Received: September 21, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Efficient VLSI Implementation of De-Blocking Filter with CSLA for H.264

Yogesh Gandasi Suresh[1]* **Sharanabasaveshwar Gangadharayya Hiremath[2]**

[1] *Department of Electronics and Communication Engineering, Visvesvaraya Technological University, Belagavi, India*
[2]*East West Institute of Technology, Electronics and Communication Engineering, India*
* Corresponding author's Email: gsyogesh@gmail.com

Abstract: The De-blocking filter is used in H.264 video codec for improving the quality of video. The De-blocking filter helps in removing the artifacts which are generated due to block based transform. In this research work, the Modified Filter Order (MFO)-Carry Select Adder (CSLA) based De-Blocking Filter (DBF) is introduced to improve the video quality, which is named as MFO-CSLA-DBF architecture. Conventional H.264 standard DBF requires more area due to a separate storage block of the horizontal and vertical data. To solve this problem, this paper implemented modified filter order architecture to reduce the memory and area for storing filter block data. The complexity of the boundary block has reduced by minimizing the boundary strength values. Furthermore, the CSLA design used for the data merging process in the DBF architecture instead of normal adder designs that also occupies less area in the MFO. The MFO-CSLA-DBF architecture improved the quality of the video by increasing Peak-to-Noise Ratio (PSNR), Structure similarity Index Matrix (SSIM) for different resolutions such as Quarter Common Intermediate Format (QCIF-176×144) and Common Intermediate Format (CIF-352×288). This MFO-CSLA-DBF architecture was implemented in the Xilinx tool by using the Verilog code for the Virtex-6 FPGA device. In this research work experimental results showed that MFO_CSLA_DBF architecture has reduced hardware utilization of FPGA for DBF from 2-3 % compared to the traditional DBF technique.

Keywords: Carry select adder, De-blocking filter, Field programmable gate array, Modified filter order, Xilinx.

1. Introduction

The video signals are frequently degraded by noise during image transmission and acquisition process. So, that noise reduction architectures are much required in the video sequences. A de-noising of the video is necessary for improving the image quality, facilitate transmission bandwidth reduction, increase compression and enhance the accuracy of the subsequent feature extraction and pattern recognition processes [1, 2]. In the past years, there were several video noise reduction techniques used for removing noise in the video signals/images, which is classified into three types such as spatial filter, temporal filter and combination of the temporal and spatial filters. For example, the Wiener filter is considered as the best technique for spatial noise filtering that achieved a high gain in the noise removal. Sometimes, the wiener filter causes serious damage to the image edge during the noise removal process which is the main drawback of the Wiener filter architecture [3]. The DBF architecture is used in the image data and de-blocking filter. The computation of the boundary strength values for each boundary depends on the horizontal and vertical direction of the input image data. It is feasible to estimate the values of boundary strength for each boundary that depends on a different direction [4]. The trilateral filter designed for Filter Depth Map (FDM) to achieve 0.8dB gain in rendering view compared to the de-blocking filter of h.264/ Advanced Video Coding (AVC) [5].

The performance of the image frames is enhanced by using DBF operations. The quantization step is used for lossy compression architectures in DBF [6]. The Block Sensitive address known as Peak Signal to Noise Ratio-B

(PSNR-B) achieved a good performance in quality assessments of the De-Blocking Images (DBI) than normal PSNR [7]. H.264/AVC approach is the most popular coding standard which is highly-efficient and provides complex video compression technique to the multimedia industry. The H.264/AVC approach has been limited in several fields due to its complexity [8, 9]. A High Efficient Video Coding (HEVC) algorithm reduces 50 percent bit rates in the encoding sequences without changing the image quality [10]. Nowadays, there are different techniques used to compress the data in video, although those are fails to provide better image quality [11, 12].

To overcome above mentioned problems, modified filter order is proposed in the DBF architecture for improving the video quality in this research. The complexity of the boundary block has been reduced by minimizing the boundary strength values. The MFO-CSLA-DBF architecture is designed for two major important things: 1. To perform a specific block boundary in DBF, 2. To determine the filtering strength. An unnecessary filtering can lead to over smoothing of the image details where lack of filtering may fail to neglect the blocking fault. The MFO-CSLA-DBF architecture has improved the video quality compared to Sample Level Filtering order [18] and Edge Filter H.264/AVC [19].

This research work is composed as follows, Section 2 presents a brief survey of recent papers based on DBF design. In section 3, briefly described the MFO-CSLA-DBF architecture by using CSLA approach. Section 4 presents the comparative experimental result of a proposed MFO-CSLA-DBF architecture and conventional architectures. The research work's conclusion is made in section 5.

2. Literature survey

Researchers have suggested several architectures based on the DBF architecture. This section presents a brief evaluation of some significant contribution to the field of DB architecture.

T.H. Teng, and C.P. Chung [13] proposed an exploiting fine-grain parallelism for H.264 de-blocking filter by operation reordering. This paper used a 4-pixel long boundary as the basis for analyzing and exploiting possible parallelism Compared to 2 Dimensional (2D) wave front for de-blocking both 1920x1080 and 1080x1920 pixel frames. The proposed method needs extra time when the number of frames are more. These features were desirable in the data intensive computing.

