Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 4661 – 4669
© TÜBİTAK
doi:10.3906/elk-1407-144

FPGA implementation of a HEVC deblocking filter for fast processing of super high resolution applications

Awais KHAN[*], **Gulistan RAJA**
Faculty of Electronics and Engineering, University of Engineering and Technology Taxila, Taxila, Pakistan

Received: 21.07.2014 • **Accepted/Published Online:** 24.08.2015 • **Final Version:** 06.12.2016

Abstract: This paper proposes the architecture of a deblocking filter (DBF) that removes blocking artifacts in new emerging High Efficiency Video Coding (HEVC). A parallel architecture for both normal and strong filtering modes of HEVC is proposed. Distributed memories and two data paths increase the parallelism and make the architecture more efficient. The proposed architecture is described by Verilog and implemented on FPGA. The architecture can realize real time to compute 4K UHD video at 30 fps by using 46.65 million clocks with total equivalent gate count of 46K. The maximum delay time for output to come after taking input for the proposed architecture is 18.514 ns and the currently operating frequency is 54 MHz.

Key words: Deblocking filter, HEVC, FPGA, 4K UHD

1. Introduction

High Efficiency Video Coding (HEVC) is the new video coding standard developed by the Joint Collaborative Team on Video Coding (JCT-VC). HEVC gives more than 40% bit rate saving over the currently used H.264 video standard [1]. To achieve this, HEVC employs new encoding tools like the new deblocking filter algorithm, which provides greater compression efficiency.

HEVC divides each frame into blocks similarly to H.264 and previous coding standards. Each frame is divided into 64 $\times$ 64 coding tree units (CTUs), which are further divided into smaller units [2]. After going through the transformation, quantization, and entropy encoding process these blocks are available in the form of bit streams. In the case of prediction, the frames are recovered back in the local decoder. During the recovery disturbance in the adjoining blocks occurs. These disturbances or misalignments are known as blocking artifacts. HEVC uses a deblocking filter (DBF) to remove or suppress these blocking artifacts. DBF helps to improve video quality by removing blocking artifacts and discontinuities introduced into the frames due to coarse quantization. The DBF filter is applied across 8 $\times$ 8 sample blocks in HEVC, whereas, in H.264, 4 $\times$ 4 sample blocks are filtered [3]. The computational complexity of the deblocking filter in HEVC is less than that of the H.264. The deblocking filter in the H.264 decoder takes one third of the time of the total computational process, while the HEVC decoder takes one fifth [4].

There are two modes of filtering in HEVC, namely normal and strong filtering. This architecture implements both filtering modes. In this paper the efficient filter architecture is designed and implemented for the strong filtering mode. Distributed memories and processing parallelism make the proposed architecture

[*]Correspondence: engr.awaiskhan1990@yahoo.com

efficient. This paper is organized as follows. Section 2 explains the DBF algorithm of HEVC. Section 3 proposes the DBF filter architecture and the experimental results are shown in Section 4. Finally the conclusions are given in Section 5.

2. HEVC deblocking filter algorithm

The filtering in HEVC is less complex and more efficient than the deblocking filter of the previous H.264 standard [5]. The DBF in HEVC employs parallel filtering to improve subjective and objective quality by suppressing and removing blocking artifacts [6]. The misalignment of the samples across boundaries causes blocking artifacts as shown in Figure 1.

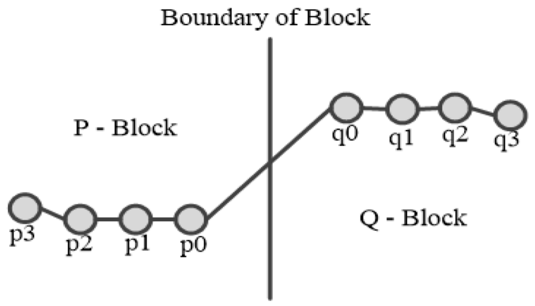

Figure 1. Misalignment of samples across boundary.

The samples of adjacent P and Q should be aligned in-line and their misplacement introduces blocking artifacts. The whole frame is divided into blocks with the help of vertical and horizontal boundaries. First horizontal filtering is done on the vertical boundaries and then vertical filtering is applied on the horizontal boundaries [7]. The division of the whole frame into blocks is shown in Figure 2.

