International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Implementation of Run Length Encoding Using Verilog HDL

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

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

Abstract: ***Run Length Encoding (RLE) compression algorithms is one of the lossless data compression algorithms. RLE is considered an easy and simple method to reduce the original data bits into a lesser number of bits. This paper proposes a modified architecture and implementation of RLE algorithm. The modification in the architecture was by applying 3-bit instead of 8-bit register as a counter to the repletion of identical consecutive data elements. The implementation of this algorithm is based on FPGA by using Verilog HDL. The proposed architectures prepared in Verilog hardware description language (HDL). The modules of Verilog HDL were simulated and synthesized using Xilinx ISE 14.7. The result showed that the compression ratio was 1.282 by using counter of 3-bit comparing to 1.0037 when the counter was of size of 8-bit .***

Keywords: RLE, Lossless Compression, Verilog HDL, FPGA

1. Introduction

Data Compression classifies into two ways lossless compression and lossy compression. Lossless compression is a kind of data compression algorithm that permits the information to be totally compressed and decompressed without any loss of the original information. The first data and the data that outcome from compression and decompression are totally the equivalent on the grounds that no part of the information is lost in the process [1]. Lossy data compression is the class of data encoding strategies that utilize estimated approximations and partial data discarding to represent the content. These methods are utilized to lessen data size for storage, dealing with, and transmitting content. Lossy compression has a higher compression ratio than lossless compression so lossy methods are most often used for compressing images, videos and sound. While lossless methods are most often used for compressing text and data files. So, the compression technique is one of the most important parts to increase system performance. To perform lossless data compression run length encoding RLE [2] is well prospered.

2. Related Work

In this section, some of RLE architectures and implementations by using FPGA are reviewed. In [3], S. Sarika, et. al, improved a method for data compression by utilizing bit stuffing to break bigger groups and reduce the number of bits to represent the run value and reduce the memory stack and thus speeds up. With a little modify in the compressor architecture, the compression ratio significantly affected. The author in [4], used lossy data compression to purpose modified run length encoding using Verilog HDL. The author proposed that if the next data is equal or greater by 1 or less by 1 than the first data it's considered in run and counter will be increased. By using the architecture that mentioned, the work in [4] achieved a high compression ratio but suffering from loss data after compression and decompression. In [5], the authors used RLE in certain way that the identical data (runs of data) are stored as a data value 8 bit and count value 16 bit that mean a maximum value of count equal to 2^{16}. That method improved the compression ratio when the input sequence has very large identical data.

3. RLE

RLE is a very simple and easy method of lossless data compression, in which the sequence of equal data is stored as a single data value and a single count. This is useful for data that contains great identical data [6]. For example, if the input data is [35,35,35,35,40,40,72,72] then the output data sequence will be [(35,4), (40,2), (72,2)]. The Compression Ratio (CR) is denoted by the mathematical formula [7] : CR $= \frac{\text{Uncompressed size}}{\text{compressed size}}$

So, the compression ratio of this example will be:

$$CR = \frac{\text{number of uncompressed data} * \text{number of bits}}{\text{number of compressed data} * \text{number of bits}}$$

$$= \frac{8*8}{6*8} = \frac{64}{48} = 1.33$$

4. Architecture of the proposed Modified RLE

The architecture of modified RLE is presented in the next subsection. Firstly, the architecture of compression phase will be described with the flowchart that clears the sequence of events followed to achieve the purpose. Secondly, the architecture module of decompression of modified RLE is described with the flowchart.

Compression and Decompression

Generally, the main goal of using RLE is to decrease the memory used for data storage purposed and to improve the compression ratio. The proposed RLE compression architecture uses four main registers for certain function for each one and with different number of bits. Table 1 shows the mentioned registers with description for their functions.

Figure1 shows the block diagram of the compression of the Modified RLE. In general, there are set of registers with different sizes where used in the architecture module proposed in this work, these registers with their description were listed in Table1.

Initially, the stream of input elements (or characters) with size of 8-bit for each is delivered in a register *I-temp* according to a specified clock. If the next input element is equal to *I-temp* then *I-counter* register will be incremented by 1. if not, then *I-temp* will copy out to register *ouput*, 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 data stream that is indicated by *data-read*. The indicator *last* is used to define the desirable output where it's set to 1 with each desirable output.