N. Kefalas, and G. Theodoridis [14] proposed a high throughput pipeline architecture to implement the H.264/AVC de-blocking filter. The proposed method extended to support higher sample bit depths and various chroma sub-sampling formats, which required by H.264 high-profile. This pipeline architecture was synthesized in 0.18 micro meter technology. The proposed method achieved 400MHz of the frequency and 16.8 Kilo gates of the area video processing. But, this method requires more hardware utilization.

A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G.V. der Auwera [15] implemented the DBF in the HEVC standard to reduce visible artifacts in the block boundaries. The de-blocking performed detection of the artifacts at the coded block boundaries. The HEVC-DBF architecture has less computational complexity and better parallel processing capabilities and reduction of the visual artifacts. This architecture requires huge overhead on slice level and has much complexity, which is the main limitation of this work.

E. Ozcan, Y. Adibelli, and I. Hamzaoglu [16] proposed the first HEVC-DBF architecture. Two parallel data paths were used in the HEVC-DBF hardware to reduce an execution time. This hardware structure was implemented in Verilog HDL. The HEVC-DBF hardware coded 30 full HD 1920x1080 video structure per second. The HEVC-DBF doesn't concentrate on distortion complexity and examining the software codecs.

S.C. Hsia, W.C. Hsu, and S.C. Lee [17] implemented a low complexity and high quality adaptive de- blocking filter for H.264/AVC system. An adaptive offset technique used here to enhance the quality of the image for H.264 de-coding. This method automatically chooses offset value to neglect artifact on blocking boundary. The proposed method achieved better subjective and objective quality than original H.264 de-blocking filter with a fixed offset value. The proposed method is not suitable for high-bit rate H.264/AVC.

To overcome the above-mentioned problems, the video quality is improved by using MFO-CSLA-DBF architecture.

3. Modified filter order based on de-blocking filter

To filter a macro block, the value of a pixel must be read many times and intermediate results of the filtering must be stored. To enhance the use of local memory, which is possible to reorder the filtering

Figure.1 Block diagram of the MFO-CSLA-DBF architecture

operations in the same way that the intermediate results are employed sooner by De-blocking filter. The processing order implemented by the H.264/AVC standard is such that vertical borders of chrominance and luminance blocks are filtered before the horizontal borders. The outputs of vertical filtering are given to the horizontal filtering, obtained intermediate results must be stored in memory. This filter processing order is more expensive in terms of memory use.

G. Correa, L. Agostini, and L.A. Cruz [18] proposed order of the filter that was developed using sample level for DBF operation in the H.264/AVC. This sample level filtering order used parallelism by decreasing dependency of the data. In this work, the filtering process involved in four parallel and independent samples simultaneously in the filtering order. The filtering order minimized the number of cycles required to filter a complete macro block, which is a significant solution to process a high-resolution video.

L.A. Ayadi *et al*. [19] proposed filtering order for better parallelism of processing the edges based on four-edges, which are capable to treat at the same time. In the filter order, the repeated numbers are used that represented the parallel processing order.

Fig. 1 shows the block diagram of the MFO-CSLA-DBF architecture. These filtering decisions are based on the values of different parameters like QP, block type, tc, β, which are generated by LUT by using QP value as an input index. An input is given to boundary strength and an input index block decides which filter has been applied to the boundaries. The filter operation takes place to smooth off the artifact portions. At the final stage, the smoothened frames are obtained as an output.

3.1 Modified filter order design

In this research work, the MFO is used for the DBF of H.264 compression standard. If the two-vertical edges have been filtered, then the horizontal edge also filtered. Each Luma Pixels (LP) have to be filtered before filtering a Chroma Pixel (CP). In Fig. 2, Y denotes the luminance macro-block, Cb denotes the blue chrominance macro-block and Cr denotes the red chrominance macro-block.

In this research, order of filtering basis pipeline concept allows CPs to Vertical filter for rearranging