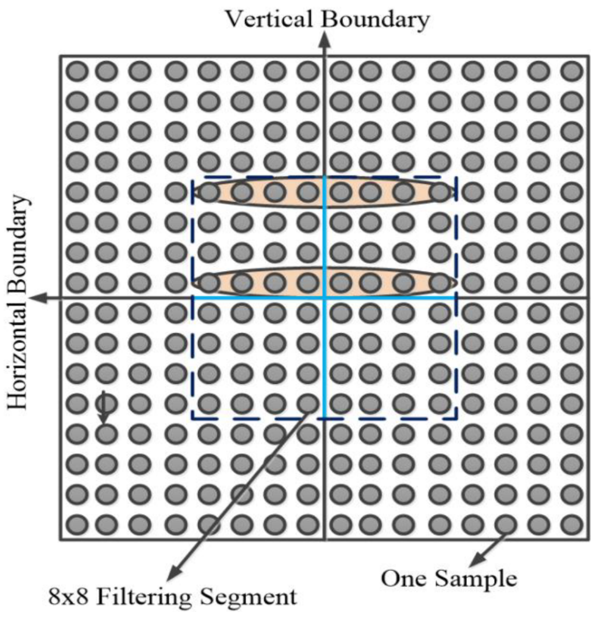

Figure 2. 8 × 8 sample blocks division.

The first and fourth lines of the 8 $\times$ 8 filtering segment are used in making the decision about filtering and modifying samples. Boundary strength (bS) is calculated to determine the type of filter to be used. There are three possible values for bS in the HEVC standard, i.e. 0, 1, and 2. The algorithm is shown in Figure 3 for assigning bS values.

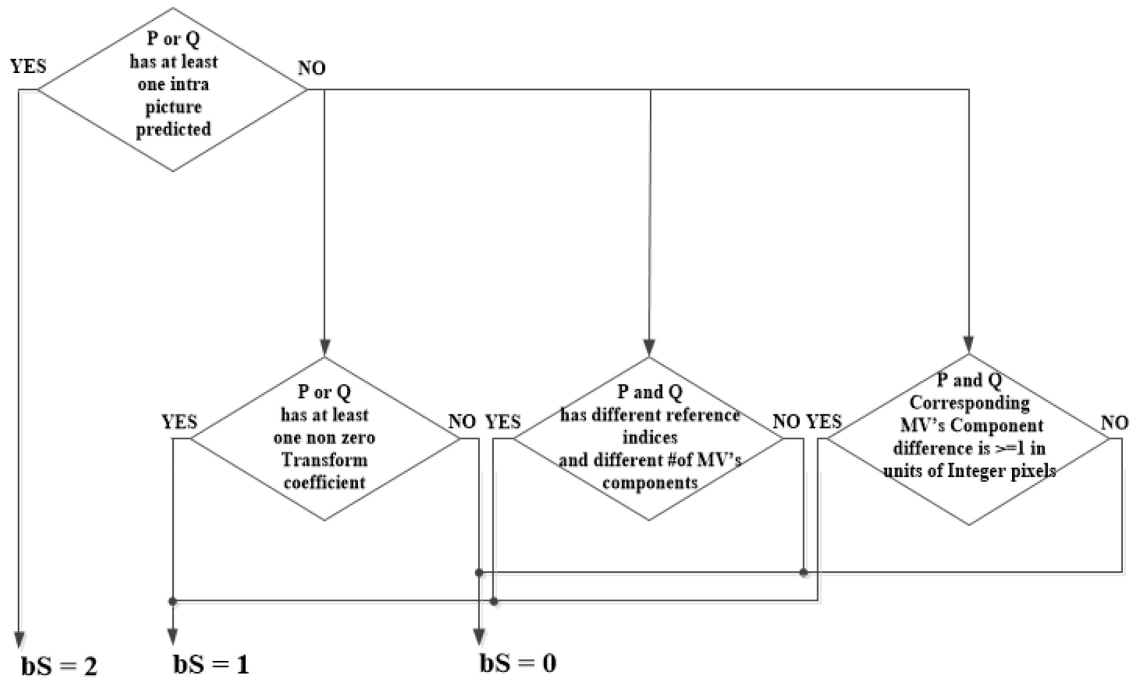
P or Q has at least one intra picture predicted
YES
NO
P or Q has at least one non zero Transform coefficient
YES
NO
P and Q has different reference indices and different #of MV's components
YES
NO
P and Q Corresponding MV's Component difference is >=1 in units of Integer pixels
YES
NO
bS = 2
bS = 1
bS = 0

Figure 3. Boundary strength (bS) calculation.

