World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:12, 2009

An Improvement of PDLZW implementation with a Modified WSC Updating Technique on FPGA

Perapong Vichitkraivin, Orachat Chitsobhuk
Computer Engineering Department, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok Thailand
Email: solidy2000@hotmail.com, kcoracha@kmitl.ac.th

Abstract—In this paper, an improvement of PDLZW implementation with a new dictionary updating technique is proposed. A unique dictionary is partitioned into hierarchical variable word-width dictionaries. This allows us to search through dictionaries in parallel. Moreover, the barrel shifter is adopted for loading a new input string into the shift register in order to achieve a faster speed. However, the original PDLZW uses a simple FIFO update strategy, which is not efficient. Therefore, a new window based updating technique is implemented to better classify the difference in how often each particular address in the window is referred. The freezing policy is applied to the address most often referred, which would not be updated until all the other addresses in the window have the same priority. This guarantees that the more often referred addresses would not be updated until their time comes. This updating policy leads to an improvement on the compression efficiency of the proposed algorithm while still keep the architecture low complexity and easy to implement.

Keywords—lossless data compression, LZW algorithm, PDLZW algorithm, WSC and dictionary update.

I. Introduction

NOWADAYS data compression is important for storing and transmitting data. With a compression, the information can be represented with less amount of data. This allows us to save the storage space and transmission bandwidth. Data compression is broadly divided into two categories. One of these is lossless compression that the original data can be recovered without any data loss. It is usually used to compress text or binary files. The other is lossy compression that some loss of data quality is so acceptable. It is usually used to compress image data files.

One of the most widely used compression methods for lossless compression is LZW (Lempel-Ziv-Welch) [1]. LZW is a dictionary based compression, which encodes input data through establishing a string table and searching the table to identify the longest possible input data string that exists in the table. The encoded output is a sequence of the matching string's address and length. It can typically compress large English texts to about half of their original sizes. However, conventional LZW algorithm requires large amount of processing time for adjusting and searching through the dictionary. The dynamic LZW (DLZW) and word-based DLZW (WDLZW) algorithms were proposed to improve searching efficiency [2]. In DLZW, the dictionary has been initialized with different combinations of characters. It is organized in hierarchical string tables. The baseline idea is to store the most frequently used strings in the shorter table, which requires fewer bits to identify the corresponding string. The tables are updated using the move-to-front and weighting system with associated frequency counter. During the compression time, after the longest matching string is recognized in the table, it is moved to the first position of its block. The table updating process is based on the least recently used (LRU) policy to ensure that frequently used strings are kept in the smaller tables. This is to minimize the average number of bits required to code a string when compared with a single table implementation. The WDLZW algorithm is a modified version of DLZW that focuses on text compression by identifying each word in the text and make it a basic unit (symbol). The algorithm encodes the input word into literal codes and copy codes. If the search for a word has failed, it is sent out as a literal code, which is its original ASCII code preceded by other codes for identification. The copy code is the address of the matching string in the string table. However, both algorithms are too complicated and not suitable for hardware implementation. To improve this, parallel dictionary LZW (PDLZW) was proposed [3]. The PDLZW is a simplified DLZW architecture suited for VLSI realization. Since not all entries of the DLZW dictionary contains the same word size, this leads to the need of the entire dictionary search for every character. Consequently, the PDLZW has designed to overcome this problem by partitioning the dictionary into several dictionaries of different address spaces and sizes. With the hierarchical parallel dictionary set, the search time can be reduced significantly since these dictionaries can operate independently and thus can carry out their search operations in parallel. The fundamental modified structures are the utilization of a virtual dictionary and the first-in first-out (FIFO). The virtual dictionary only takes up a part of address space with no hardware cost. The first-in first-out (FIFO) could help to simplify the hardware implementation for the dictionary update policy. The selection of an updating algorithm depends on several factors such as compression ratio, complexity and ease of implementation. One of the effective dictionary updating techniques suitable for the hardware implementation of LZW is windowed second chance updating technique (WSC) [4]. WSC partitions a dictionary into windows and uses a one-bit flag associated with each phrase in the dictionary to keep track of an update. This helps in reducing the time complexity to select a phrase to be updated.

In this paper, we present a modified WSC dictionary updating technique for PDLZW. Instead of using a one-bit flag, the proposed architecture uses a three-bit flags associated with each address in the dictionary to keep track of the least referred address to be updated while implementing the freeze policy for the most frequently used addresses. The proposed architecture not only takes the benefits of the parallel structure of the hierarchical variable word width dictionary from PDLZW but also allows the efficient updating from the modification of the WSC algorithm. This leads to a better compression ratio while still maintains the same time complexity as WSC.

