2016 International Siberian Conference on Control and Communications (SIBCON)

Lossless Compression Algorithm For Use In Telecommunication Systems

Valery A. Kokovin
State University "Dubna", branch "Protvino"
Protvino, Moscow reg., Russia
kokovin@uni-protvino.ru

Saygid U. Uvaysov
National Research University Higher School of Economics
Moscow, Russia
s.uvaysov@hse.ru

Svetlana S. Uvaysova
National Research University Higher School of Economics
Moscow, Russia
uvay@hse.ru

***Abstract*— The paper describes basic methods of data compression without loss and analyzes their advantages and disadvantages. There is a hardware implementation on FPGA of compression algorithm for stream processing of information. This algorithm can be used in applications related to telecommunications networks of distributed control systems.**

Keywords—lossles compression algorithm; FPGA; telecommunication system.

I. INTRODUCTION

Designing distributed control systems for automation of technological processes related to the development of telecommunication systems (TCS). Velocity of data transfer between the individual subsystems and their volumes in TCS determine the velocity of reaction of the entire control system. Different compression methods are used to improve performance of TCS. Methods of lossless compression are used if data to generate the control actions of the control object are transmitted, for example, information from sensors or event. If informative data is transmitted, such as image or audio, to the operator during compression process, then losses are acceptable.

The paper deals with a simple but effective method for lossless data compression, which is realized in hardware and is based on a Field-Programmable Gate Array (FPGA). The project was a part of the task of timer subsystem diagnostics of the linear accelerator LU-30 by IHEP accelerator complex [1]. Each subsystem has certain timing means, which are necessary to provide real-time control and synchronization of technological processes in the accelerator. The timer subsystem of the linear accelerator is a part of the global timer system [1]. The algorithm has been successfully used in a control system by distributed technological line. This distributed technological line is a teaching and research stand at the base of a conveyor line. [2]. Using the component base based on FPGA for management and diagnostic tasks allows scaling of encoded information, and parallelizing of data streams to improve performance and control processes for data streams treatment in FPGA in real-time [3].

Section 2 provides an overview of data compression methods without loss. Section 3 provides a statement of the problem, analyzes the initial conditions of the problem. A suggested block diagram of the encoder and details for its implementation are discussed in Section 4. The algorithm for data compression and discussion of results are presented in Section 5. Conclusion and suggestions for further work on the algorithm are given in Section 6.

II. METHODS FOR DATA COMPRESSION WITHOUT LOSS

There are many different methods for data compression without loss. The paper in [4] gives a detailed analysis of methods with and without loss. The paper outlines basic ideas and concepts of the main methods. Lossless compression methods are most interesting in distributed control systems for implementing control algorithms for transmitted control data. An important characteristic of compression is efficiency of compression. By efficiency we mean not only the degree of compression (the ratio of the size of original data and the size of compressed data), but also the speed of the compressing device called "compressor". A compressor can be designed as a software or hardware product. One of the most widely used methods for data compression without loss is RLE (Run Length Encoding). RLE- run-length coding, various options of Huffman algorithm and arithmetic coding. Let us briefly consider these methods:

- RLE (Run Length Encoding) length coding. The main idea of coding is to replace the series of repeated sequences of bytes by one or more bytes that encode these sequences by counting the number of repetitions of the original bytes.
- Huffman algorithm is widely used in various archivers. There are several varieties of the algorithm. Canonical or statistical Huffman algorithm [4] is bi-coding the input data block. On the first pass, the frequency of occurrence of identical bytes (characters) in the block is determined and character encoding is performed so that the more frequently appearing symbols are associated with a shorter length codes. On the second pass, a binary tree is constructed. The

978-1-4673-8383-7/16/$31.00 ©2016 IEEE

coding table must be placed into the output file. The disadvantage of this algorithm is that it is impossible to encode data as a stream.

- An adaptive or dynamic Huffman algorithm eliminates the discussed problems of the canonical compression algorithm. The paper in [5] gives the basic principles of the single-pass algorithm. This algorithm allows for hardware implementation of the compressor.
- Arithmetic coding refers to the class of entropy algorithms [4], so that the frequency distribution of the incoming characters must be known prior to coding. There are variations of this method of data compression with adaptive algorithm [6].

A comparative analysis of the discussed lossless compression algorithms is given in [7]. A RLE algorithm is rather simple. It is convenient for hardware implementation; however, in the worst case, the output may have a size greater than the input data size in half. Judging by compression ratio, the arithmetic algorithm is strictly preferred, but it has lower speed of compression and decompression.