Luma filtering is applied when the bS value is 1 or 2, whereas chroma filtering is applied only for the bS value of 2. After determining bS values the following conditions are checked in order to decide whether to apply filtering or not:

$$bS > 0 \tag{1}$$

$$|p_{2,0} - 2p_{1,0} + p_{0,0}| + |p_{2,3} - 2p_{1,3} + p_{0,3}| + |q_{2,0} - 2q_{1,0} + q_{0,0}| + |q_{2,3} - 2q_{1,3} + q_{0,3}| < \beta, \tag{2}$$

where p and q are the samples of the adjacent P and Q blocks, respectively. The value of β is selected corresponding to the quantization parameter (QP) value given in the standard [8]. After the decision about applying filtering, the mode of filtering is select according to the following pseudo code:

```
    If ($|p_{2,i} - 2p_{1,i} + p_{0,i}| + |q_{2,i} - 2q_{1,i} + q_{0,i}| > \frac{\beta}{8} \&\& |p_{3,i} - p_{0,i}| + |q_{3,i} - 2q_{1,i} + q_{0,i}| > \frac{\beta}{8} \&\& |p_{3,i} - p_{0,i}| > 2.5t_c$)
        Apply Strong Filtering;
Else
        Apply Normal Filtering;
```

where the condition will be checked for the first sample row (i = 0) and the fourth sample row (i = 3) both. Here t_c threshold parameter can also be determined from the standard. In the case of luma strong filtering mode, three samples on each side of the boundary are modified, whereas for luma normal filtering only two consecutive samples across the boundary are filtered as shown in Figure 4.

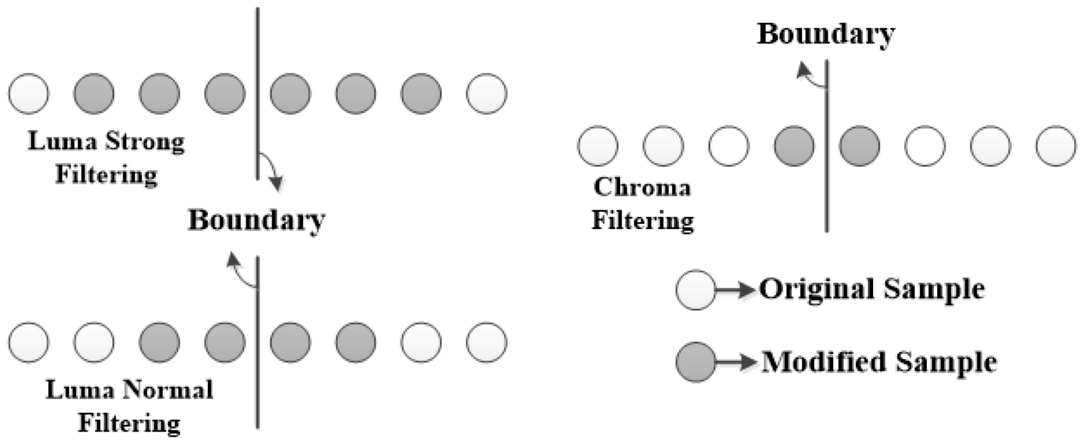

Figure 4. Modification of samples after applying filtering.

