DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

Video Compression Based on FPGA Using SIMD Architecture

Hayder Waleed Shnain[1], Mohammed Najm Abdullah[2], Hassan Awheed Jeiad[3]

[1, 2,3]Department of Computer Engineering, University of Technology, Iraq

120361@student.uotechnology.edu.iq[1] , 120002@ uotechnology.edu.iq [2] , 120004@ uotechnology.edu.iq [3]

Abstract— *Recently, video files and images have became the dominant media material for transmitting or storing across different applications that are used by different people. So, there was a serious need to find more effective and efficient video compression techniques to reduce the large size of such multimedia files. This paper proposes SIMD based FPGA lossless JPEG video compression system with the facility of scalability. Generally, the proposed system consists of a software side and a hardware side. The digital video file is prepared to be processed by the hardware side frame by frame on the software side. The hardware side is proposed to consist of two main processing circuits, which are the prediction circuit for calculating the predicted value of each pixel in the certain frame and the encoding circuit that was represented by a modified RLE (Run-Length-Encoder) to encode the result obtained through subtracting the predicted value from the real value for each pixel to produce the final compressed video file. The compression ratio obtained for the proposed system is equal to 1.7493. The throughput improvement for the two and four processing units basing on SIMD architecture was 100 MP/s and 200 MP/s, respectively. The clock results showed that the number of clocks required had become 50% and 25% when using two processing units and four processing units, respectively, from the number of clocks using single processing units.*

Index Terms— *Video Compression, Lossless JPEG, RLE, FPGA.*

I. INTRODUCTION

Recently, the embedded system tends to be more complex and combine multiple functions in one system. Moreover, many applications have massive data such as multimedia, medical imaging, radar, etc. The need for robust architecture increased with high executive performance, and a chip with multiple cores became very common, and especially single instruction multiple data (SIMD). It applies the same set of instructions to various data elements. It is appropriate in multimedia applications like image and video processing applications [1][2].

The process that reduces the amount of data and thus reduces the space needed to store this data is called data compression [3]. There are two essential types of data compression lossless compression and lossy compression. Lossless compression can compress data and then restore the original data without any change after the decompression process [4][5], while lossy causes some loss of information, which leads to not recovering the original data after decompressing. Lossy has a higher compression ratio than it is in lossless [6][7]. To understand the principle of video compression technology, it is necessary to understand the structure of the video where any video consists of a set of successive still images (frames), and it is often the temporal interval for any video is (25 fps) or (30 fps) [8]. Each frame consists of a matrix arranged in columns and rows. Moreover, there is a high nexus between consecutive frames as the video is compressed by removing this close association between these frames [9].

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

In 2015, T. Inatsuki et al. [10] the authors suggested real-time lossless and near-lossless video compression and implement it by using JPEG-Ls and DPCM algorithm and using Huffman coding as an entropy encoder; and this system achieves a compression ratio of 1.818, and the maximum data throughput is 148.5 Mpixles/s.

In 2016, I. D. F. Silva et al. [11], the authors offered video compression with scalability by using SIMD architectures based on FPGA where using low complexity lossless compression for images (LOCO-I) algorithm. The compression ratio was 1.88 as maximum, and the data throughout for single-core was 19.69 Mpixels/s and the maximum data throughout achieving when using ten core where it was 196,9 Mpixels/s.

In 2017, A. H. Hussein et al. [12], the authors offered modified run-length encoding to compressed an image. The authors suggested a technique to improve the compression ratio, as they assumed that if the difference between adjacent pixels was less than (10), meaning that if the current pixel is equal to or greater by ten or smaller by ten than the previous pixel, the two pixels are considered identical. They have applied this method to a group of pictures, and the compression ratio ranged between (5.48-1.07).

In 2018, K.J. Lin et al. [13], the authors proposed lossless compression for electrocardiogram (ECG) signal, where the author used the lossless JPEG algorithm for compression and used huffman coding as an entropy encoder. The compression ratio that achieved was 2.7, with latency through the encoder is 50 ns.

In 2019, S. M. Hardi at el. [14], the authors implemented run-length encoding on two types of images (color and gray). The compression ratio obtained when this algorithm was applied to gray an image ranging from (2.5 - 4.3), while for color images, the compression ratio using run-length encoding ranged between (0.5 - 0.7).

This paper is organized as follows: Section II describes the proposed system. Section III explains the implementation and the results of the test video. Section IV gives the conclusions..

II. THE PROPOSED SYSTEM

This paper proposed a system that is used to compress the video by using a modified lossless JPEG algorithm and exploiting SIMD architecture to improve data throughput. Figure 1 shows the block diagram of the proposed system.

a) Video Source

Video is an electronic way of showing moving visual media. There are two types of videos: digital video and analog video. Generally, any video contains a set of successive frames, and each frame consist of three-layers (Y, Cb, Cr) where y is luminance, Cb is the blue difference, and Cr is the red difference in the case of color videos, and each layer is a group of integer numbers arranged in rows and columns. The number of frames varies from video to others, and it is usually 24 or 30 frames per second.

b) Compressor

The compressor or encoder is suggested to convert the video source into a compressed video. It involves four main stages, as shown in *Fig*.2. Both the software and the hardware are used to design the compressor of the proposed system. Frames are extracted from the video by using the software, while the hardware is used in designing the other stages of the compressor, which include pre-processing, prediction process, and coding with a modified RLE.