All of the above algorithms are universal compression algorithms. The universality of an algorithm can be viewed as an advantage. However, emergence of a large number of methods and algorithms [4] suggests that most effective compression algorithms must be tailored for a particular task. By efficiency, we mean not only the compression degree but also the speed of the compressor. So on the market there are more devices that perform data compression on a hardware platform, usually on FPGA. For example, Huffman and RLE are used in hardware compressors [8.9] for image compression in the JPEG standard [10]. These compressors are implemented using IP (Intellectual Property)-cores based on FPGA belonging to companies Altera [11] and Xilinx [12]. By buying an IP-core, a designer can adapt it to characteristics of the task, thereby enhancing the performance of their system. Moreover, it is possible to change the data compression algorithm "on the fly" by changing configuration of FPGA thus adapting to changing conditions of the system operation.

III. Problem Statement

A timer Diagnostic Tool (TDT) was developed as a part of the diagnosis of technological timer (TT) linac URAL30 accelerator complex IHEP [13]. TDT controls input and output signals of the timer for operational decision-making of the accelerator operator. TT challenge is to synchronize the basic processes in a linear accelerator. TT is synchronized with the frequency of 1 MHz and generates 64 channels of sync with the timed delay, which is defined by the operator. Starting TT is given by outer 32 pulse T1..T32 (Fig.1 upper oscillogram), which forms the structure of the super cycle accelerator [1]. The period of this synchro series T_s is 60 ms.

On Fig.1, the lower oscillograms show generation of output channel pulses to the channels K2, K4, K13 and K14 with predetermined clock delays *tkn*, where *n* is the number of the channel.

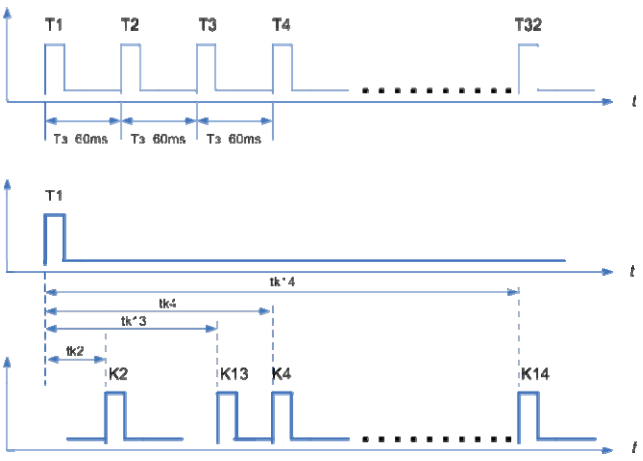

Fig.1 Input and output synchronization impulses TT

In each cycle, only one pulse for each channel is generated (or not formed) at a specified time during the normal operation of the timer. Each signal from the TT output is fixed in the TDT input register, and at that moment, time from the next start of *tkn* (e.g. T1) to the channel impulse K1..K64 is stored in a special memory. Discretization of time measurement for each channel (e.g., *tk4*) is 0.125us. The maximum length, *tk*, is limited by the time between neighboring trigger pulses and is 60000 us, and discrete of the timer delay, defined by the operator, is 1us.

Based on the above parameters, one can estimate the amount of data entering TDT. It is necessary to fix the 64-bit channel data, 24 bit of timestamp counter (19 bits is enough, but it should be rounded up to a byte) for 2 s. The estimated volume of recorded data considering the possible false positives can be up to 100 Mb / s. The considered problem requires stream processing method with certain block size of data. As an element base with the hardware implementation of the encoder, FPGA is useful since it allows you to parallelize processing and improve the efficiency of compression. The paper describes only the compression algorithm and hardware implementation for transmission over a telecommunications network. Decoding data on the receiving side is simplified because of the possibility to parallelize input stream.

IV. Structural Diagram Of The Encoder

Fig.2 shows structure of the encoder TDT that includes a synchronization of channel's data TT with internal generator TDT, entry in the intermediate memory data buffer and time stamps, data compression and writing to the network controller. Structure of the encoder (fig.2) includes:

- Clock Pulse Generator (CPG) forming from the input signal *f_gen* (from the internal oscillator SJT) two-phase synchro series where pulses *fp1* and *fp2* are shifted to 180 degrees relative to *f_gen*.