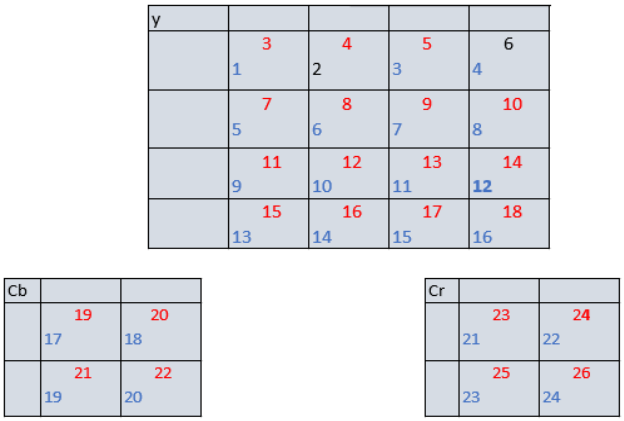

Figure.2 Modification of the Filter Order

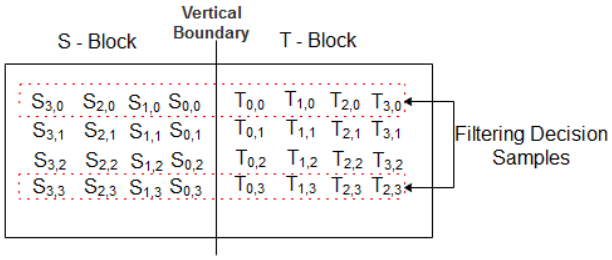

Figure.3 Boundary of the 4x4 blocks

the filter order. While the LPs are still left to remain in the pipeline and the number of the slice registers required for filtering will be mitigated. Furthermore, the amount of memory requirement is reduced for storing 4x4 filter blocks by using MFO approach. Fig. 2 shows the modification of the filter order for the MFO-CSLA-DBF architecture. The MFO-CSLA-DBF architecture decreases the blocking devices due to block-based encoding with strong quantization by applying modified samples along with Horizontal/Vertical boundaries. The filtering is applied separately in 4x4 blocks that are shown in Fig. 3. The normal and strong filtering modes modify the two and three Luma samples through each and every boundary. The MFO is a significant functionality in the DBF process. The performance of the DBF developed with a filtering order is discussed in the following sections.

3.2 Boundary strength modification

The boundary strength is calculated for the boundaries; such as transformed unit boundary or predictive unit boundary. This boundary strength takes values such as 0, 1, or 2. The block boundaries with boundary strength values such as 1 or 2 are filtered for Luma Component (LC). Hence, there is no filtering in static areas, which avoids sub-sequent filtering of the similar area. The image LPs are copied from one image to another image with residual equal to zero causes over smoothing. For

Chromo Component (CC), boundaries with boundary strength values similar to 2 are filtered. Thus, the filtered block boundaries have at least one of two adjacent blocks that are intra predicted. The boundary strength modification improves by the extra computation complexity on the H.264 standard, which is the major advantage of this technique.

3.3 Filtering decisions and operations

The filtering decision approach is required for the filtering of two given 4x4 blocks. Fig. 4 shows the merged data path using CSLA design, which represented that the conditions (1), (2), (3), (8) and (9) shares the same Eq. and input samples. The partial results from the conditions (2) and (3) are used for conditions (1), (8) and (9). In this work, S considers as X block and T consider as Y block used for the below-mentioned Eq. from (1) to (10). Thus, the MFO-CSLA-DBF architecture uses hardware reuse design to provide a merged data path for those Eq. using CSLA approach. This research work replaced the multiplications by CSLA and shift operations to employ low hardware resolutions. Furthermore, shortening the conditions (1), (2), (3), (8) and (9) as z1, z2, z3, z8 and z9, respectively. The data paths employed for Eq. (4) and (5) are equal, the only difference is the input sample. The MFO-CSLA-DBF architecture includes two instances of this data path to evaluate both z4 and z5 for the Eq. (6) and (7). The Eq. (10) is an additional filtering decision applied to z10 used for 4-rows 4x4 blocks after the normal filtering operations.

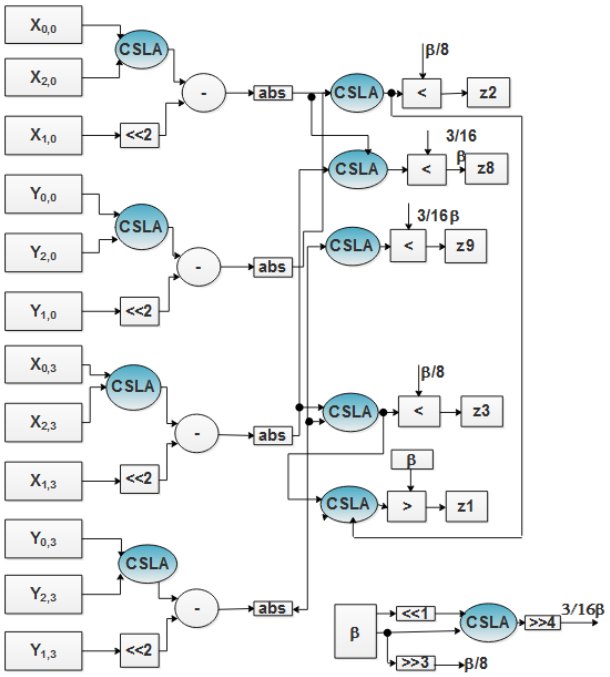

Figure.4 Merged data path using CSLA design

$$|x_{2,0} - 2x_{1,0} + x_{0,0}| + |x_{2,3} - 2x_{1,3} + x_{0,3}| + |x_{2,0} - 2y_{1,0} + y_{0,0}| + |y_{2,3} - 2y_{1,3} + y_{0,3}| > \beta \quad (1)$$

$$|x_{2,i} - 2x_{1,i} + x_{0,i}| + |y_{2,i} - 2y_{1,i} + y_{0,i}| < \beta/8 \quad (2)$$