3. Proposed architecture design

The proposed architecture executes both normal and strong filtering modes. First the mode of filtering is decided by checking the conditions; then respective filtering is applied. The equations for normal filtering are shown below:

$$\Delta_0 = Clip3[(9\,(q_0 - p_0) - 3\,(q_1 - p_1) + 8 \gg 4] \tag{3}$$

$$p_0' = p_0 + \Delta_0 \tag{4}$$

$$p_1' = p_1 + Clip3[(((p_2 + p_0 + 1) \gg 1) - p_1 + \Delta_0) \gg 1] \tag{5}$$

$$q_0' = q_0 - \Delta_0 \tag{6}$$

$$q_1' = q_1 + Clip3[(((q + q_0 + 1) \gg 1) - q_1 - \Delta_0) \gg 1] \tag{7}$$

Strong filtering utilizes maximum resources and takes more time than normal filtering as it has a greater number of equations and also it needs more adders, subtracts, shifters, and multipliers. The filter equations for luma strong filtering are given below:

$$p_0' = p_0 + Clip3[(p_2 + 2p_1 - 6p_0 + 2q_0 + q_1 + 4) \gg 3] \tag{8}$$

$$p_1' = p_1 + Clip3[(p_2 - 3p_1 + p_0 + q_0 + 2) \gg 2] \tag{9}$$

$$p_2' = p_2 + Clip3[(2p_3 - 5p_2 + p_1 + p_0 + q_0 + 4) \gg 3], \tag{10}$$

where p0′ , p1′ , and p2′ are the filtered sample values of the original p0, p1, and p2 samples, respectively. To modify the corresponding Q block sample values the p and q are replaced with each other in Eqs. (8)–(10). Chroma filtering is applied only for the strong filtering case in which the pixel adjacent to the each side of the boundary is modified according to the following equations:

$$p_0' = p_0 + Clip3[(((q_0 - p_0) \ll 2) + p_1 - q_1 + 4) \gg 3] \tag{11}$$

$$q_0' = q_0 + Clip3[(((q_0 - p_0) \ll 2) + p_1 - q_1 + 4) \gg 3] \tag{12}$$

The key features that make our proposed architecture faster and more efficient are parallel filtering and distributed memories. There are 16 dual port rams used as distributed memories that help to store and process 16 samples in one clock cycle. Dual port rams are used because their memory cells can read/write simultaneously at different addresses. The parallel filtering allows two rows of samples to be filtered simultaneously. The view from above of the proposed implemented DBF hardware is shown in Figure 5.

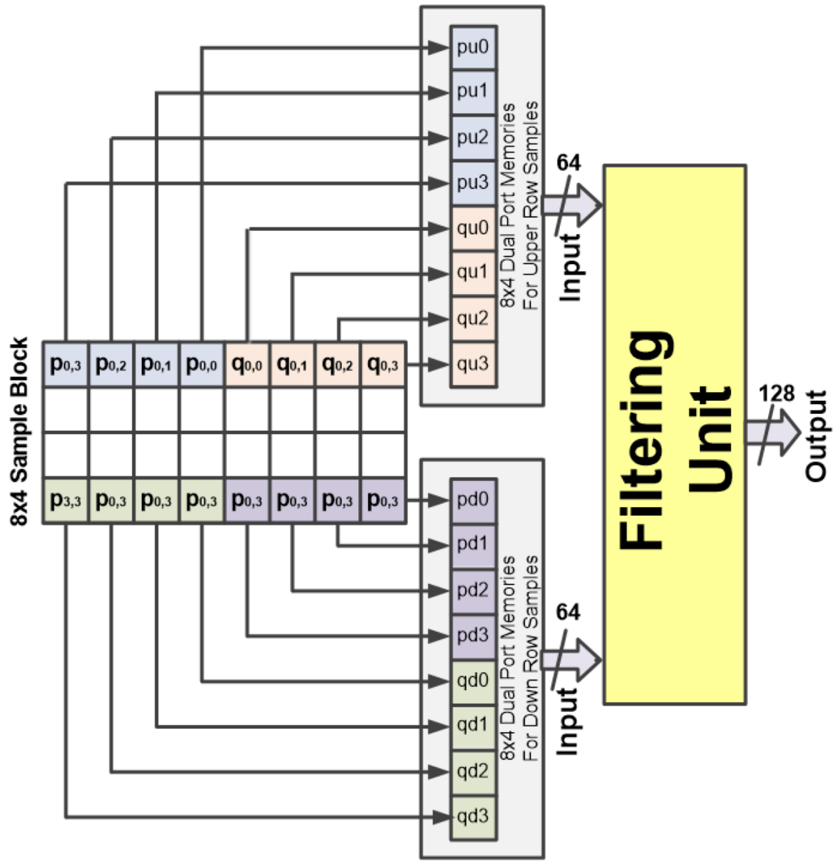

Figure 5. Top view of proposed architecture.