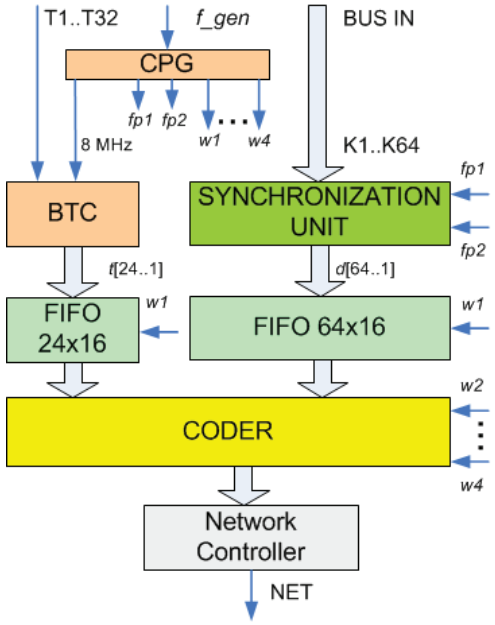

Fig.2 TDT encoder structure

- Bit Time Counter (BTC) is a timestamp counter on 24 bits (variable *t*[24..1]), which is run in each cycle by pulses T1..T32.
- Synchronization Unit performs channel timing of pulses to the internal synchro series of the encoder. The accuracy is determined by the frequency of *f_gen*. At 100 MHz oscillator frequency accuracy will be +/- 10ns.
- The intermediate memory buffer is made on FIFO (First In, First Out) and serves to align the data rate *d*[64..1] from Synchronization Unit output and data inputted to the encoder.
- CODER is an encoding device and serves to compress the data for the proposed algorithm.

The encoded data is provided to the network controller, which can be any network device having desired performance. In addition, the reporting algorithm can be implemented with additional memory.

V. Implementation Of Algorithm

In the implemented algorithm, there are elements which were discussed above when compression methods were reviewed. The data set which was received from the FIFO on the CODER, contains a 64-bit word channelized data. A logical "1" in any discharge is a recorded impulse with a time stamp of the 24-bit counter BTC, and the position of the "1" in the bit grid uniquely identifies the channel number TT. The entire 88-bit word is split into *m* bytes. In our case, $m = 11$. The work of CODER consists of two consecutive stages. The first stage of the work is to sort data with timestamp bytes for zero. The sequence of actions for implementation of the algorithm is shown on Fig.3. The start of the algorithm is associated with the arrival of the next starting pulse. After data channel synchronization and loading into FIFA, each byte is compared with the zero constant. Based on this comparison, a new word is formed of a variable length. In the second stage of the algorithm the channel frame is formed. The channel frame has two bytes for an overhead (the marker) and significant bytes for the channel signals. Markers are located at the lowest address of the channel frame. The markers are recorded after data is compared with the zero constant. The format of markers is presented in Tab.1 and Tab.2. Markers include information about significant bytes:

- If the byte of the channel signal has at least one '1', the corresponding bit is written as a marker '1' and this byte (channel signals) is stored in memory or transmitted to the network controller. There should be at least one '1' byte.
- The low byte of the time stamp is always recorded in the memory, so in Bit 0 (Tab. 1) there always will be '1'.
- Two most significant bytes of the time stamp are transmitted only if they have changed from the previous frame channel.

The next step of the algorithm is to load the generated channel frames in the network controller. Information about measured time interval between two successive start pulses, such as T1 and T2, can be added into the frame channel.

