Lossless Compression Methods for Real-Time Images

Rand B. Mohammed[1] & Roelof van Silfhout[2]

[1]Mechatronics Engineering Department, Faculty of Engineering, Tishk International University, Erbil, Iraq
[2]School of Electrical and Electronic Engineering, University of Manchester, Manchester, United Kingdom
Correspondence: Rand B. Mohammed, Tishk International University, Erbil, Iraq.
Email: rand.basil@tiu.edu.iq

Doi: 10.23918/eajse.v6i2p169

Abstract: This paper proposes and implements two lossless methods, to compress real-time greyscale medical images, which are Huffman coding and a new lossless method called Reduced Lossless Compression Method (RLCM), both of which were tested when applying a random sample of greyscale medical images with a size of 256x256 pixels. Different factors were measured to check the compression method performances such as the compression time, the compressed image size, and the compression ratio (CR). The system is fully implemented on a field programmable gate array (FPGA) using a fully hardware based (no software driven processor) system architecture. A Terasic DE4 board was used as the main platform for implementing and testing the system using Quartus-II software and tools for design and debugging. The impact of compressing the image and carrying the compressed data through parallel lines is like the impact of compressed the same image inside a single core with a higher compression ratio, in this system between 7.5 and 126.8.

Keywords: Embedded System, Greyscale Image, Real-Time System, Huffman Coding, Lossless Compression Method, Medical Images

1. Introduction

The image is a matrix of pixels organised in rows and columns, usually defined according to two factors, the size of the image and the size of each pixel; each pixel carries a specific value within the image. Image quality is measured by resolution, i.e., the number of 8/16-bits pixels per square inch. An image is made up of a foreground and a background. The foreground is usually the most important part of the image, since it refers to a certain object, and it is usually the most useful part in any research and application. The second part of the image is the background, which generally corresponds to the image in a unique and specific way and is less important than the foreground (Lei, 2013; Roy et al., 2015; Tang, 2010).

When describing an image, many factors such as colour format and image size are important. The image size is based on the number of pixels in each image and the dimension in which the image is produced, i.e., 2D. The colour of the image is the most frequently used feature to retrieve an image, and it is classed as either grey or coloured. The greyscale image is based on two main colours, black and white, along with all the shades in between. The coloured image, on the other hand, has many formats to define it: the RGB format is the main colour space, but other formats such as YUV (Saboori, 2015; Zhongshui QU, 2010; Lei, 2013) and YCbCr (Chen, 2010) can be transformed based on RGB.

Received: October 10, 2020
Accepted: December 20, 2020
Mohammed, R.B., & van Silfhout, R. (2020). Lossless Compression Methods for Real-Time Images. *Eurasian Journal of Science & Engineering*, 6(2), 169-188.

The greyscale image, is a digital image, mostly seen nowadays in different applications such as medical applications, depending on the pixel size. The greyscale images often come with the size of the pixel equal to 8-bits, which provides 256 grey shades to define the image. However, in technical uses such as medical imaging the pixel comes in larger sizes like 10-bits, 12-bits, and 16-bits. The pixel value of the greyscale image represents intensity. Different techniques can be used to calculate the distribution of the pixel's intensity for any set of pixels such as the histogram. The histogram is the most popular technique used to display the number of pixels at each intensity value, which helps to find the most frequent pixels inside the image and to understand the characteristics of the image (Javier, 2003). The histogram for the greyscale images, as shown in Fig. 1 (a, b), is produced for the whole image which displays the pixel values (intensities) with their count number. Each image has a unique histogram that presents the distribution of its pixel values, for example, as shown in Fig. 1 (a), the range of pixel intensities for the MRI image was between 0 and 255, while the average value was 63. While for the X-ray image, as shown in Fig. 1 (b), the range of pixel intensities was between 0 and 214 while the average value was 119. For this MRI image, most of the pixel intensities were surrounded by two values as shown in Fig. 1 (a), while the pixel intensities of the X-ray image were spread over a wide range of intensity values.

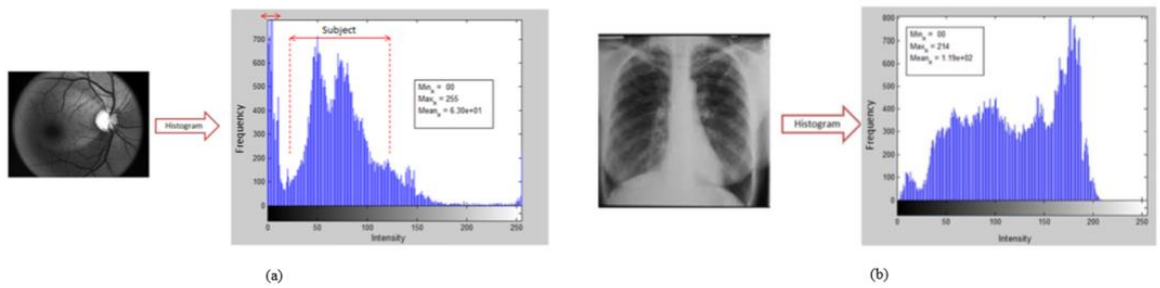


Figure 1: The histograms for (a) the MRI and (b) the X-ray medical images

However, the histogram for the coloured images shows the brightness for each pixel with its count number in the whole image produced either as 2D for each colour, red, green, and blue, or as a 3D with three axes, each of which represents one channel, red, green, or blue.

Many formats are available to use to define data, starting from the simplest through to the most complex: data, image, audio or speech and video. Images are of two types: offline images, which are received from an external device and stored on a storage device to apply some processors in the future whereby the time factor is not important, and real-time images that require applying an immediate processor to the receiving data and then directly forwarding them without being stored. The processing time is the most important aspect of real-time systems in terms of instability and security, which means the full processor must be completed within a time limit to avoid any noticeable delay in the system. For example, with a real-time camera the full processor that is applied to each single image must be completed before generating the next image from the camera (Senturk et al., 2015).

2. Medical Images

Medical imaging has developed over the years and resulted in major imaging advancements; for instance, this technology can be applied before actual surgery and different types of technology can be adapted to different types of medical images. Medical images are mostly used in clinical studies, diagnosis, and treatment planning, to diagnose and treat diseases as well as to reveal the internal body

structure hidden by the skin and bones. Medical images are also used to identify abnormalities by establishing a database of normal anatomy and physiology, which can provide information about the human structure by offering a wide variety of functional processes while producing valuable information about oxygen consumption, cell metabolism and blood supply and circulation (Ritter et al., 2011; Lee, 2015).

Different diagnosis machines such as Ultrasonography (USG) or Ultrasound images, Magnetic Resonance Images (MRIs), X-Ray, Computed Tomography (CT), and Capsule Endoscopy (CE), etc. are used to produce medical images in a digital form for human organs (Longanathan, 2011). Based on the diagnosis machine, the images are produced either in the form of a sequence of digital images or a single image. In radiology, for example, a 2D image is produced for the X-ray images produced by using X-ray beam that passes through them, while the CT consists of a tube and a detector, both rotating around the body to generate multiple images as 3D images. Medical images are mostly greyscale images such as the X-ray, the MRI and the CT. Fig. 2 shows examples of different greyscale images taken by different diagnosis machines. The accuracy of medical images is very important, because any error in diagnosing patients can be very harmful.

Some systems require applying image processing, e.g., image compression, to speed up the enhance operation when analysing the medical images, the limitation in the storage space, and the required time to carry these images over the network. For example, the CT generates a set of images between 200 to 400 images, each image with a size equal to 512x512 of 16-bits, requires storing a big memory space (Ritter, et al., 2011; Lee, 2015). However, when applying compression to the medical images, choosing a technique must be accurate and all the details in the image should be retained for future recovery of the data file (Adiwijaya, 2013).

