FPGA Based Unicode Data Compression Using Static Dictionary Technique

Venkata Ratnam Anappindi
Department of ECE
National Institute of Technology
Warangal, India
venkataratnam.anappindi@gmail.com

Dr.P.Prithvi
Department of ECE
National Institute of Technology
Warangal, India
prithvi@nitw.ac.in

Abstract— **With the increasing demands for large amounts of data in the digital era for information storage, processing and transfer, the demand for smaller and faster data storage memories have also exponentially increased. In order to avoid the bottleneck between the larger data storages that require larger bandwidths for data and avoid loss of information we have to choose correct data compression techniques which reduces the redundant data storage and in turn reduces the hardware required for data storage and processing. There are two different types of data compression techniques/algorithms: (1) Lossy data compression algorithms (2) Loss less data compression algorithms. While Lossy compression algorithms are faster, they involve loss of data/information to certain level during de-compression or reconstruction. Whereas Lossless compression algorithm are relatively, they can perfectly reconstruct the complete information from the compressed data. We already have several data compression software and tools like 7-zip, WinZip etc., which compress files, audio, and video at software level. The aim of this paper is to present a hardware-based technique to perform Unicode data compression on text data using static dictionary technique where the input data stream is scanned, indexed for the positions of characters with a reference character and replace with certain symbols which are the ASCII values of the Unicode characters in the text and their differences from the static dictionary designed based on a hash table reducing the redundant data and thus reducing the data size.**

Keywords—LZMA, Unicode Encoding, FPGA, Data Compression

I. INTRODUCTION

The technique of compressing the data have been for a long time in software perspective which eases the data storage, data transmission and processing in many applications and domains like communication, digital data storage. With the adverse increase in the digital technologies in the current data driven era, there is an estimate of nearly 44 zettabytes of data generated in the world from all sorts of domains by the dawn of 2020 And this number is expected to reach 175 zettabytes by 2025. All these data are important for developing IoT based technologies to Machine Learning algorithms and Artificial Intelligence, from telemarketing based on content suggestions to generating cryptocurrency.

With such huge amounts of data being generated we cannot tolerate to have redundancy in any of such data as with proper compression algorithms and encoding techniques we can save millions of dollars spent on handling these data at datacenters and reduce the traffic over the internet which proportionally increases the speeds at which perform operations over internet.

We say the software-based compression techniques are slower because the encoding speed of software-based coder is slow compared to the arrival time of real time data i.e., the software based encoder are sequential in nature and has to wait for the arrival of the real time data before processing or encoding any other data frame. This makes software methods of data compression quite sloppy compared to the hardware based solutions and we can achieve high amounts of parallel data processing and encoding at hardware level [1].

Also, hardware-based techniques offer high level of security to the data being encoded and by having dedicated units of encoders, memory buffer for output data storage and decoders.

There are two different methods for data compression:

(1)Lossy data compression and

(2)Lossless Data compression.

We choose to preform lossless data compression-based technique on FPGA in this paper.

Lossy data compression algorithms are used where information loss is tolerable like in case of audio and video data storage and processing. Whereas for files like spreadsheets, records, word processing files, we need to use lossless data compression algorithms to compress the redundant data and reconstruct or extract exact duplicate of the original data.

Lossless data compression is generally implemented using one of two different types namely statistical modeling or dictionary-based compression.

Statistical Modeling: reads and encodes a single symbol at a time depending on the probability of character appearance.

Dictionary Based Modelling: This type of modelling uses single code to replace strings of symbols. Here coding problem is significantly reduced, leaving the model supremely important.

There are also other models called adaptive models in which data need not be scanned before generating any coding to generate statistics. Instead, the statistics are continually modified as new characters are read in and coded, but the drawback of these models is that they are completely unaware of the data. As the name indicate, these models adapt to the local conditions quickly to the data stream and begin adjusting the compression ratios only after a few thousands of bytes. A dictionary-based compression scheme reads in input data and looks for groups of symbols that appear in a dictionary. A pointer or index into the dictionary can be output instead of the code for the symbol if a string match is found. The compression ratio depends on the amount of match occurs. But dictionary-based methods introduce overhead because the dictionary needs to be transmitted along with the text.

There are various high-end data compression algorithms available from legacy Huffman encoding methods to the latest Lempel Ziv Marcov chain Algorithm (LZMA). Huffman coding is quite popular in communications domain which is a variable to fixed length coding which means variable length codes are assigned to the input characters based on the frequency of occurrence. The most frequently occurring data will be assigned the smallest code whereas the least frequently occurring data will be assigned the largest code.

