HDL IMPLEMENTATION OF DATA COMPRESSION: LZW ALGORITHM

Simrandeep kaur, Academic & Consultancy Services Division (ACSD),
Centre for Development of Advanced Computing Mohali, Punjab India[1];
V.Sulochana Verma, Project Consultant, member of ACSD,
Centre for Development of Advanced Computing (C-DAC) Mohali, Punjab, India[2];

Abstract

This paper presents LZW data compression algorithm which is implemented using finite state machine technique. The proposed algorithm enhances the performance by using less number of bits than their ASCII code, utilizing content addressable memory arrays. Thus the text data can be effectively compressed using the proposed algorithm. Simulation results using Xilinx tool shows an improvement in lossless data compression scheme. In addition to this, the proposed technique results in reduced storage space by 60.25% and increased compression rate by30.3%.

Keywords - Compression rate, English and Binary text.

Introduction

Data compression is often referred to as coding, where coding is general term showing any special representation of data which satisfies a given need. Information theory is defined as the study of efficient coding. Data compression may be viewed as a branch of information theory in which the primary objective is to minimize the amount of data to be transmitted. Data compression has an important role in the area of data transmission and data storage. It plays a key role in information technology. The reduction of redundancies in data representation in order to decrease data storage requirement is defined as data compression. It used to less usage of resources such as data space or transmission capacity. Data compression is classified as lossless and lossy compression. Lossless compression is used for text and lossy compression for image.

The first data compression technique "Morse code" was invented in 1838. Morse code was used in telegraph. In 1977, Abraham Lempel and Jacob Ziv suggested the basic idea of pointer-based encoding. In 1980, Terry Welch invented LZW algorithm which became the popular technique for general-purpose compression systems. It was used in programs such as PKZIP as well as in hardware devices. Lempel-Ziv-Welch proposed a variant of LZ78 algorithms, in which compressor never outputs a character, it always outputs a code.

To do this, a major change in LZW is to preload the dictionary with all possible symbols that can occur. LZW compression replaces strings of characters with codes.

Merits of Data Compression

1. It reduces the data storage requirements.
2. It also provides rich-quality signals for audio data representation.
3. Data security can also be greatly enhanced by encrypting the decoding parameters and transmitting them separately from the compressed database files.
4. Data compression obviously reduces the cost of backup and recovery of data in computers system by storing large back-up database files in compressed form.

Data Compression Model

The block diagram of data compression model is described in figure 1.

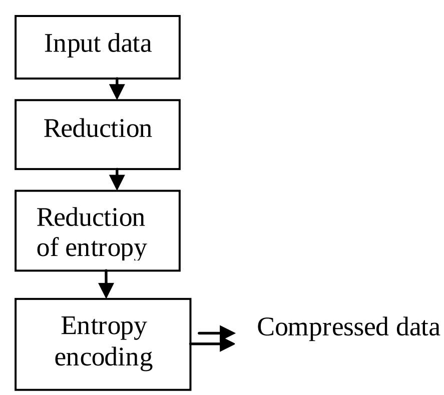

Figure 1: Data compression model

A data compression model consists of three major stages which are redundancy, reduction in entropy and entropy encoding.

Data Compression Algorithm: LZW (Lempel- Ziv Welch) Algorithm

There are many algorithms which have been used for data compression like Huffman and Lempel-Ziv-Welch (LZW), arithmetic coding. LZW algorithm is the most popular algorithm. LZW algorithm is just like a greedy approach and divides text into substrings. Like the LZW algorithm proposed in [2] LZW algorithm has both compression and decompression techniques which is explained as

LZW Compression Algorithm

LZW compression algorithm is dictionary based algorithm which always output a code for a character. Each character has a code and index number in dictionary. Input data which we want to compress is read from file. Initially data is entered in buffer for searching in dictionary to generate its code. If there is no matching character found in dictionary. Then it will be entered as new character in dictionary and assign a code. If Character is in dictionary then its code will be generate. Output codes have less number of bits than input data. This technique is useful for both graphics images and digitized voice

