IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 21, 2019, accepted July 16, 2019, date of publication July 22, 2019, date of current version August 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930149

High-Throughput Deblocking Filter Architecture Using Quad Parallel Edge Filter for H.264 Video Coding Systems

PRAYLINE RAJABAI C. AND SIVANANTHAM S., (Senior Member, IEEE)
School of Electronics Engineering, Vellore Institute of Technology, Vellore 632014, India
Corresponding author: Sivanantham S. (ssivanantham@vit.ac.in)

ABSTRACT With the increasing demand in electronic gadgets expecting better video quality for multimedia applications, various coding standards evolved for the past two decades and optimization on the architectures of the various modules used in the video codec is most popular. In this paper, an efficient architecture for deblocking filter used to smoothen the pixels of the decompressed video data is proposed, which utilizes both pipelining and parallelism. The filtering process follows a sequential order as filtering vertical edges of luma block and chroma block followed by the horizontal edges of the luma block and chroma block. Three pipeline stages are used and four edges, either vertical or horizontal are filtered in parallel. Internal buffers which hold the sub-blocks read from the external frame buffers are accessed in a ping pong fashion to filter the adjacent sub-edges and thus reducing the external memory access cycles. Due to parallelism with novel edge filtering order, self-transposing mechanism, and ping pong buffer access, the throughput is increased. The proposed quad parallel edge deblocking filter architecture is implemented using Synopsys 90 nm library. It achieves a target area of 19.8 K and can process a Macro Block in 58 clock cycles.

INDEX TERMS Deblocking filter, H.264/AVC, throughput, VLSI architecture, video compression.

I. INTRODUCTION

Repercussions of emerging trends and advancements in the field of video technology and the electronics industry for the past two decades increased the amount of image/video data produced from the still-image/video camera. This image/video data has to be either stored or transmitted based on the application requirement. In both cases, there is a strong requirement to compress the data since the size of the video data is very huge. Research on video compression started late in the 1920s [1] and due to the gradual advancements in video technology extensive research on video compression started in early 1980s, and still, research on video compression is going on to efficiently optimize the compression and decompression standard of the video data. The Joint Video Team/ Joint Collaborative Team, collaboration of the Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T) Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) introduced the two video coding standards H.264 and H.265 which will meet the cutting edge requirements of the video technologies in the electronics industry. Consumers of electronic gadgets always expect the end-product to be operated at very high speed, low-power, and with much lesser complexity. Hence a highly efficient video codec which can be suitably operated based on the customers/end-users requirement is inevitable.

Pixels are the pieces of information on the color and intensity at a particular point in an image. The collection of pixels constitute an image and the collection of images over a period of time is known as video. These pixels are highly correlated spatially within an image frame and for a video, pixels are highly correlated with respect to time since the video is the collection of frames over a period of time. Hence a video data is both spatially and temporally correlated. Compression of the video data is done by exploiting the spatial and temporal redundancies. Fig. 1 shows the block diagram of a video codec. The various blocks involved in the codec are the transformation unit, quantization unit, inverse quantization unit, entropy coder, inverse transform unit, deblocking filter, motion estimation and motion compensation unit. Deblocking filter is one of the most critical units among

The associate editor coordinating the review of this manuscript and approving it for publication was Yun Zhang.

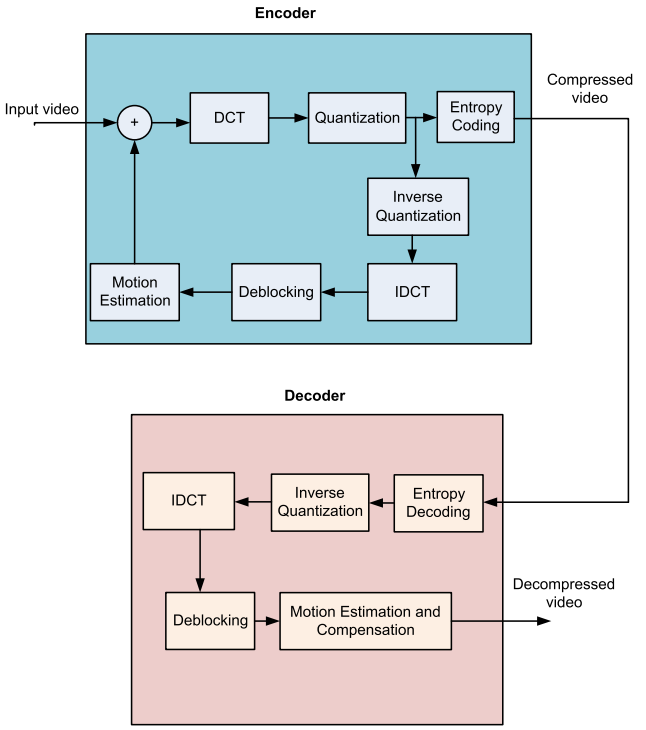

FIGURE 1. Block diagram of video codec.

the various blocks since it involves complex computations consuming around one-third of the computational complexity [2] compared to other blocks also it requires frequent access of external memories for filtering operations.

II. DEBLOCKING FILTER ALGORITHM FOR H.264/AVC

The H.264/AVC video coding standard utilizes the lossy compression technique of block-based Discrete Cosine Transform (DCT). Due to this, the reconstructed video frames appear to be rough, having a visual effect of non-smooth block boundaries in the image frame. Deblocking filter algorithm is used to improve the visual quality of the decompressed image. Deblocking filter is used both in the coder (prediction) and decoder (decompression). In any video codec, the video frames are split into non-overlapping blocks for processing. In H.264 coding standard, the frames are split into Macro Blocks (MB) of size 16×16 for luma and 8×8 for chroma Cb and Chroma Cr. These MBs are further divided into smaller sub-blocks of size 4×4. Deblocking filter is applied to the reconstructed video frames to improve the visual quality of the video where the vertical edges of every 4×4 block in a MB is filtered followed by the horizontal edges of these sub-blocks [3]. The luma MB is first filtered both vertically and horizontally, followed by Chroma Cb and Chroma Cr [4]. Thus all the MBs in a frame are filtered in raster scan order. Filtering decision and the filtering strength of a DBF are decided based on the Boundary Strength (BS), α and β [5]. The BS value for H.264 coding standard ranges from 0 - 4 where the value 0 indicates no filtering, values 1 - 3 denote weak filtering and 4 denotes strong filtering to be performed.