$$|x_{3,i} - x_{0,i}| + |y_{0,i} - y_{3,i}| < \beta/8 \quad (3)$$

$$|x_{0,i} - y_{0,i}| < 2.5t_C \quad (4)$$

$$x'_0 = (x_2 + 2x_1 + 2x_0 + 2y_0 + y_1 + 4) \gg 3 \quad (5)$$

$$x'_1 = (x_2 + x_1 + x_0 + y_0 + y_1 + 2) \gg 2 \quad (6)$$

$$x'_2 = (2x_3 + 3x_2 + x_1 + x_0 + y_0 + 4) \gg 3 \quad (7)$$

$$|x_{2,0} - 2x_{1,0} + x_{0,0}| + |x_{2,3} - 2x_{1,3} + x_{0,3}| < {}^{3}/_{16}\beta \quad (8)$$

$$|x_{2,0} - 2y_{1,0} + y_{0,0}| + |y_{2,3} - 2y_{1,3} + y_{0,3}| < {}^{3}/_{16}\beta \quad (9)$$

$$|\delta_0| < 10\text{tc} \quad i = 0,3 \quad (10)$$

After calculating the filtering decisions, this unit calculates the filtering operations used for strong and normal filters (for Luma and Chroma filter). The normal filter modifies 1 or 2 samples between the block boundaries. That calculates the delta values such as delta x0, delta x1 and delta x2, and these offset values are added to the original samples (x0, x1, y0 and y1) in order to generate the final filtered sample (x0, x1, y0, and y1). That offsets value is applied to the 4-rows of samples of 4x4 blocks S and T. The delta x0 operation is calculated at the normal filter in the filtering decision process and the modified x0 and y0 samples are close to the boundary. The delta x1 and y1 operations are modified as x1 and y1 samples. The strong filter modifies the three-samples through the block boundary. That calculates the delta values such as delta s0, delta s1 and delta s2, and these offsets are added with data path by using CSLA to the original samples like x0, x1, x2, y0, y1, and y2 in order to generate the final filtered samples such as x0, x1, x2, y0, y1 and y2. That input is applied to the 4-rows of samples with a 4x4 block of S and T.

3.3.1. Carry select adder design

The major advantage of CSLA is, less propagation delay, which is obtained by employing the parallel stage that result taken from several pairs of RCA. The RCAs generate their temporary sum and carry for the CSLA architecture by considering the carry input to be zero and one respectively. Fig. 5 shows the block diagram of the CSLA. The CSLA design used in the MFO design for adding data path instead of the normal adder. It achieves fast arithmetic operation of various data processing approaches. The major aim of this CSLA is to reduce the area in the MFO-CSLA-DBF architecture. The CSLA operates in many computational designs to cut the Carry Propagation Delay (CPD).

The CSLA design uses the Binary-to-Excess Converter (BEC) and Ripple Carry Adder (RCA) with Cin=0 to achieve low area. The CSLA design uses few numbers of logic gates to derive the BEC logic instead of N-bit Full Adder (FA). The major advantage of BEC logic function is the least number of logic gates compared to the N-bit FA structure. In the Fig. 6 shows the operation of the CSLA architecture which consists of the four FA design, RCA, BEC and Multiplexer (MUX). For example, CSLA operation: As input A=0100 and B= 0101. In initial stage, A_0=0, B_0=1, A_1=0, B_1=0 are given to the input of the RCA circuit. The group2 has one 2-b RCA, which has two FA for Cin= 0. The first FA output: sum is 1 and carry is 0 that carry is given to the input of another second FA. The second full adder sum is 0 and carry also 0, which carry output is given to the input of Mux. While the RCA output is 01.

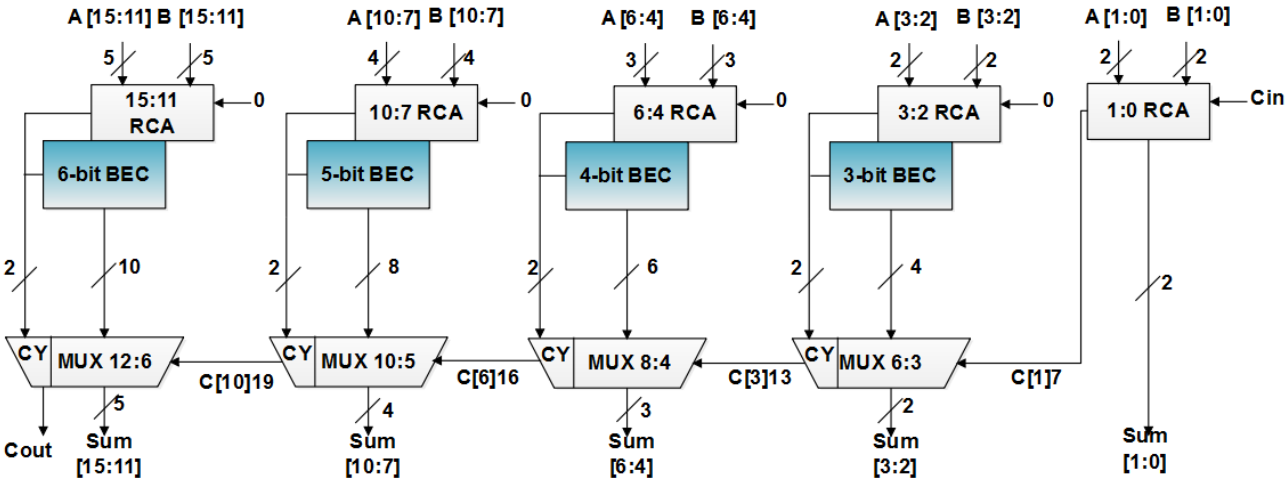

Figure.5 Block diagram of CSLA design.