String j, char c;
j- get input character
while (there is still input character)
ch- transfer input string to ch .
if(ch is in dictionary)
Generate its codeword;
else
update ch and get next character to ch and
again search data in dictionary;
if(it is not present in dictionary) then
add that string to dictionary;
end if;

Compression example: consider a string "BAABAABB" is given to LZW algorithm. Figure 2 shows the steps done by LZW to generate the output code is "1211211C". In following example when input string (BAABAABBC) is given as a text to LZW compression algorithm. Initially every single character will save in buffer. When 'B'is move to buffer "parse string" then it will replace by 1.Character has its own ASCII code of 7 bit. In case of B, it has 65 as ASCII code. But in dictionary it will replace by 1.So, less number of bits will be used to represent character. Similarly, AA will move forward and generating its code which is also fewer bits than original. BAA is saved in buffer its code is generated from both AA and B's code words that is defined as 12. At last when full string has been searched in dictionary then its output will be generated as 1211211C.

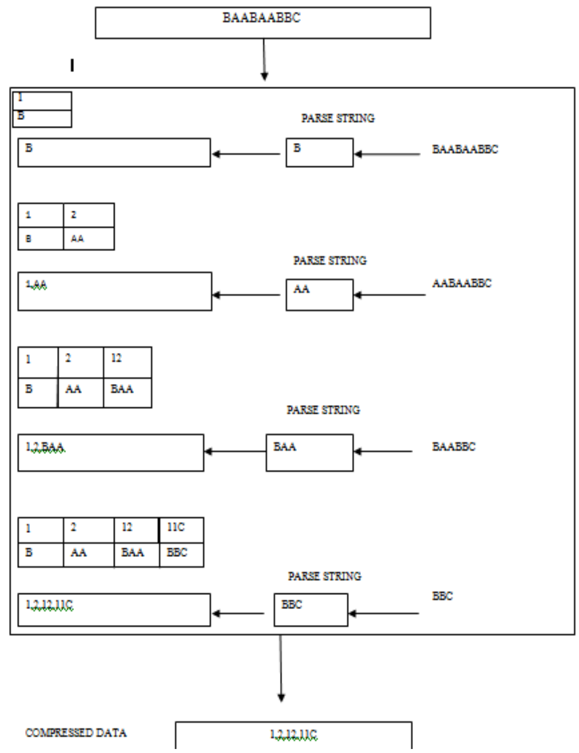

Figure 2: Example of LZW algorithm

LZW Decompression Algorithm

In LZW decompression algorithm, it needs to take the stream of code output from the compression algorithm, and use them to exactly recreate the input stream. Decompression algorithm is shown as:

ch = output code
while (there is still data to read)
code =get input character;
if (code is not in the dictionary)
entry =get translation of code;
else
entry=get translation of output code;
output entry;
ch =first character in entry
add output code + c to the dictionary
output code = code;

In decompression algorithm, code will be searched in dictionary and its character will be output.

Implementation of LZW Algorithm

Implementation of LZW Algorithm

The proposed finite state machine diagram of LZW algorithm is shown in figure 3.

