Military Technical College
Kobry El-Kobba,
Cairo, Egypt

11-th International Conference
on Aerospace Sciences &
Aviation Technology

AN IMPLEMENTATION OF THE RUN-LENGTH ENCODE ALGORITHM USING FPGA

Gouda I. Salama*, Fawzy ELtohamy Hassan*, M. Sharrawy**, and Ramy M. Bahy*.

ABSTRACT

This paper presents a real time implementation of Run-Length Encode (RLE) using FPGA as one of image compression algorithms. The RLE algorithm can be implemented either on commercial DSP or as an ASIC but due to the huge development in the FPGA field, it is recommended to use the FPGA technology. The design steps from design entry to files which are needed for the download process are developed.

KEYWORD: Run-Length Encode, FPGA, Image compression

1- INTRODUCTION

Image compression process is the reduction of the number of bits required to represent a digital image [1], Run-Length Encode (RLE) is considered one of a simple image compression algorithms which depends on the reduction of the repeated image gray levels values into only two values, the first value represents the gray level value and the second one represents the repetition of this gray level value [2]. The speed is one of the main factors in evaluation of a compression algorithm. In some applications such as On-Board satellite image compression, the speed of the compression algorithm is mandatory due to real time nature of the satellite missions (simultaneous imaging and downlinking). Therefore, hardware-based solutions are considered [3-4-5]. The design steps will be accomplished by using two well known packages:**FPGA advantage for HDL design, release 5.2** and **Xilinx ISE, release 5.2i**.

In this paper, all design steps which includes design entry, functional simulation, synthesis and timing simulation [6] are introduced. The RLE design description is introduced in the next section, the simulation results are explained in section 3 and finally conclusion is presented in the last section.

* Egyptian Armed Forces
** Faculty of Computers and Information, Helwan University, Egypt

2- RLE DESIGN DESCRIPTION

Fig.1. illustrates the main inputs and outputs to the proposed design. The image data is entered to the design line by line or precisely pixel by pixel then the RLE processing is performed on these line pixels. The *DATA_in* refers to the **8**-bit input image data, where, every pixel is stored in one byte (gray scale image). The *Data_out* is the output data in an **8**-Bit format. The *Rep-out* refers to the repetition of the output data. Both of *Data_out* and *Rep_*out is called the output compressed image data. The function of these signals is given in Table 1.

Table 1. Main Block inputs/outputs

Port	Function
DATA_in	Enters the pixels value into the RLE processing circuit (*processing*)
Ena	**Enable** of the circuit (Ena='1' the circuit is 'ON' and ready for processing steps, Ena='0' the cicuit is 'OFF' and no processing sequence occurs).
Rst	**Reset,** when rst='1' the circuit is in reset state and no processing occurs, when rst='0' the circuit is ready for processing procedures.
Clk	**Clock,** in every clock rising edge a new pixel value is entered.
Data_out	The output data (pixels value or gray levels values).
Rep_out	The repetition of the output data.

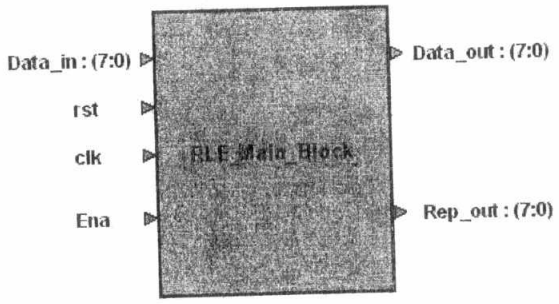

Fig.1. The Main Block of the RLE Algorithm

In Fig. 2, the details of the RLE block are shown. It comprises:
1- The RLE processing circuit.
2- The Data output RAMs (RAM 1 and RAM 2)
3- The Inverter.