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

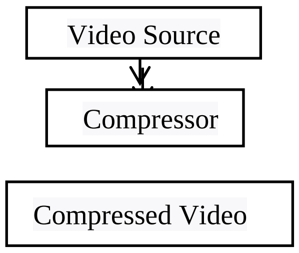


FIG. 1 : BLOCK DIAGRAM OF PROPOSED SYSTEM.

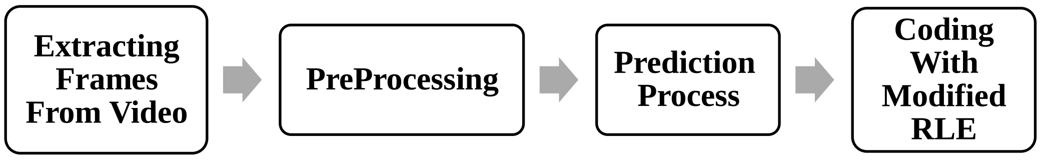


FIG. 2: STAGES OF COMPRESSOR OF THE PROPOSED SYSTEM.

1) Frames Extraction

The first step to compress the video is dividing it into frames and deal with each frame separately to exploit the phenomena of the correlation between the successive frames or the repetition found in the frame itself. This operation is done by using a software side.

2) Pre-Processing

It is the process of preamble each frame to perform a prediction process through creating three adjacent pixels for top and left border pixels where before the pre-processing, there are no adjacent pixels that can be used to find the predictive value to the top and left pixels for each layer of the frame. It's applied through the addition of padding that consists of a row and column of zeroes to the top and left to each layer of each frame of the video. Figure 3 show the frame with a three-layer before and after the pre-processing. It is applied in FPGA by creating two registers, the first one is the row_p register, which represents the number of rows and increases with each row, and the second is the col_p register, which represents the number of columns and increases with each column. The zero value is given to registers that are used to calculate the predictive value of pixel when the value of row_p or col_p is 1.

3) Prediction process

It tries to increase the number of successive identical data. Three adjacent pixels were selected, namely A, B, and C for current pixel X for which the predictive value is to be found, as shown in Figure 4. The predictive value of each pixel will be calculated as follows[15] :

$$X' = A + B - C \qquad (1)$$

$$P = X - X' \qquad (2)$$

where X' is the estimated value and P is the predicted value

In general, there are a set of registers with different sizes where used in the architecture module proposed of the prediction process. These registers with their description were listed in Table1. The sample *x* in Table 1 denotes to X-axis and sample *y* denote to Y-axis. It is noticed from Table 1 that the size of *I-A*, *I-B*, *I-C*, *I-X*, *din1*, *din2* is an 8-bit because the pixel value ranges from 0 to 255 while the size of *I-D* and output is 9-bit in anticipation if the result is greater than 255. For example, if the value of *I-A* is 230, the value of *I-B* is 200, and the value of *I-C* 170, then the value of *I-D* will be 260. So It is needed for 9-bit to store the value.

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

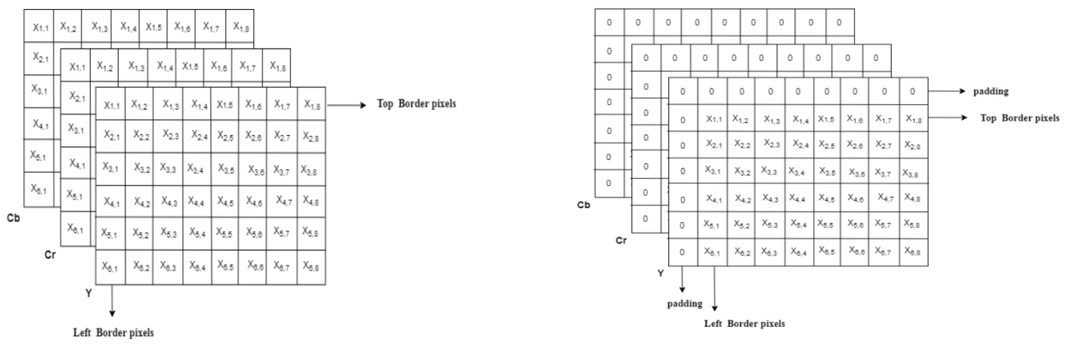


FIG. 3: THE FRAME BEFORE AND AFTER PRE PROCESSING FOR 6*8.