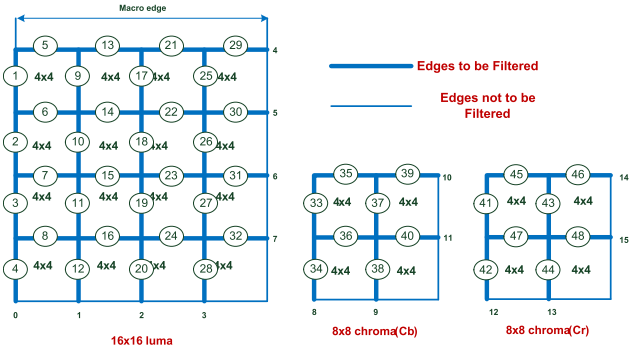

FIGURE 2. Basic filtering order for H.264/AVC video coding standard.

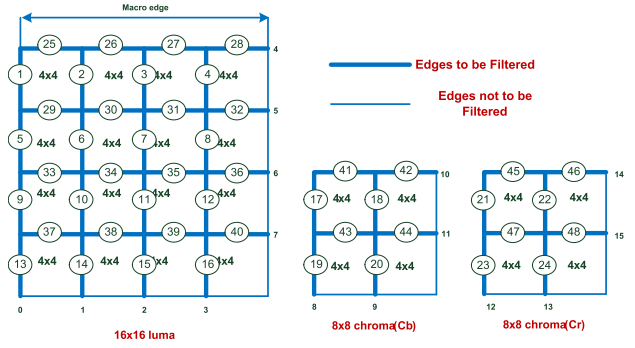

FIGURE 3. Sequential filtering order.

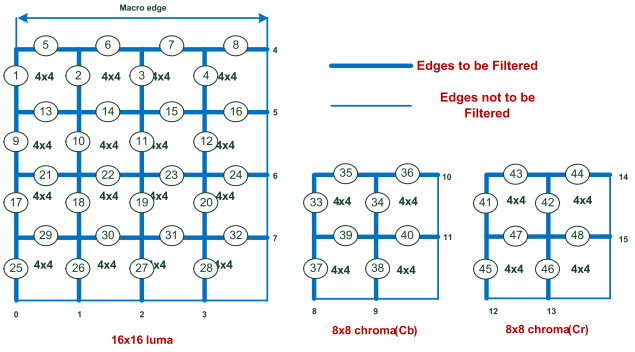

FIGURE 4. Hybrid filtering order implemented in [6].

In the literature, various filtering orders are studied. Fig. 2 shows the basic filtering order, which is the standard filtering order for a Macro Block in H.264/AVC video coding standard for the Luma and the Chroma Cb and Cr blocks. However, the filtering orders can be classified into two categories as i) Sequential and ii) Hybrid. In sequential filtering order shown in Fig. 3, the vertical edges are filtered first followed by the horizontal edges, whereas in hybrid filtering order shown in Fig. 4, vertical and horizontal edges are filtered in a mixed fashion. Filtering order profoundly affects the throughput of the DBF architecture. The throughput of the DBF architecture can be calculated as in (1).

$$Throughput\ (kMB/s) = \frac{Frequency\ (kHz)}{Procesing\ time\ (cycles/MB)} \quad (1)$$

III. REVIEW OF DEBLOCKING FILTER ARCHITECTURES

Deblocking filters of various coding standards are implemented in hardware and optimizations of the filter architecture is still under research to render video with the highest degree of quality to the end-user. Quality of video data can be measured by mean squared error (MSE) and peak signal-to-noise ratio (PSNR) as fidelity metrics. Also, PSNR and MSE are the data metrics which are not sure on the distortion or the content. Though with sound values of PSNR and MSE, it is difficult to justify the quality of the decompressed/decoded video data. Human Visual System (HVS) varies from one viewer to another viewer. Distortions may appear more for one viewer and less for another viewer [7]. Also, the video data has to be streamed at a proper rate so that the video can be viewed without flickering. So, to render the video data without flickers, the computations involved in various blocks have to be done at a faster rate or high speed for real-time video applications. Deblocking filter is one of the blocks which improves the video quality by performing complex computations and frequent memory access. The computational complexity of the Deblocking filter for H.264/AVC is one-third of the whole decoder [8].

In literature, hardware architectures of the deblocking filter for H.264/AVC are implemented with different levels of pipelining and parallelism. The design of the hardware architecture plays a significant role in the computational complexity, area and the throughput of the codec. A highly efficient architecture can perform the filtering operations with minimum computations and low latency, which improvises the throughput occupying a lesser area, consume less power with optimum frequency. In the H.264/AVC video coding standard, the deblocking filtering operation is performed by splitting the reconstructed image frame into blocks of 16×16 called Macro Block (MB). Luminance Block is filtered first, followed by the Chrominance Cb and then Chrominance Cr. Each MB is further split into sixteen 4×4 pixel sub-blocks, and the edges of each 4×4 pixel blocks are filtered as the vertical edges followed by the horizontal edges.

In [9], a memory-efficient architecture is implemented where the system throughput is improved by using a hybrid filtering order and pixel reuse. This architecture utilizes two single-port SRAM of size 96×32 and 2N×32 to store the current block and the neighboring data (N represents the width of the coded frame). The processing cycles per MB is reduced to 250 cycles/MB compared to the architectures implemented in [10], which requires 504 cycles/MB. The architecture based on AMBA is implemented in [11]. It uses efficient memory organization and requires 646 processing cycles/MB, which is high and does not suit for real-time applications. In [12], two-stage pipelining with hybrid filter ordering is implemented in FPGA. At the worst case, it requires 6144 processing cycles/MB and can support only CIF frame (352×288) resolution.