The size of each memory block is 8 × 4 units. Two lines of samples (upper and lower pixels) are taken as input at one clock cycle and they are filtered in parallel. The first row samples of the P block are stored in pu0∼pu3 memory units and Q block samples in qu0∼qu3. Similarly, the fourth row samples of P and Q blocks are stored in pd0∼pd3 and qd0∼qd3 memory units. The first and fourth rows are stored in different memory units so that they can be used in the next filtering blocks without the need to store them again. Memories are organized in such way that the filtering unit and control unit take 16 bytes or 128 bits of input and after processing give 128 bits of filtered output. The block diagram of the proposed architecture is shown in Figure 6.

The sample data are delivered to the filtering and control units after storing in memory units. The filtering unit applies normal and strong filtering to the samples and then sends it to the output multiplexers (MUXs). The control unit takes 128 bit sample values and additional β, t_c, and QP values as input and then it decides whether to apply filtering or not based on the conditions and it also chooses between the chrominance and luminance filtered data to show on output through the SEL line of MUXs. Sel0 decides on b/w filtering mode and Sel1 b/w luma and chroma output. The inner view of the control unit is shown in Figure 7.

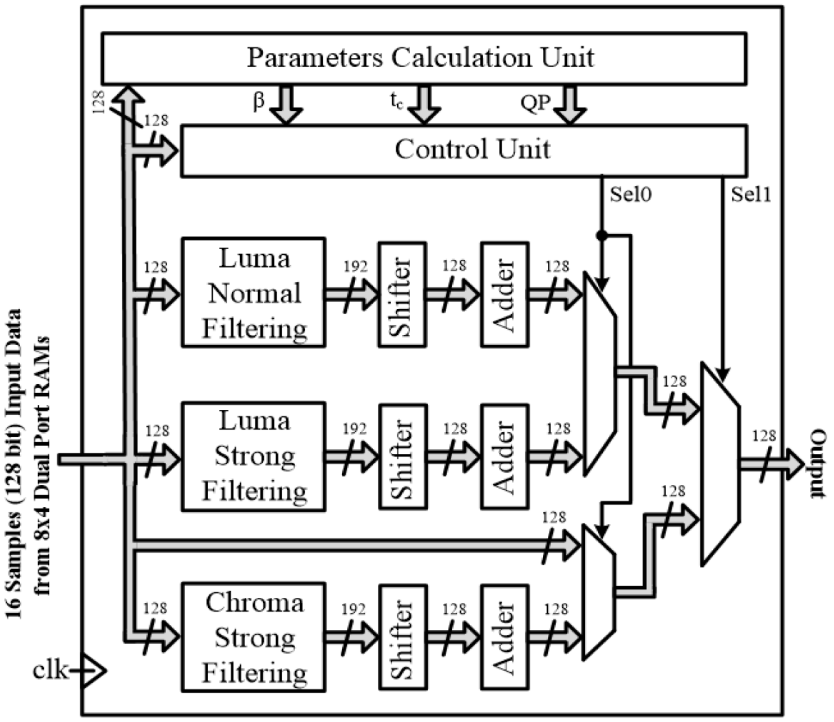

Figure 6. Proposed deblocking filter architecture.

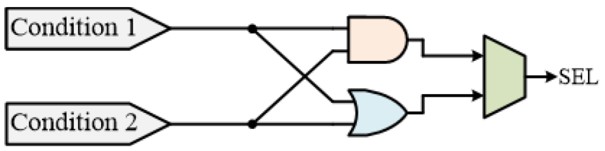

Figure 7. Internal view of control unit.

In Figure 6, the filtering unit consists of strong and normal luma filtering and chroma filtering blocks. After that there is the shifter and adder block. Shifting is done separately because it helps to decrease the gate count and in the adder block the resulting data after shifting and clipping are added to the original sample values. The output of the filtering unit is sent to the MUXs and the control unit decides which output to show. The inner view of the single filtering unit for first row samples is shown in Figure 8.