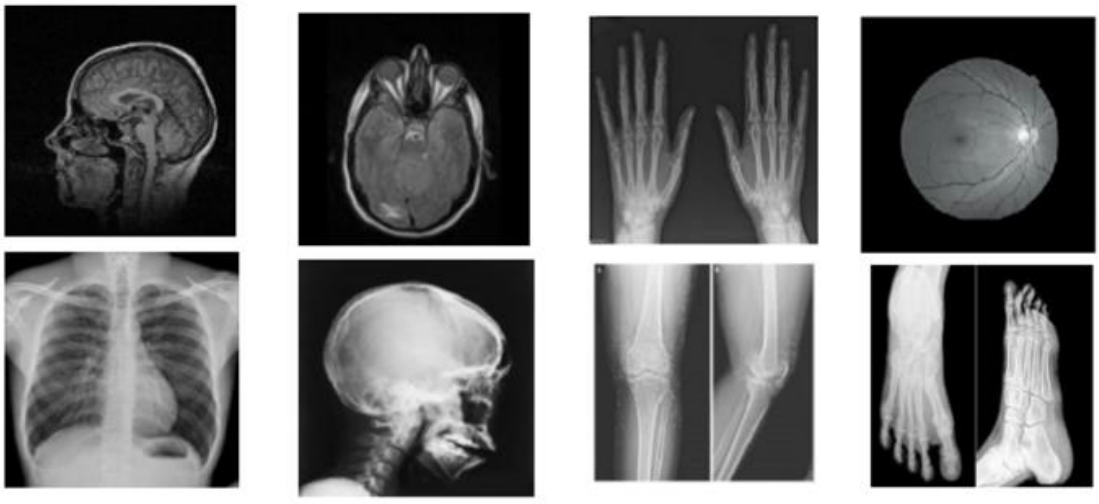


Figure 2: Examples of greyscale medical images

3. Image Compression

Compression is, in general terms, data-encoding with a transformation method used to compress the size of a data file (Akhtar et al., 2014). Data/Image compression is required in applications with storage limitations, such as portable devices, as well as it is useful in digital image processing, because it helps to reduce any redundancy in the original image and reduce the consumption of expensive resources, since the original image is retrieved from the compressed image by decompressing it accordingly. The

uncompressed image requires a wide bandwidth for transmission and occupies a large amount of storage space (Arthur, 2012; Devaraj et al., 2005). Data compression has many advantages; for instance, it helps to save bandwidth, storage space, cost and the time required for transmitting data between two places. Conversely, data compression has a downside, to reuse the data, the compressed data must be decompressed, which adds extra processing that may be detrimental to some applications (Arthur, 2012; Akhtar et al., 2014).

Image compression can be classified into two types: lossy and lossless. Both the lossy and lossless compression methods support many compression techniques. Run-length encoding, Huffman coding, and arithmetic coding are used for the lossless compression method, while dictionary-based algorithm, Lempel-Ziv-Welch (LZW), JPEG and MPEG are applied to the lossy compression method (Ashraf et al., 2006; Arthur, 2012; Akhtar et al., 2014).

Lossy compression is one of the compression methods that are mostly applied to analogue data stored digitally, such as graphics, videos, and sound files. It differs from the lossless compression method in that it accepts losses of some data during the compression operation. The lossy compression method is a good candidate for compressing large volumes of data, but it might lead to a significant reduction in image resolution, which may not be suitable for some types of images such as medical images. The reduction in image resolution in the lossy compression method by throwing away some of the image data is not suitable for medical image application, because each part of the image could carry important information about the patient which could be harmful. Lossy compression is mainly based on transforming the image into the frequency domain using algorithms such as Fast Fourier Transform (FFT), Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transformations (DWT) (Nelson, 1991; Telagarapu et al., 2011; Katharotiya et al., 2011).

The lossless compression method can be referred to as noiseless coding since it can recover the original image fully and completely from compressed data. The main drawback of this method, however, its low compression ratio. In lossless image compression, the compression ratio is normally between 2 and 10 (Gonzalez, 2002). In the lossy compression method, the compression ratio is improved by throwing away most redundant data without losing much quality, and it is usually used for applications where it is not clear whether an amount of data is lost; for example, it can be used for compressing images, videos, and sound files. Both the lossy and lossless compression methods support many compression techniques. Run-length encoding, Huffman coding, and arithmetic coding are used for the lossless compression method, while dictionary-based algorithm, Lempel-Ziv-Welch (LZW), JPEG and MPEG are applied to the lossy compression method (Ashraf et al., 2006; Arthur, 2012; Akhtar et al., 2014).

However, in some cases two different compression methods are applied to compress the same image, to take the advantages of both by keeping the important details of the image as well as getting a high compression ratio, namely the hybrid compression method such as JPEG2000 (Firoozbakht, 2010).

4. Lossless Compression Method

Lossless compression is mainly used with medical images to reduce the large original image size and at the same time to keep every detail. Different techniques can be used with the lossless compression method, all of which are guaranteed to generate an exact, identical copy of the original image after decompression. Some of these methods have been developed to match real-time tasks and telemedicine, such as lossless JPEG and a lossless version of JPEG2000; these methods can provide

images with high compression ratios through complex systems and high energy compaction (Arthur 2012). In general, the lossless data compression technique involves two processors, modelling and coding (Nelson, 1991). Modelling is used to feed the model with good probabilities to create a good compression system. Coding, on the other hand, is used to produce codes for the symbols based on the number of their possibilities inside the image, or predict the value for each pixel based on the value of the neighboring pixels, such as Huffman coding, which is based on checking the appearance probability for each symbol in a message and then creating a table based on these, by providing a shorter code for the symbol with higher probabilities, and providing a larger code for the symbol with lowest probabilities (Nelson, 1991; Bahmanyar et al., 2004; Ashraf et al., 2006).

Many researchers have been working to improve the lossless compression method either by improving the way to implement the lossless technique or by developing a new compression method to meet their application requirements. For example, some of them are focusing on changing the colour space to another space that helps to keep the image quality and at the same time have a good compression ratio as done by Kim (2012), who used the lossless compression method to compress a coloured image, by developing a transformation method called the reversible colour transform (RCT) to convert the colour image from the RGB colour space into YCuCv, following one of the lossless compression methods. For instance, JPEG2000 was used to compress Y, while Cu and Cv were compressed with a new method developed by the author. Based on this method, which was applied for testing images, the maximum reduction in the file size was 15.11%. Other researchers combined two compression techniques to get a good compression ratio. For instance, Edmundson (2012) developed a fast compression method by combining the lossy and lossless compression methods to improve the compression ratio and provide a useful image descriptor. This new method is based on adding the Huffman tables to the JPEG header that can "be optimised on a per image basis". However, Arif (2012) developed a different technique to compress medical images by combining two effective methods, namely Huffman coding and run-length coding, both of which are lossless compression methods, to reduce the delivery time and the data volume, save the power in scanning applications, and achieve maximum compression. Jimenez-Rodriguez (2013) developed a new method called the visual lossless coding, which is based on the ability to recognise the important parts of the image, which are compressed by using JPEG2000. However, the other parts with less important details are compressed by using the lossy coding technique. The main achievement of this method is that the decompressed image is identical to the original image, the fast-coding process, the high compression ratio, and the lower channel bandwidth which is used to transmit the images. Mudassar (2012) applied a sequence of lossless compression methods to compress the greyscale MRI and colour CE medical images taken from MRI-TIP database, with an image size equal to 256x256, with 8-bits for each colour component. Two factors are measured: The Compression Ratio (CR) that defines the space saving at the memory space, and the quality of the decompressed image, by checking the percentage of the image recovery after being decompressed as compared to the original image; the peak signal-to-noise ratio (PSRS) was used to find it. The algorithm was first tested on individual images, secondly on the MRI images, thirdly on greyscale MRI image sequence, and finally on four-colour CE image sequence. The CR was 4.481 for the MRI image sequence, between 2.775 and 4.377 for the CE coloured image, and 66.09% for the PSNR.

