Journal of Computer Science 3 (1): 25-27, 2007
ISSN 1549-3636
© Science Publications 2007

Lossless Compression Using LFSRs

R. Muthiah, K. Neelakantan and Somi Sebastian
TIFAC-CORE, SASTRA, Thirumalaisamudram, Thanjavur, Tamil Nadu, India
GDA Technologies Ltd., Bangalore, India

Abstract: While lossy compression techniques are used for images, sound. where feature reproduction is more important than the actual data, it is necessary to use lossless compression for certain files containing data, programmes. Lossless compression techniques are not as effective as lossy ones and usually require a lot of computation for both compression and decompression. We presented a novel lossless compression techniques based on linear feedback shift registers. The advantage of this technique is that decompression is fairly simple and can be implemented without extensive computations. The decompression algorithm is also useful for implementation in hardware. In this paper we presented the details of the LFSR based algorithm. The results on various types of data such as images, sound files and text files presented and the results compared against existing lossless compression techniques.

Key words: LFSR, **l**ossless compression, berlekamp algorithm

INTRODUCTION

In today's world of multimedia and internet, image compression [1, 2, 3, and 6] poses a great challenge to both academic community and the industry people. The huge amount of data with limited storage capacity and restricted communication bandwidth makes compression an absolute necessity. Although standard compression like DCT [8] performs well at low compression ratio, at low bit rates it has been found to introduce blocking artifacts, which tend to be visually disturbing. On the other hand sub band based coding techniques line DWT has been found to be free from shortcoming. At low bit rates DWT leads to a uniform degradation of the sequence.

This paper focuses on lossless data compression without the use of the dictionary based algorithm [7]; rather it makes use of the Berlekamp algorithm and gets the image data encoded and the reproduction of the image data using the LFSR. The new architecture enables the use of a seed for a variable, number of tester cycles, such that each seed covers the maximum number of bits specified, resulting in a significant compression ratio. The compression technique is very easy to implement and the hardware overhead is very easy to implement than the conventional one.

Lossy data compression is named for what it does. One applies lossy data compression to a message; the message can never be recovered exactly as it was before it was compressed. When the compressed message is decoded it does not give back the original message. Data is lost. Because lossy compression can not be decoded to yield the exact original message, it is not a good method of compression for critical data, such as textual data. It is not useful for digitally sampled analog data[5]. In lossless data compression file the original message can be exactly decoded. Lossless data compression works by finding repeated patterns in a message and encoding those patterns in an efficient manner. For this reason, lossless data compression is also referred as redundancy reduction.

Many of the lossy compression use the discrete Fourier transforms [5], the disadvantage of this approach are computationally expensive. Hardware implementation is also complicated and expensive. This process is proving to rounding errors. For all critical application like bio-medical application we go for lossless image compression , as accurate reproduction of the compressed data is needed in the receiver side else it could be fatal. For other non critical application like video communications, voice communications etc., we go for lossy compression as even if we loose few bits out of many while decompressing the compressed data, still we can retrieve the original data and it will not cause anything fatal, but it is not the same when it is bio-medical application and hence we prefer lossless compression technique for such critical application.

Lossless algorithms are those which reproduce the original image after compression. WinZip, RAR, TAR are some of the well known lossless algorithms used for compression. Win Zip Algorithm is file archive for Microsoft windows. It uses the PKZIP format and can also handle a number of other archive formats. WinZip uses the dictionary encoding technique for its compression it is similar to the LZ77 [7], where the repeated sequences of input file are encoded and stored in a dictionary, later if these sequence are repeated in the input file its checked from the dictionary and encodes the incoming data sequences.

High compression ratio can be achieved by increasing the size of the dictionary and also the input file size. RAR and TAR are also some lossless compressions that are used in UNIX environments and they are also adopted to use for windows environment.