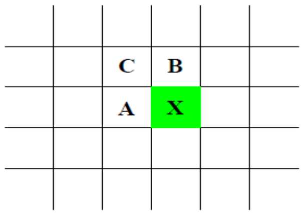


FIG. 4: THREE NEIGHBORING SAMPLES AROUND PIXEL X THAT NEEDS TO BE PREDICTED.

TABLE 1: DESCRIPTION OF REGISTERS USED IN PREDICTION.

Register	Description	# of bits
I-A	Intermediate register, stores the value of pixel in position (*x-1*,*y*) from pixel X	8
I-B	Intermediate register, stores the value of pixel in position (*x*,*y+1*) from pixel X	8
I-C	Intermediate register, stores the value of pixel in position (*x-1*,*y+1*) from pixel X	8
I-D	Intermediate register, used to store the result of X' = A + B - C	9
I-X	Intermediate register, and stores the value of pixel X	8
din1	Stores the value of the each pixel from row1 before entering the system	8
din2	Stores the value of the each pixel from row2 before entering the system	8
output	Stores the predictive value of current pixel which calculated by X- X'	9

Figure 5 shows the block diagram of the prediction circuit. Firstly, registers *din1* and *din2* store the value of the pixels before entering the system. In the first clock, the pixel input in *din1* and *din2* stores in register *I-B* and register *I-X*. At the next clock, the value of *I-B* shift into register *I-C*, and the value of *I-X* shift to register *I-A* and stores the value of *din1* and *din2* into *I-B* and *I-X* and calculates the predictive value and stores the result in register *I-D*. This mechanism continues with every processing clock until the end of the current frame, and it is repeated in the next frame.

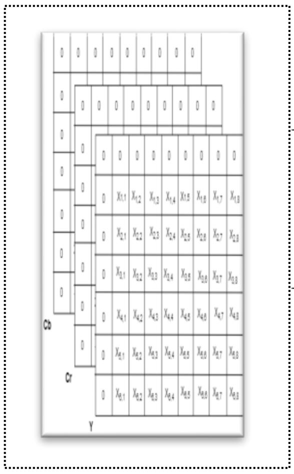
4) Coding with Modified RLE

RLE is used in video compression to store a consecutive set of pixels that have the same value (identical pixels) in a single value and a single counter. RLE is always effective in image compression of a binary image while it is not useful in color images. Moreover, there is the possibility of enlarging the image size instead of reducing it. The latter notice refers to the case when RLE is used directly to encode the color image with its raw pixel values and without any prior process like the prediction for these values of image pixels. In this paper, RLE was used after a prediction procedure for pixels, as mentioned earlier, with a 3-bit size for the repetition counter. For example, if RLE with 3 bit for a counter is applied to the sequence of 8-bit data: [83,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,12,0,0,0,0]. The encoded output for the traditional RLE that

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

uses 8 bits for the repetition counter will be: [(83,1) (0,9) (3,1) (0,9) (12,1) (0,4)] while the encoded output for the modified RLE that based on 3-bit counter will be: [(83,1) (0,7) (0,2) (3,1) (0,7) (0,2) (12,1) (0,4)]. It can be noticed that in traditional RLE, there is a reduction in the sequence from 25 digits that uses 8 bits to represent each of them to 12 digits, and these 12 digits were divided into two groups. The first group consists of 6 digits of the size of 9 bits. The second group is the rest six digits of the size 8 bits. Thus, the number of bits reduced from 200 bits for the original sequence to 102 bits for the encoded sequence by using the traditional RLE. On the other hand, for modified RLE, there is a reduction in sequence size from 25 digits with 8 bits for each to 16 digits, and these 16 digits were divided into two groups. The first group consists of 8 digits, which are the first number of each of the two values enclosed in parentheses in the encoded sequence. These eight digits will be represented by 9 bits, and it is the real value of the repeated predictive value of the pixel. So, the second group is the rest eight digits of the encoded sequence that represents the repetition counter, which is proposed in this work to be represented by just 3 bits for each. That means there will be a reduction in the number of bites required to represent the original sequence from 200 bits to 96 bits only. Generally, there are sets of registers with different sizes where used in the architecture of modified RLE, Table 2 lists these registers with their description. Figure 6 shows the block diagram RLE circuit. Initially, with the first clock, the predictive value of a pixel with a size of 9-bit that store in *I-D* store in a register *I-temp* according to a specified clock. In the next clock, the second predictive value store in *I-D* is a comparator with *I-temp* if they are equal, an enable is active to permit increasing the *I-counter* register by 1. if not, then *I-temp* will copy out to *dout* register, while *I-co*unter will copy out to *counter,* then store the next data element in *I-temp* and so on until reaching the end of the pixel stream. The flag *last* is used to define the desirable output, where it's set to 1 with each desirable output. The modified RLE uses a register *counter* with 3 bits only to store the number of repetitions, so the highest value of the *counter* is 7. That means the proposed architecture of modified RLE suggests that the maximum repetition of a certain data element is 7. In fact, this will lead to decrement the number of bits in the resulted output data stream. The results showed that using *a counter* with larger than 3 bits will reduce the compression ratio of the modified RLE.