Parallel In Parallel Out (PIPO) style architecture is implemented in [13] by processing four pixels in parallel. The edges of each 4×4 block are scheduled in horizontal-vertical interleaved fashion and thus uses a transpose memory to transpose the pixel data when the filtering edge changes either from horizontal to vertical or from vertical to horizontal. This architecture consumes 300 clock cycles/MB with the area of 13.41K in .25μm technology excluding the dual-port RAM of size 16×32. It also supports real-time video of resolution 2048×1024 at 73.73MHz. Horizontal-vertical interleaved scheduling of 4×4 pixel block edges is also followed in [14]. Due to the hybrid scheduling of edges for deblocking operation, this architecture requires only two buffers to store the intermediate block generated from the filtering block. It also saves the memory access required to process the left, top and the right edge in a 4×4 pixel block and requires only the 4×4 block above the current block to be buffered instead of storing the whole macroblock as in conventional architectures. This architecture requires 232 clock cycles to process a MB. In [4], two identical filters are implemented to process both the horizontal and vertical edges in parallel. Since horizontal and vertical edges are scheduled in parallel, two transpose buffers are used to transpose the block of 4×4 pixels from rows to columns and from columns to rows. This architecture requires 110 clock cycles to process a MB with the architectural area of 13.63K in .25μm technology. A parallel deblocking filter architecture which utilizes six filters is implemented in [15]. A novel filtering order is implemented in this design. Even though it can process filtering of six edges in parallel, all six filters are not simultaneously used fully when processing a MB due to data dependencies. This results in the wastage of resource utilization as well as the architectural area will be huge. Since six filters are used the number of clock cycles to process a MB is considerably less of 49 cycles/MB, and hence it can be used for real-time applications. A five-stage pipeline and single filter architecture is implemented in [16], with hybrid filtering order and efficient memory organization. This architecture required 204 clock cycles to process a MB. Hybrid and sequential filtering order is also implemented with four-stage pipelining in [6]. This architecture requires few clock cycles lesser than [16] to process a MB. Most of the hybrid filtering architectures use two transpose buffers. Scalable deblocking filter architectures are implemented in [17] and [18]. Scalable architectures can support different video configurations by configuration setting or by reconfigurability to support the different demands of the consumer. A novel filter architecture is implemented in [2] which uses 6-stage pipeline architecture, which can filter four edges in parallel and can process a MB in 64 clock cycles.

IV. QUAD PARALLEL EDGE DEBLOCKING FILTER ARCHITECTURE

The Deblocking filter for H.264 is highly adaptive based on the BS parameter, α and β values (α and β depend on the Quantization Parameter (QP)) and clipping threshold (tc or C1). The decision of filtering and the strength of the filter to be applied depends on these parameter values. It is

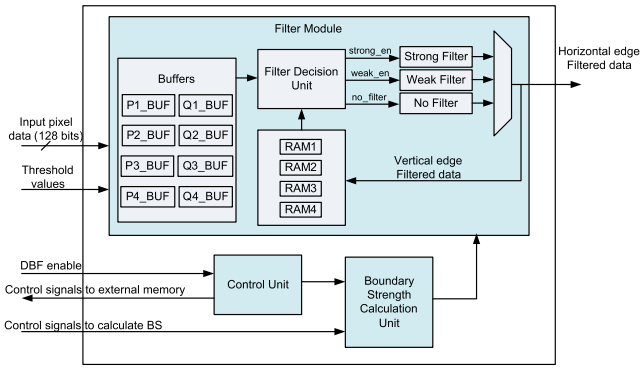

FIGURE 5. Architecure of proposed deblocking filter.

studied from the literature that pipelined architecture with parallel edge filter outperforms compared to other architectures. The Quad Parallel Edge Deblocking Filter(QPEDBF) architecture shown in Fig. 5 has i) BS calculator unit ii) Filter unit iii) 8×16 internal ping pong buffer and a vi) control unit. The filtering operations are performed in three stages as i) Data Read and BS calculation ii) Filter decision iii) filtering of pixels and data write.

A. CONTROL UNIT

This module is used to control the operations of the deblocking filter architecture. The FSM in the control unit generates control signals to enable modules like BS calculator unit and filter unit. It also generates the control signals to read the data from the external memory for vertical edge filtering and to read the data from the internal memory for horizontal edge filtering. All data path is designed to be of 128 bit to enable the read and write of one 4×4 sub-block (16 pixels × 8 bits). The filtering process is controlled by the control unit, as shown in Fig. 6. The control unit controls five different operations performed in three-pipeline stages where the memory read and boundary strength calculation happens at the initial stage, filter decision and filtering of the sub-edges are performed at the second stage and the memory write happens at the third stage.

B. BS CALCULATOR UNIT

The BS calculator unit gets the coding information of the incoming video data like whether the block of pixels are inter/intra coded, whether the block has non-zero transform coefficients or the motion vector is greater than 4 or the frame is coded with the different reference frame. Based on this information, the BS calculator unit calculates the BS value for each edge of a 4×4 block as given in Fig. 7. The BS value for H.264 ranges from 0 - 4 where the value 0 denotes no filtering, 1 - 3 denotes weak filtering, and 4 denotes strong filtering to be applied to the edge of a 4×4 block. The value of BS varies adaptively for each 4×4 block, and the two adjacent 4×4 blocks share the same BS value. The BS value for the MB in the top and the left border of the frame is set to 0.

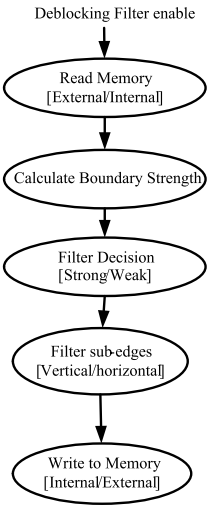

FIGURE 6. Control unit flow chart.

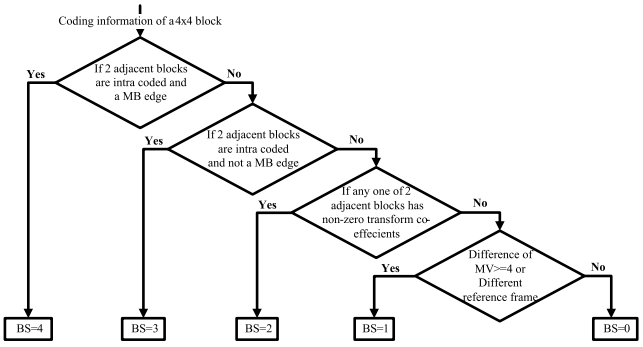

FIGURE 7. Boundary strength computation.