The rest of the paper is organized as follows. Section 2 describes the principle of both PDLZW and WSC algorithms. The proposed algorithm and its hardware architecture are presented in section 3 while section 4 discusses the simulation results and FPGA implementation cost. Finally, the conclusions are given in Section 5.

II. Related Work

A. PDLZW Algorithm

PDLZW algorithm is a LZW based implementation using a parallel dictionary set. It partitions one large dictionary into several small variable-word-width dictionaries. Searching in parallel through small dictionaries would require less amount of processing time than searching sequentially through one large-address-space dictionary. The FIFO (first-in first-out) is used as the dictionary update technique for simple hardware implementation. The architecture of PDLZW compression is showed in Fig. 1. In PDLZW, the input string is shifted into the shift register. All of the input characters in the shift register are searched from all CAMs (content addressable memory) simultaneously. The matched address having the largest number of bytes matched is returned as the output codeword. This could be implemented using the priority encoder and multiplexer. The longest string matched along with the next character is stored into the next entry pointed by the update pointer (UP) of the next-level dictionary.

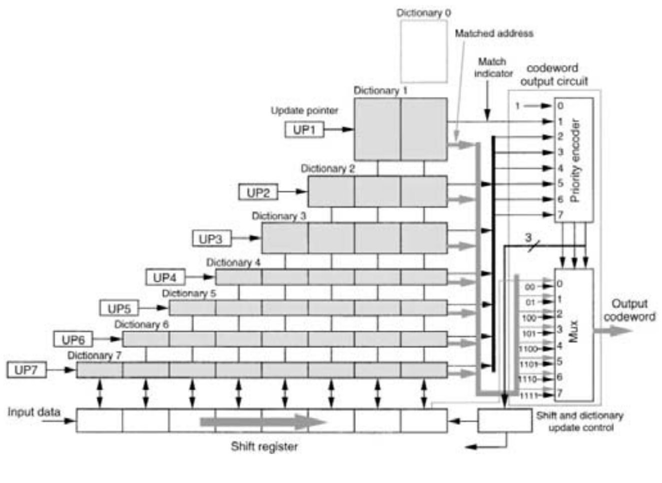

Fig. 1. The architecture of PDLZW compression processor

B. Windowed Second Chance Updating Technique (WSC)

Dictionary updating is a part of the LZW compression that has a direct effect on compression ratio. The main propose of dictionary updating is to find the address for the new data to be updated into the dictionary when it is full. WSC is one of the effective dictionary updating techniques suitable for the hardware implementation of LZW. It partitions a dictionary into windows with a size of K phrases. A one-bit flag is associated with each phrase. When the phrase is referred, the flag is set to 1. The updating occurs at the address that never been referred before where the flag is 0. However, if the dictionary is full, three rules for updating the flag bit are executed as followed.

1) More than one flag in the window are 0: Select the first phrase with flag 0, update the phrase, and set its flag.
2) Only one flag is 0: Update the phrase with flag 0, set its flag, and reset all the other flags in the window.
3) No flag is 0: Update the first phrase, set its flag, and reset all the other flags in the window.

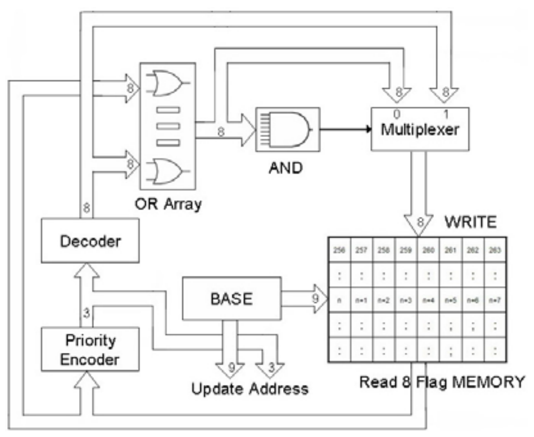

Fig. 2. The architecture of WSC Updating for K of 8

After the updating, the base pointer will point to the next window. The architecture of WSC updating is showed in Fig. 2 and Table 1 show the truth table of the priority encoder and decoder. The architecture consists of a memory to store the flags, a priority encoder to find the offset of the first flag 0, a decoder to determine the flag to be updated, an OR gate array to set the flag for the updated phrase and a multiplexer to select the new flag.