The most popular 7-zip data compression software uses the LZMA algorithm in which a sliding dictionary is used to scan the incoming data and update the literals in the dictionary and replacing the data with the literals in the dictionary on the fly thus performing the compression[2 - 3].

Whatever the algorithm may be, we always must deal with the tradeoff between the compression ratio and the speed of compression which are inversely proportional to each other. If we prioritize higher compression ratio where,

Compression ratio = (total number of bits at input) / (total number of bits at output)

We must compromise the speed with which the data is compressed which in turn reduces the performance of the system

[4 - 5].

In this paper, we present the results of performing data compression on a text data using static dictionary technique and a special way of indexing the frequency and position of characters based on a reference character that is frequently repeated.

II. LITERATURE SURVEY

A. LZ77

It is commonly referred to as LZ77 and is the first compression algorithm proposed by Lempel and Ziv. It uses a dictionary which consists of list of all strings stored into a window that is captured from the previously read input stream. When new group of data is read in, the algorithm searches for a match in previously read data. If matches are found they are encoded into pointers and sent as output. The drawback if this algorithm is that it is effective only when the input data is highly repetitive or redundant.

B. LZMA

Lempel-Ziv-Markov chain-Algorithm used in 7zip, is a very popular lossless data compression algorithm which has overtaken the Huffman coding technique in terms of compression ratio and speed. This algorithm is efficient with unknown data stream. In this any sequence of source output is uniquely parsed into phrases of varying length and these phrases are encoded using code words of equal length.

C. RUN LENGTH ENCODING

In this the data is stored as a single value and count rather than the original data. For example, if the input sequence is 44, 44, 44, 44, 45, 45, 27, 27, 26, 26 then the output sequence will be (44, 4), (45, 2), (27, 2), (26, 2). From the above example, the compression ratio is better for the longer runs of data [6 -7].

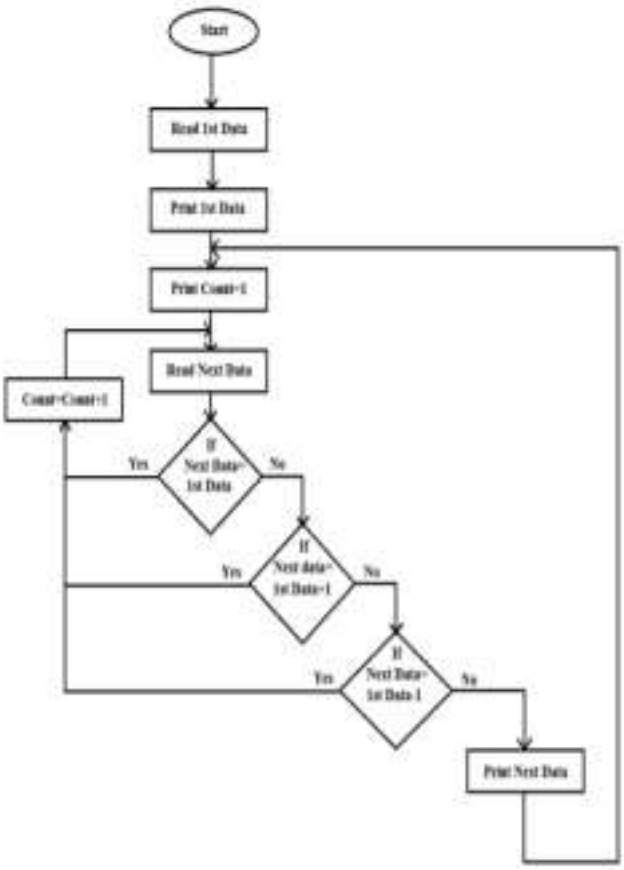

Fig.1 Run Length Encoding Flow chart

Figure.1 shows the flow chart for how the run length encoding works. This type of encoding technique is very useful in case of Bo-medical signal processing like processing, storage, and transmission if digitized ECG signals.

III. UNICODE DATA COMPRESSION

In the algorithm that we propose for text data compression, we expect the characters in input data stream to be Unicode encoded so that they can be replaced with their ASCII values from the dictionary we create [8 -9] .

After a single character instance (say for example 'space' in text) is chosen as an index, the output is encoded as below: • ASCII value of the character.