C. FILTER UNIT

The filter unit has buffers, internal dual-port memory and filter modules. Based on the control signal from the control unit, the filter unit enables the corresponding filter module and filters the edges of the sub-blocks based on BS value calculated by the BS calculator unit, threshold values and the pixel values stored in the buffers. In literature, the edges of the MB are filtered with different ordering styles like vertical edges are filtered first followed by horizontal edges or vertical and horizontal edges are filtered parallel in a hybrid fashion. The edges to be filtered is shown in Fig. 8. In this work, we utilize sequential filtering order where four vertical edges or horizontal edges are filtered in parallel, as shown in Fig. 9.

D. FILTER MODULES

The filter modules used in the filter unit, are of two types i) weak filter and ii) strong filter. Based on the BS value and the filter decision equations, the corresponding filter is enabled by the filter unit and the pixels are filtered. The weak filter modifies either one or two pixels and the strong filter modifies up to three pixels in each row on both side of a 4×4 sub-block edge based on the threshold values of α and β.

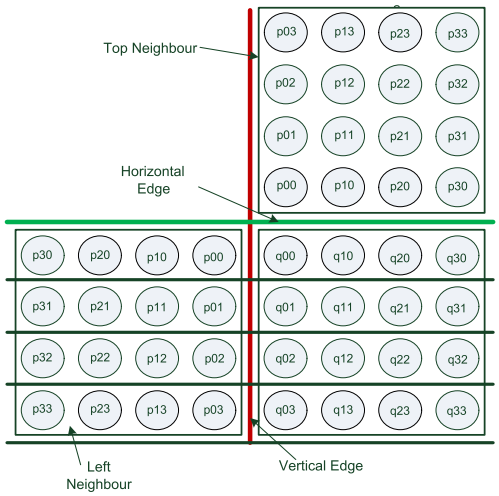

FIGURE 8. Edges involved in Filtering.

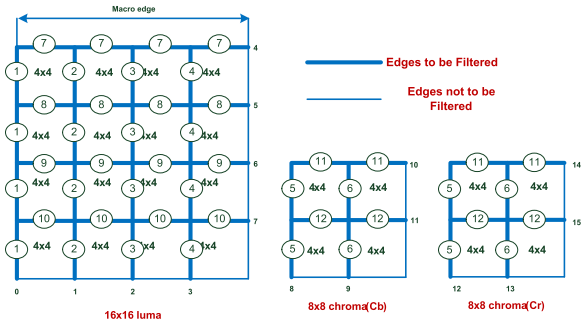

FIGURE 9. Proposed filtering order.

So for a weak filtering maximum of eight pixels are modified and for strong filtering maximum of twelve pixels are modified in a 4×4 block. Therefore, for a 16×16 MB, which has sixteen 4×4 sub-blocks, a maximum of 128 pixels are modified for weak filtering, and 192 pixels are modified for strong filtering and no pixels are modified for no filtering.

1) NO FILTER

Filtering of pixels is performed based on [19]. Pixels are not filtered if BS value is equal to 0 and any one of the equations (2), (3), or (4) is not satisfied.

$$|p_0 - q_0| < \alpha \tag{2}$$
$$|p_1 - p_0| < \beta \tag{3}$$
$$|q_0 - q_0| < \beta \tag{4}$$

2) WEAK FILTER

If all the conditions (2), (3), and (4) are satisfied and if the computed BS is a non-zero value ranging between 1 - 3, then the conditions (5) and (6) are checked. If (5) is satisfied then p_0 and p_1 are modified as in (8) and (9), and if (6) is satisfied then q_0 and q_1 are modified as in (10) and (11). If (5) is not satisfied, then p_0 alone is modified as in (8), and if (6) is not satisfied, then q_0 alone is modified as in (10). Thus for weak

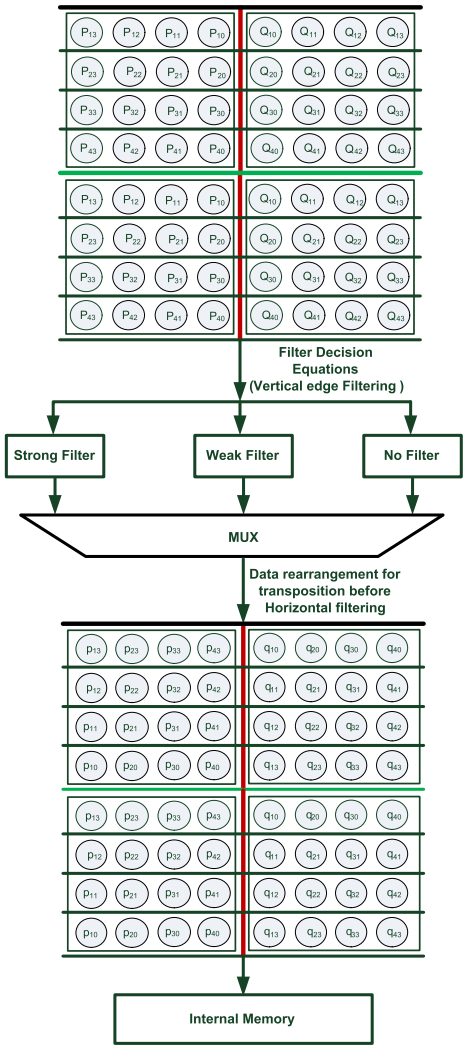

FIGURE 10. Filter unit showing vertical edge filtering.

filtering, one to two pixels on both the sides of the edge is modified.

$$|p_2 - p_0| < \beta \tag{5}$$
$$|q_2 - q_0| < \beta \tag{6}$$
$$|p_0 - q_0| < 0.25\alpha + 2 \tag{7}$$
$$p_0' = p_0 + \Delta_0 \tag{8}$$
$$p_1' = p_1 + \Delta_{p1} \tag{9}$$
$$q_0' = q_0 - \Delta_0 \tag{10}$$
$$q_1' = q_1 + \Delta_{q1} \tag{11}$$

where, Δ_0, Δ_{p1} *and* Δ_{q1} are given in (12), (13) and (14)

$$\Delta_0 = Min(Max(-c_0, \Delta_{0i}), c_0)$$
$$\Delta_{0i} = (4(q_0 - p_0) + (p_1 - q_1) + 4) >> 3 \tag{12}$$
$$\Delta_{p1} = Min(Max(-c_1, \Delta_{p1i}), c_1)$$
$$\Delta_{p1i} = (p_2 + ((p_0 + q_0 + 1) >> 1) - 2p_1) >> 1 \tag{13}$$
$$\Delta_{q1} = Min(Max(-c_1, \Delta_{q1i}), c_1)$$
$$\Delta_{q1i} = (q_2 + ((p_0 + q_0 + 1) >> 1) - 2q_1) >> 1 \tag{14}$$