Corresponding Author: R. Muthaiah, TIFAC-CORE, SASTRA, Thirumalaisamudram, Thanjavur – 613402, TamilNadu, India

The use of the BerleKamp algorithm [4] is to compute the linear complexity of a given sequence and find the length of the shortest LFSR, which can be produce that sequence. The measure therefore speaks to the difficulty of generating, a particular sequence. We denote Lk({si}>0) to be the L.C of the sequence s0,s1………………sk-1 and C(k)(x) to be the characteristic polynomial of an Lk stage LFSR that generates si,0<i<sk-1. In general c (k) (x) is not unique for a give Lk-stage LFSR. We define the L.C of all zero vector by length k to be 0 and c (k) (x) =1.
Example: Using the Berlekamp Massey Algorithm, find the characteristic polynomial of the lowest possible degree that will generate the sequence 1, 1,0,1,0,0,1,0.

Step (1) we have $s_0 = 1$. By theorem 4.1, $L_1 = 1$ and $c^{(1)}(x) = 1+x$
Step (2) $s_1 = 1$. Verify $c^{(2)}(x) = c^{(1)}(x)$. Then $L_2 = 1$ and $c^{(2)}(x) = 1 + x$
Step (3) $s_2 = 0$. Verify $c^{(3)}(x) \neq c^{(2)}(x)$. Then $L_3 = \max\{L_2, 3 - L_2\} = 2$ implies our feedback polynomial $c^{(3)}(x)$ is of degree 2. We have $L_0 = 0 < L_2$ and $L_1 = L_2 = 1$. Thus we choose m = 0, and $c^{(0)}(x) = 1$ by definition. Thus $c^{(3)}(x) = x^{2-1}c^{(2)}(x) + x^{2-2}c^{(0)}(x) = x(1 + x) + 1 = 1 + x + x^2$ by the Berlekamp Massey Algorithm.
Step (4) s3 = 1. Verify $c^{(4)}(x) = c^{(3)}(x)$. Then $L_4 = 2$ and $c^{(4)}(x) = 1 + x + x^2$.
Step (5) $s_4 = 0$. Verify $c^{(5)}(x) \neq c^{(4)}(x)$. Then $L_5 = \max\{L_4, 5 - L_2\} = 3$ implies our feedback polynomial $c^{(5)}(x)$ is of degree 3. We have $c^{(m)}(x) = c^{(2)}(x) = 1 + x$. Then $c^{(5)}(x) = x\,c^{(4)}(x) + c^{(2)}(x) = x(1 + x + x^2) + (1 + x) = 1 + x^2 + x^3$.
Step (6) $s_5 = 0$. Verify $c^{(6)}(x) = c^{(5)}(x)$. Then $L_6 = 3$ and $c^{(6)}(x) = 1 + x^2 + x^3$.
Step (7) $s_6 = 1$. Verify $c^{(7)}(x) = c^{(6)}(x)$. Then $L_7 = 3$ and $c^{(7)}(x) = 1 + x^2 + x^3$.
Step (8) $s_7 = 0$. Verify $c^{(8)}(x) \neq c^{(7)}(x)$. Then $L_8 = \max\{L_7, 8 - L_7\} = 5$ implies our feedback polynomial $c^{(8)}(x)$ is of degree 5.
We have $c^{(m)}(x) = c^{(4)}(x) = 1 + x + x^2$. Then $c^{(8)}(x) = x^2c^{(7)}(x) + c^{(4)}(x) = x^2(1 + x^2 + x^3) + (1 + x + x^2) = 1 + x + x^4 + x^5$.

LFSR's[3] are commonly used in application where pseudo-random bit streams are required. An LFSR is a mechanism for generating a sequence of binary bits. The register consists of series y cells that are set by an initialization vector that b, most often, the secret key. The behavior of the register is regulated by a clock and at each clocking instant, the counters of the cells of the register are shifted left by one position, and the cursor of a subject of the call contacts is placed in the right most cells.

