2.1 BACKGROUND OF THE STUDY

The purpose of this chapter is to study and analyze famous lossless data compression algorithm, called LZW. The main objective of the study is to analyze the several LZW implementations which reduce the computational complexity, varieties of enhancement and optimization which are done on LZW. This chapter of the thesis also summarizes the improved architecture of the LZW data compression algorithm and discusses how the computational cost is reduced using various previous innovative approaches.

2.1.1 GRAMMAR BASED LZ OPTIMIZATION

LZ grammar based compression [68], the algorithm run in \( O(n \log |\Sigma|) \) time and \( O(\log n) \) ratio where the given string length \( n \) over the alphabet \( \Sigma \). A new way of reducing code redundancy in greedy-parsing dictionary coders called phrase elimination, is proposed by Zhen Yao [69]. Mathematical estimation of the expected number of dictionary phrases after the elimination is verified with empirical statistics. He has found the balanced code in LRC family and improved around 3% over LZW with balanced code and 5% over traditional LZW.

2.1.2 MULTIPLE DICTIONARY LZW (MDLZW) OR PARALLEL DICTIONARY LZW PDLZW

To enhance the performance of dictionary based algorithm, two methods are suggested [55]: i). to studying how to choose the minimum dictionary size to minimize the time taken and maximize compression, ii). Modifying the dictionary building process to reduce the ultimate dictionary size. The performance of LZW has improved using three modifications [82]. The first two enhancements eliminate the frequent flushing of the dictionary. This removes the computational complexity of LZW [124]. The third enhancement improves the compression ratio. Two pass LZW [70] offline algorithm uses separate static dictionary for each file. Tire is constructed in the first phase and actual Encoding is carried in the second phase. The tire is attached to the compressed file. The tire like implementation reduces the computational cost. A unique dictionary is partitioned into hierarchical variable word-width dictionaries. This allows us to search through dictionaries in
parallel. Moreover, the barrel shifter [83] is adopted for loading a new input string into the shift register in order to achieve faster speed. However, the original Parallel Dictionary LZW (PDLZW) (initially this dictionary called Multiple Dictionary LZW (MDLZW) uses a simple First in First Out (FIFO) update strategy, which is replaced with a new window based updating technique, implemented to classify the differences, how often each particular address in the window is referred. The freezing policy is applied to the address which is most often referred and it will not be updated until all the other addresses in the window have the same priority. This guarantees that the more often referred addresses will not be updated until their time comes; this leads to an improvement on the compression efficiency and achieves low computational complexity. A new Two-Stage architecture that combines the features of both Parallel-dictionary –LZW (PDLZW) [62] and an approximated adaptive algorithm. Another two stage compression algorithm combines the features of PDLZW and Arithmetic Coding [76] [77]. In this architecture order list is used instead of tree based structure. The approach replaces the hardware cost, the compression ratio and time cost is reduced. A multi-processor based algorithm is called Bi Directory LZW (BDLZW) [57]. The implementation is approximately like PDLZW. As the algorithm runs on multi-processor system such as CELL, this can minimize the computation time. The algorithm reads string from both the ends and loading two threads fairly and easily. When comparing the running time, ten percentage is improved than the conventional PDLZW when there are two threads running on CELL architecture. The parallel dictionary of LZW is implemented with the parallel VLSI architecture [20] that can improve the throughput, the compression ratio and reduces the time complexity because of the parallel search using the VLSI architecture. The algorithm combines the features of both parallel dictionary LZW (PDLZW) and an approximated Adaptive Huffman (AH) algorithm [93] [91]. The algorithm achieves compression ratio but cost of computation is slightly increased when compared with traditional PDLZW. High speed Low-complexity register transfer logic (RTL) design and implementation [96] of LZW algorithm on Xilinx vertex in device family for high bandwidth applications. This offers high throughput and lower power requirements. Another hardware implementation using the enhanced content-Addressable-Memory (CAM) [81] cells to accelerate the implementation of LZW [92]. By enhancing the efficiency of Systolic-Array and FPGA approaches increase the speed and compression ratio [63]. LZW [33] based algorithm for both complete and practical bit streams compression for vertex FPGA’s. Only the compression ratio is improved. LZWA is a memory-based VLSI architecture [1] of text compression, binary tire is used to represent the dictionary of LZWA, using the properties of
the tire can easily mapped into the memory, this leads to the replacing of fixed length code with variable length prefix binary codes. This approach slightly reduces the time and increases the compression efficiency [99]. The windowed second chance (WSC) updating technique uses partition dictionary in to several windows to minimize the computational complexity [36]. An implementation of LZW with CAM and WSC [11] by priority encoder/decoder uses one character clock for compression or decompression.