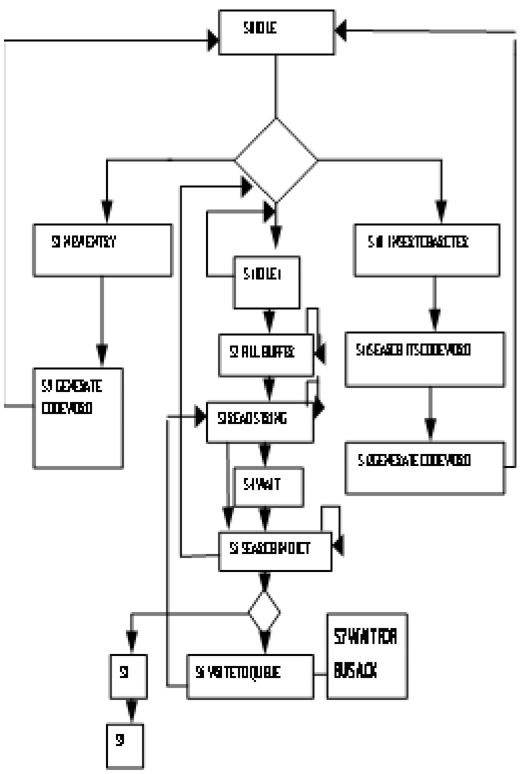

Figure 3: Finite state machine Diagram of LZW algorithm

LZW algorithm initially has idle state. New character has been added to dictionary when no longer match will found in search process. LZW algorithm is execute state S8 for performing adding operation in dictionary. Dictionary is based on content access memory technique which has both content as well as code in it.

Decompressor has reviewed same process since it is possible to have input codes for searching in dictionary to recreate its original string. Individual character's code can be also viewed in dictionary

Table 1: Specifications of FSM state for LZW Algorithm

State	Description
S0 idle	Initial state reset the system
S1 idle1	Initialization of signal
S2Fill buffer	Transfer text from file to buffer
S3Read string	Read character by character for searching
S4 wait	For waiting
S5 search in dict	For searching in dictionary by signal character
S6Write to queue	Save output to output buffer
S7 wait for ack	Wait for Bus acknowledge
S8New Entry	Adding new entry
S9Generate codeword	To generate codes
S10	Insert single character
S11search its codeword	Check in dictionary
S12 generate codeword	Display codeword
Decompression	For performing decompression

Improvement of the Dictionary Storage Method

LZW algorithm is mainly used for compressing character but not numeric. Every character has ASCII code which is of 7 bits. But in our proposed algorithm we have to replace character with 5 bit code in dictionary to improving data compression rate.

Experimental Results

LZW Compression algorithm is modelled in VHDL. The syntax of the RTL design is checked by using Xilinx tool.

Simulation Results

In the proposed work, the simulations results are done using Xilinx ISE Simulator. Simulation results show an improvement in lossless data compression scheme. In addition to this, the proposed technique results in reduced storage space by 60.25% and increased compression rate by 30.3%.

LZW Compressor Result

Figure 4 shows that input is given to LZW Compressor through text file. "Connect the input to logic one & two & three++*"string is entered to it. Input string having collection of special characters, alphabets. Whole text will transfer to buffer "data read" when data_write=1, load=1, clear=1.Rd_b=0, wr_b=1, data_write=0 and lzw_search=1 are given to start searching process to find longest match in content access memory arrays. There are two main counters which are used for searching process. "Count1" is used for searching character in dictionary. If character is present in dictionary then its code is saved in other buffer that is "de11". "Count" buffer is shifted to next value and start to point next character present in input data. All searched characters will save in "check" buffer. Once the content of check buffer is equal to content of "data read" buffer then searching process indicate to completed and their codes will save in "de11" buffer which is shown in figure5 compressed output is generated through file shown in figure 7.

Input text – connect the input to logic one & two & three.
Output text-121145146157301714251425 15

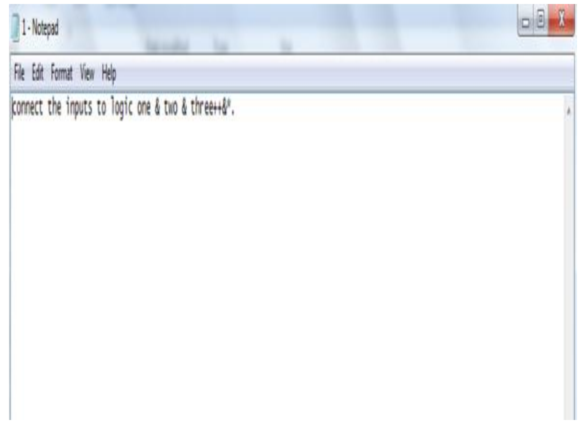

Figure 4: Enter data through file which we want to compress

Given Input text – connect the input to logic one & two & three.