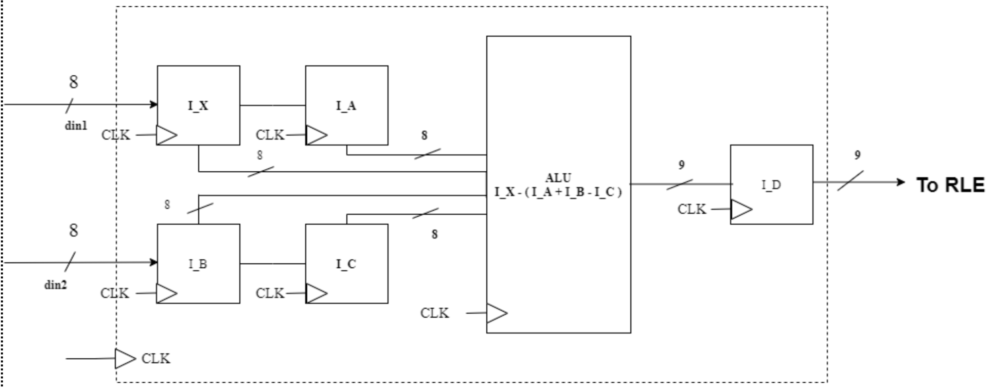


FIG.5: BLOCK DIAGRAM OF PREDICTION CIRCUIT.

TABLE 2: DESCRIPTION OF REGISTERS USED IN MODIFIED RLE.

Register	Description	# of bits
I-temp	Intermediate register, stores the current pixel that is compared to the next pixel	9
I-counter	Intermediate register, stores the intermediate count of identical consecutive pixels	3
last	Flag used to define the Desirable output. Where it's set to 1 with each Desirable output	1
counter	Stores the final count of identical pixels	3
dout	Stores the value of identical pixels	9

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

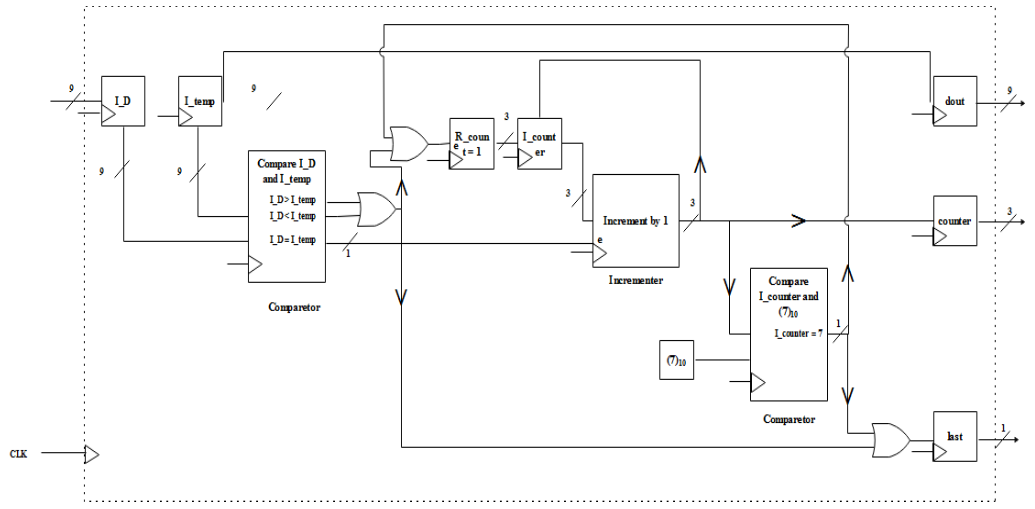


FIG. 6 : BLOCK DIAGRAM OF RLE CIRCUIT.

5) SIMD-based compression process

Essentially, the compression process of the proposal comprises three secondary processes, which are the pre-processing, prediction process, and encoding with the modified RLE. The pre-processing of each of the extracted frames of the video file is a straight-through, vertically dividing the certain frame into two equal partitions in the case of two processing units SIMD (2PUSIMD) is applied. Padding bits are attached to the left and upper sides of the borders of each of the two partitions. The same idea is followed for the case of four processing units SIMD (4PUSIMD) via dividing the single frame into four equal partitions and attaching the padding bits to the left and upper sides of the four partitions.

The prediction circuit of the proposal, which explained in details in section II is duplicated to be comprised of two identical prediction circuits for the case of 2PUSIMD was planned to be tilized. Each of the two prediction circuits is fed with one half of the previously padded frame of a video file. In the case of 4PUSIMD is applied, four copies of the original prediction circuit were constructed with the feeding of each of them by one-quarter of the under processing video frame**.** The modified RLE circuit of the proposal, which explained in details in section II, is duplicated to be comprised of two identical modified RLE circuits for the case of 2PUSIMD was planned to be utilized. Each of the two modified RLE circuits is fed with one-half of the predictive values of the frame of a video file. In the case of 4PUSIMD is applied, four copies of the original modified RLE circuit were constructed with the feeding of each of them by one-quarter of the under processing video frame.