New dictionary updating technique is proposed for PDLZW [83]. The barrel shifter is adopted for loading a new input string into the shift register in order to achieve faster speed. However, the original PDLZW uses 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 [100]. The freezing policy is applied to the address most often referred, which will not be updated until all the other addresses in the window have the same priority. This guarantees that the more often referred addresses will not be updated until their time comes. This updating policy leads to improve the compression efficiency of the proposed algorithm, while still keeping the architecture low complexity and easy to implement.

To improve the compression ratio and reduce the time complexity of LZW multiple dictionary is used [39], because this kind of implementation is faster than prediction by pattern matching (PPMC), but only the compression ratio is experimented [3] by the author. A two level N-array tree implementation [48] is achieved high encoding and speed for LZ relevant algorithms. The speed purely depends on the tree hierarchy used to reduce the search complexity in dictionary D.

2.1.3 Hybrid LZW implementations

The paper [75] proposes two stage algorithms that carry advantages of PDLZW and Arithmetic Coding (AC) and compares its performance with deflate, which a well-known two - stage algorithm that combines the features of LZ77 and Huffman Coding. The PDLZW is designed 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. Arithmetic coding replaces a stream of input symbols with a single floating - point output number [17]. The cascading of two algorithms that combines the
features of both PDLZW [77] and Arithmetic Coding, and also compares this with deflate which is a cascading of LZ77 and Huffman Coding.

LZW-RLE [79] algorithm is built by combining the features of Run length Encoding (RLE) with LZW and free branches deleting method is analyzed and stimulated. The algorithm achieves better compression ratio but high time complexity, because it takes the sum of time required for RLE and LZW i.e., $O(n \log |D| + n)$ time. Another hybrid RLE and LZW [122] based algorithm is designed and pretreatment approach is done before the encoding. The compression algorithm solved low compression ratio and high computational cost.

A new dictionary based algorithm divides file into chunks of predefined length before compressing [4]. The concept is borrowed from GP-Zip, and GP is calculated for Arithmetic Coding (AC), Lempel Ziv Welch (LZW), unbounded prediction by partial matching (PPMD), Run Length Encoding (RLE) and Boolean Minimization (BM), in addition two transformation algorithms namely BWT and MTF are used. The file is compressed and composed of heterogeneous data fragments. This method is capable of achieving best compression ratio but time complexity is not good. To find the optimal compression algorithm K-means Clustering [5] is used with GP-Zip.

The accelerated dictionary loading algorithm [73] is used to improve the compression ratio and compression speed. The lossless image compression algorithm is developed using the Huffman with LZW dictionary Duplication Free Runs Length Coding (DFRLC) [116] [12]. To reduce the computation cost of LZ variants, LZ combines with BP network [30]. A recursive [84] way of finding the longest match in LZW is increased the compression ratio but this approach leads to increase the computational cost. A reduced computational complexity model using LZW for online prediction [39], and this approach does not have an offline component and fit in memory with good prediction accuracy are accepted for modeling the user navigation in the web. LZW based variable-Sized-Block [16] method requires only simple lookup and update when necessary. This approach reduces the computational time.

2.1.4 WORD BASED LZW IMPLEMENTATIONS

A new algorithm [47], word-based dynamic Lempel-Ziv (WDLZW) for universal (lossless) data compression is introduced. The novel feature is that the algorithm is optimized
for the compression of natural language data, in which all the spaces between words are deleted whenever copy codes or literal codes are sent out. Therefore better compression rates can be achieved. The algorithm can still compress alternative forms of data [60] [61]. The structure, operation and implementation of the WDLZW are described. A comparison with other algorithms when compressing a wide range of data forms is reported. For text-based information WDLZW offers attractive performance. For other forms of data, WDLZW provides compression rates similar to those of dynamic Lempel-Ziv systems. In the survey of word based text compression algorithm, the word bases LZW compression algorithm discussed [74], all algorithms are normally functioning on alphabets or symbols, but this approach uses the symbols or the words of English or string of alphanumeric characters and non-alpha numeric characters. The computational complexity of this approach is purely based on the number of string in the input text i.e., $O(n \log |D|)$, where n is number of words in the X and $|D|$ the length of dictionary.

Fixed-Word-width dictionary of LZW compression algorithm uses very large address space; this approach increases computational complexity. Instead of using unique fixed word dictionary, a hierarchical variable –Word-Width dictionary [114] set containing several small address space dictionaries with increasing word width. The approach of variable-word-width dictionary technique reduces the time or computational complexity.

LZW index code word is replaced by binary index [64] to enhance the compression ratio but the time required is approximately same as original LZW. Self-Embedding document water marking technique using LZW [95] algorithm uses seven bits per character for indexing. The dictionary vocabulary indices need not be adapted regularly. This approach reduces time taken for the process.