5. Methodology

Generally, images are compressed to increase system bandwidth and to save on storage. For example, when the image is compressed with a compression ratio equal to 2, the size of the compressed image

will be half the size of the input image, and the time required to carry the compressed image is almost half the time required to carry the uncompressed image (original image). However, if this image is compressed inside four cores instead of one core and four parallel lines are used to carry the compressed image, then the time required to carry the compressed image through four parallel lines will be one-fourth the time required to carry the compressed image through one line, which is already half the time required to carry the uncompressed image. That means the time to carry the image (that compressed inside four cores with a compression factor of two) through four parallel lines is almost one-eighth of the time spent when carrying the uncompressed image through a single line. This time is equal to the time required to carry the compressed image through one line when the image is compressed inside a single core with a CR equal to 8. And the amount of the pixels carried during one second in time (pixel rate) through four parallel lines is almost eight times more than the amount of the pixels for the uncompressed image that carried through a single line, compressing data through multiple cores instead of single core called parallel system.

The compression system will be a part of bigger system, so there is a time to compress the image that is called compression time as well as there is a time to carry the image to the next step which is called transmission time. Based on the transmission time and the pixel rate, the impact of compressing the image inside parallel cores is like the impact of compressing the image inside a single core with a higher compression ratio: compressing the image inside eight cores with a compression ratio equal to 2 is like compress the image inside one core with a compression ratio equal to 16.

With parallel system, when eight compression systems connecting in parallel, and all of them work together to compress data, compression data will be eight times faster than using a single compression system. With each period, the full system can compress the eight packets of the image, at a total size of 64K pixels (64x1024x16-bits). Moreover, on the other side, of the system will consists of multiple de-compression systems connected in parallel, to decompress data and generate the original image eight times faster than when using a single decompression system. Two compression methods were applied, Huffman coding and Reduced Lossless Compression Method (RLCM).

5.1 The Huffman Coding

The Huffman coding is a lossless compression technique used to compress images, without affecting image quality. When the image is decompressed, it provides an exact copy of the original one. In the lossless compression method, the compression ratio is low compared to that in the lossy method. In essence, the Huffman coding is based on converting a 2D image into a 1D image and replacing the most abundant pixel in the image with a new value that has the smallest size, while the least frequent pixel in the image is replaced with another, albeit larger, assignment.

Based on Huffman coding, the images are divided into packets which are sent to the compression system at a size of 8K pixels (8192 pixels), to create a new, initial table that contains new assignment data. The system will check the most popular pixel inside each packet of the image and replace the value with a new assignment from the initial table. The most common pixel in each packet will receive a newer value from the initial value that is smaller in size, which is usually placed in the first location in the table. All the packets will be checked, and all the original values replaced with a newer assignment value. After completing this replacement process for all the images and creating a newly compressed image based on a new assignment value, the system generates a unique look-up table for each packet, which contains the value for all the original pixels with their new assignments. These are

then all sent directly to the next part of the system and are used in the opposing de-compression system, to generate the original packets.

The Huffman coding has advantages and disadvantages. The main advantage is that it is an easier way to compress and decompress images, by keeping the quality of the image, also it is easy to implement. The first of the main disadvantages is that it requires creating a unique table for each image that will add more size to the compressed image, which will ultimately affect the compression ratio. The second disadvantage is that if the image does not contain many repeated pixels, the created table will be large and add more size to the output compressed image. In the worst case, the table created beside the compressed image will be larger than the original image, in which case there is no point in compressing the image, because the time required to carry the original image data without compression is less than the time required to carry the compressed image, which therefore defeats the main objective of compressing data. To avoid this situation, which may lead to losing the main benefits of compressing the system, a smart control that can be added between the read data and the compression part. This smart system checks the image before sending it out to be compressed, by checking the number of the repeated pixels. If the number is less than or equal to the default value, which can be set by the designer or the user and depends on the image size, the data will be forwarded to the compressing system; otherwise, it will send the data out either directly to the transport layer without compression or to another compression method. With this control, the system has a built-in smart device which can decide whether to compress an image, to achieve the best quality in all cases.

5.2 The Reduced Lossless Compression Method (RLCM)

The RLCM is a new compression method based on the main idea behind the lossless technique, namely focusing on compressing an image without affecting its quality. Then, after decompressing the image, an exact copy of the original image is regenerated. The main objective of this method, compared to Huffman coding, is improving the compression ratio for all image types and sizes. The RLCM creates for each image two temporary locations, one called a compressed image and the other called a table, both of which are important for the decompression process in reproducing the original image.

RLCM works by scanning the image and selecting one of its pixels as an index value, which can be either the first pixel inside each image or the most frequently used pixel. The index value is assigned to the first location in the table, and the rest of the pixels in the image are compared with the index value – if the pixel value is equal to the index, 0 is assigned to the same location in the compressed image; otherwise, the pixel value, which refers to intensity, is added to the table and assigned 1 in the compressed image, because the images are greyscale images. Each pixel inside the image is replaced by either 0 or 1 in the compressed image, which means the size of the output compression image is equal to the size of the original image, albeit defined in bits instead of pixels. For example, if the original image size were 1024 pixels, the output compression image would be 1024 bits. Beside the compressed image, there is a unique table that contains the original pixel value for each image. In its first location, it contains the index that is replaced by 0 in the compressed image, while the rest of the table defines the other pixels that are organized in correct order and are assigned as 1 inside the compressed image. The compressed image and the table are forwarded to the next step in the system, and they are required by the application layer on the receiver side, to decompress the data and generate the original image.

On the other side, de-compresses the image based on the compressed image and its table. First, the system will check the data in the compressed image, to see if it is equal to zero, and then assign the

index value of the table to the same location in the original image; otherwise, if it is equal to 1, the next value of the table will be read and the correct value assigned to the generated image. This checking operation continues until the whole compressed image has been scanned.

Like any other method, the RLCM has its advantages and disadvantages. The main advantage is improving the downside of the lossless technique by compressing the image with a high-quality ratio, without adding an extra size to the original data. This method can compress images with a compression ratio between 0.97 and 16. However, one of the disadvantages of this method is that if the first pixel is not the most frequent and is used as the index for the image, the table will be larger than if the most frequent pixel is used for this purpose. The solution in this case is to check the image before compressing it, to find the most frequent pixel and assign it as an index value, and then complete the compression process as described above. Another possible disadvantage is that sometimes the image does not have many repeated pixels, in which case the size of the generated table will be nearly equal to the original image, and the output will be the compressed image with the generated table. However, while the compressed image is the same size as the original, and the pixels are replaced with bits, the output image and its table will be almost equal to the original image without compression and much smaller than the output compression image in the same situation in Huffman coding.

To get a clear comparison between the Huffman Coding and the RLCM compression methods and their impact on the medical images, a set of 300 greyscale medical images are applied which refer to different parts of the human body, such as the brain, the eye, the hand, the foot, the chest, the thumb, and the spinal cord. These images are taken in a factor of greyscale image and do not want to see color image because most of the medical images such as the X-ray are only dealing with greyscale. The images were taken using two different medical diagnosis techniques e.g., X-ray and MRI which are the most common types of medical images. These images are generally collected as greyscale images in which each pixel has either 8 or 16-bit precision. The full range of images selected together give a good overview of typical images that would be taken for diagnosing a medical issue for a human patient. The images used are open access and thus widely available to the community without issues such as a requirement for ethical approval. The available number of the open access greyscale medical images was three hundred, and it was a sufficient sample size to identify the main trends in our image compression algorithm. This was justified by taken one hundred images then expanded to three hundred images. The input size of these images is 256x256 pixels. Different factors have been calculated for each selected method to check the RLCM performance.