III. IMPLEMENTATION AND RESULTS

The proposed compression system was implemented by using MATLAB and FPGA, where Matlab 2012a that compatible with Xilinx ISE 14.7 is used. The specifications of the FPGA board that used are XC3S500E as a device model with system gates equal to 500 K, block RAM bits equal to 360 K, and clock frequency of 50 MHz. The system's implementation is carried out by using two sides: the software side, represented by Matlab 2012a, and the hardware side, which illustrates by the FPGA board. The software side is responsible for dividing videos stored in the laptop into frames and dealing with each frame separately. The individual frame is rearranged as two inputs stored in the workspace file in preparation for completing the rest of the proposed system using FPGA. To connect between the software side and FPGA side Xilinx System Generator was used, which works to join the ISE 14.7 platform and Matlab 2012a platform provided by Xilinx-ISE 14.7**.** It is specifically focused on Xilinx FPGAs, enabling the developers

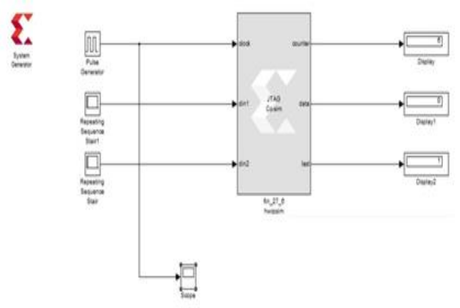
Received 24/10/2020; Accepted 4/2/2021

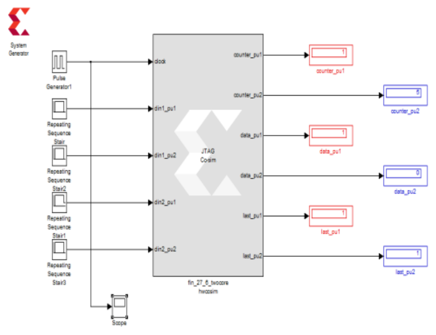
DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

to work in the Simulink environment and to generate parameterized cores, particularly optimized for Xilinx FPGAs. The connection between the laptop and FPGA board is done by using Universal Serial Bus (USB) cable. *Fig*.(7.a,b,c) shows the hardware implementation of Lossless JPEG algorithm by using PUSIMD and 2PUSIMD and 4PUSIMD.

In Fig. (7.b), the red color represents an output of processing unit one, while the blue color represents an output of processing unit two. In *Fig*.(7.c), the red color represents an output of processing unit one, the green color represents an output of processing unit two, the blue color represents an output of processing unit three, and the pink color represents an output of processing unit four.

To assess the performance of the proposed system, it was applied to five different types of standard uncompressed videos that were considered for testing by other researchers[16][17]. The five video files are Akiyo, Foremen, Cartoon, Flower, and Bridge, which are shown in *Fig*. 8.


(A) PUSIMD


(B) 2 PUSIMD

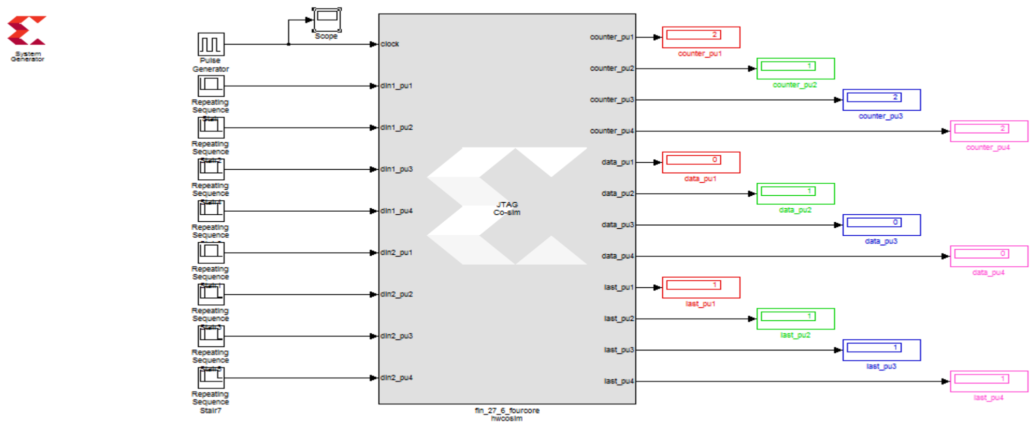

(C) 4 PUSIMD

FIG. 7: HARDWARE IMPLEMENTATION HARDWARE OF LOSSLESS JPEG ALGORITHM BY USING PUSIMD AND 2PUSIMD AND 4PUSIMD.

A)AKIYO

(B) FORMEN

C) CARTOON

(D) FLOWER

(E) BRIDGE

FIG. 8: FIVE TEST STANDARD VIDEOS.