Efficiency and capability of LZW++ algorithm [66] are discussed. The LZW++ technique is a modified and enhanced version of existing LZW technique. The modification of the existing LZW is used to produce LZW++ technique. LZW reads one by one character at one time. Differ with LZW++ technique, where the LZW++ read three characters at one time. The LZW++ algorithm takes $O\left(\frac{|X|}{3} \times \log |D|\right)$ time, where X is the input sequence with the length $|X|$ and D is the dictionary with the length $|D|$.
2.1.5 Hardware LZW Implementations

Hardware implementation of data compression algorithms is receiving increasing attention due to exponentially expanding network traffic and digital data storage usage. Several serial one-dimensional and parallel two-dimensional systolic-arrays are proposed for Lempel-Ziv data compression [107]. A VLSI chip is implemented for optimal linear array is fabricated and tested. This array architecture is scalable also, multiple chips (linear arrays) can be connected in parallel to implement the parallel array structure and provide a proportional speedup.

2.1.6 Data Structure Implementations of LZW

The paper [122] implements an LZW algorithm based on a tree-like data structure in C programming and offers two optimization schemes, including using pointer-trace and variable-length code. The results show that these schemes extremely improve the compression efficiency with reduced computation cost.

LZW compression and Set Partitioning In Hierarchical Trees (SPIHT) code [36] are used to obtain a low bit rate and high quality image compression. In the process, an adaptive phase modulation (APM) mechanism and Discrete Fourier Transformation (DFT) are adopted for secret data embedding. The nearest phase modulation technique is used to improve imperceptibility of cover image. Quantized hashing search technique [91] is used to lookup the values from the LZW dictionary and the Computational Cost are reduced with the implementation of Quantized hashing technique.

A new compression method used to compress trees and graphs, by building the LZW tree in LZW compression is proposed [105] and replaces the parsed sub trees by dictionary indices to form the compressed tree, this approach, is very useful to compress undirected and directed acyclic graph. The another LZW[102] based algorithm accelerates and improves the space requirements of known technique is based on the reduction of graph-theoretic problem by reducing size of the graph, without affecting the efficiency of the solution and improves any static dictionary scheme.

Data-Hiding LZW [45] (DH-LZW) is a process of data hiding and it is very much compactable with LZW. The computational cost of this approach is slightly more than the original LZW, because of the data hiding process due to the additional data hiding in the encoding phase.
The LZW implementation in C is discussed [28] using the tree like structure and pointer concept also replaced the fixed length code with variable length code. The C implementations increase the compression efficiency and reduce the computational cost. Four dimensional approaches to compress sub bands of image using the LZW are proposed [101]. The temporal prediction and encoding of stream is done in Matlab.

2.1.7 LEMPEL ZIV WELCH GENETIC ALGORITHM

LZWGA [71] (Lempel Ziv Welch Genetic Algorithm) is designed to compress the chromosomes sequences. Chromosomes are dealing with binary strings, when the length of binary string increases, the search space also increases. LZWGA algorithm reduces the search space of original problem $9.90 \times 10^{301029}$ points. When compressing, that is reduced to $8.37 \times 10^{166717}$ points. A multi mutation in LZWGA [108] results a new method can solve One Max and trap problem 46.3 percentage, faster mover over, this method can reduce the size of the compressed chromosome by 54.8 percentage. A DNA compression method is using the hybrid approach of LZW and Huffman coding is proposed [78]. LZW-MIMIC (LZW Mutual Information Maximizing input clustering algorithm is developed and discussed [118].

2.1.8 MEDICAL DATA COMPRESSION AND LZW

ECG (Electro Cardiogram) signal compression [65] based on local extreme extraction, adaptive hysteretic and Lempel Ziv Welch coding has been verified using eight of the most frequent normal and pathological types of cardiac beats and Multi-Layer Perception (MLP) neural network trained with original cardiac patterns and tested with reconstructed ones. Aspects regarding the possibility of using the principal component analysis (PCA) to cardiac pattern classifications are done [15]. A new compression measure called “Quality Score” takes in to account both reconstruction errors and compression ratio are achieved. The library based encoder LZW used in a Low power capsule endoscopic system. The library size can be set by the user and output data rate can be controlled according to the band width requirement. Simulation carried out with several endoscopic images. The algorithm is named as LZW-Flush [86] algorithm. For lossless medical compression, adaptive LZW algorithm [113] is proposed. Dictionary Lempel-Ziv-Welch (PDLZW) method and the wavelet transform method are used to attain very efficient and faster electrocardiogram (ECG) data compression [46]. Slow and less compression ratio is the limitation of wavelet transform, this issue is solved by their approach and they compare the compression of ECG with different types of wavelets.
2.1.9 Other LZW Enhancements

A new algorithm based on LZW to compress Dynamic-Data [85] (the size data which keeps increasing rapidly). The method permits to update and modify the document with minimum overhead. Further the method allows data to be appended to already compressed text; this approach reduces the repeated compression cycle, and this method directly leads to reduce the time. The implementation satisfies compression ratio over existing LZW.