Figure 5: Searching process (Searching each character from dictionary)

Simulation for LZW Compression algorithm observed on Xilinx tool. When 350 bits entered to LZW compression algorithm then it is transmitted to 119 bits and clock rate for simulation is 493 ps.

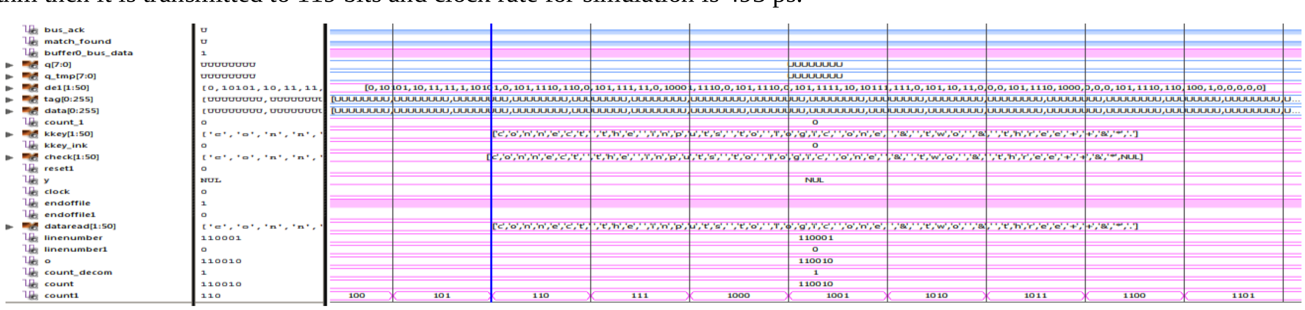

Figure 6: Complete data compression process

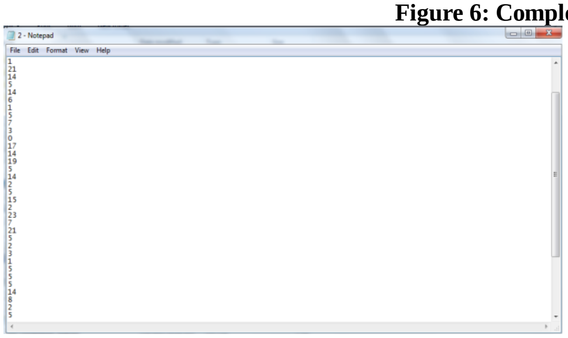

Figure 7: Compressed output generate on file

Output text-12114514615730171425142515

RTL view of LZW Compressor

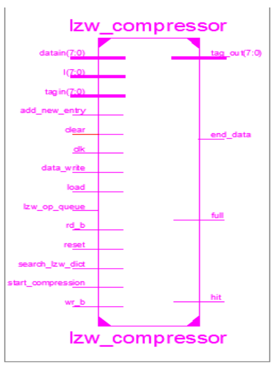

Figure-8: RTL view of LZW Compressor

This RTL view shows the signals which are used for proposed LZW data compression algorithm. Reset, clock, start_compression used for initialization of data compression. Load, data_write, wr_b, rd_b are signals used for buffer in LZW algorithm.

Search_lzw is for searching data in dictionary. The signal description of this proposed algorithm is shown in table 2.

Signal description of LZW Compressor

Table 3: Input/output signals with Remarks

Name	Description
Reset	To reset
Clock	Provide clock
Start_compression	Signal for start compression
Data_write	Signal for write data
Load	For data load in buffer
Clear	Clear buffer
Wr_b	Signal for write and read
Rd_b	Signal for write and read
Search_lzw	For searching
Add_new_entry	For adding new data
Data_in	Enter value

Verification and Synthesis

For system verification, we successfully execute proposed LZW algorithm. Test case for finite state machine is generated in VHDL. The synthesis result of LZW compression algorithm is summarized in table 4. The synthesis report shows device utilization summary.