III. The Proposed Algorithm and Its Hardware Architecture

In this paper, we have proposed an improvement of the PDLZW technique with a modified WSC for updating strategy. In PDLZW compression algorithm, a unique dictionary is partitioned into hierarchical variable word-width dictionaries. The input string is shifted into the shift register and used to search through all the partitioned dictionaries in parallel to find the longest match. Then, it returns the index as the encoded output and store the new string in dictionary. Shifting input into the shift register leads to slow encoding speed. In this paper, the barrel shifter is adopted for loading a new input string into the shift register in order to achieve a faster speed. However, the original PDLZW uses a simple FIFO update strategy, which is not efficient. The WSC method provides an efficient update technique, which aims to reduce the complexity in selecting the address to be updated. However, one of the problems in WSC is that there are only a few chances for the flag to be ’0’. They are at least referred to once. In this case, the first address in the window will be updated even it is referred more than the others in the same window. Therefore, we present a new updating technique to improve the WSC technique using a three-bit flag associated with each address in the dictionary. A three-bit flag is used to differentiate the frequently referred addresses from the least referred ones. When the address is referred, the flag is increased by 1. The freezing policy is applied to the address having the value of a three-bit flag greater than three. When the dictionary is full, the flag associated with each address in the window is examined the number of times that address has been referred. The least referred address would expect to be updated before. The hierarchical update strategy has been executed based on the value of the flag as followed.

1) More than one address in the window has never been referred: Select the first address never been referred and set the flag of this address to 1.
2) Only one address has never been referred: Select that address, set the flag to 1 and reset the flags of the other addresses in the window except the freezing address.
3) All the addresses are referred at least one time: Select the first address that is referred less than two times, set the flag to 1, and reset the flags of the other addresses in the window except the freezing address.
4) All the addresses are referred at least two times: Select the first address that is referred less than four times, set the flag to 1, and reset the flags of the other addresses in the window except the freezing address.
5) All the addresses are frozen: Select the first address in the window.

The block diagram of the proposed algorithm is shown in Fig. 3 while the hardware implementation of the proposed updating technique is shown in Fig. 5.

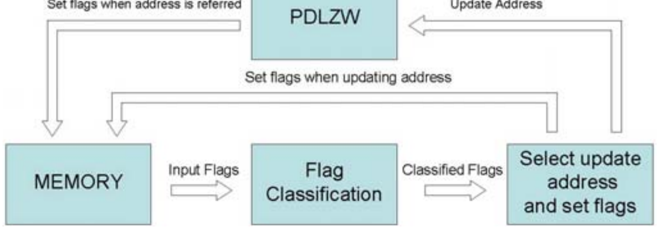
Set flags when address is referred
PDLZW
Update Address
Set flags when updating address
MEMORY
Input Flags
Flag Classification
Classified Flags
Select update address and set flags

Fig. 3. The block diagram of the proposed algorithm

The window size of 8 is used in this paper. Each address is associated with a three-bit flag. The range of the updated addresses pointed by BASE is from n to n+7 in the window. This requires the 8x3 = 24-bit flags for each window. The 24-bit flags are classified into three groups; A, B and C using the array of OR gates. The flags would be classified into group A, if its associated address is referred to at least once. If the address is referred to at least two and four times, its flag would be classified into group B and C respectively.

The OR Array A is used to evaluate the group flags from a set of 24-bit flags while the 16-bit flags of the most 2 significant bits of a three-bit flag are analyzed by the OR Array B. For group C, only the 8-bit flags of the most significant bit are considered. The group flag is set to ’1’ if the flags are classified into its group. The multiplexer then selects the group flag data from group A, B, and C using ”sel” signal. The ”sel” will select data from group A if at least one group flag of group A is ’0’. However, if all the group flags of group A are set to ’1’ and at least one group flag of group B becomes ’0’, the output will be from group B. Otherwise, the output is from group C. The group flag output from the multiplexer is decoded by a priority encoder to select the update address. The update address is sent to PDLZW for updating that address to a new string. The example of classification of a three-bit flag from each updated address is presented in Fig. 4. The example flags are ”001010000100100011000001”. The group flags of group A, B, and C are ”11011101”, ”01011100”, and ”00011000” respectively. In this case, the update address is selected from the data in group A and the third address in this window is updated. Then, the flags are set into ”001010001100100011 000001”.