The first factor is the compressed image size, which determines the output data after applying the compression method. This includes the table and the compressed image which vary for each image and for the same image after applying two different techniques. If the whole image is compressed inside one core as described earlier, there is only one value for the compressed image size that can be read directly from the running system. However, if multiple parallel cores are used to compress the image, there are multiple values which refer to the output compressed image, from each compression core, the total compressed image size value is the summation of the output compressed size of all the parallel cores.

The second factor is the compression ratio (CR), which is calculated by using equation (1). The uncompressed image refers to the input image size which is, in our system, equal to 65536-pixels and its fixed size for all the images will be tested, while the compressed image size refers to the output image size that varies depending on the input image.

$$Compression\ Ratio\ (CR) = \frac{Uncompressed\ Image\ Size}{Compressed\ Image\ Size} \quad (1)$$

The third factor is the execution time, which is related to the time required to do the whole process. In this case, to compress the whole image because there are two cases to compress the data, there are two ways to find the system execution time. First, when the whole image is compressed inside one core, the execution time is the time required to compress the whole image for both the pipeline and the sequential techniques (see Fig. 3 and 4). Secondly, when dividing the image into multiple packets, each packet is compressed at one core within a specific time, in which case the execution time for them is equal to the longest time taken at one of the cores. Two ways are available for compressing the image:

1. Firstly, compressing the whole image inside a single core, and reading data from it based on either the pipelined technique or the sequential technique. The pipelined technique involves more than one processor at the same time. While the system is transferring the previous packet (for example packet-1), it compresses the next packet (for example packet-2). The total execution time for the system is based on the sum of the execution time for each row, which is either the sum of the compression time or the sum of the transmission time depending on which time is longer. With both the RLCM and the Huffman Coding, the compression time is much longer than the transmission time due to the large number of processor cycles at the compression core compared with the number of cycles at the transmission core. However, according to the sequential technique, there cannot be more than one processor at the same time. For example, to compress and transmit data, the system can either compress the whole image and transmit it, or divide the image into multiple packets, then until the first packet is compressed and transmitted, the system starts compressing the second packet and keeps doing the same processor until finishing all the images. The total time required to compress and transmit the whole image will be the summation of the compression time and the transmission time. In conclusion, the pipelined technique is more efficient than the sequential technique, because the sequential technique is too slow, which makes it unsuitable to run for the real-time system (Mohammed, 2017).
2. The second available technique for compressing the image is by dividing the image into eight sub-images called packets, 8K-pixels each. Each packet is compressed inside a single compression core, which means there are eight single systems all connected in parallel and share the same source of data. Carrying data from each single core is based on the pipelined technique. Compressing data by using parallel cores making the execution time eight times shorter than that when using the first case. However, the execution time at each core can be either the compressed time for each core or the transmitted time, depending on which one is longer. Inside each core, two ways are available to compress the sub-image: compressing all the sub-images and sending them out to the next core, which is like the Huffman coding, and compressing each row and transmitting it alone as is the case in the RLCM. In the first case, the delay between generating images until reading them at the next core will be the whole time spent at the compression core to compress all the sub-images with a size of 8K-pixels. However, with the second case, the delay will be the time required to compress the first row from the sub-image when its size is 256-pixels. The main reason behind choosing the size of the data processed each time to be equal to the row size (i.e., 256 pixels) is that the RLCM compression method was designed to compress 256 pixels at a time. In compressing the image inside one core, the system takes 256 processor-cycles to complete the compression, while the

system takes 32 processor-cycles to compress the same image inside eight parallel cores; when the processor-cycle is the time to compress each row from the image (Mohammed, 2017).

6. Implementation

The lossy compression technique is applied to a general image that has limitations in storage size while the quality of the image is not as important as the compression ratio. While, the medical images, which are read from a camera or a sensor, depending on the user, are compressed at one stage, without affecting the image itself, the lossless compression method is used in this instance.

The compression and decompression are apart from bigger system, for example they are implemented inside the application layer in the TRIO model that described in (Mohammed, 2018). The compression will be implemented at one side that called transmitter, and the decompression is implemented at the other side called receiver, as shown in Fig. 3.

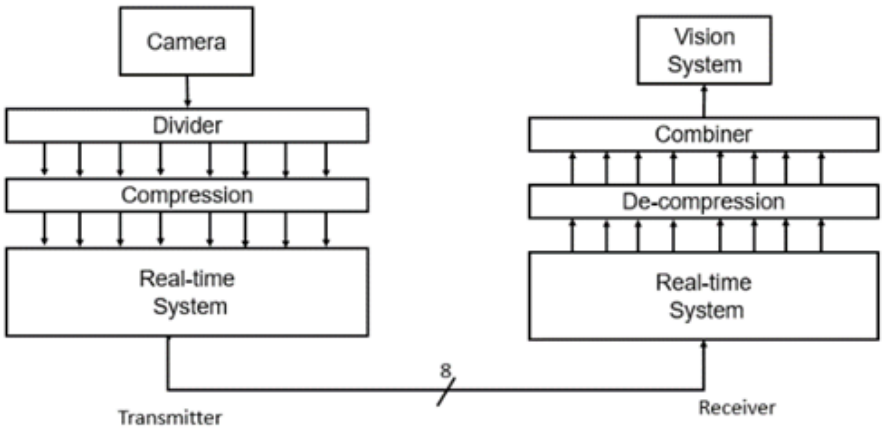


Figure 3: Eight cores connected in parallel based on the pipelined technique to compress and transmit data

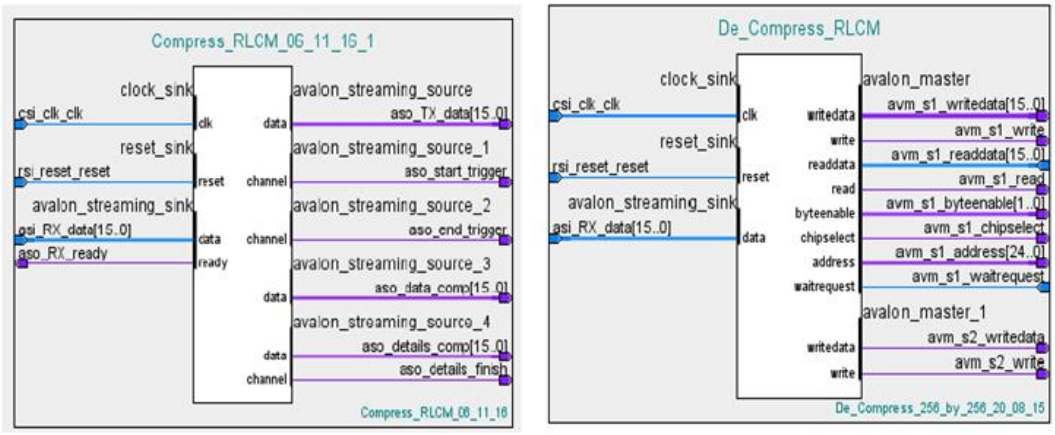
Practically, starting with the divider on the transmitter side and the combiner on the receiver side, the system consists of multiple individual systems connected in parallel by sharing the same memory location at both sides. The compression and de-compression systems are built as an intermediate step to connect between the vision device and the rest of the system.

On the transmitter side, each compression system has two interface ports, one connected to a bus in the divider core, which is used to read data, and the other interface port connected to the next layer, i.e., the real-time system. The compressed image is carried over this bus. However, on the receiver side, each de-compression system has two interface ports, one connected to the final layer of the real-time system, from which the data derives, and the other connected to one of the combiner buses, which carries the image after it has been decompressed.