The modified RLE uses register *counter* with 3 bits only to store the number of repetition, 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 *counter* with larger than 3 bits will reduces the compression ratio of the modified RLE.

The compression ratio of the modified RLE can be studied through applying the same example presented in section 3 above. The elements of input data [35,35,35,35,40,40,72,72] contains 8 elements and the size of each element is 8 bits, then the output data sequence will be [(35,4), (40,2), (72,2)] according to the proposed RLE. The first number in the sub parentheses represents the element itself while the second one is the count of repetitions for that element. In fact, the result is the same of that example shown above but the difference is in the number of bits required to represent the count numbers of the resulted data stream. Here we just need 3 bits instead of 8 bits that was needed to represent the count and the compression ratio will be:

$$CR = \frac{\text{number of uncompressed data} * \text{number of bits}}{\text{number of compressed data} * \text{number of bits}}$$

$$= = \frac{8*8}{3*8+3*3} = \frac{64}{33} = 1.93$$

comparing to 1.33 that represent CR when the number of bits was equal to 8 for representing the repetition of certain element. Figure 2. shows the flow chart of the compression algorithm considered by the proposed architecture of modified RLE. Similarly, Figure 3. Demonstrates the block diagram of the decompression of the Modified RLE that uses the same registers used in the compression phase but with opposite order. In the same way, Figure 4. illustrates the decompression algorithm for the architecture of modified RLE. Really, the way that the decompression takes on is the reverse of that used by compression side and is detailed in Figure 3.

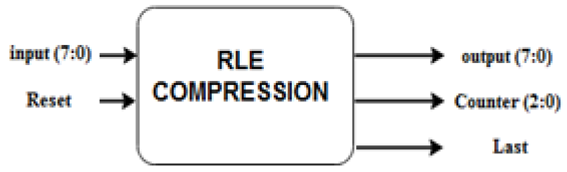


Figure 1:Block diagram of modified RLE compression

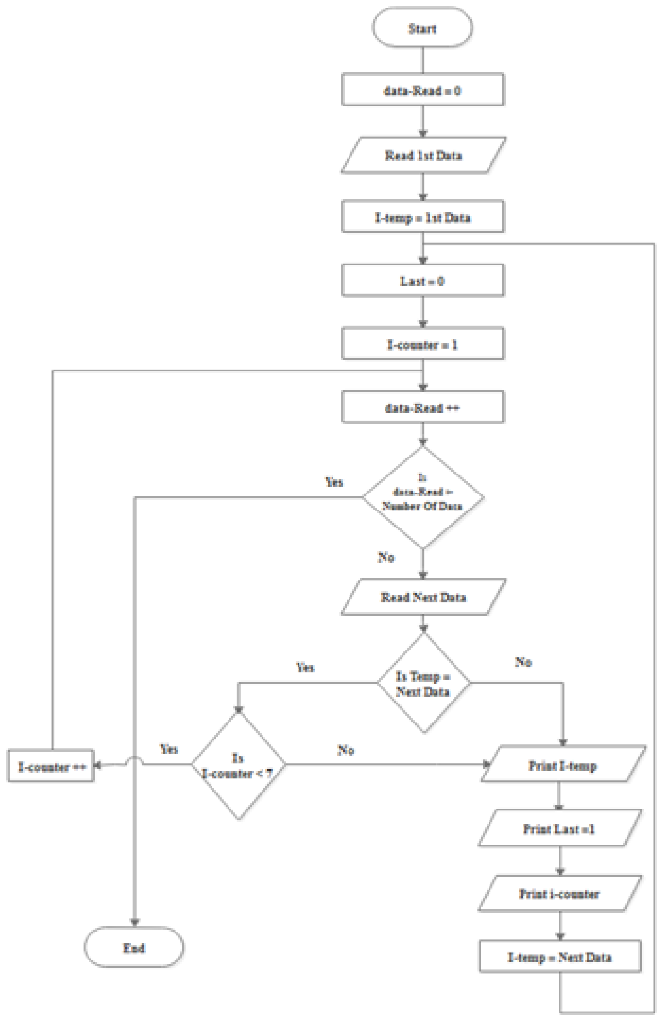


Figure 2: Flowchart of the compression.