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

All the five mentioned videos have an AVI (Audio Video Interleave) file format, a duration of 10 seconds with 30 frames/second, and the number of total frames for each video file was 300 frames. The size of each frame was 480*640 pixels with a color model of type YCrCb. The Compression Ratio (CR) is denoted by the mathematical formula [18]:

$$\text{Compression Ratio (CR)} = \frac{Uncompressed\ size}{compressed\ size} \tag{3}$$

a) Compression Ratio for Single Frame

This subsection and the next one are dedicated to make a study for ensuring that using the modified version of RLE that uses 3-bit as a size for repetition counter with a prediction which was proposed by this paper is better than the traditional RLE when applied for the video coding to obtain better compression ratio. Essentially, the traditional RLE is based on using 8-bit as a size for the repetition counter and is used for coding of any type of data file without any considerations for the specification and nature of that data file. In this work, it is founded that the size of the repetition counter can be taken less than 8-bit. In fact, it can be just a 3-bit size, and this reduction in the counter size will improve the obtained compression of the video file.

Here, three different suggestions for RLE were taken to study. The first is RLE with a size of 8-bit of repetition counter. Secondly, RLE is considered with an 8-bit counter size with prediction. Thirdly, RLE with a 3-bit counter size with prediction. However, these three suggestions were applied to five individual frames that were taken from the five test video files to evaluate how efficient they are in terms of reducing the size of video files. The obtained compression ratio for the three suggestions is shown in *Fig*. 9.

It can be observed that the lowest value for the compression ratio is obtained for 8-bit RLE without prediction. This is because the prediction process increases the number of identical pixels and the highest value obtained of the compression ratio for RLE with a counter size of 3 bits with applying the prediction approach of the pixels. It has been found that in most cases, the number of identical pixels does not exceed 7. Also, it is noticeable that the compression ratio varies from one frame to another due to the difference in the spatial details.

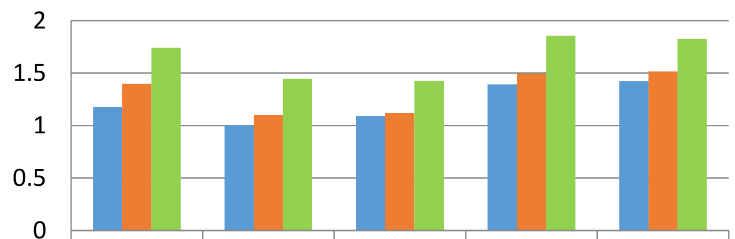
b) Compression Ratio for Video Files

The same scenario of the three RLE variants of the previous subsection has been applied to the five video files, which are Akiyo, Foremen, Cartoon, Bridge, and Flower, for evaluating the performance of the proposed Lossless JPEG that is based on the modified 3-bit RLE with prediction. This is done by taking the 300 frames of each video sample and then applying the proposal on the three layers (Y, Cr, Cb) of each frame of the certain video file. Figure 10 shows the compression ratio obtained for each video type when the three RLE variants are applied.

By looking at *Fig.* 10, it could be noticed that the lowest compression ratio was achieved when using the traditional RLE and note the extent of the improvement achieved. Firstly, by using an 8-bit for RLE after the prediction process. Secondly, by using 3-bit for RLE after the prediction process.

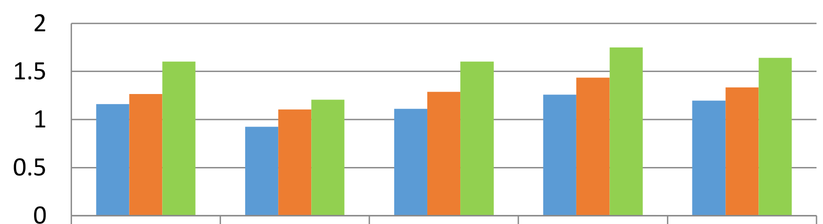
Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6



	Akiyo	Foremen	Cartoon	Bridge	Flower
🟦 RLE(8-bit)	1.18	1.002	1.09	1.3925	1.4235
🟧 RLE(8-bit) with prediction	1.4	1.1	1.119	1.4959	1.5151
🟩 RLE(3-bit) with prediction	1.7412	1.4463	1.42483	1.8558	1.8266

FIG.9: COMPRESSION RATIO FOR DIFFERENT COUNTER SIZE OF RLE AND PREDICTION.