Two lossless methods are applied, one based on the Huffman coding and the other based on a new, novel compression method called the reduced lossless compression method (RLCM). On the other side, the image will be decompressed based on a similar compression method, to fully recover the original image without any loss of detail. For both compression methods, the data read from the memory are divided into sub-packets, each of which is individually compressed, carried through the

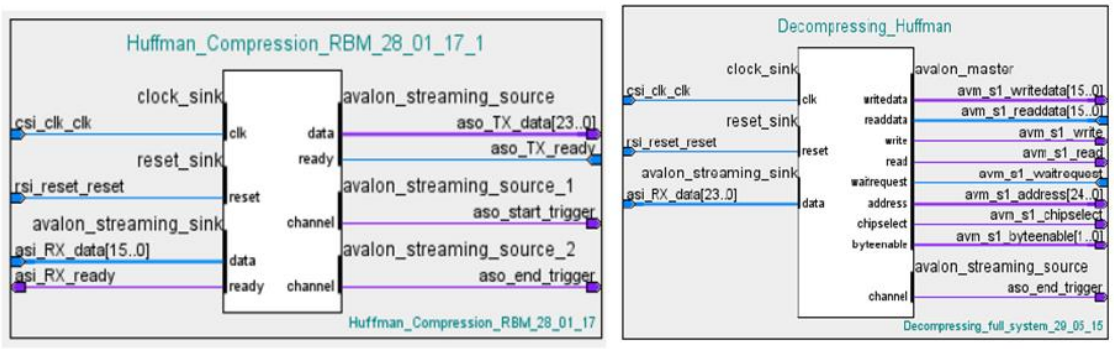
system, received on the receiver side, and then decompressed. The time required to complete the whole process as described above must be equal to or less than the time required to read a new image to get the system to run in real time; otherwise, a significant delay will occur between reading and displaying the data.

The compression system was built as an embedded system inside the FPGA; the model was used to test both compression methods, by reading greyscale medical images. The model was implemented as custom cores and connected inside the Qsys builder. The main cores were compress and decompress cores, which were based on either the Huffman or the RLCM compression methods. These cores were created using Qsys builder by writing a Verilog-HDL code and then converting it into a reusable IP block, as shown in Figs. 4 and 5. The created IP-blocks are available to use with any future system and will connect with any model. The compression and decompression cores for both methods included different interface buses, to connect with other system's cores. The main two ports were clock and reset, which were responsible for synchronizing the system connected and exported directly to an external clock and reset source. The other interface buses were Avalon-ST and Avalon-MM; both methods in the compressing core were used to connect the compressing core with the source of data, while in the decompression core they were used to connect the decompressing core with the destination device. The Avalon-ST source was used to carry a compressed image from the compressed core, and the Avalon-ST sink was used to carry a compressed image into the decompression core. The Avalon-ST source and sink buses were different for both methods – for Huffman coding the length of the compressed image was fixed depending on the length of the original image, which means the compression and decompression cores were already initialized with a fixed length for the image, while the width was 24 bits, to carry the compressed image with a table. For the RLCM the length was changeable based on the number of repeated pixels inside the image, because compressed core has two buses with Avalon-ST, one of which is a 16-bit data bus that carries a compressed image with a table, while the other one is a 16bit control bus that carries the length of the compressed image which has been used by the decompressed core to decompress the image.



(a) (b)

Figure 4: The IP blocks inside Qsys builder for: (a) the RLCM compression core (b) the RLCM decompression core



(a) (b)

Figure 5: The IP blocks inside Qsys builder for: (a) the Huffman coding compression core (b) the Huffman coding decompression core

Both compression systems were designed to compress/decompress a packet size equal to 65536 words. The input data can be either general data such as patient data-base, or medical image if the system used with medical application. For a larger image/packet it had to be divided into sub-packets, and the total compression time was the processing time for the sub-packets multiplied by the number of sub-packets inside the full image. For 2D images the system included a core that converts 2D image into 1D image by reading them as row-by-row and then forwarding on to another core to divide it into sub-packets.

Both methods were built and implemented as a full hardware embedded system, then they were applied individually to the FPGA board to be tested. Thereafter, they were successfully compiled in the Quartus-II software. A Stratix-IV GX FPGA device was used inside a Terasic DE4 development board that shown in Fig. 6, and a signal tap analyser tool was used to debug the design via the FPGA device. The system was also tested by running it in a software in MATLAB.

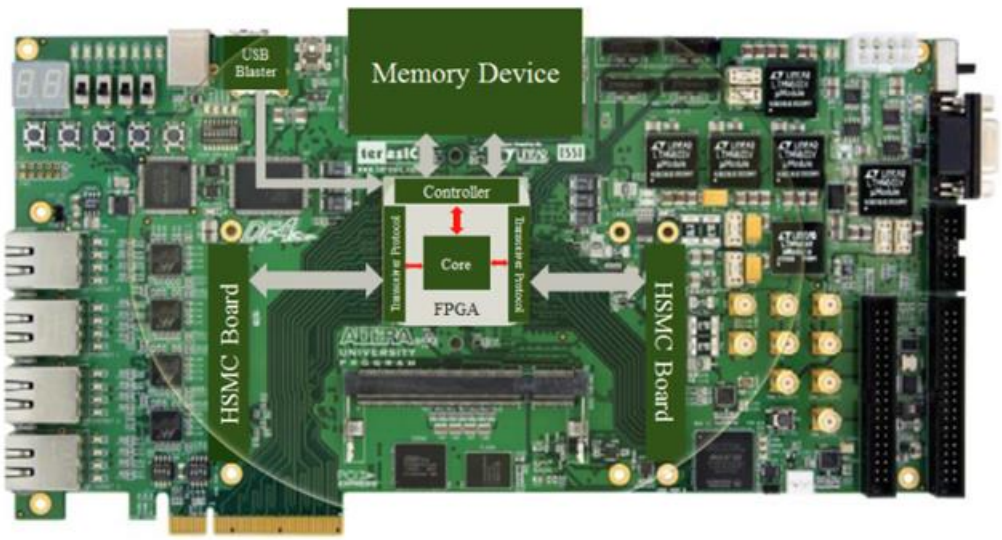


Figure 6: DE4 board (Mohammed, 2017)

7. FPGA and MATLAB Testing

A sample of medical images was applied to the system by uploading these images to the memory, and then running the system to interact with these data. Various medical images were taken to test the system, all of which were greyscale images, to check and evaluate the compression system performance. The schematic diagram of this test case is shown in Fig.7 for both compression techniques, i.e., the Huffman coding and the RLCM.

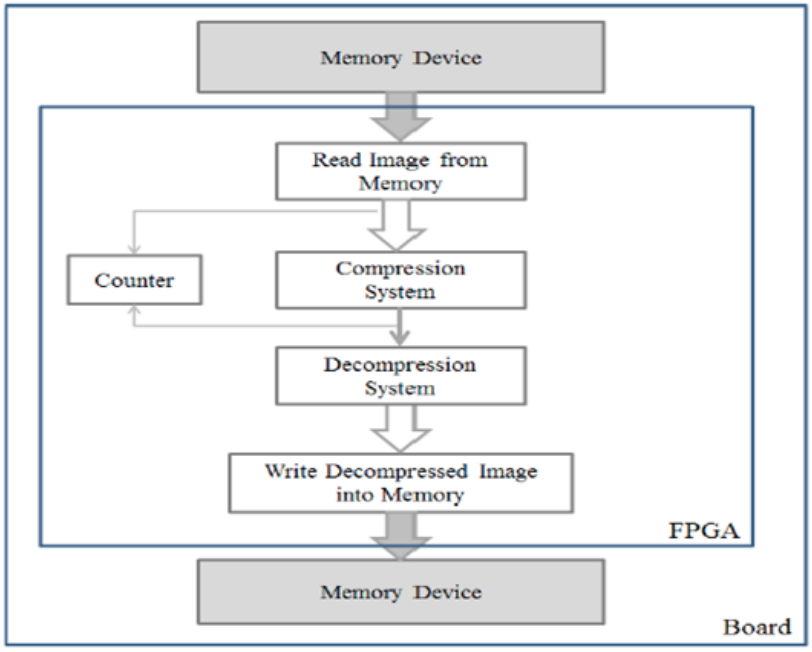


Figure 7: The test case schematic diagram for reading greyscale medical images from a fixed memory location

The images showing different parts of the human body (for example the brain, the head, a hand, a foot, an eye, the chest, and the spinal-cord) were taken from Google. They were taken using different medical diagnosis techniques such as the X-ray, the CT, and the MRI. The images are greyscale images with a size of 256 x 256 pixels, with 8/16-bits per pixel. By applying a sample set of the images to the system and running it in the hardware, equations will be established for each of the required performances: the compression ratio (CR), the compression time, and the execution time. The taken images have been tested in the hardware system, are carried different characteristics. They have been chosen based on their CR that cover the range of the CR for the RLCM compression algorithm, which is between 1 and 16; the CR for any such images will be within this range. The compression system was a main part of the system; therefore, the performances and their equations will be depended on the CR as will be presented in the next section.