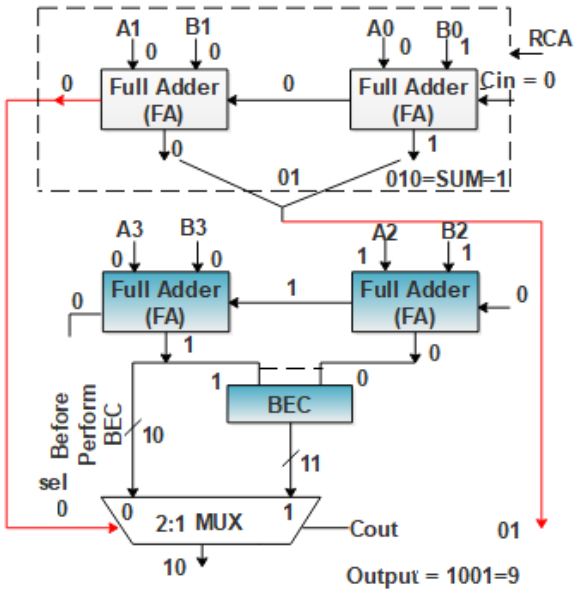

Figure.6 Operation of the Carry select adder

The second stage, the A_2=1, B_2=1 are given to the first FA, which sum represents 0 and carry represents 1. In second FA, A_3=0, B_3=0 and first FA carry=1 values are given to an input of the second FA, here sum is 1 and carry is 0. This FA output is given to the input of the BEC. The BEC's main operation is a one-bit incremental operation. If given the input (binary 10) of the BEC that the output is 11. The BEC and second stage FA outputs 10, it is given to the input of MUX. If selection line is 0 at the time MUX output is 10 otherwise MUX output is 11. Finally, the concatenation of the first stage output (01) and MUX output (10), which is delivered the 1001. The input arrival time is less compared to the multiplexer selection input arrival time.

4. Result and discussion

In this research work, the MFO-CSLA-DBF architecture has been implemented in the Verilog. The simulation outputs have been verified by using Xilinx tool 14.4 ISE simulator. This MFO-CSLA-DBF architecture simulated in the MATLAB tool, which is similar to MFO-CSLA-DBF hardware architecture and confirmed the frame quality by offering the reconstructed frames. The Register Transfer Level (RTL) simulation outputs has been detected to appropriately match the MATLAB simulation outputs. Here, simulation outputs denote reconstructed frame quality, which is produced after operation of the DBF. The Verilog RTL program is synthesized in Xilinx 14.4 tool and mapped to a FPGA Virtex-6 device, which is shown in the Table 3 and 4.

4.1 Mean square error

The MSE is calculated by the average of the square input image and output image pixels. The MSE is defined in Eq. (11)

$$\text{MSE} = \frac{1}{\text{NM}} \sum_{m=0}^{M-1} e(m,n)^2 \tag{11}$$

Here, $e(m,n)$ is the error difference between an input and distorted images.

4.2 Peak signal to noise ratio

The PSNR is a mathematical measure of the image quality based on the pixel difference between 2-images. The PSNR is defined in Eq. (12)

$$\text{PSNR} = 10\log \frac{s^2}{\text{MSE}} \tag{12}$$

Here, $S = 255$ is used for an 8-bit image.

4.3 Structural similarity index matric

The SSIM is the degree of similarity between digital images that helps to compute the received quality of video signal. The SSIM is a measure between image x and y which is shown in Eq. (13). Here, C_1 and C_2 represents constants.

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{\text{xy}} + C_2)}{(\mu x^2 + \mu y^2 + C_1)(\sigma^2 x + \sigma^2 y + C_2)} \tag{13}$$

In Table 1 and 2, the values of MSE, PSNR and SSIM are computed for 3- different images like Akiyo, Foremen and Hall Monitor that are taken from Vermont Information Processing (VIP) database. From this Table 1 and 2, it is clearly concluded that the MSE value reduced and PSNR and SSIM values increased using the MFO-CSLA in the DBF architecture compared to Sample Level Filtering order [18] and Edge Filter-H.264/AVC [19] architectures. So, MFO-CSLA-DBF architecture improves the quality of the video by maximizing PSNR and SSIM value. The Table 3, and 4 represents the FPGA performance of the existing and MFO-CSLA-DBF architecture on various resolutions. In the Sample Level Filtering order [18], the DBF architecture has been designed using more number of multipliers and registers, which occupied more area in the DBF architecture. In Edge Filter-H.264/AVC [19], a programming to target parallel structure was a bit difficult due to a number of the gates used for multi-core architecture. To overcome this problem, the MFO-CSLA-DBF filter architecture has been developed in this paper for H.264/AVC standard. In MFO-CSLA-DBF filter architecture, FPGA concentrated on reducing the hardware utilization. The CSLA design is used in the DBF architecture for data merging, which occupies less area because Slice LUTs, slice registers, power consumption and delay are reduced in the MFO-CSLA-DBF architecture compared to the exiting architectures. Hence, the MFO-CSLA-DBF architecture has improved the quality of the image in video compression.