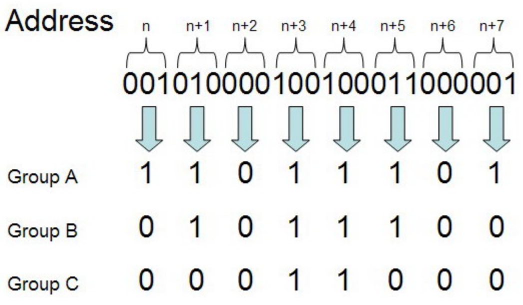

Fig. 4. Example of classify three-bit flags into three groups

The reset condition is determined by the decoder, OR Array C, and AND gate C and transmitted to the Update Bit Control to reset all the flags in the window. After the updating, Base will point to the next window. The truth tables of the priority encoder and decoder are adopted from the WSC technique and presented in Table 1.

TABLE I
PRIORITY ENCODER AND DECODER TRUE TABLES

Source Flag								Priority Encoder			Decoder							
F7	F6	F5	F4	F3	F2	F1	F0	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	1
X	X	X	X	X	X	0	1	0	0	1	0	0	0	0	0	0	1	0
X	X	X	X	X	0	1	1	0	1	0	0	0	0	0	0	1	0	0
X	X	X	X	0	1	1	1	0	1	1	0	0	0	0	1	0	0	0
X	X	X	0	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0
X	X	0	1	1	1	1	1	1	0	1	0	0	1	0	0	0	0	0
X	0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1

The basic concept of the proposed algorithm is to freeze the address that is referred at least four times before the flags are reset. With the new hierarchical updating strategy, the proposed algorithm could not only improve the efficiency in updating the dictionary but also be able to increase the compression ratio compared to other techniques.

IV. Algorithm Simulation and FPGA Implementation

To evaluate the compression performance of the proposed algorithm, a variety of eleven different types of files from the Canterbury Corpus and two large files from The Large Corpus are used [5]. We compare the compression ratios of the three algorithms, conventional PDLZW algorithm, PDLZW with WSC algorithms and PDLZW with the proposed updating algorithms. A 1K-address space dictionary is used for each algorithm with a window size of 8. All the algorithms are implemented on XILINX (Spartan III XC3S500) FPGA. Table 2 shows the comparison of the compression ratio for each technique. From the results in table 2, the average compression ratio of the PDLZW with the proposed updating algorithm is greater than that of the conventional PDLZW algorithm about 5%, and 2.3% greater than PDLZW with WSC algorithm. It can be seen that the PDLZW with the proposed updating technique outperforms the other techniques. With the new updating technique, we can effectively utilize the dictionary. This leads to an improvement in the compression efficiency of the proposed algorithm while still keep the architecture low complexity and easy to implement. Moreover, the implementation of the barrel shifter instead of the regular shift register helps the proposed architecture to achieve a faster speed compared to the conventional PDLZW.

The compression rate is defined as the number of input bits which can be compressed in one second but it depends on the amount of the input data to be matched and the structure of the dictionary. For the PDLZW, the hierarchical variable word-width is adopted where the minimum and maximum sizes of a string matched are one byte and 8 bytes respectively. Therefore, the minimum compression rate is calculated in the case of only one byte matched per compression while the maximum compression rate is calculated in the case that all 8 bytes are matched for each compression. The maximum clock rate of the proposed architecture is 49.4 MHz. The time required for compression operation is 4 clocks. Consequently, the minimum compression rate = [49.4 MHz / 4] x 1 x 8 bits = 98.8 Mbits/s, thus the maximum compression rate = [49.4 MHz / 4] x 8 x 8 bits = 790.4 Mbits/s. Table 3 illustrates some of the hardware differences between the FPGA implementation of proposed architecture, the conventional PDLZW and the PDLZW with WSC.

V. Conclusion

In this paper, we have presented a new dictionary updating technique for the hardware implementation of PDLZW data compression and its hardware architecture. The proposed architecture employed the barrel shifter to the PDLZW architecture in order to achieve a faster speed. Moreover, we have introduced a new dictionary updating technique based on WSC to improve the compression ratio while still keep the proposed architecture low complexity and easy to implement. The proposed updating technique is a window based tracking technique using a three-bit flag instead of a one-bit flag in order to better classify the difference in how often each particular address in the window is referred. The freezing policy is applied to the address having the value of a three-bit flag greater than three, which is considered to be the address most often referred and would not be updated until all the other addresses in the window have the same priority. This guarantees that the more often referred addresses would not be updated until their time comes. This updating policy leads to an improvement on the compression efficiency of the proposed algorithm. It has been shown from the test results that the PDLZW with the proposed updating technique gives a better compression ratio of 5% greater than that of the conventional PDLZW with a simple FIFO updating technique and 2.3% greater than that of the PDLZW with WSC technique.