The sample set was run in the hardware system and used to establish the performances' equations. These equations were concluded based on the sample of the image that covered the system boundaries based on the CR values, which means these equations are covering the boundaries for each performance (compression-time and execution-time), so it is expected the performance of any greyscale image will be within this range, which makes these equations can be applied widely to find the performance for any such image.

Different factors were measured to check the system performances: the CR, the compressed image size, and the system execution time. These factors were calculated for each image after applying the RLCM and the Huffman coding. The CR was calculated by dividing the size of the input image by the size of the compressed image, while the size of the compressed image was found for each image after the compression method was applied using the Signal-Tap Analyzer by monitoring the internal system signals, and the execution time was measured by inserting an embedded counter that started counting when the first pixel was read and stopped counting when the final pixel of the image was compressed and sent out to the physical medium. The embedded internal-counter had two input triggers, start-trigger, and stop-trigger; the start-trigger was triggered to start counting when the first pixel read from the memory while the end-trigger was triggered to stop counting when the last pixel was sent out to the next part of the system.

In this system, the concluded equations will be applied later to the rest of the images in MATLAB to analyze them and to find their performances in a similar way for running them based on the hardware. The performances' values are therefore similar for both hardware and MATLAB, this has been justified by considering another sample of six greyscale images, as a test sample, which has been chosen randomly from the 300-images, the test sample has been tested at both hardware and MATLAB approaches, the performances are same for both approaches. Accordingly, it is concluded that to know the performances, of such greyscale images, we can use either the MATLAB analysis or the hardware system.

However, testing the system in MATLAB is mainly based on four things: reading the images from its source, finding their histograms, compressing the image using either the RLCM or the Huffman coding, and calculating the required performance according to the equations when running the system as a hardware. The performances are the compressed image size, the compression ratio, the compression time, and the execution time. In addition to the previous performances, with any medical image one of the main performances guarantees the lossless compression method, which means the decompressed image must be an identical copy of the original image. To display a decompressed image and compare it with the original image, MATLAB software will be used. A 300 medical images are applied to the system built in MATLAB. This system reads and displays the original image, finds the histogram, compresses the image, decompresses it, and then displays it.

8. Experimental Results

With most compression techniques, the quality of the compression method is determined by the compression ratio and space-saving. However, in this project both techniques were used in real-time systems because the quality of a compression method is measured based on four factors: the image compressed size, the compression ratio, the execution time, and the quality of the decompressed image compared with the original image, particularly in relation to whether anything has been lost from the image which is known as the lossless compression method. The image compressed size is determined by monitoring, at the signal tap analyzer, the size of the output signals from the compression core. The second factor, the compression ratio (CR), is determined by dividing the size of the input image to the size of the compressed image. The third factor, the execution time, is determined by assigning an internal counter inside the system. The last factor, the quality of the compression method, is determined by comparing the decompressed data with the original. When running the system in hardware, the quality is checked by using the signal tap analyzer to compare the read data from the source with the written data inside the memory after being decompressed. When testing the system in MATLAB, on

the other hand, the output compressed size, the compression-ratio, the compression time, and the transmission time are calculated based on the results from the hardware test, while the error rate is found by comparing the decompressed image with the original image after implementing the compression and decompression system in MATLAB.

Different medical images have been taken to investigate the performance of the Huffman and the RLCM compression methods and their ability to compress greyscale medical images. These greyscale medical images that are related to different parts of the human body by uploading their pixel's intensity to the memory to interact with the rest of the system. Some of these images are tested in the hardware system by implementing it as an embedded system. Based on these empirical results, equations were used to predict the performance, the compressed image size, the CR, the compression time and the transmission time. These equations are used to calculate the features of a wider sample of images in MATLAB.

The model that is used to test both methods in the hardware is shown in Fig. 8. By initializing the on-chip memory with the image data (pixel intensities), each method was designed using the Qsys builder tool. After successfully compiling in the Quartus-II software, each method was implemented as an embedded hardware system inside the FPGA device and debugged by the host PC using the signal tap analyzer to monitor the assigned signals P1, P2, P3, P4, P5 and P6. For each image applied to the system, there are three factors measured: the compressed image size, the CR, and the execution time. P2 relates to the size of the compressed image, P6 the output value of the counter size which is used to measure the execution time, and the CR by dividing the size of the input packet to the size of the compressed image.

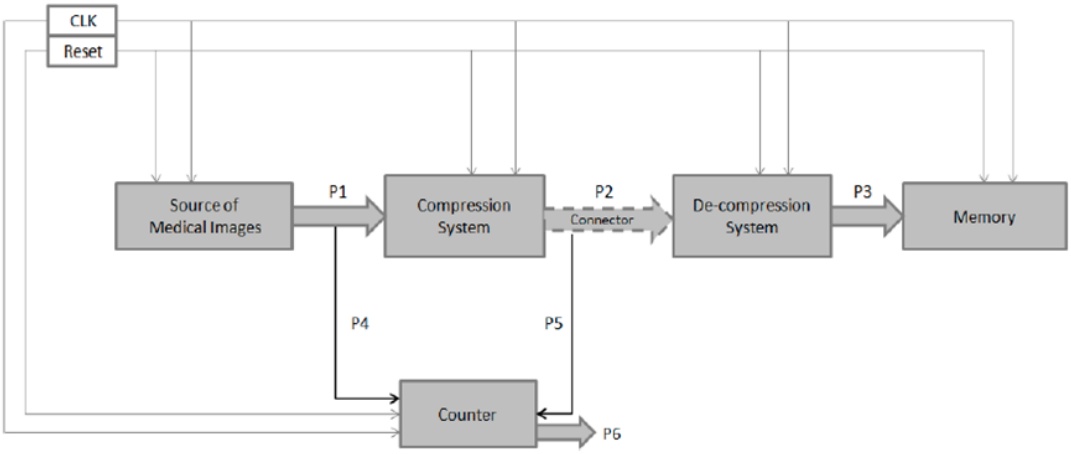


Figure 8: A schematic diagram of the basic compression and decompression system test when reading greyscale medical images

With this testing case, the whole image is compressed inside a single core by using either the Huffman coding or the RLCM, but the transmitting data from this core are based on the pipelined technique, which means that the system will transmit each row, after being compressed, to the next core and start compressing the next row until finishing the whole image. The compression time is always higher than the transmission time since the number of processors at the compression core is much larger than the number of processors at the transmitter core, and that while the system is based on the pipelined

technique to read data from the compression core, the execution time will be the higher of these two values, viz. the compression time.