4. Implementation results of the proposed architecture

The proposed architecture design is first tested on MATLAB 2013. The simulation of the proposed architecture is checked on ModelSim 10.2c and then implemented and synthesized on FPGA using Xilinx ISE 9.1i. The Virtex-5 family and device XC5VLX330T are used to implement the design. The results of implementation are shown in Table 1.

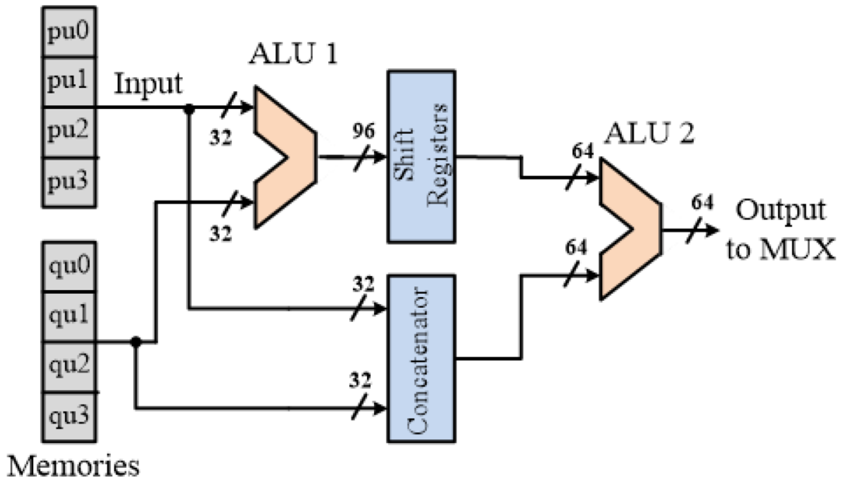

Figure 8. Filter unit's internal structure.

Table 1. Results of proposed architecture.

FPGA family	Virtex-5 (XC5VLX330T)
Speed grade	–2
Max. output delay	18.514 ns
Number of slices	343
Number of LUTs	1211
Cycles per 16 × 16 block	48
Total equivalent gate count	46k

The maximum output delay or latency after giving input is 18.514 ns. The total equivalent gate count is estimated implementing the architecture in Xilinx ISE 9.1i. Various inputs are given to the proposed architecture to verify the implementation and corresponding output is compared with the simulation results. The computation of one 4 × 8 sample block takes 2 clock cycles in the proposed architecture for horizontal filtering and also 2 cycles for one 8 × 4 sample block for vertical filtering. In each block one clock cycle is for input fetching and one for giving output after applying filtering. There are 388,800 blocks of 4 × 8 size of samples including luma and chroma blocks for a single 4K UHD frame and total number of blocks is 11,664,000 at 30 fps. Our architecture use 46.65 million clocks to process this frame and therefore it needs to run on 46.65 MHz frequency minimum. From the delay information we can calculate the operating frequency of our architecture, which comes out to be 54 MHz. The graphical representation of clock cycles distribution is shown in Figure 9.

There will be total 8 blocks of 8 × 4 samples in horizontal filtering and 4 × 8 for vertical filtering and so it will take 8 clock cycles for input to be stored in dual port RAMs. Then horizontal and vertical filtering for luma and chroma components will take 8 cycles each. Then there will be 8 cycles for the control unit to determine output through MUXs. The comparison of gate counts and clock cycles of the proposed deblocking filter with existing architectures is given in Table 2.

It can be observed from Table 2 that the proposed architecture has a lower gate count than that from [9] and [10] but not [5] and [11]; however, clock cycles required by [5] and [11] are much greater than the proposed one. [10] achieved fewer clock cycles at the cost of using large size memories and higher gate count. The resolution used in [5] is 1080p and is four times less than that of all the other architectures. We have achieved