Table 4: Device Utilization Summary

Number of Slices	3606 out of 6144 58%
Number of Slice Flip Flops	4097 out of 12288 33%
Number of 4 input LUTs	4190 out of 1288 34%
Number of IOs:	30
Number of bonded IOBs:	30 out of 240 12%
IOB Flip Flops:	1

Conclusion

In order to get better compression rate. The proposed dictionary based LZW algorithm can replace their codes with 5 bits instead of 7 bits ASCII code. The proposed LZW algorithm is evaluated by finite state machine technique. With this technique we have observed that storage space is reduced up to 60.25% and compression rate improved up to 30.3%. We analyze compression rate with different number of input bits on Xilinx tool.

Acknowledgment

I would like to thanks Dr. Dilip Kumar H.O.D (Academic & Consultancy Services Division), Mrs. Vemu Sulochana, Mrs Manjit kaur, Er.Balwinder Singh for giving their suggestions. I would also like to thank our friends Miss Shikha Singh and Tapsi Singh for their advice for writing the paper. I would like to thank my family for their support.

References

[1] Parvinder Singh, Manoj Duhan and Priyanka "Enhancing LZW Algorithm to Increase Overall Performance", India Conference, 2006 Annual IEEE Sept. 2006.

[2] Ming-Bo Lin, Jang-Feng Lee, G. E. Jan, "A Lossless Data Compression and Decompression Algorithm and Its Hardware Architecture" VLSI IEEE Transactions on Volume: 14 2006.

[3] J.Ziv and A.Lempel, "Compression of individual sequences via variable length coding," IEEE Trans. Inf Theory, vol 24, pp. 530-536, 1978.

[4] M-B Lin, "A parallel VLSI architecture for the LZW data compression algorithm," J. VLSI Signal Process. vol. 26, no. 3, pp. 369–381, Nov.2000.

[5] Parvinder Singh, Sudhir Batra, and HR Sharma, "Evaluating the performance of message hidden in 1st and 2nd bit plane", WSEAS Trans. on Information Science an Applications, issue 8, vol 2, pp. 1220-1227, August 2005.

[6] H. Park and V. K. Prasanna, "Area efficient VLSI architectures for Huffman coding," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. vol. 40, no. 9, pp. 568–575, Sep. 1993.

[7] S. Khalid, "Introduction to Data Compression", 2nd San Mateo, CA: Morgan Kaufmann, 2000.

[8] Henriques and N. Ranganathan, "A parallel architecture for data compression," pp. 260–266 in Proc. 2nd IEEE Symp.Parall. Distrib. Process 2005.

[9] Huan Zhang, Xiao-ping Fan, Shao-qiang Liu Zhi Zhong "Design and Realization of Improved LZW Algorithm for Wireless Sensor Networks", International Conference on Information Science and Technology March 26-28, 2011.

Biographies

Simrandeep Kaur received the B.Tech degree in Computer Science Engineering from the Punjab technical university, Punjab in 2010, and pursuing M.Tech degree in VLSI Design from Centre of Development and Advance Computing Mohali, Punjab .Currently, She is doing her thesis work on data compression technique. Her topic of interest is data compression, security system, data structure and embedded system. Email-simrandeepkaur25@yahoo.com.

V.Sulochana Verma received M.Tech degree in VLSI Design from NIT Hamirpur Himachal Pradesh in 2009. Currently, She is a Project Consultant and member of Academic & Consultancy Services Division in C-DAC Mohali, Punjab, Her teaching and research areas include analog circuit design, interconnects and data compression techniques. Email-suchivlsi@gmail.com.