The CR, compressed image size and execution time were found for a random sample of the medical images when applying both compression methods. For the RLCM, both the transmission and compression time values depend on the CR. The compression time is measured based on the number of cycles inside the core that are required to compress the image, which is equal to the reading value from the embedded counter, this value represents the system execution time. The RLCM method has a CR between 0.97 and 16, which is normal for the lossless compression method. The compression time mainly depends on the value of the CR. With the high CR, the execution time will be short while with the low CR, the system takes a longer time to do the full processor so that we can conclude a formula for a direct relation between the compression time and the CR. According to this formula, the compression time mainly depends on three factors: the size of the input image (in this case it is fixed to 256x256-pixels), the value of the input frequency, and the CR. Based on this formula, the compression time, can be calculated for any input image; by considering the actual compression time can be differ from the calculated compression time by ±70µs.

$$Compression\ Time\ (sec) = \frac{Size\ of\ the\ Input\ Image\ (pixel) * N}{Input\ Frequency\ (Hz)} \qquad (2)$$

When $N = 2.1885 * CR^{-0.302}$

Both compression methods are applied to a wider sample of greyscale medical images in MATLAB. The system in MATLAB reads the images from the computer, finds their histograms, compresses them, and finally finding their performances which represent the compressed image size, the CR, the compression time, and the transmission time based on the equations that are found when testing the system in the hardware and applying a random sample of greyscale medical images. The RLCM is designed to work with a pixel size equal to 16-bits, with the taken samples of the medical images, the pixel size was 8-bits, so every two neighboring pixels with 8-bits in size are read and combined to generate a single pixel with 16-bits in size to be compressed as a single pixel at the compression-core, then this pixel is split into two pixels after it has been decompressed at the decompression-core.

At the Huffman coding, as Fig. 9 shows the outputs from MATLAB, the CR value was between 0.5 and 2.7, while the transmission time was between 0.3ms and 0.5ms. However, the time required to compress the whole sample of images (301 images) was 5.214s, which takes around 17.3ms to compress each image. Comparing the compression time value with the required time to read a new image from its source, which is 1.3ms, the Huffman coding cannot be the ideal method to compress images in real-time systems due to its low CR and slow processing time.

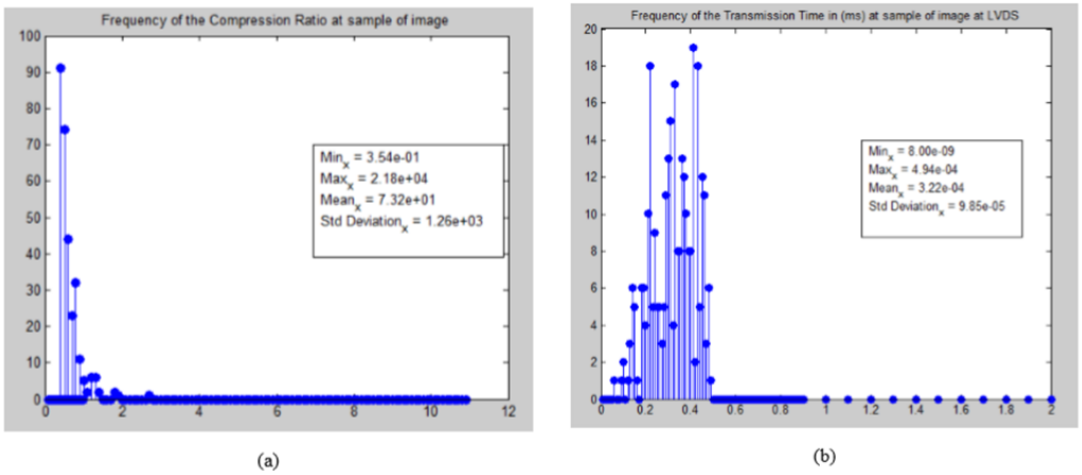


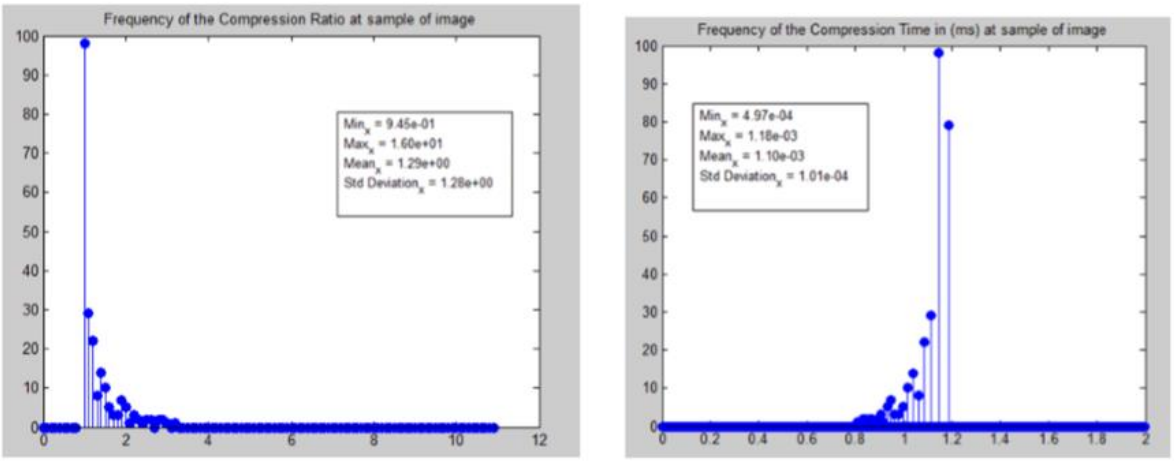
Figure 9: The Output Histograms for (a) the Compression Ratio and (b) the Transmission Time when applying a sample of greyscale medical images to be compressed using the Huffman coding in MATLAB

However, at the RLCM coding, as Fig. 10 shows the outputs from MATLAB, the CR value was between 0.98 and 15.99, while the compression time to compress each of them individually varied between 0.497ms and 1.18ms based on the CR value as shown in equation (2), when the total compression time to compress 301 images was 1.902s. However, when considering the difference between the measured compression time and the calculated compression time (which was ±70 µs), the compression time can be in a range between 0.425ms and 1.25ms. The CR value mainly depends on the types of the input image. The taken sample of the medical images was the MRI and the X-ray. When running the system, the value of the CR was checked for each image, the lowest CRs (around 0.98) were mainly for the chest, the head, and the neck X-ray images, while the other X-ray images such as the hand, the foot, and the legs were compressed with higher CRs. However, most MRI images were compressed with a CR more than or equal to 2. As shown in Fig. 10, the average value of the CR was 1.29 and that of the compression time was 1.1ms. That means in a term of CR, the RLCM is a suitable compression method for all MRI images and some of the X-ray images (in order of their CRs were between 1.5 and 16), while the other X-ray images were compressed in a size almost equal to their original size (without compression). While when comparing both characters (the CR and the compression time) to the Huffman coding characters and the time required to generate a new image, the RLCM would be an ideal method to compress the greyscale medical images in a real-time system for all medical image types.

While when compressing the image inside multiple cores instead of one and carrying the compressed data through parallel lines, the compression-time was reduced almost to one-eighth of the time spent when compressing the whole image inside one core (the compression time was reduced from 1.18ms to 0.148ms), and the compression ratio to compress the image inside eight parallel cores is similar to the impact of compressed the same image inside a single core with a higher compression ratio, in this case between 7.5 and 126.8.

To check if the compression methods were generating an identical image after it is decompressed, the same module shown in Fig. 10 was built, but in this case, it was implemented in MATLAB instead of Quartus-II, which includes the decompression process in addition to the compression core tested earlier. The MATLAB was used in this case because of its ability to display the image by using the

available functions and instructions, which offered an easy way to compare the original image with the decompressed image. The decompressed image in Fig. 12 was identical to the original image in Fig. 11, which proves that the RLCM was a compression method ideally used for medical applications.