Table 1: Description of registers used in modified RLE

Register	Description	# of bits
I-temp	Intermediate register, and stores the current element that is compared to the next element	8
I-counter	Intermediate register, stores the intermediate count of identical consecutive elements	3
data-Read	Stores the count of the total number of elements read	16
last	Indicator used to define the Desirable output. Where it's set to 1 with each Desirable output	1
counter	Stores the final count of identical elements	3
output	Stores the value of identical elements	8
input	Stores the value of the each element before entering the system	8

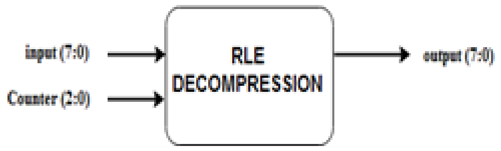


Figure 3: The Block Diagram of Modified RLE Decompression

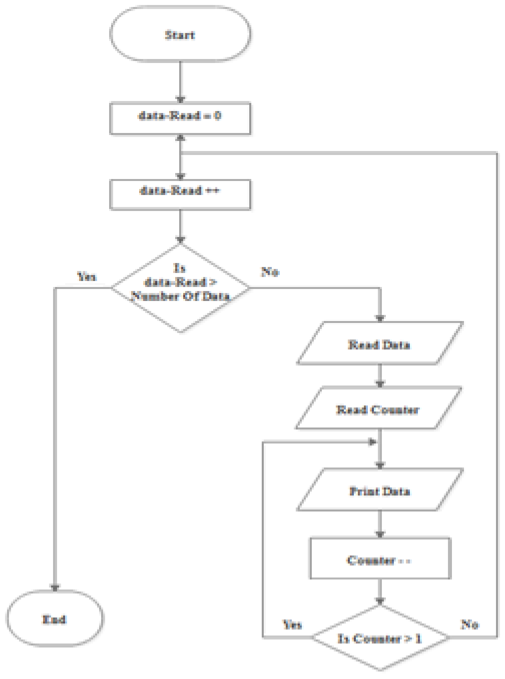
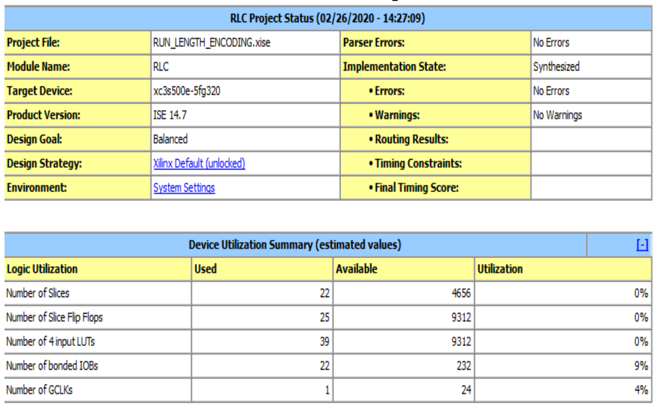


Figure 4: Flow chart of decompression.

5. Implementation and results

Modified RLE compression and decompression architectures were designed using Verilog HDL. The designed modules are simulated and synthesized using Xilinx ISE 14.7. In this technique, a finite-state machine (FSM) is used. The maximum operating frequency of compression RLE used is 171.737MHz. This design is implemented in XA Spartan®-3E FPGA platform. XPS synthesis report of implementation of modified RLE compression is presented in Table 2. Also, Figure 5 shows the output signals of the Modified RLE compression module.