A modified LZW [67] compression algorithm for image compression is proposed. Before applying the LZW compression the image is segmented into its color components, for example Red Green Blue (RGB) and separately performs the compression on each segments, the compression ratio of proposed method is better than the standard LZW method by 55.6% for colored image, and 102% for gray scaled images. But the time taken for this approach is three times more than standard LZW i.e., \(O(3 \cdot |X| \cdot \log|D|)\). Voice data integrated LZW (VDILZW) [35] is used both code word of complete Hangeul [35] and the variable length code word methodology in compression. This algorithm is specially developed for voice compression.

A new approach is designed for the variable size dictionary based compression methods [71]. This approach replaces the flat codes in the original LZW compression algorithm, and removes nearly three to seven percentage of redundancy. The experimentation is done on artificial Canterbury corpus and this approach runs in \(O(N \log N)\) time [71]. Chang Hong Lin, Yuan Xie and Wayne Wolf [37] have proposed LZW-based on selective code compression schemes that use branch blocks as the compression unit. The compression ratio is about 83% and 75% respectively, this approach achieves Low power and requires smaller memory to store compressed source code. These schemes have less decompression overhead and larger decoding bandwidth with compatible compression ratio [119]. Parallel decompression could also be applied to these methods to achieve faster decompression; the algorithm runs in \(O(\sum_{i=1}^{K}(n_t \cdot \log|D_t|))\) time, where \(K\) is the number of block, \(n_t\) is the length of each block, \(|D_t|\) indicates the length of each dictionary. Gonzalo Navarro [55] has improved the performance of LZW compression by improving the data structure of LZW. He introduced a data structure called LZ-index based on the LZW-tire. His algorithm requires \(4n \log_2 n (1 + o(1))\) space and \(O(m^3 \log \sigma + (m + R)\log n)\) in worst case and \(O(\sigma m \log u + \sqrt{uR})\) average search time.
2.2.10 **KNN and Its Enhancements**

K-Nearest Neighbor Algorithm (KNN) is the most popular algorithm in data mining. An improved KNN algorithm, which uses different numbers of nearest neighbors for different categories, is proposed [56]. For evaluating Reverse k-Nearest-Neighbor (RKNN) queries and its various bio chromatic RKNN queries on 2-dimensional location data, a new algorithm proposed called FINCH (Fast RKNN processing using INCH) that can compute a RKNN query search region (from which candidates draw the query result). In our RKNN evaluation algorithm called FINCH [120][54][66][18], a new approach is developed for text categorization, in which the importance of discriminating words is learned using mutual information and weight adjustment techniques using KNN [26][7][25]. An improved KNN algorithm for text categorization, which builds the classification model by combining constrained one pass clustering algorithm and KNN text categorization is developed and discussed [106][21]. RS-tree-based incremental Nearest Neighbor Algorithm is developed to solve the limitation of R-tree [24]. The thesis [24] explores the application of Compute Unified Device Architecture (CUDA) for data classification using the k-Nearest Neighbor (KNN) Algorithm. The limitations of conventional KNN algorithm are solved by combining the features of Genetic Algorithm (GA), and KNN. The limitations are calculation complexity due to the usage of all the training samples for classification, the performance is solely dependent on the training set, and there is no weight difference between samples [112]. The computational cost of K means algorithm is reduced using parallel computing [94][37]. The distances of the nearest neighbors can be taken into account. In this sense there is a close connection to LOESS, a local regression technique. In addition the enhancement possibilities to use nearest neighbor for classification in the case of an ordinal class structure [52][80]. A new strategy is introduced to re-sample the historical data with perturbations using KNN [67].

2.2 **Summary**

The LZW data compression algorithm is a dictionary based data compression algorithm. The computational cost of the algorithm purely relates with the data structure used by the LZW dictionary. The base algorithm selected for the research is MDLZW or PDLZW. The literature shows several kinds of optimization in order to reduce the computational cost. Primarily the implementation can be classified into two: one is improving the ability of algorithm using some algorithmic techniques and another is with the dedicated hardware. The
second method has several disadvantages such as it is purely hardware depended, so to run such implementation of algorithm it should have a dedicated or sophisticated hardware. This approach also leads to increase the economic barrier. So the focus of the thesis falls on the first method.