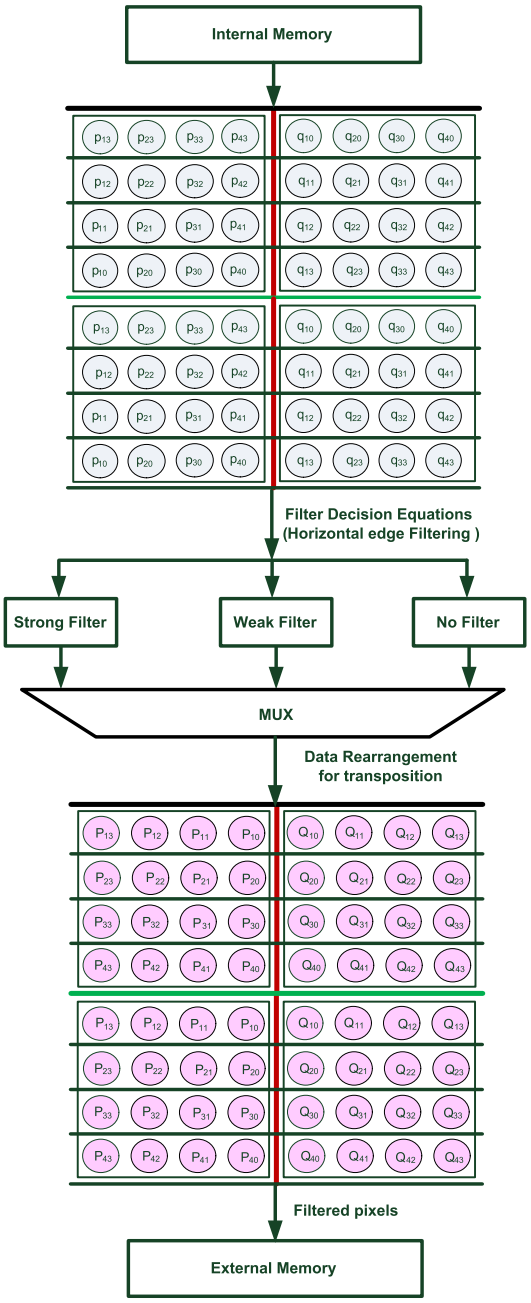

FIGURE 11. Filter unit showing horizontal edge filtering.

3) STRONG FILTER

If all the conditions (2), (3), and (4) are satisfied and if the computed BS is equal to 4, then condition (5) and (6) are checked. If both (5) and (6) are satisfied and if (7) is satisfied then three pixels on both the sides of the edges are modified as in (15) - (20). If both (5) and (6) are satisfied and if (7) is not satisfied then only one pixel on both the sides of the edges is modified as in (21) and (22). If (5) is satisfied and if (6) is not satisfied, then the pixels p_0, p_1, p_2 and q_0 alone are modified as in (15) - (18). If (6) is satisfied and if (5) is not satisfied then the pixels q_0, q_1, q_2 and p_0 alone are modified as in (15), (18) - (20). Thus if BS is equal to 4, then two to three pixels on both the sides of the edges are modified.

$$p_0' = (p_2 + 2p_1 + 2p_0 + 2q_0 + q_1 + 4) >> 3 \quad (15)$$
$$p_1' = (p_2 + p_1 + p_0 + q_0 + 2) >> 2 \quad (16)$$
$$p_2' = (2p_3 + 3p_2 + p_1 + p_0 + q_0 + 4) >> 3 \quad (17)$$
$$q_0' = (q_2 + 2q_1 + 2q_0 + 2p_0 + p_1 + 4) >> 3 \quad (18)$$
$$q_1' = (q_2 + q_1 + q_0 + p_0 + 2) >> 2 \quad (19)$$
$$q_2' = (2q_3 + 3q_2 + q_1 + q_0 + p_0 + 4) >> 3 \quad (20)$$
$$p_0' = (2p_1 + p_0 + q_1 + 2) >> 2 \quad (21)$$
$$q_0' = (2q_1 + q_0 + p_1 + 2) >> 2 \quad (22)$$

E. OPERATION

The QPEDBF, initially reads eight 4×4 sub-blocks from the external memory and stores the data into eight internal buffers shown in Fig. 5 during the read stage. The BS calculation unit computes the corresponding BS value at the second stage. The filter decision unit is then enabled by the control unit which compares the intensity of the pixel values stored in the buffer with the threshold values based on the filter decision equations during the third stage of the filtering operation. At the fourth stage, the appropriate filter (strong, weak or no filter) is enabled and the filtering process takes place as shown in Fig. 9. In this technique, four sub-edges can be filtered in parallel. Once the initial four vertical edges are filtered, the next four 4×4 sub-block data is read again from the external memory and stored in the P and Q buffers in a ping-pong fashion. Thus all the vertical edges of the Luma block followed by the vertical edges of the chroma(Cb and Cr) blocks are filtered. The vertical edge filtered data is stored in an internal dual-port memory after rearranging the data to transpose the 4×4 sub-block. Four internal dual-port memory is used to store the vertically edge filtered data of a MB. Once the vertical filtering of all the edges are done, the data from the internal memory is fetched and filtered for horizontal edges. After horizontal edge filtering the filtered data is rearranged again to transpose the 4×4 block. Fig. 10 and Fig. 11 shows the operation of the filter module within the filter unit. During the filtering process, the weak filter modifies one or two pixels on either side of a 4×4 block edge and a strong filter modifies upto 3 pixels on either side of a 4×4 block edge as given in the below equations. The filtered data is then written to the external buffer at the fifth stage.

