** (2018) 276 (DOI: 10.1109/TCE.2018.2867812).
[14] H. Kim, *et al.*: "An efficient architecture of in-loop filters for multicore scalable hevc hardware decoders," IEEE Trans. Multimed. **20** (2017) 810 (DOI: 10.1109/TMM.2017.2759506).
[15] A. M. Kotra, *et al.*: "Comparison of different parallel implementations for deblocking filter of hevc," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2013) 2721 (DOI: 10.1109/ICASSP.2013.6638151).
[16] L. Liu, *et al.*: "Implementation of in-loop filter for hevc decoder on reconfigurable processor," IET Image Process. **11** (2017) 685 (DOI: 10.1049/iet-ipr.2016.0143).
[17] W. Shen, *et al.*: "A combined deblocking filter and sao hardware architecture for hevc," IEEE Trans. Multimed. **18** (2016) 1022 (DOI: 10.1109/TMM.2016.2532606).
[18] S. Baldev, *et al.*: "Design and implementation of efficient streaming deblocking and sao filter for hevc decoder," IEEE Trans. Consum. Electron. **64** (2018) 127 (DOI: 10.1109/TCE.2018.2812518).
[19] W. Jiang, *et al.*: "A novel parallel deblocking filtering strategy for hevc/h.265 based on gpu," Concurr. Comput. **28** (2016) 4264 (DOI: 10.1002/cpe.3751).
[20] Y. Dai, *et al.*: "A cnn-based in-loop filter with cu classification for hevc," IEEE Visual Communications and Image Processing (VCIP) (2018) 1 (DOI: 10.1109/VCIP.2018.8698616).
[21] W.-S. Park and M. Kim: "Cnn-based in-loop filtering for coding efficiency improvement," IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP) (2016) 1 (DOI: 10.1109/IVMSPW.2016.7528223).
[22] S. Park and K. Ryoo: "Hardware design of hevc in-loop filter for ultra-hd video encoding," Lecture Notes in Electrical Engineering **518** (2019) 405 (DOI: 10.1007/978-981-13-1328-8_52).
[23] P.-K. Hsu and C.-A. Shen: "The vlsi architecture of a highly efficient deblocking filter for hevc systems," IEEE Trans. Circuits Syst. Video Technol. **27** (2017) 1091 (DOI: 10.1109/TCSVT.2016.2515306).
[24] W. Zhou, *et al.*: "A high-throughput and multi-parallel vlsi architecture for hevc deblocking filter," IEEE Trans. Multimed. **18** (2016) 1034 (DOI: 10.1109/TMM.2016.2537217).
[25] X. Ye, *et al.*: "A cost-efficient hardware architecture of deblocking filter in hevc," IEEE Visual Communications and Image Processing Conference (VCIP) (2014) 209 (DOI: 10.1109/VCIP.2014.7051541).
[26] J. Bae: "Register array-based VLSI architecture of H.265/HEVC loop filter," IEICE Electron. Express **10** (2013) 20130161 (DOI: 10.1587/elex.10.20130161).
[27] G. Tang, *et al.*: "An sram-free hevc deblocking filter vlsi architecture for 8k application," IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT) (2018) 1 (DOI: 10.1109/ICSICT.2018.8565040).
[28] R. Peesapati, *et al.*: "Design of streaming deblocking filter for hevc decoder," IEEE Trans. Consum. Electron. **63** (2017) 1 (DOI: 10.1109/TCE.2017.014949).
[29] L. Jiang, *et al.*: "A parallel implementation of deblocking filter based on video array architecture for hevc," IEEE Seventh International Green and Sustainable Computing Conference (IGSC) (2016) 1 (DOI: 10.1109/IGCC.2016.7892592).
[30] I.-T. H.265, "Series h: Audiovisual and multimedia systems - infrastructure of audiovisual services - coding of moving video, high efficiency video coding," Recommendation ITU-T H.265 (2018) http://handle.itu.int/11.1002/1000/13433.