• Number of repetitions is character in the entire text. • nth position of the character after kth occurrence of the chosen index.

Where, n =1,2,3,4,5…….

k= 0,1,2,3,4,5……

This method provides us to send only the ASCII value of the character, it's frequency of occurrence and it's position with respect to the chosen index, which profoundly reduces the size of the input data stream.

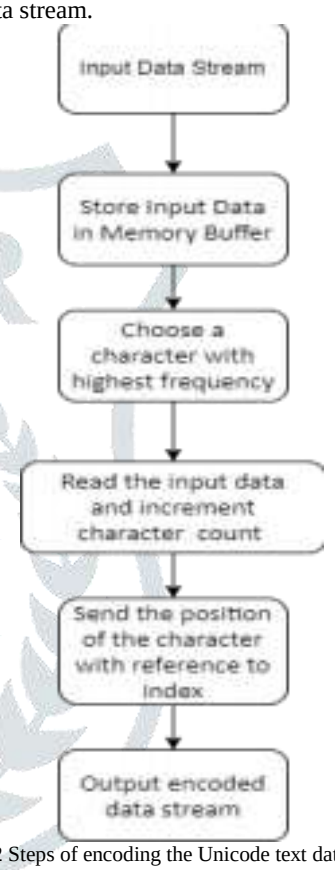

Fig.2 Steps of encoding the Unicode text data

A. THEORETICAL CALCULATION

Consider the below example:

"This paper discusses about Unicode data compression"

Consider a file in which above text is repeated 10 times. In that text file we have 570 characters each of 8 bit size which sums upto 4560 bits in total. The entire text consists of only 15 different characters including the "space" character. As per the algorithm we desire to transmit ASCII code for each character in 8 bit format considering only english alphabets are present, followed by the frequency of the character occurrence and the position of the character from the kth space.

For all the 14 characters, not including space as it is chosen as index, we expect to trasnmit them in below fromat:.

TABLE I. ENCODED SIZE FOR EACH CHARACTER IN TEXT FILE

ASCII value of the character(8 bit)	Frequency of character (8 bit)	Position of character (8 bit)	Total 24 bits for each character

We have a total of 14 characters and 15 x 16 = 360 bits. We transmit the "space" character in the similar fashion except that it is index itself. This show us that we achieve a compression ratio of 5.6 which is not very much impressive compared to today's high end algorithm but could achieve higher compression ratio with improvising the techniques used.

B. HARDWARE SETUP

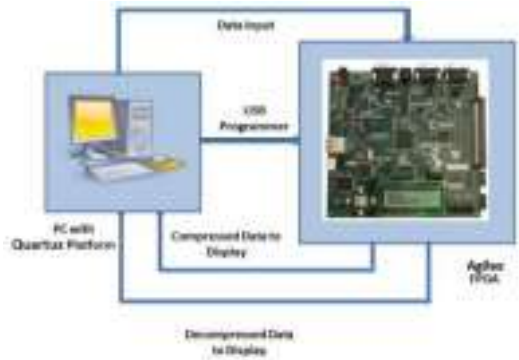

Fig.3 Experimental Hardware setup

The experimental hardware setup is shown in Figure. 3. The proposed compression algorithm is programmed on Quartus Prime platform and is tested on Agilex E-tile FPGA USB programmer. The input data is sent from the Quartus platform to the FPGA and the compressed data is sent back from FPGA to the PC.

IV. RESULTS

Our aim is to convert the text in the file into respective ASCII values and encode those ASCII values as per the proposed algorithms