Table 2: XPS Synthesis Report of the implementation of Modified RLE Compression



RLC Project Status (02/26/2020 - 14:27:09)			
Project File:	RUN_LENGTH_ENCODING.xise	**Parser Errors:**	No Errors
Module Name:	RLC	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	No Warnings
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values) [-]			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	22	4656	0%
Number of Slice Flip Flops	25	9312	0%
Number of 4 input LUTs	39	9312	0%
Number of bonded IOBs	22	232	9%
Number of GCLKs	1	24	4%

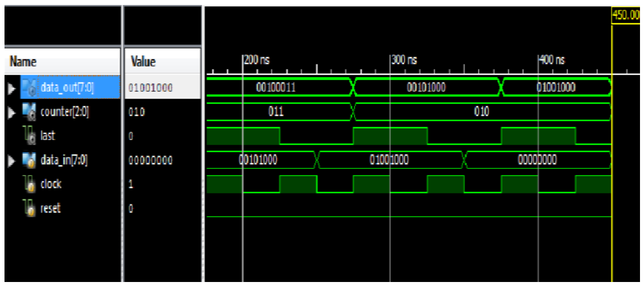


Figure 5: Output Signal Of RLE Compression Module

Figure 6 shows the Register-transfer level of modified RLE compression architecture.

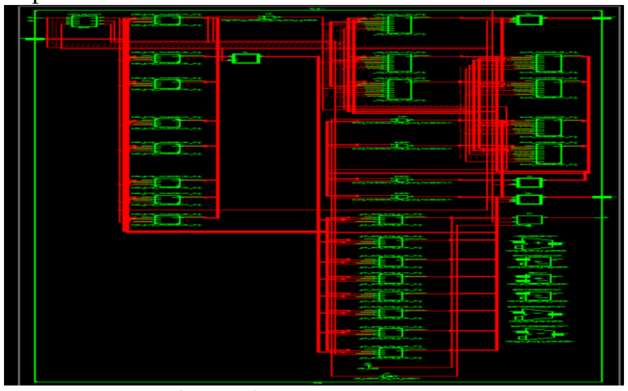
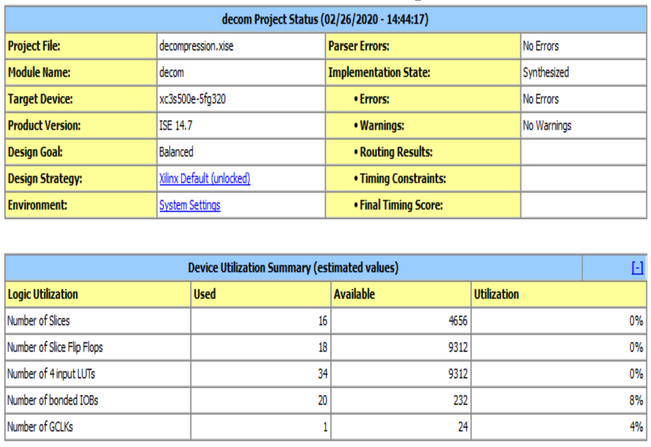


Figure 6: RTL of Modified RLE Compression Architecture

The maximum operating frequency of decompression RLE applied is 308.356MHz. This design is implemented in XA Spartan®-3E FPGA platform. XPS synthesis report of implementation of modified RLE decompressionis presented in Table 3. Figure 7 shows the output signal of run-length encoding decompression module.

Table 3: XPS Synthesis Report Of The implementation Of Modified RLE Decompression



decom Project Status (02/26/2020 - 14:44:17)			
Project File:	decompression.xise	**Parser Errors:**	No Errors
Module Name:	decom	**Implementation State:**	Synthesized
Target Device:	xc3s500e-5fg320	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	No Warnings
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values) [-]			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	16	4656	0%
Number of Slice Flip Flops	18	9312	0%
Number of 4 input LUTs	34	9312	0%
Number of bonded IOBs	20	232	8%
Number of GCLKs	1	24	4%

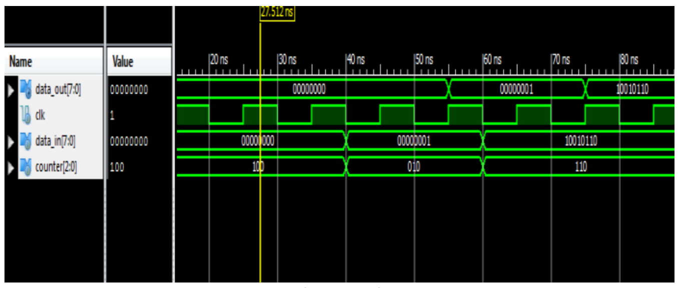


Figure 7: Output Signal Of Modified RLE Decompression Module

Figure 8 shows the Register-transfer level of run-length encoding decompression architecture.