Thus, the required hardware cost is comparable to the other techniques

References

[1] T.A. Welch, ”A Technique for High-Performance Data Compression,” *IEEE Computer*, vol. 17, no. 6, pp. 8-19, June, 1984.

[2] J. Jiang and S. Jones, ”Word-based dynamic algorithms for data compression,” *Proc. Inst. Elect. Eng.-I*, vol. 139, no. 6, pp. 582-586, Dec 1992.

[3] 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.

[4] Ch. Su, Ch. Fan and J. Ch. Yo, ”Hardware efficient updating technique for LZW CODEC design”, *1997 IEEE International Symposium on Circuits and Systems*, pp. 2797-2800, June 1997.

[5] The Canterbury Corpus file for testing new compression algorithms. Available: http://corpus.canterbury.ac.nz/index.html.

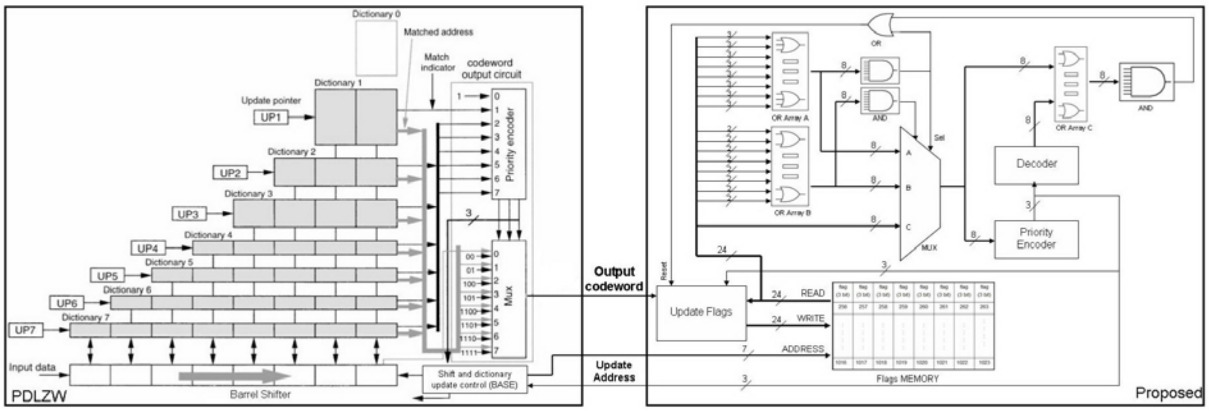

Fig. 5. The VLSI architecture of the proposed algorithm

TABLE II
COMPRESSION RATIO ACHIEVED ON TEST FILES

File	Size (Byte)	PDLZW	PDLZW + WSC	PDLZW + Proposed
alice29.txt	152089	1.792076	1.866695	1.922652
asyoulik.txt	125179	1.689667	1.757237	1.822674
cp.html	24603	1.806222	1.851764	1.878092
fields.c	11150	2.087038	2.140115	2.174549
grammar.lsp	3721	1.992503	2.014073	2.020910
kennedy.xls	1029744	3.289470	3.375214	3.334555
lcet10.txt	426754	1.776273	1.848380	1.939306
plrabn12.txt	481861	1.703185	1.775440	1.861563
ptt5	513216	4.291104	4.337572	4.386884
sum	38240	1.918836	1.952390	1.942719
xargs.1	4227	1.651172	1.685743	1.713070
bible.txt	4047392	2.040628	2.118654	2.276404
world192.txt	2473400	1.624985	1.673976	1.775846
Average	717813	2.127935	2.184404	2.234556

TABLE III
A COMPARISON OF DIFFERENT HARDWARE IMPLEMENTATIONS

	Number of Slices		Number of Slices Flip Flops		Number of 4 input LUTs		Number of BRAMs		Clock cycle	Maximum clock rate
PDLZW	3796	11%	1424	2%	6902	10%	96	92%	4 - 11	53.8 MHz
PDLZW + WSC	3819	11%	1476	2%	7073	10%	97	93%	4 - 11	47.0 MHz
PDLZW + Proposed	3826	11%	1475	2%	7168	10%	97	93%	4	49.4 MHz