Table 1. Comparison of PSNR and SSIM value for existing and MFO-CSLA-DBF architectures of QCIF 176×144 resolution

Frames	Sample Level Filtering order [18]		Edge Filter-H.264/AVC [19]		**MFO-CSLA-DBF**	
Performance	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Akiyo	39.93	0.96207	39.25	0.95208	40.124	0.94258
Foreman	39.02	0.95346	39.96	0.94349	40.133	0.94169
Hall Monitor	39.85	0.92187	39.65	0.93189	40.551	0.93119

Table 2. Comparison of PSNR and SSIM value for existing and MFO-CSLA-DBF architectures with CIF 352×288 resolution

Frames	Sample Level Filtering order [18]		Edge Filter-H.264/AVC [19]		**MFO-CSLA-DBF**	
Performance	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Akiyo	45.14	0.995319	45.23	0.990122	46.74	0.990056
Foreman	45.88	0.990286	45.92	0.980318	46.10	0.980254
Hall Monitor	46.02	0.998356	45.99	0.972357	46.86	0.971367

Table 3. Comparison of FPGA performance for existing and MFO-CSLA-DBF architecture with QCIF 176×144 resolution

Target FPGA Device	**Methodology**	**Power Consumption (W)**	**Delay (nano seconds)**	**Slice LUT**	**Slice Register**
Virtex-6	Sample Level Filtering order [18]	2.781	6.958	2147	488
	Edge Filter-H.264/AVC [19]	2.701	6.899	2122	470
	MFO-DBF	2.654	6.842	2112	462
	MFO-CSLA-DBF	2.423	6.654	2101	452

Table 4. Comparison of FPGA performance for existing and MFO-CSLA-DBF architecture with CIF 352×288 resolution

Target FPGA Device	**Methodology**	**Power Consumption (W)**	**Delay (nano Seconds)**	**Slice LUT**	**Slice Register**
Virtex-6	Sample Level Filtering order [18]	2.795	6.984	2285	508
	Edge Filter-H.264/AVC [19]	2.780	6.972	2273	500
	MFO-DBF	2.656	6.847	2121	469
	MFO-CSLA-DBF	2.576	6.765	2115	455

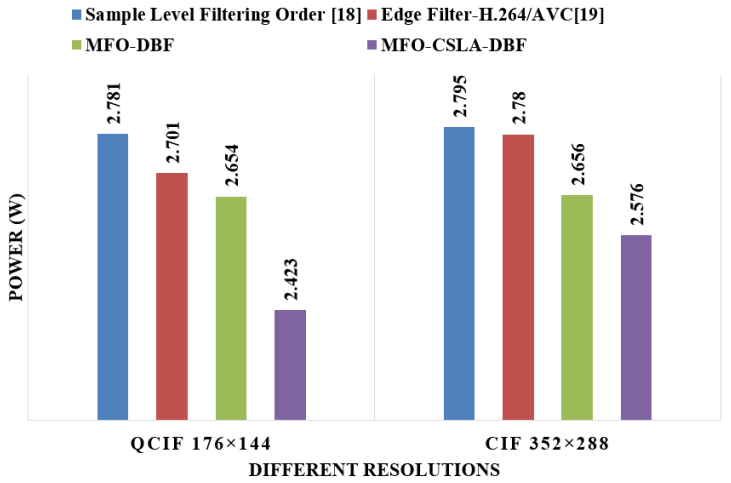

Figure.7 Comparison of the power consumption performance of Virtex-6 FPGA device for existing and MFO-CSLA-DBF architectures

In this research work, both existing and MFO-CSLA-DBF architecture was implemented, and the results of those are tabulated. The Table 3, 4 shows the performance of FPGA for existing and MFO-CSLA-DBF architectures on QCIF 176×144 and CIF 352×288 resolution. The MFO-CSLA-DBF architecture is suitable for Virtex-6. From this devices, Virtex-6 has considered as a high configuration device which provide better performance than less configuration devices: Virtex-4, Virtex-5. From the Table 3 and 4 it is clearly concluded that power consumption, slice LUTs, slice registers and delay are minimized in MFO-CSLA-DBF architecture compared to the existing architectures in the higher FPGA Virtex-6 device. The operating frequency value of the MFO-CSLA-DBF architecture is 110.14 MHz. FPGA performances have been taken from Xilinx tool.