Usage of internal buffers in the QPEDBF architecture reduces the external memory access cycles and thus improves the throughput of the architecture. Due to data rearrangement, the vertically filtered data is self-transposed before being stored into the memory for horizontal filtering. Again, after horizontal edge filtering the data is self-transposed before the filtered data is sent out to the external memory. Hence we get the original 4×4 pixel block with the pixel being modified based on the filtering strength. Thus it avoids the usage of transpose buffers. Also, due to sequential filter ordering, this architecture complies the H.264 video coding standard.

TABLE 1. Comparison of implementation results.

	[20]	[21]	[16]	[22]	[23]	[2]	Proposed
Coding Standard	H.264	H.264	H.264	H.264	H.264	H.264	H.264
Processing Order	Hybrid	Hybrid	Hybrid	Hybrid	Hybrid	Sequential	Sequential
RAM Size	16×32 (Dual port)	1×32×32 (Dual Port) 2×96×32 (Single Port)	2×96×32, 2N×32 (Single Port)	None	640 bytes SRAM	288 bytes	384 bytes (Dual Port)
Technology	0.18μm	90nm	0.18μm	0.18μm	0.18μm	StratixIII	90nm
Area (K)	12.3	26.1	21.5	3.9	41.6	15.9	19.8
Processing Time Cycles/MB	192	108×16	204	540	48	64	58
Throughput (kMB/s)	365	434	980	185	2812	2343	2586
Level of Parallelism	2	2	1	3	4	4	4
Pipeline stages	Non-pipelined	8	5	3	-	6	3
Frequency(MHz)	70	750	200	100	135	150	150
Transpose Buffers	Yes	Yes	Yes	Yes	Yes	Yes	No

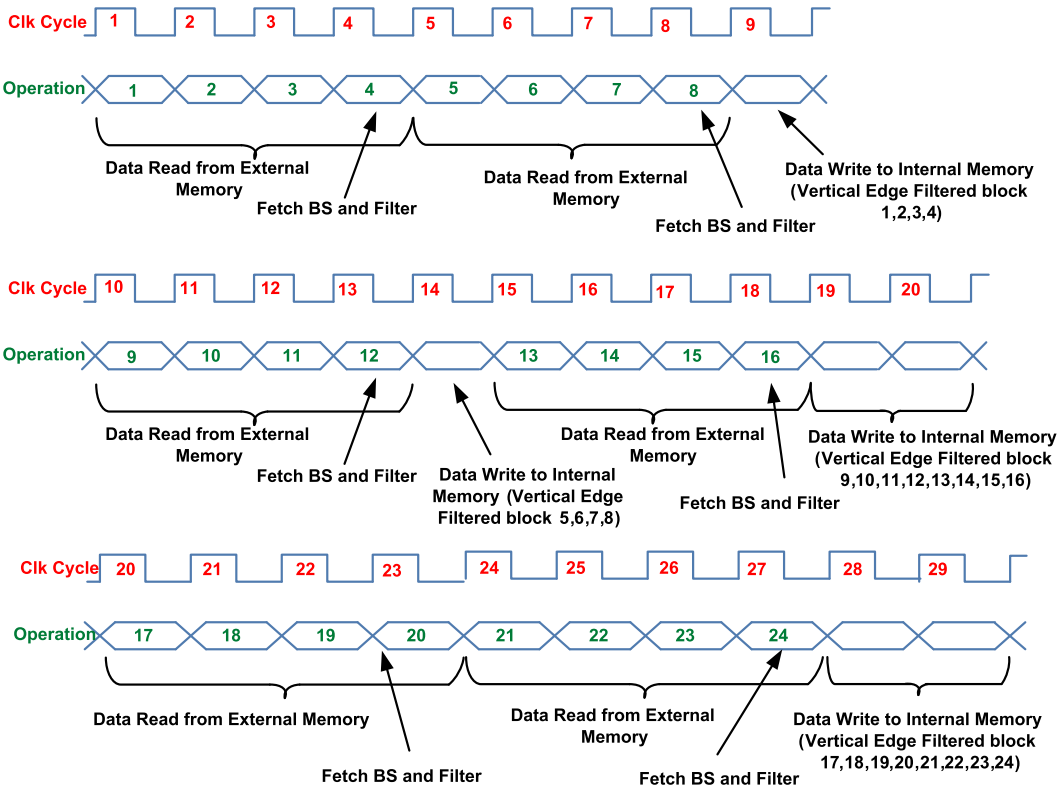

FIGURE 12. Clock cycles required for vertical edge filtering.

V. RESULTS

The QPEDBF architecture is implemented using Verilog HDL and the functional verification is done by simulating the RTL using Modelsim ALTERA. The functional simulation shows that this architecture can filter a MB in 58 clock cycles. The data from the external memory is read in the order, as shown in Fig. 14. Each number in Fig. 14 indicates a 4×4 sub-block. The operation performed for each clock cycle is shown in Fig. 12 and Fig. 13. Initially, for the first four clock cycles, four 4×4 sub-blocks of reconstructed pixel data from the external memory is read and stored in the internal buffer. BS for each sub-edge is computed while the data is