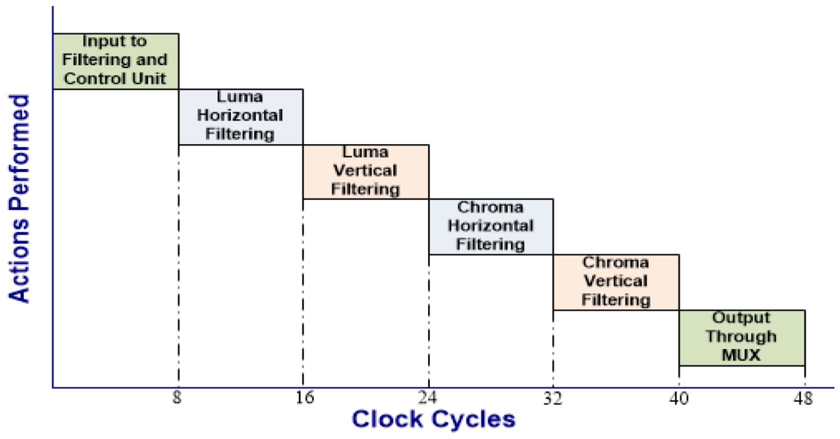

Figure 9. Clock cycle distribution for 16 × 16 blocks in the proposed architecture.

Table 2. Proposed deblocking filter architecture comparison.

DBF hardware	Ozcan et al. [5]	Bae [9]	Weiwei et al. [10]	Tikekar [11]	Proposed architec-ture
Platform	FPGA	ASIC	FPGA	FPGA	FPGA
Resolution	1080p	4K UHD	4K UHD	4K UHD	4K UHD
Memory type	Dual port SRAM	Register array	Dual port SRAM	SRAM	Dual port SRAM
Gate count (K)	16.4	54	75	44	46
Clock cycles millions/frame	108	94.4	28	200	46.65

a faster architecture in our design than [5] but at the stake of larger gate count. The main reason for the better results of the proposed architecture is the use of small size distributed memories and parallel filtering.

5. Conclusion

We proposed an efficient deblocking filter architecture in this paper. Use of distributed memories reduced the data accessing time for computation, and filtering of luminance and chroma samples in parallel reduced the cycles per CTU. Filtering of a single 16 × 16 block takes 48 clock cycles with parallel processing of two sample rows. Equivalent gate count achieved is 46K with maximum delay experienced of 18.514 ns. For the filtering of a 4K UHD frame at 30 fps, 46.65 million clocks are required but due to latency and delays practically the architecture is working at 54 MHz. The proposed architecture has reduced complexity and can be used by applications that require high speed.

References

[1] Nguyen T, Marpe D. Performance analysis of HEVC-based intra coding for still image compression. In: Picture Coding Symposium (PCS); 7–9 May 2012; Krakow, Poland: IEEE. pp. 233-236.

[2] Sullivan GJ, Ohm J, Han W, Wiegand T. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE T Circ Syst Vid 2012; 22: 1649-1668.

[3] Norkin A, Bjøntegaard G, Fuldseth A, Narroschke M, Ikeda M, Andersson K, Zhou M, Auwera GV. HEVC deblocking filter. IEEE T Circ Syst Vid 2012; 22: 1746-1754.

[4] Vanne J, Viitanen M, Hamalainen TD, Hallapuro A. Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs. IEEE T Circ Syst Vid 2012; 22: 1885-1898.

[5] Ozcan E, Adibelli Y, Hamzaoglu I. A high performance deblocking filter hardware for High Efficiency Video Coding. IEEE T Consum Electr 2012; 59: 714-720.

[6] Kotra AM, Raulet M, Deforges O. Comparison of different parallel implementations for deblocking filter of HEVC. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 26–31 May 2013; Vancouver, BC, Canada: IEEE. pp. 2721-2725.

[7] Raja G, Khan A, Khan AK, Yousaf MH. Performance analysis of HEVC in-loop filter. Life Sci J 2013; 10: 331-336.

[8] Bross B, Han WJ, Sullivan GJ, Ohm JR, Wiegand T. High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Last Call), JCTVC-L1003-v34 2013; 12th Meeting, Geneva.

[9] Bae J. Register array-based VLSI architecture of H.265/HEVC loop filter. IEICE Electron Expr 2013; 10: 1-9.

[10] Shen W, Shang Q, Shen S, Fan Y, Zeng X. A high-throughput VLSI architecture for deblocking filter in HEVC. In: IEEE International Symposium on Circuits and Systems (ISCAS); 19–23 May 2013; Beijing, China: IEEE. pp. 673-676.

[11] Tikekar M. Circuit Implementations for High-Efficiency Video Coding Tools. MSc, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, 2012.