A basic LFSR consist of 3 Components.
(i) Input sequence
(ii)Feedback (tap sequence)
(iii)Output

Let the input sequence of length n the (S0, S1 …Sn-1). The feedback is thus a linear function. given by

$$f(s) = \sum_{i=0}^{n-1} c_i s_i$$

where c_o, c_1, c_n are constant coefficients

The output of the LFSR is determined by the initial values S0, S1, S2………..sn-1 are the linear recursion relationship

$$s_{k+n} = \left(\sum_{i=0}^{n-1} c_i s_{i+k} \right),\ k \geq 0$$

Or equivalently,

$$\sum_{i=0}^{n} c_i s_{i+k} = 0,\ k \geq 0$$

Example: Let the LFSR be of length 4 with initial state (or input sequence) – 0,1,1,0. If $c_o = c_2 = c_3 = 1$, $c_1 = 0$, then
$s_4 = c_0s_0 + c_1s_1 + c_2s_2 + c_3s_3 = 1.0 + 0.1 + 1.1 + 1.0 = 1$
$s_5 = c_0s_1 + c_1s_2 + c_2s_3 + c_3s_4 = 1.1 + 0.1 + 1.0 + 1.1 = 0$
$s_{k+4} = c_0s_k + c_1s_{k+1} + c_2s_{k+2} + c_3s_{k+3} = s_k + s_{k+2} + s_{k+3}$

Table 1: LFSR States

Time	FSR	States Output
0	0,1,1,0	1
1	1,1,0,1	0
2	1,0,1,0	0
3	0,1,0,0	0
4	1,0,0,0	1
5	0,0,0,1	1
6	0,0,1,1	0
7	0,1,0,1	1

Proposed Algorithm: This paper presents the LFSR based algorithm which is a non-dictionary based algorithm [2]. In this algorithm, we use the Run length coding and berlekamp algorithm [4]. The continuous sequences of repeated binary sequences are coded using the run-length technique. The other sequence i.e., the discrete sequence or scrambled sequences of words are taken as one continuous sequence and it is given to the berlekamp algorithm to compute the polynomial and linear complexity of the sequence. The obtained linear complexity is the minimal number of LFSR [3] required to compute or generate the sequence at the receiver end, based on the polynomial and linear complexity the minimum required number of bits is taken from the original sequence and it is given the LFSR at the receiver side and hence the original data can be retrieved.

RESULTS AND DISCUSSION

This study presents the LFSR based algorithm, which is non-dictionary based algorithm. When compare to exiting scheme like WinZip and TAR method files, it is plus or minus 5 % to 8% less compression ratio because of fixed length , while we increasing the length we can get the excellent compression ratio and also it is easy to realize in the hardware when compare to WinZip and TAR files.

Table 2: Comparison of Image and text files

S.No.	Data type	File size	Win Zip compression	LFSR Compression
1	Image	15 kb	92%	84%
2	Image	10kb	89%	80%
3	Image	1kb	90%	83%
4	Text	12kb	85%	79%
5	Text	9kb	89%	80%
6	Text	4kb	78%	69%

Further Improvements: This paper presents the compression using the fixed length of data. The compression can be achieved more by increasing the length of the sequence. The future work will be testing for the compression ration by increasing the length of the scheme.

CONCLUSION

This non-library based compression technique has an advantage of easy hardware realization. This technique uses only the library bit streams to encode the data and does not use any other codes for the encoding purpose. More enhancements have to be brought out in this kind of technique which is going to be the future image compression method.

REFERENCES

1. Gonciari, P.T., Al-Hashimi, 2003, B M., Nicolici, Variable-length input huffman coding for system-on-a-chip test. IEEE Trans. on Computer-Aided Design, 22: 783-796.
2. Jaganathan, S., P. Nagabhushan, R.K. Rajangan, R. Seshaiah, 1997. A number theory based compression. J. Spacecraft Technol., 7: 1.
3. Linear Feedback ShiftRegisters, 2002. http://www-math.cudenver.edu/ 2002 ~wcherowi/ courses/m5410/ m5410fsr.html, pp: 1– 8.
4. http://www.math.nus.edu.sg/~matlhh/UROPS/reports/ Berlekamp.DOC
5. Jovan, D.j. Golic, Renato Reniococci, 1997. Edit distance correlation attack on the alternating step generator. Advances in Cryptol., pp: 499-512.
6. Anil, K.J., 1981. Image data compression: A review. Proc. IEEE, 69: 349-389.
7. Wolff, F.G. and C. Papachristou, 2002. Multiscan based test compression and hardware decomposition using LZ77. Proc. Intl. Test Conf., pp: 331-338.
8. http://www.cs.cf.ac.uk/Dave/Multimedia/ node231.html