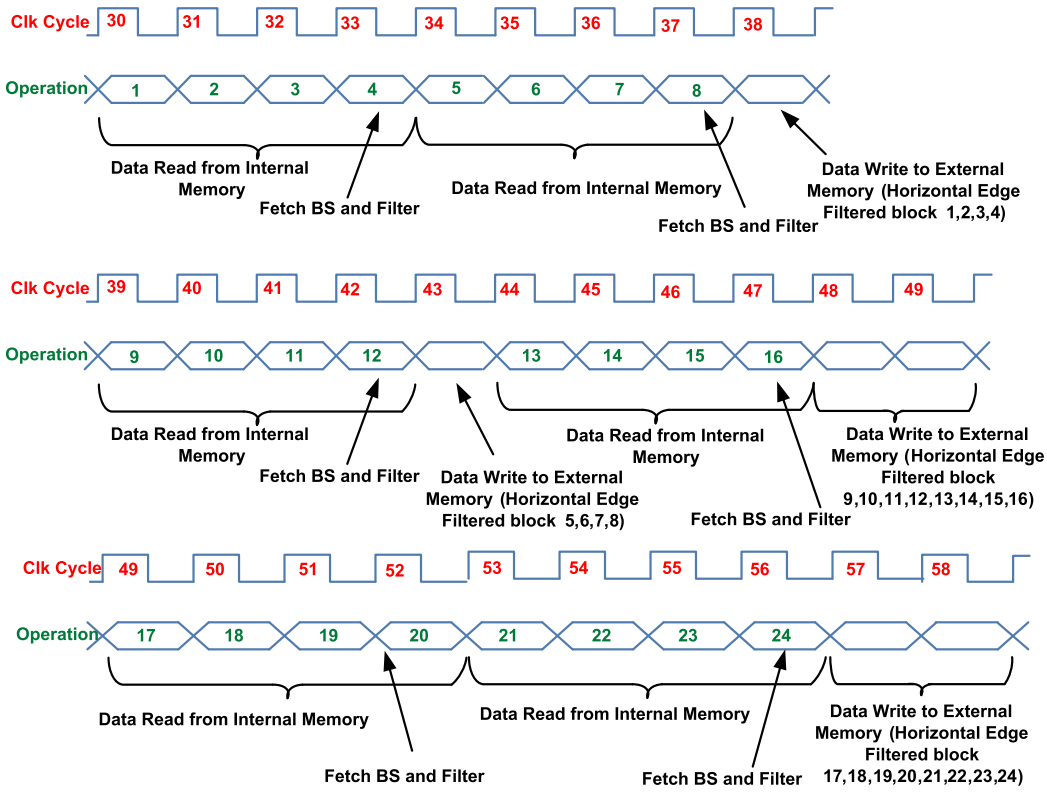

FIGURE 13. Clock cycles required for horizontal edge filtering.

being read from the external memory and the computed BS value is fetched in the fourth clock cycle, and the filter is enabled to filter the sub-edges mentioned as '1' in Fig. 9. Then from the fifth clock cycle to eighth clock cycle next four 4×4 sub-blocks of reconstructed pixel data from the external memory is read and stored in the internal buffer. BS values for these sub-edges are fetched, and the filter is enabled to filter the next four sub-edges mentioned as '2' in Fig. 9. At the ninth clock cycle, the first four 4×4 vertically filtered data is written to the internal memory. The process is repeated until all the vertical edges are filtered, which requires 29 clock cycles. Also, when data is written consecutively for two clock cycles during the 19th and 20th clock cycle, the data is simultaneously read at 20th clock cycle for the vertical filtering of chroma sub-edges. The same procedure is repeated for horizontal edge filtering starting from 30th clock cycle and during horizontal edge filtering the data is read from the internal memory. Again 29 clock cycles are required to filter the horizontal sub-edges, and overall the design requires 58 clock cycles to filter a MB.

The design is also synthesized using Synopsys DC compiler targeting for the 90nm library. This architecture can operate at a frequency of 150MHz and achieves a target area of 19.8K gates. Table 1 shows the results of the implemented architecture in comparison with the existing architectures. Two groups of four internal buffers of 128 bits stores the data read from the external memory, and these buffers are accessed in a ping pong fashion to access the P and Q sub-blocks of

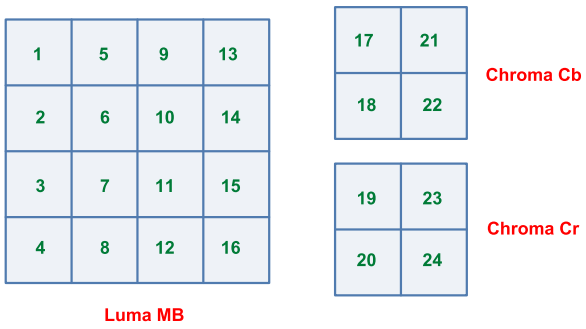

FIGURE 14. Order of data read from external memory.

pixels for filtering. This novel technique reduces the external memory access cycles, which reduces the processing time and increases the throughput. The filter output after vertical edge filtering is transposed and stored in the internal memory, which is then fetched and horizontal edge filtering is performed. The filter output after horizontal edge filtering is also transposed, and the filtered data is written to the external memory. The transposed output of the edge filters avoids the use of transpose buffers, and this reduces the usage of the target area and increases the throughput. The throughput of this work is computed as in (1), which is 2586kMB/s. The throughput achieved is 10.36% higher than [2] and 8% lesser than [23]. Though the throughput is 8% lesser, the area consumed by this architecture is half than that of [23]. Also, the internal memory usage is half compared to [23]. As this

work is implemented in hardware alone, the visual quality of the filtered images is not measured. Also, the design follows the sequential filtering order as mentioned in the H.264/AVC specification; the filtering process will not degrade the quality of the filtered images.

VI. CONCLUSION

This paper presents the DBF algorithms and hardware architectures for H.264/AVC video codec. The major challenges in designing the DBF are the memory organization to reduce the memory access cycles and the restructuring of the pixel blocks due to high data dependencies. It is seen that the processing time decreases as the number of pipeline stage increases in the DBF architecture with the area compensation. It is also noted that pipelined with parallel edge filter architecture achieves optimized results compared to pipelined only architectures in terms of processing cycles, throughput, area and frequency. It also presents a novel hardware architecture for the DBF used for H.264 coding standard using pipelined and four parallel edge filters to remove the blocking artifacts. The proposed architecture is implemented in Synopsys using 90nm library and achieves the target area of 19.8K and can process a Macro Block in 58 clock cycles. Since QPEDBF achieves higher throughput and comparatively lesser area, this architecture is suitable for real-time applications.