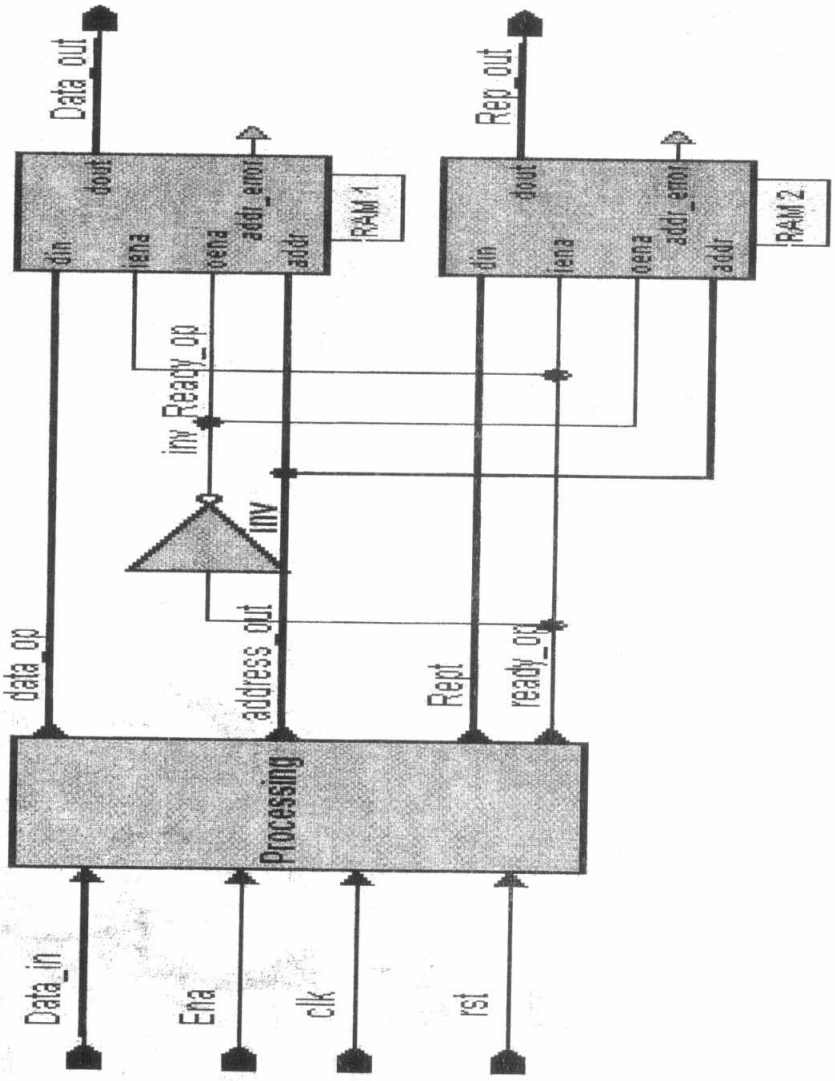

Fig.2. The Block diagram of the RLE design

The **RLE processing circuit** (*Processing*) shown in Fig. 2, is the main core of the design. The processing circuit becomes in the '*ON*' state when the *Ena* becomes '**1**' which means that the data (*Data_in*) is ready for RLE processing. At every clock rising edge (*clk*), a new input data is entered to the RLE processing circuit and counted. The RLE processing design procedures are written in VHDL code. In the design, the input enable signal '*iena*' of the two output memories (RAM 1 and RAM 2) are connected to the ready output signal (*Ready*_OP) of the RLE processing circuit. The *Ready*_OP signal equals '**0**' as long as there is no change in the values of the input data (*Data_in*) which means that there is no input values can be written or stored into the two memories. The '*oena*' signal of the two output memories is the

inverse of the *iena* signal, where, the *Ready_OP* of the RLE processing circuit is connected with the output enable '*oena*' signal of the two output memories via an inverter (The inverter input is *Ready_op* signal and its output is *Ready_op_inv* signal) as illustrated in Fig. 2. Once, there exist a change in the input data values (pixels values) the previous operation is reversed i.e. the *Ready_OP* will be equal '**1**' and sequentially the input enable '*iena*' signals of RAM 1 and RAM 2 will be equal '**1**', this means that the two memories (RAM 1 and RAM 2) are in writing state and are ready to store the output values from the RLE processing circuit. The *Ready_OP* equals '**0**' only when the whole input data (*Data_in*) is completely arrived to the RLE processing circuit (*processing*). When *Ready_op* becomes '**0**', the *oen* of the two memories (RAM 1 and RAM 2) becomes '**1**' and the two memories become in reading state and ready to exit the compressed data (*Data_out* and *Rep_out*).

Data output RAMs(RAM 1 and RAM 2):the outputs pixel values are stored in RAM 1 (*Data_op*) and their number of repetitions are stored in RAM 2 (*Rept*). RAM 1 and RAM 2 are in writing state when *ien* equals '1' and *oen* equals '0' and vice versa in reading state. In the design the memories size are assumed to be 256 bytes.

Inverter (*INV*) inverts the state of RAM1 and RAM 2 from witting state to reading state and vice versa.

3- SIMULATION RESULTS

The functional simulation is done by using **ModelSim 5.5e**. The simulation example is performed on 256 pixels values (can be considered as the first scanned image line). In the presence of the following values:

150 150 150 150 150 150 150 150 150 100 100 100 100 227

The output data with value 150 is repeated eight times and the output data with value 100 is repeated three times this means that the output of the previous input data sequence after applying the RLE algorithm becomes **150 8 100 3**. These outputs can be stored in only 4 bytes instead of 13 bytes (i.e. Compression Ratio (CR) =2.6:1). The functional simulation results are shown in the two wave forms of Fig.3.