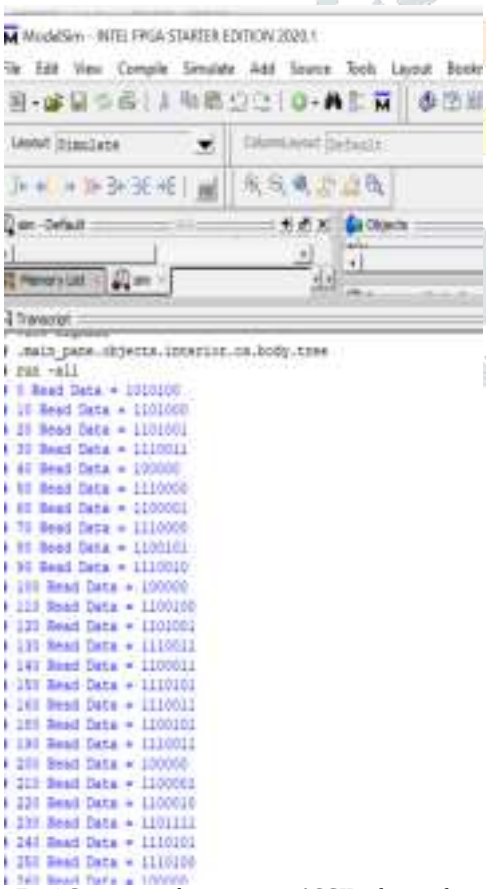

Fig:4 Converting characters into ASCII values and reading using Modelsim

The above figure shows the converted ASCII values of each character in the text file, loaded into memory buffer and read using Modelsim console.

Fig 5 Original size of the text file

Above is the original size of the file in windows machine.

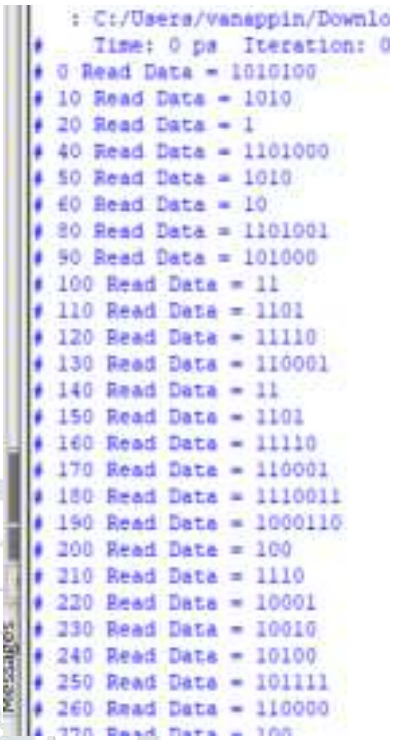

Fig. 6 Reading the encoded values using Modelsim

Figure.6 shows the values being read using Modelsim. Here in the encoding compresses the data size of original file.

Fig.7 Compressed size of file after encoding

As per the Figure. 5 and 7 we have compressed the file size from 519 bytes to 156 bytes which is 3 times compression. The ratio of compression increases exponentially as the size of original text file increases.

V. CONCLUSION

The proposed Unicode compression algorithm programmed on Agilex FPGA can achieve a compression ratio of 1.9 and can have a better performance than the LZMA algorithm because of the usage of static dictionary technique. In this paper we have presented the compression technique with improved speed with a tradeoff on compression ratio. In future, we expect to improve the compression ratio of the algorithm up to 4 with the usage of special encoding techniques.

REFERENCES

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

[2] X. Zhao and B. Li, "Implementation of the LZMA compression algorithm on FPGA," 2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC), Hsinchu, Taiwan, 2017, pp. 1-2, doi: 10.1109/EDSSC.2017.8126506.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] "Hardware Implementation of LZMA Data Compression Algorithm", *International Journal of Applied Information Systems (IJAIS)*, vol. 5, no. 4, March 2013, ISSN ISSN: 2249-0868 [4] http://mattmahoney.net/dc/dce.html#Section_523

[5] http://www.unicode.org/notes/tn31/tn31-2.html#LZ77 [6] A. Avila, R. Santoyo and S. O. Martinez, "Hardware/software implementation of the EEG signal compression module for an ambulatory monitoring subsystem," 2006 International Caribbean Conference on Devices, Circuits and Systems, Playa del Carmen, Mexico, 2006, pp. 125-129, doi:

10.1109/ICCDCS.2006.250848. [7] S. Nasehi and H. Pourghassem, "EEG Signal Compression Based on Adaptive Arithmetic Coding and First-Order Markov Model for an Ambulatory Monitoring System," 2012 Fourth International Conference on Computational Intelligence and Communication Networks, Mathura, India, 2012, pp. 313-316, doi: 10.1109/CICN.2012.103.

[8] M. T. Islam, R. R. Ema, M. R. Islam and T. Islam, "A Lossless Bit Isolation Algorithm for Data Compression by Using Static Dictionary," 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), Rajshahi, Bangladesh, 2019, pp. 1-5, doi: 10.1109/IC4ME247184.2019.9036662.

[9]) M. R. Islam, M. A. Mahmood and M. T. Islam, "A Dynamic 5 Bit Data Compression Scheme by Using Set Change Table (5BDC)," 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018, pp. 221-226, doi: 10.1109/CEEICT.2018.8628160.