	Akiyo	Foremen	Cartoon	Bridge	Flower
🟦 RLE(8-bit)	1.1592	0.9247	1.1114	1.2593	1.1972
🟧 RLE(8-bit) with prediction	1.2662	1.1047	1.2886	1.4351	1.3343
🟩 RLE(3-bit) with prediction	1.6028	1.2079	1.6031	1.7493	1.6411

FIG. 10: COMPRESSION RATIO FOR VIDEO FRAMES.

c) The Effect of the Size of Repetition Counter of RLE

To study the effect of the size of the repetition counter of RLE on the compression ratio, Lossless JPEG with a different counter size of RLE was used to encode an individual frame. RLE with 2 through 8 bits counter size was applied to five 480x640 color frames, which are Akiyo, Foremen, Cartoon, Bridge, and Flower. The results are shown in *Fig*. 11. It can be noticed that the highest value of the compression ratio is obtained when the RLE counter size is 3 bits. That means, according to the taken sample frames, it is suitable for the counter size to be 3 bits to give the highest compression ratio since the length of almost sequence of identical pixels not exceeds 7 in general.

d) Effect of SIMD

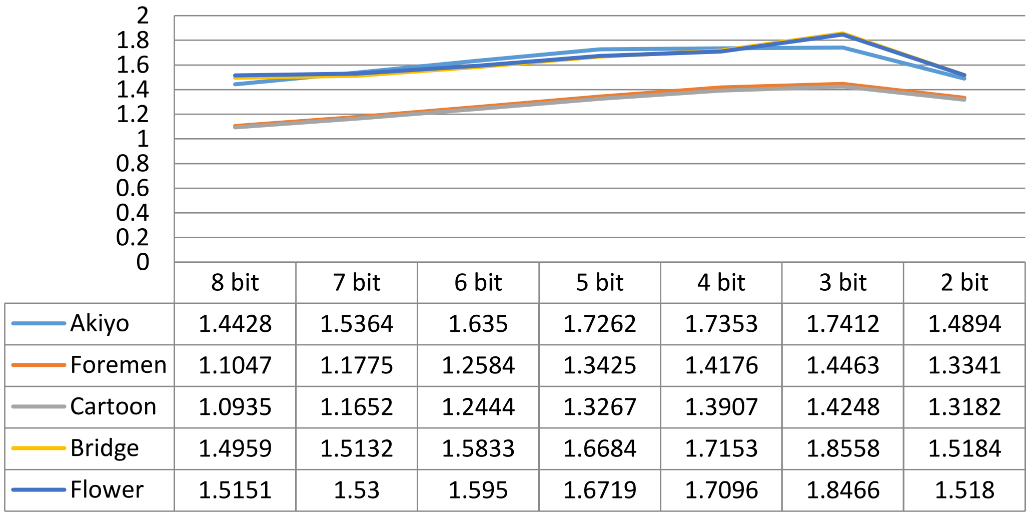
The effect of using the SIMD architecture in the compression ratio is shown in Table 3, where simple N represents the number of processing units that are used. It has been noticed that the compression ratio suffers from a decrease in its value whenever more the processing units are used due to the fact that the identical pixel chain has been broken by dividing the frame by the number of the used processing units.

e) Evaluation for SIMD parameters

By using the ISE platform, the highest frequency of the proposed system that obtained is 81.7 MHz, but the maximum frequency provided by the Spartan 3E FPGA board is 50MHz, so the result is calculated according to this frequency. Table 4 shows the effect of using the SIMD architecture in some parameters, which are the number of clocks and the time needed to the compressed frame with size 480*640 (YCrCb), maximum data throughput, logical elements which are used, and the percentage of wasted clocks.

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6



	8 bit	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit
Akiyo	1.4428	1.5364	1.635	1.7262	1.7353	1.7412	1.4894
Foremen	1.1047	1.1775	1.2584	1.3425	1.4176	1.4463	1.3341
Cartoon	1.0935	1.1652	1.2444	1.3267	1.3907	1.4248	1.3182
Bridge	1.4959	1.5132	1.5833	1.6684	1.7153	1.8558	1.5184
Flower	1.5151	1.53	1.595	1.6719	1.7096	1.8466	1.518

FIG. 11: COMPRESSION RATIO FOR LOSSLESS JPEG WITH DIFFERENT COUNTER SIZES OF RLE.

TABLE 3: THE INFLUENCE OF USING SIMD ARCHITECTURE IN COMPRESSION RATIO.

Frame	Compression Ratio		
	N = 1	N = 2	N = 4
Akiyo	1.7412	1.7396	1.7358
Foremen	1.4463	1.4443	1.4416
Cartoon	1.4248	1.4236	1.4214
Bridge	1.8558	1.8012	1.7978
Flower	1.8466	1.7947	1.7913

TABLE 4: THE EFFECT OF USING SIMD IN SOME PARAMETERS.

Number of PU$_s$	Number of Clocks	Time (ms)	Maximum Data Throughput (Mpixels / s)	Logical Elements	The Percentage of Wasted Clocks
1	923041	18.460825	50	279	0.0015611
2	462241	9.244820	100	551	0.0031174
4	231841	4.636820	200	1043	0.0062154

IV. CONCLUSIONS

In this paper, the main gained conclusions is a hardware implementation of a Lossless JPEG algorithm based on FPGA has been accomplished. The results show that the system's performance based on FPGA offers fewer process times than software processing time, where the software is more than hardware by (554*100)% time. Using prediction with modified RLE increases the compression ratio. The possibility of obtaining an acceptable compression ratio by using the Lossless JPEG with simple and uncomplicated compression algorithms when compared to other more complicated compression algorithms. The introduced modified RLE improved the obtained video compression ratio by just reducing the number of bits for the repetition counter of the traditional RLE from eight to three bits for the application of video compression. The adoption of SIMD architectural increases the maximum data throughput but consume more logical resources.