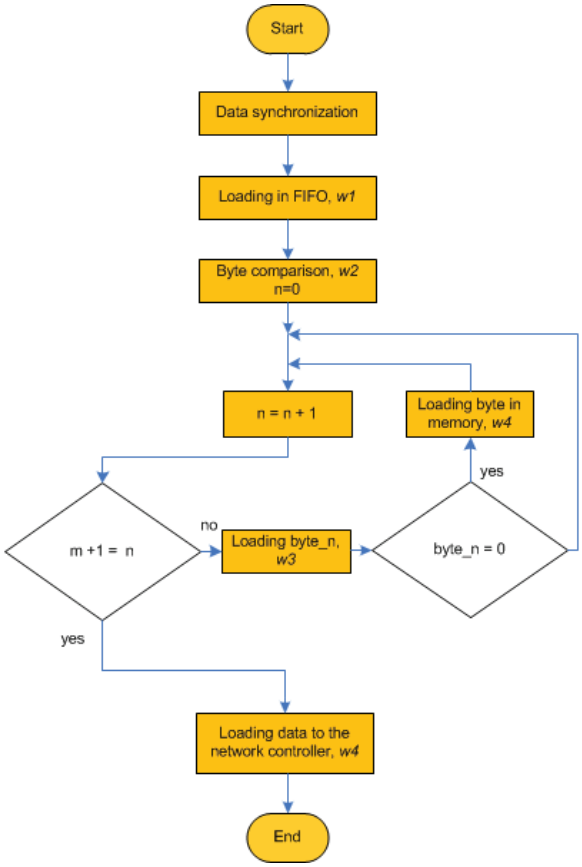

Fig.3 Algorithm of encoding work

TABLE I. MARKER 1

	Low Byte							
	Bit 7	***Bit 6***	***Bit 5***	***Bit 4***	***Bit 3***	***Bit 2***	***Bit 1***	***Bit 0***
Value	X	X	X	X	X	X	X	1
Description	5th byte data	4th byte data	3rd byte data	2nd byte data	1st byte data	3rd byte time stamp	2nd byte time stamp	1st byte time stamp

TABLE II. MARKER 2

	Most Significant Byte							
	Bit 7	***Bit 6***	***Bit 5***	***Bit 4***	***Bit 3***	***Bit 2***	***Bit 1***	***Bit 0***
Value	X	X	X	X	X	X	X	X
Description	reserve	reserve	reserve	reserve	reserve	8th byte data	7th byte data	6th byte data

VI. CONCLUSION

The trial operation of the diagnostic tools showed stable performance of SDT. The maximum estimated compression ratio by using the developed algorithm is about 20. Real compression ratio which depends on the timing of channel signals, did not exceed 10. This algorithm is scalable: the bitness of compression ratio will increase, and at the same time the processing speed will decrease. Because of high processing speed, the ease of implementation, and the resulting compression ratio, we can talk about a rather good efficiency of the algorithm. The disadvantage of this algorithm is that the encoder for the output, in the worst case, may have a size greater than the input data resolution. Further work on the algorithm may include additional data compression using a sliding window method [14].

REFERENCES

[1] V.Komarov, G.Antonichev, L.Kim, V.Kokovin, N.Krotov, V.Kuznetsov, Yu.Milichenko, N.Radomsky, V.Voevodin, “Modernization of U-70 general timing system”, Proceedings of ICALEPCS-2005, Geneva, Switzerland, October 10-14, 2005.

[2] V.Kokovin, A.Evsikov, “Streaming techniques use management automation in manufacturing plants”, Sovremennoe mashinostroenie: Nauka i obrazovanie: Materialy 4-j Mezhdunar. nauch.-prakt. konferencii. / Pod red. M.M. Radkevicha i A.N. Evgrafova. – SPb.: Izd-vo Politehn. un-ta, 2014. – pp. 1275 – 1284,(rus)

[3] V. Kokovin, S. Uvaysov, “Diagnostic port for scanning the selected objects in the electronic means on FPGA”, Kontrol'. Diagnostika, Izdat. dom "Spektr". 2015. № 12. pp. 54 – 59 DOI: 10.14489/td.2015.12.pp.054-059

[4] D.Vatolin, A.Ratushnjak, M.Smirnov, V.Jukin, “Metody szhatija dannyh. Ustrojstvo arhivatorov, szhatie izobrazhenij i video”-M.: DIALOG-MIFI, 2003.-p.384.

[5] D. E. Knuth, “Dynamic Huffman coding,” J. Algorithms, vol. 6, pp.163–180, 1985.

[6] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, and R. B. Arps, “An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder”, IBM J. Res. Dev., vol. 32, pp. 717-726, 1988.

[7] Anmol Jyot Maan "Analysis and Comparison of Algorithms for Lossless Data Compression", International Journal of Information and Computation Technology, vol. 3, numb.3, 2013, pp.139-146.

[8] http://www.cast-inc.com/ip-cores/images/jpeg-c/index.html

[9] http://www.visengi.com/products/jpeg_hardware_encoder

[10] http://jpeg.org/

[11] https://www.altera.com/

[12] http://www.xilinx.com/

[13] S. Ivanov, "Accelerator complex u70 of ihep: present status and recent upgrades", Proceedings of RuPAC-2010, Protvino, 2010.-pp. 27-31.

[14] Jacob Ziv, Abraham Lempel, A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, 23(3):337-343, May 1977.