(a) (b)

Figure 10: The Output Histograms of the Compression Ratio and Compression Time when applying a sample of greyscale medical images to be compressed using the RLCM in MATLAB. The upper figure: frequency of the compression-ratio of a sample of images, and the lower figure: the frequency of the compression-time in ms of a sample of greyscale medical images

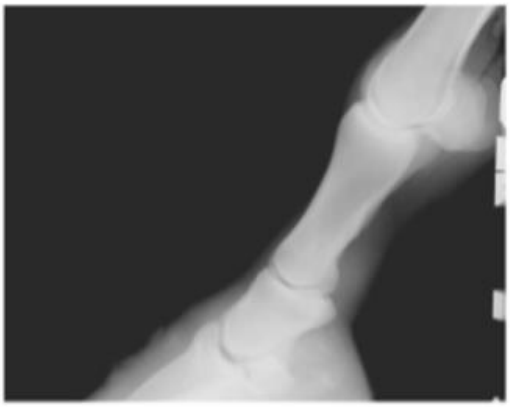


Figure 11: The original row image

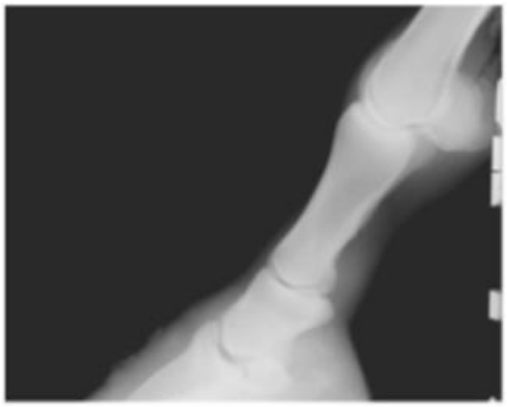


Figure 12: The decompressed image

9. Conclusion

One of the most significant outcomes and achievements of the present research was successfully building, implementing, and testing the compression system by using different types of greyscale medical images. Two different compression algorithms were applied, the Huffman coding and the RLCM, both of which were tested when applying a random sample of greyscale medical images with a size of 256x256 pixels. Different factors were measured to check the compression method performances (when compressing the image inside a single core) such as the compression time, the compressed image size, and the CR. For the Huffman coding, the compression time was 17.3ms and

the CR is between 0.5 and 2.7. However, for the RLCM the compression time was 1.1ms and the CR was between 0.97 and 16; for the X-ray image the CR was between 0.97 and 1.5 while for the MRI image the CR was more than 2. In view of these values, the RLCM matches the real-time processing due to the short compression time at all the CR values. The compression time was between 0.497ms and 1.18ms at the RLCM when the average value was 1.1ms and at the Huffman coding, the time required to compress each image was 17.3ms almost 16 times slower than the RLCM. The impact of compressing image inside parallel cores and carrying the compressed data through parallel lines is like the impact of compress the same image inside a single core with a higher compression ratio: compressing the image inside eight cores with a compression ratio between 0.97 and 15.9 is like compress the image inside one core with a compression ratio between 7.5 and 126.8.

References

Adiwijaya, P., Faoziyah, F., Permana., & Wirayuda, T. (2013). Tamper detection and recovery of medical image watermarking using modified LSB and Huffman compression. Lodz, IEEE, pp. 129 – 132.

Akhtar, N., Khan, S., & Siddiqui, G. (2014). A novel lossy image compression method. Bhopal, IEEE, pp. 866 - 870.

Arif, A. S., Mansor, S., Karim, H. A., & Logeswaran, R. (2012). Lossless compression of fluoroscopy medical images using correlation and the combination of Run-Length and Huffman Coding. Langkawi, IEEE, pp. 759 - 762.

Arthur, A., & Saravanan, V. (2012). Efficient medical image compression technique for telemedicine considering online and offline application. Dindigul, Tamilnadu, IEEE, pp. 1 - 5.

Ashraf, R., Akbar, M., & Jafri, N. (2006). Diagnostically lossless compression-2 of medical images. Arlington, VA, IEEE, pp. 28 - 32.

Bahmanyar, R., Datcu, M., & Rigoll, G. (2014). Comparing the information extracted by feature descriptors from EO image using Huffman Coding. Klagenfurt, IEEE, pp. 1 - 6.

Chen, G. (2010). Application of processing techniques from color image to grey image. San Juan, PR, IEEE, pp. V2-372 - V2-375.

Devaraj, K., Munukur, R. K., & Kesavamurthy, T. (2005). Lossless medical-image compression using multiple array technique. Hong Kong, IEEE, pp. 837 – 840.

Edmundson, D., & Schaefer, G. (2012). Fast JPEG image retrieval using optimised Huffman tables. Tsukuba, IEEE, pp. 3188 - 3191.

Firoozbakht, M. (2010). Compression of digital medical images based on multiple regions of interest. St. Maarten, IEEE, pp. 260 - 263.

Gonzalez, R., & Richard E. W. (2002). Digital image processing. Pearson Education International.

Javier Nunez-Garcia, Vassilios Mersinias, Kwang-Hyan Cho, Colin P. Smith and Olaf Wolkenhauer, 2003. A Study of the Statistical Distribution of the Intensity of Pixels within Spots of DNA Microarrays: what is the Appropriate Single-Valued Representative?

Jimenez-Rodriguez, L., Auli-Linas, F., Marcellin, M. W., & Serra-Sagrista, J. (2013). Visually lossless JPEG 2000 decoder. Snowbird, UT, IEEE, pp. 161 - 170.

Katharotiya, A., Patel, S., & Goyani, M. (2011). Comparative analysis between DCT & DWT techniques of image compression. *Journal of Information Engineering and Applications.*

Katti, R. S., & Ghosh, A. (2009). Security using shannon-fano-elias codes. Taipei, IEEE, pp. 2689 - 2692.

Kim, S., & Cho, N. I. (2012). A lossless color image compression method based on a new reversible color transform. San Diego, CA, IEEE, pp. 1 - 4.

Lee, L.K., & Liew, S.C. (2015). A survey of medical image processing tools. Kuantan, IEEE, pp. 171 – 176.

Lei, H. (2013). A new retrieval method based on YUV Clour information for digital libraries. Hong Kong, IEEE, pp. 128 - 130.

Mohammed, R. B. (2017). *Novel scalable and real-time embedded transceiver system*. PhD thesis, University of Manchester

Mohammed, R. B., & van Silfhout, R. (2018). Real-time transceiver system based on rapid-io protocol. *Eurasian Journal of Science & Engineering*, 4(2).

Raza, M., Adnan, A., Sharif, M., & Haider, S.W. (2012). Lossless compression method for medical image sequences using super-spatial structure prediction and inter-frame coding. *Journal of Applied Research and Technology*, 618 - 628.

Nelson, M. (1991). The Data Compression Book. s.l.: IDG Books Worldwide, Inc.

Ritter, F. (2011). Medical Image Analysis. IEEE Pulse, pp. 60 - 70.

Roy, S., Mitra, A., & Setua, S. K. (2015). Color & greyscale image representation using multivector. Hooghly, IEEE, pp. 1 - 6.

Ruan, X., & Katti, R. (2006). Reducing the Length of Shannon-Fano-Elias Codes and Shannon-Fano Codes. Washington, DC, IEEE, pp. 1 – 7.

Saboori, A., & Abolfazl Hosseini, S. (2015). Color image watermaking in YUV color space based on combination of DCT and PCA. Tehran, IEEE, pp. 308 – 313.

Burak, S., Carlo, G., Bernd, T., & Beaulieu, G.C. (2008). Region of interest based medical image compression. CiteSeerx.

Senturk, A., Senturk, Z. K., & Kara, R. (2015). Comparison of real time image transfer in wireless multimedia sensor networks. Bursa, IEEE, pp. 1226 - 1228.

Tang, J. (2010). A color image segmentation algorithm based on region growing. Chengdu, IEEE, pp. V6-634 - V6-637.

Telagarapu, P., Naveen, V. J., Lakshm, A., & Santhi, G. V. (2011). Image compression using DCT and wavelet transformations. *International Journal of Signal Processing, Image Processing and Pattern Reconfiguration*, 61 - 74.

Zhongshui, Q.U., & Jianwei, W. (2010). A color YUV image edge detection method based on histogram equalization transformation. Yantai, Shandong, IEEE, pp. 3546 - 3549.