Received 24/10/2020; Accepted 4/2/2021

DOI: https://doi.org/10.33103/uot.ijccce.21.1.6

REFERENCES

[1] M. Baklouti et al., FPGA-based many-core System-on-Chip design: Embedded Hardware Design (MICPRO),Elsevier, 2015, pp.38.10.1016/j.micpro.2015.03.007. hal-01144977.

[2] W. Y. Lo, D. P. K. Lun, W. C. Siu, W. Wang, and J. Song, "Improved SIMD architecture for high performance video processors", IEEE Trans. Circuits Syst. Video Technol., vol. 21, no. 12, pp. 1769–1783, 2011, doi: 10.1109/TCSVT.2011.2130250.

[3] P. Yellamma and N. Challa, "Performance Analysis Of Different Data Compression Techniques On Text File," Int. J. Eng. Res. Technol., vol. 1, no. 8, pp. 1–6, 2012.

[4] A. Gupta, A. Bansal, and V. Khanduja, "Modern lossless compression techniques: Review, comparison and analysis," Proc. 2017 2nd IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT2017, no.February,2017,doi: 10.1109/ICECCT.2017.8117850.

[5] V. Singh, "A SURVEY ON LOSSLESS TEXT DATA," no. May, pp. 1–5, 2018.

[6] M. Singh, S. Kumar, S. Singh, and M. Shrivastava, "Various Image Compression Techniques: Lossy and Lossless," Int. J. Comput. Appl., vol. 142, no. 6, pp. 23–26, 2016, doi: 10.5120/ijca2016909829.

[7] David Salomon, Data Compression: The Complete Reference. Springer-Verlag, London, 2007.

[8] Iain E. Richardson," The H.264 Advanced Video Compression Standard", Wiley Publishing, 2010.

[9] Thyagarajan, K. S, "Still image and video compression with MATLAB",Hoboken: Wiley-Blackwell. ,(2011).

[10] T. Inatsuki, M. Matsuura, K. Morinaga, H. Tsutsui, and Y. Miyanaga, "An FPGA implementation of low-latency video transmission system using lossless and near-lossless line-based compression," Int. Conf. Digit. Signal Process. DSP, vol. 2015-Septe, pp. 1062–1066, 2015, doi: 10.1109/ICDSP.2015.7252041.

[11] I. de Faria Silva, L. Souza e Silva, C. Alves Carneiro, Z. M. Assis Peixoto, and F. Magalhaes Freitas Ferreira, "An FPGA-Based SIMD Architecture for Video Compression with Scalable Throughput," IEEE Potentials, vol. 35, no. 1, pp. 32–37, 2016, doi: 10.1109/mpot.2014.2312426.

[12] A. H. Husseen, S. S. Al-juboori, and R. J. Mohammed, "Image compression using proposed enhanced run length encoding algorithm," Ibn AL-Haitham J. Pure Appl. Sci., vol. 24, no. 1, 2017.

[13] K. J. Lin, H. H. Huang, and Y. Y. Lin, "An FPGA Implementation of Lossless ECG Compressors Based on Multi-Stage Huffman Coding," 2018 IEEE 7th Glob. Conf. Consum. Electron. GCCE 2018, pp. 410–413, 2018, doi: 10.1109/GCCE.2018.8574652.

[14] S. M. Hardi, B. Angga, M. S. Lydia, I. Jaya, and J. T. Tarigan, "Comparative Analysis Run-Length Encoding Algorithm and Fibonacci Code Algorithm on Image Compression," J. Phys. Conf. Ser., vol. 1235, no. 1, 2019, doi: 10.1088/1742-6596/1235/1/012107.

[15] ITU-T. ISO DIS 10918-1, "Digital compression and coding of continuous-tone still images (JPEG) ", Recommendation T.81.

[16] Z. Haitham and M. K. Mahmood Al-Azawi, "Video Compression Based on Motion Compensation and Contourlet Transform," *2018 3rd Sci. Conf. Electr. Eng. SCEE 2018*, no. Mc, pp. 90–94, 2018, doi: 10.1109/SCEE.2018.8684077.

[17] E. Akyol, D. Mukherjee, and Y. Liu, "Complexity control for real-time video coding," Proc. - Int. Conf. Image Process. ICIP, vol. 1, pp. 77–80, 2006, doi: 10.1109/ICIP.2007.4378895.

[18] S. Katsigiannis, D. Maroulis, and G. Papaioannou, "A GPU based real-time video compression method for video conferencing," in 2013 18th International Conference on Digital Signal Processing (DSP), 2013, pp. 1–6.

Received 24/10/2020; Accepted 4/2/2021