REFERENCES

[1] M. Jacobs and J. Probell, "A brief history of video coding," in *Proc. ARC Int.*, Jan. 2007, pp. 1–6.
[2] L. A. Ayadi, T. Dammak, H. Loukil, M. A. Benayed, and N. Masmoudi, "A novel deblocking filter architecture for H.264/AVC," *J. Signal Process. Syst.*, vol. 89, no. 2, pp. 281–292, Nov. 2017.
[3] H. Yin, H. Jia, J. Zhou, and Z. Gao, "Survey on algorithm and VLSI architecture for MPEG-like video coder," *J. Signal Process. Syst.*, vol. 88, no. 3, pp. 357–410, Sep. 2017.
[4] F. Tobajas and G. M. Callicó, P. A. Pérez, V. De Armas, and R. Sarmiento, "An efficient double-filter hardware architecture for H.264/AVC deblocking filtering," *IEEE Trans. Consum. Electron.*, vol. 54, no. 1, pp. 131–139, Feb. 2008.
[5] M. Li, J. Zhou, D. Zhou, X. Peng, and S. Goto, "De-blocking filter design for HEVC and H.264/AVC," in *Proc. Pacific Rim Conf. Multimedia*, 2012, pp. 273–284.
[6] G. Khurana, A. A. Kassim, T. P. Chua, and M. B. Mi, "A pipelined hardware implementation of in-loop deblocking filter in H.264/AVC," *IEEE Trans. Consum. Electron.*, vol. 52, no. 2, pp. 536–540, May 2006.
[7] S. Winkler and P. Mohandas, "The evolution of video quality measurement: From PSNR to hybrid metrics," *IEEE Trans. Broadcast.*, vol. 54, no. 3, pp. 660–668, Sep. 2008.
[8] B. K. N. Srinivasarao, I. Chakrabarti, and M. N. Ahmad, "High-speed low-power very-large-scale integration architecture for dual-standard deblocking filter," *IET Circuits, Devices Syst.*, vol. 9, no. 5, pp. 377–383, Sep. 2015.
[9] T.-M. Liu, W.-P. Lee, T.-A. Lin, and C.-Y. Lee, "A memory-efficient deblocking filter for H.264/AVC video coding," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 2140–2143.
[10] Y.-W. Huang, T.-W. Chen, B.-Y. Hsieh, T.-C. Wang, T.-H. Chang, and L.-G. Chen, "Architecture design for deblocking filter in H.264/JVT/AVC," in *Proc. IEEE Int. Conf. Multimedia Expo*, vol. 1, Jul. 2003, p. I-693.
[11] S.-Y. Shih, C.-R. Chang, and Y.-L. Lin, "An AMBA-compliant deblocking filter IP for H.264/AVC," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 4529–4532.
[12] M. Parlak and I. Hamzaoglu, "An efficient hardware architecture for H.264 adaptive deblocking filter," in *Proc. 1st NASA/ESA Conf. Adapt. Hardw. Syst.*, Jun. 2006, pp. 381–385.
[13] C.-C. Cheng, T.-S. Chang, and K.-B. Lee, "An in-place architecture for the deblocking filter in H.264/AVC," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 53, no. 7, pp. 530–534, Jul. 2006.
[14] K.-Y. Min and J.-W. Chong, "A memory and performance optimized architecture of deblocking filter in H.264/AVC," in *Proc. IEEE Int. Conf. Multimedia Ubiquitous Eng.*, Apr. 2007, pp. 220–225.
[15] M. Kthiri, P. Kadionik, and H. Lévi, H. Loukil, A. B. Atitallah, and N. Masmoudi, "A parallel hardware architecture of deblocking filter in H264/AVC," in *Proc. 9th Int. Symp. Electron. Telecommun.*, Nov. 2010, pp. 341–344.
[16] 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.
[17] R. Khraisha and J. Lee, "A scalable H.264/AVC deblocking filter architecture using dynamic partial reconfiguration," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Mar. 2010, pp. 1566–1569.
[18] T. Cervero, A. Otero, and S. López, E. de la Torre, G. M. Callicó, T. Riesgo, and R. Sarmiento, "A scalable H.264/AVC deblocking filter architecture," *J. Real-Time Image Process.*, vol. 12, no. 1, pp. 81–105, Jun. 2016.
[19] *Advanced Video Coding for Generic Audiovisual Services*, document Recommendation ITU-T H.264, 2017. [Online]. Available: http://handle.itu.int/11.1002/1000/13189
[20] K.-Y. Min and J.-W. Chong, "A memory efficient architecture of deblocking filter in H.264/AVC using hybrid processing order," in *Proc. Int. SoC Design Conf. (ISOCC)*, Nov. 2009, pp. 67–70.
[21] N. Kefalas and G. Theodoridis, "An 8K-UHD capable 8-stage pipeline deblocking filter for H.264/AVC," in *Proc. 6th Int. Symp. Commun., Control Signal Process. (ISCCSP)*, May 2014, pp. 570–573.
[22] C.-B. Wu, L.-H. Wang, and Y.-L. Chou, "Hardware-and-memory-sharing architecture of deblocking filter for VP8 and H.264/AVC," *IEEE Trans. Consum. Electron.*, vol. 63, no. 3, pp. 216–224, Aug. 2017.
[23] K.-H. Chen, "48 Cycles-per-macro block deblocking filter accelerator for high-resolution H.264/AVC decoding," *IET Circuits, Devices Syst.*, vol. 4, no. 3, pp. 196–206, May 2010.

PRAYLINE RAJABAI C. received the B.E. degree in electronics and communication engineering from Madurai Kamaraj University, Madurai, India, in 2003, and the M.E. degree in VLSI design under the Faculty of Information and Communication Engineering, Anna University, Chennai, India, in 2014. She is currently an Assistant Professor (Senior) with the School of Electronics Engineering, Vellore Institute of Technology, Vellore, India. She was with Wipro Technologies, Bengaluru, as the Technical Lead, from 2010 to 2012, and also with Arasan Chip Technologies Ltd., Bengaluru, and Tuticorin as the Front-end Design Engineer, from 2004 to 2010. Her research interests include FPGA/ASIC Implementation of video compression and encryption algorithms and reconfigurable architectures for multimedia applications.

SIVANANTHAM S. received the B.E. degree in electronics and communication engineering from the University of Madras, India, the M.Tech. degree in VLSI design from SASTRA University, India, and the Ph.D. degree in electrical engineering from the Vellore Institute of Technology (VIT), India, in 1997, 2002, and 2014, respectively. He is currently an Associate Professor with the School of Electronics Engineering, and the Assistant Dean of Academic Research with the Vellore Institute of Technology, Vellore, India. He worked as the Assistant Director of the International Relations Office, from 2014 to 2015. He has served as the Leader for VLSI and Embedded System Division, Vellore Institute of Technology, from 2007 to 2009. Previously, he was associated with J. J. College of Engineering and Technology, Tiruchirappalli, India, and the Bannari Amman Institute of Technology, Satyamangalam, India, as a Faculty with the Department of Electronics and Communication Engineering. His areas of research interests include the design for testability, reconfigurable architectures, and low power VLSI design. He is a member of the IEICE, VLSI Society of India (VSI), and the Indian Society for Technical Education (ISTE).