In Fig. 7, 8, and 9 show the comparison graph of

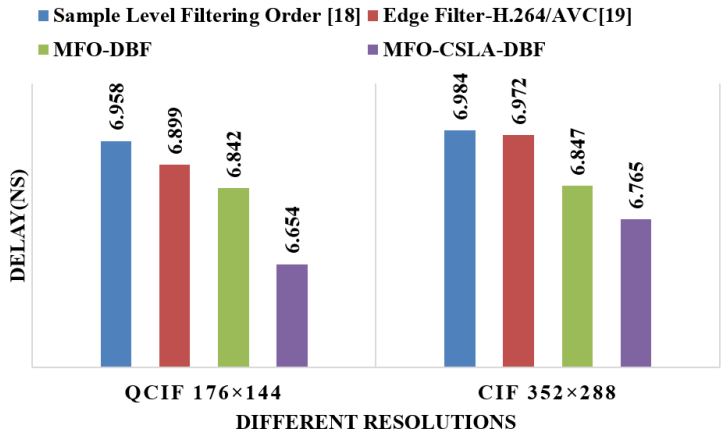

Figure.8 Comparison of delay performance of Virtex-6 FPGA device for existing and MFO-CSLA-DBF architectures

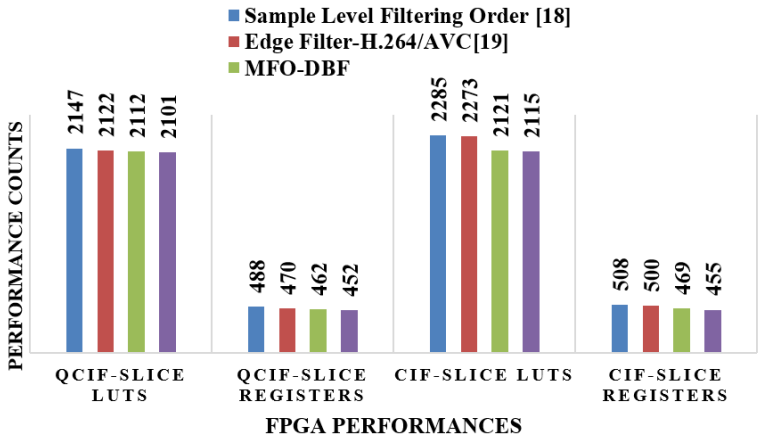

Figure.9 Comparison slice LUTs and slice registers count of Virtex-6 FPGA for existing and MFO-CSLA-DBF architectures

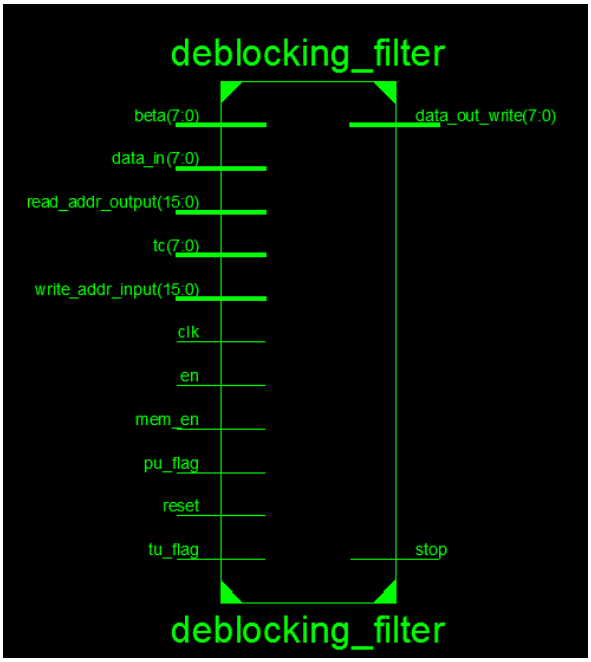

Figure.10 Top module of the RTL schematic of the MFO-CSLA-DBF architecture

the Virtex-6 FPGA performances such as power consumption, delay, slice LUTs and slice registers on QCIF 176×144 resolution and CIF 352×288 resolution. From these Fig. 7, 8, and 9, it's clearly

shown the performances of the FPGA are reduced compared to the conventional DBF architecture. In this research work, the CSLA approach has been used in MFO-DBF design for data merging, which reduces hardware utilization of the entire filter design.

In the Fig. 10 shows the RTL schematic of the MFO-CSLA-DBF architecture, which is taken from Xilinx tool software by using the Verilog Code. In this research work, separate code for each and every block such as filter architecture, filter decision, DBF, has been developed in the experimental setup. Finally, the performance parameters such as power consumption, Slice LUTs, Slice registers, delay and operating frequency take from the Xilinx tool by using Verilog code.