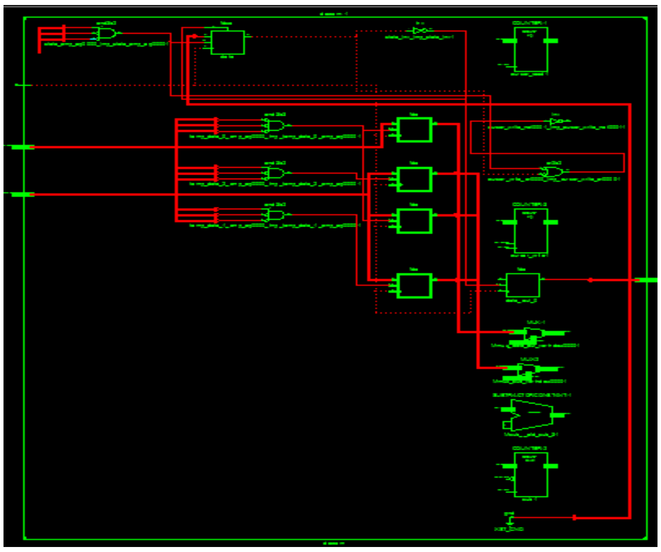


Figure 8: RTL OF Modified RLE Decompression Architecture

The proposed architecture depending on the modified RLE was applied to the Akiyo News image size of 480 * 640 * 3 that is shown in figure 9 and the compression ratio was obtained with respect to different sizes of register *counter*. The results were presented in Table 4. From Table 4 its can be noted that the highest value obtained of the compression ratio is 1.2878 when the counter size is a 3 bit .

Figure 9: Akiyo News Image

Table 4: Compression ratio for different sizes of register *counter*

The number of bits of *counter*	Compression Ratio
8	1.0037
7	1.0664
6	1.2112
5	1.1305
4	1.2722
3	1.2878
2	1.2062

6. Conclusion

In this paper, modified RLE is proposed to design more efficient compression and decompression by using Verilog HDL. The designed modules are simulated and synthesized using Xilinx ISE 14.7. The given data sequence is encoded by using 8 bits for data value and 3 bits for counter value to improve the compression ratio, thus, it improved the performance of the system.

References

[1] S. Joseph, N. Srikanth, J. E. N. Abhilash, “A Novel Approach of Modified Run-Length Encoding Scheme for High Speed Data Communication Application,” International journal of Science and Research, ISSN: 2319-7064, Vol. 2, Issue 12, December.

[2] Held, Gilbert,“Data Compression: Techniques andApplications, Hardware and Software Considerations”,second edition, John Wiley & Sons, New York, NY, 1987.

[3] S. Sarika and S. Srilali, “Improved Run Length Encoding Scheme For Efficient Compression Data Rate,”, S. Sarika et al Int. Journal of Engineering Research and Applications ISSN : 2248-9622, Vol. 3, Issue 6, Nov-Dec 2013, pp.2017-2020 .

[4] Swetha Annangi, , “Rtl Design Of Efficient Modified Run Length Encoding Architectures Using Verilog HDL,” Volume 8, Issue 1, January - February 2017, pp. 52–57, Article ID: IJECET_08_01_006.

[5] Zaid Haitham and Maher K. Mahmood Al-Azawi, , “Video Compression Based on Motion Compensation and Contourlet Transform,” 2018 Third Scientific Conference of Electrical Engineering (SCEE), University of Technology – Iraq .

[6] Akhtarl M.B., Qureshi A.M. , and Islam Q., “OPTIMIZED RUN LENGTH CODING FOR JPGE IMAGE COMPRESSION USED IN SPEC RESEARCH PROGRAM OF IST.(978-1-61284-941-6/11/$26.00 ©2011 IEEE) .

[7] 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.

[8] Xilinx “XA Spartan-3E FPGA Family”, [Online] Available: http://www.xilinx.com/products/silicon devices/fpga/xaspartan-3e/ .