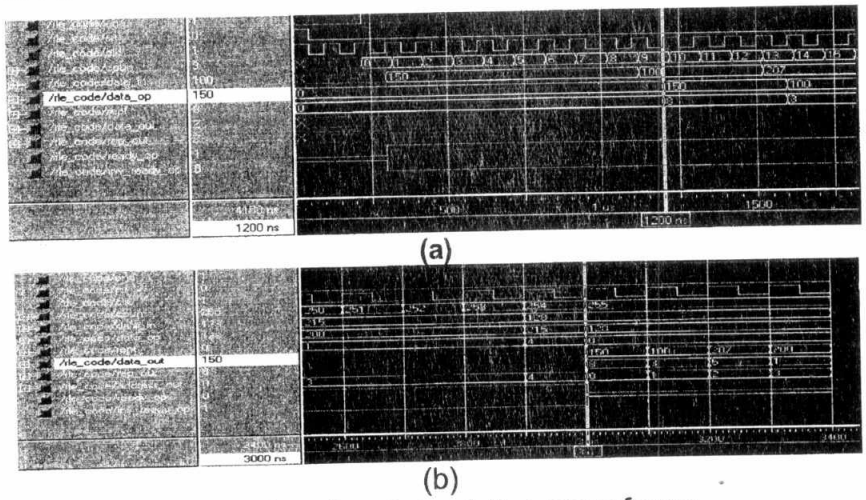

Fig.3. Functional simulation wave forms

(a) Sample wave form indicates the output from the RLE processing circuit
(b) Sample wave form illustrates the output from the RAM 1 and RAM 2

The simulation results shows that *Ready_op* signal equals '1' and *inv_ready_op* signal equals '0' as long as the last input data is not entered to the RLE processing circuit i.e., *coun*<255 (*coun* is an internal signal that counts the input data). It worth notifying that there are no delays in output values since the simulation is a functional simulation.

The design *synthesis* is done using **Leonardo Spectrum**. *Xilinx-SpartanII-xc2s100pq208* is selected as FPGA chip with clock frequency 20 MHZ (sampling rate) [7] to download the proposed design. Form the synthesis step, we obtain the area report which shows that **97 CLBs** are used in the design, and the time report which indicates that the output data arrival occurred after **16 n sec** (this means that the time taken to transfer from one input pixel to the next following one is **16 n sec** (ideal simulation). The design is then converted into its gates level to be ready for timing simulation (real simulation).The *place and route* step is done using **Xilinx ISE5.2i** in order to make the design suitable for the timing simulation where time delays between different gates are taken into consideration. After timing simulation of the previous functionally simulated data, it is noticed that the output data from RAM 1 and RAM 2 (*Data_out* and *Rep_out*) is not exiting exactly with the clock (*clk*) rising edge as illustrated in Fig.4. but delayed by **12 n sec** this means that the real output data arrival time occurred after **28 n sec** (not **16 n sec** as in functional simulation).

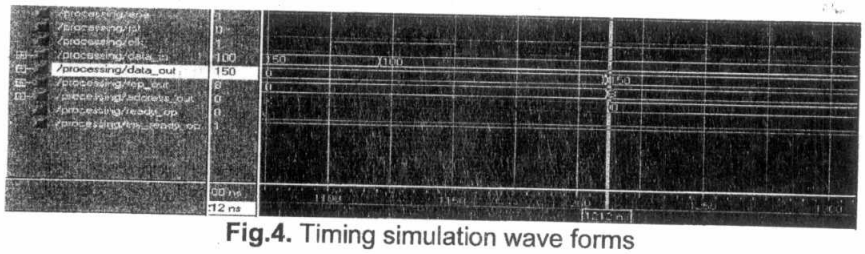

Fig.4. Timing simulation wave forms

After finishing synthesis and timing simulation steps, an EDIF (Electronic Digital Interchange Format) file is obtained; this file is used for downloading the design on the selected chip.

4- CONCLUSION

From the area report and time report obtained from the synthesis step, the hardware utilization of the selected FPGA chip is given in Table 2 and the real data arrival time is given in Table 3.

Table 2. FPGA area report for RLE design

	Used	Available	Utilization
CLBs	97	1200	8.08%
IOs	35	140	25.00%
Function Generators	194	2400	8.08%
Dffs or Latches	65	3120	2.08%

Table 3. FPGA time report after timing simulation for RLE design

Data required Time (n sec.)	50
Data arrival time (n sec.)	28
Slack (n sec.)	22

Due to the large memory requirements, large FPGA must be used. This means that the logic utilization will be low. From the time report , it is clear that the RLE design using FPGA is suitable for real time applications where, the processing time (**28 n sec**) is less than the sampling time (1/sampling rate = $\frac{1}{20MHZ}$ = **50 n sec**)

5- REFERENCES

[1] Rafel C.Gonzalez , Richard E.Woods, "*Digital image processing*", Second edition, university of Tennesse, MedData Interactive, (2002).

[2] Guy E.Blelloch ,"*Introduction to Data compression* ", computer science department, Carnegie Mellon University, October 16,(2001).

[3] R M Susilo, T R Bretschneider, *"On Real time Satellite Image Compression of X-sat"*, School of computer engineering, Nanyang Technological University, (2003)

[4] Tom Kaminski, "A Hardware Implementation of Arithmetic Compression", Final Report for 24.433 Thesis Project for the faculty of Computer Engineering at the University of Manitoba",March, (2001).

[5] Scott Hauk and William D.Wilson "Run-length Compression Techniques for FPGA Configurations", Department of Electrical and Computer Engineering, Northwestern University, (1999).

[6] Desigining with FPGA Advantage, Mentor Grahic, student workbook, software V5.2, January (2002).

[7] Memec Spartan II "LC Users Guide V1.0", July 21, (2003).

COMPUTER APPLICATIONS

CONTENTS