5. Conclusion

In this paper, a low complexity of boundary strength filter and modified filter order were proposed to enhance video frame quality by improving PSNR and SSIM values. With the help of MFO-CSLA-DBF architecture, the complexity of DBF was reduced. The MFO-CSLA-DBF architecture has been designed in Verilog, synthesized on Xilinx Virtex-6 FPGA. The key achievements of the MFO-CSLA-DBF architecture was reduced the hardware utilization, power and delay. The PSNR and SSIM values improved in MFO-CSLA-DBF architecture for different frames such as Akiyo, Foreman, and the hall monitor has been presented. Performance reduction of FPGA on QCIF 176×144 resolution is 8.70 % of power consumption, 2.74 % of the delay, 0.52 % of the slice LUTs, 2.16 % of slice registers. Performance reduction of FPGA on CIF 352×288 resolution is 3.01 % of the power consumption, 1.19 % of the delay, 0.28 % of slice LUTs, 2.98 % of the slice registers. In the future work, the optimized adder design can be used for the modified de-blocking filter design, which can effectively reduce the hardware utilization and that de-blocking filter can highly reduce the noise in the video compression.

References

[1] R. Arjunan, V. Vijaya, and V. Kumar, "Adaptive spatio-temporal filtering for video denoising using integer wavelet transform", In: *Proc. of International Conf. on Emerging Trends in Electrical and Computer Technology*, pp. 842-846, 2011.

[2] S. Yu, M.O. Ahmad, and M.N.S. Swamy, "Video denoising using motion compensated 3-D wavelet transform with integrated recursive temporal filtering", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.20, No.6, pp.780-791, 2010.

[3] A.B. Yahya, J. Tan, and L. Li, "Video noise reduction method using adaptive spatial-temporal filtering", *Discrete Dynamics in Nature and Society*, 2015.

[4] H.S. Kim, Y.H. Joo, and K.P. Choi, "De-blocking filtering method of image data and de-blocking filter", *U.S. Patent*, 8,184,713, 2012.

[5] S. Liu, P. Lai, D. Tian, and C.W. Chen, "New depth coding techniques with utilization of corresponding video", *IEEE Transactions on Broadcasting*, Vol.57, No.2, pp.551-561, 2011.

[6] L. Santos, S. Lopez, G.M. Callico, J.F. Lopez, and R. Sarmiento, "Performance evaluation of the H. 264/AVC video coding standard for lossy hyperspectral image compression", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Vol.5, No.2, pp.451-461, 2012.

[7] C. Yim and A.C. Bovik, "Quality assessment of deblocked images", *IEEE Transactions on Image Processing*, Vol.20, No.1, pp.88-98, 2011.

[8] Y. Zhang, C. Yan, F. Dai, and Y. Ma, "Efficient parallel framework for H. 264/AVC deblocking filter on many-core platform", *IEEE Transactions on Multimedia*, Vol.14, No.3, pp.510-524, 2012.

[9] M. Li, J. Zhou, D. Zhou, X. Peng, and S. Goto, "De-blocking Filter Design for HEVC and H. 264/AVC", In: *Proc. of International Conf. on Multimedia*, pp.273-284, 2012.

[10] J. Zhu, D. Zhou, G. He, and S. Goto, "A combined SAO and de-blocking filter architecture for HEVC video decoder", In: *Proc. of the 20th International Conf. on Image Processing*, pp.1967-1971, 2013.

[11] K. Messaoudi, A. Yahi, N. Messaoudi, El-Bay Bourennane, and S. Toumi, "Adaptive Hardware Implementation for the Deblocking Filter Used in H. 264/AVC Using System Generator", In: *Proc. of International Conf. on Embedded Systems in Telecommunications and Instrumentation*, 2016.

[12] G. Suganya and K. Mahesh, "A Survey: Various Techniques of Video Compression", *International Journal of Engineering Trends and Technology*, Vol.7, No.1, pp.10-12, 2014.

[13] T.H. Teng, and C.P. Chung, "Exploiting fine-grain parallelism in the H. 264 deblocking filter by operation reordering", *Future Generation Computer Systems*, Vol.37, pp.76-87, 2014.

[14] N. Kefalas and G. Theodoridis, "A high performance 5 stage pipeline architecture for

the H. 264/AVC deblocking filter", *Integration, the VLSI Journal*, Vol.49, pp.65-77, 2015.

[15] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G.V. der Auwera, "HEVC deblocking filter", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.22, No.12, pp.1746-1754, 2012.

[16] E. Ozcan, Y. Adibelli, and I. Hamzaoglu, "A high performance deblocking filter hardware for high efficiency video coding", *IEEE Transactions on Consumer Electronics*, Vol.59, No.3, pp.714-720, 2013.

[17] S.C. Hsia, W.C. Hsu, and S.C. Lee, "Low-complexity high-quality adaptive deblocking filter for H. 264/AVC system", *Signal Processing: Image Communication*, Vol.27, No.7, pp.749-759, 2012.

[18] G. Correa, L. Agostini, and L.A. Cruz, "Sample-Level Filtering Order for High-Throughput and Memory-Aware H. 264 Deblocking Filter", *ISRN Signal Processing*, Vol.2012, 2012.

[19] L.A. Ayadi, T. Dammak, H. Loukil, M.A. Benayed, and N. Masmoudi, "A Novel Deblocking Filter Architecture for H. 264/AVC", *Journal of Signal Processing Systems*, Vol.89, No.2, pp.281-292, 2017.