==========
Engineering & Technology in India www.engineeringandtechnologyinindia.com
Vol. 1:3 April 2016
==========

Amended Golomb Coding (AGC) Implementation for Reconfigurable Devices

Saranya G., M.E.
Loganya R., M.E. Student

==========

Abstract

An efficient lossless compression technique to reduce the configuration time of reconfigurable devices known as AGC (AMENDED GOLOMB CODING) is proposed. The time for loading of the configuration data from outside the chip often bottlenecks the system performance for some dynamically reconfigurable applications. Reducing the amount of configuration data with compression technique is one of the efficient approaches to improve the configuration speed. In this paper existing lossless compression technique known as Golomb Codes is amended so that compression efficiency for reconfigurable devices is still enhanced. In AGC the codeword is generated by appending the optimal prefix (amending the original Golomb's prefix), hole bit and tail in such a way that compression efficiency is enhanced.

Key words: AGC, Compression Efficiency, Golomb codes, Lossless compression, optimal prefix.

I. INTRODUCTION

Compression is a way of reducing the original bits required to transfer the source of information while preserving the original content at the decompression side for the purpose of effective transmission and enhancing the storage capacity. Compression reduces the amount of data storage space and data transmission time. Compression is performed by a program that uses a formula or algorithm to determine how to shrink the size of the data. The same program can be used at the receiver side for decompression [4].Text compression can be as simple as removing all unneeded characters, inserting a single repeat character to indicate a string of repeated characters, and substituting a smaller bit string for a frequently occurring bit string. Compression can be performed on the data content or on the entire

transmission unit, including header data. Data compression is usually of two types: lossy and lossless [11]. Lossless compression is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data. Lossless compression algorithms reduce file size with no loss in image quality. The original data is retrieved as it is at the decompression stage. This is because file is only "temporally thrown away" and not discarded hence we could read the original data without loss of information. While the advantage of this is that it maintains quality the main disadvantage is it doesn't reduce the file size as much as lossy compression. Lossy compression permits reconstruction only of an approximation of the original data, though this usually improves compression rates (and therefore reduces file sizes). Lossy compression also looks for 'redundant' pixel information, however, it permanently discards it. This means that when the file is decompressed the original data isn't retrieved. Lossy compression isn't used for data. Lossy is only effective with media elements that can still 'work' without all their original data. The Golomb code can be applied, if numbers of unknown size to be saved, but the actual application is in data compression .Golomb code can similarly efficient as the Huffman code [8] to be, but is more economical in the sense it occupies less memory, and is easier to implement and faster in execution. The proposed AGC is still effective since it enhances the compression achieved by GOLOMB and it could be compared with previous lossless compression [5].

The remainder of this paper is organized as follows. Section II gives a brief review of basic GOLOMB compression. Section III presents proposed AGC and the generation of runlength and codes based on optimal prefix and tail is discussed. In Section IV, programs were executed to verify the validity of our proposed AGC and compared with previous techniques. Finally, conclusions are drawn in Section V.

II. BASIC GOLOMB COMPRESSION

Golomb coding is lossless data compression algorithm [1] .It is a practical and powerful implementation of Run-Length Encoding of binary streams. Golomb coding algorithm contains tunable parameter M, run length N which means count of continuous number of 0"s followed by 1. In Golomb Coding, the group size, *m*, defines the code structure. In order to have simplicity in development and testing, the Golomb coding parameter *m* is set to 4.

Once the parameter *m* is decided, a table which maps the runs of zeros until the code is ended with a one is created [7]. A run length of multiples of *m* are grouped into *Ak* and given the same prefix, which is (*k* – 1) number of ones followed by a zero. The prefix is generated as per unary coding. Unary coding is an entropy encoding that represents a natural number n, with n 0"s followed by 1 if natural number is non- negative or with (n-1) 0"s followed by 1 if natural number is strictly positive. In Golomb we get unary code by (*k* – 1) number of ones followed by a zero. A tail is given for each members of the group, which is the binary representation of zero until (*m* – 1). The code word is then produced by combining the prefix and the tail.

III. PROPOSED AMENDED GOLOMB CODING (AGC) COMPRESSION FOR RECONFIGURABLE DEVICES

The proposed AMENDED GOLOMB CODING COMPRESSION makes use of the existing GOLOMB to enhance the compression efficiency still. The first step in AGC is to create a table which maps the runs of zero until the code is ended with one is created. In AGC, the group size, M, defines the code structure. Thus, choosing the m parameter decides variable length code structure which will have direct impact on the compression efficiency [7]. For simplicity M is chosen as 4.Determination of the run length is shown as in Fig.1. A run length of multiples of M are grouped into Ak and given the same prefix as per the algorithm. A tail is given for each members of the group, which is the binary representation of zero until (M – 1). The codeword is then produced by combining the prefix and the tail. AGC will modify the prefix optimally of basic Golomb in such a way that it still enhances compression. Therefore in AGC coding the code word would be generated by joining optimal prefix and tail.

Data	01 0000001 0001 000001 0011					
Subset	01	000000	000	00000	001	1
Run-Length	1	6	3	5	2	0

Fig. 1. Determination of Runlength

Generation of optimal prefix and tail is based on the algorithm given below. The codeword is obtained by joining optimal prefix and tail. Based on the algorithm and M value the code word could be generated. The main advantage of AGC is that we need not to keep track of table or tree as of existing lossless techniques [5].That is it is easier way to implement. The algorithm of AGC is given below:

Golomb coding is implemented using following 3 steps.

1. Fix the parameter *M* to an integer value.
2. For *N*, the number to be encoded, find a. quotient = *q* = int[*N*/*M*]

   b. remainder = *r* = *N* modulo *M*
3. Generate Codeword

   a. The Code format :< hole bit> <Quotient Code>

<Remainder Code>, where

b. Quotient Code (optimal group prefix in unary coding)

i. If 0 □ q □ 1 assign prefix as „q".

ii. If 2 □ q □ 5 assign prefix as binary representation of q-2 in 2-bit representation.

iii.If 6 □ q □13 assign prefix as binary representation of q-6 in 3-bit representation and so on..

c. Remainder Code (in truncated binary encoding) (tail)

i. If M is power of 2, code remainder as binary format.

So $\log_2$ (M) bits are needed. (Rice code).

ii. If M is not a power of 2, set b= [$\log_2$ (M)]

a .If r < 2^b-M Code r as plain binary using *b*-1 bits.

b. If r > 2^b-M Code the number r + 2^b-M in plain binary representation using b bits.

Using Fig. 2, binary strings can be divided into subsets of binary strings. Based on the above algorithm and the run Length from Fig.1 optimal prefix and tail is generated. Codeword is generated by appending optimal prefix and tail. In original Golomb prefix generation would be

as follows:

For M and runlength N find quotient q where

q = int [N/M].

In codeword Quotient Code would be: Quotient Code (in unary coding)

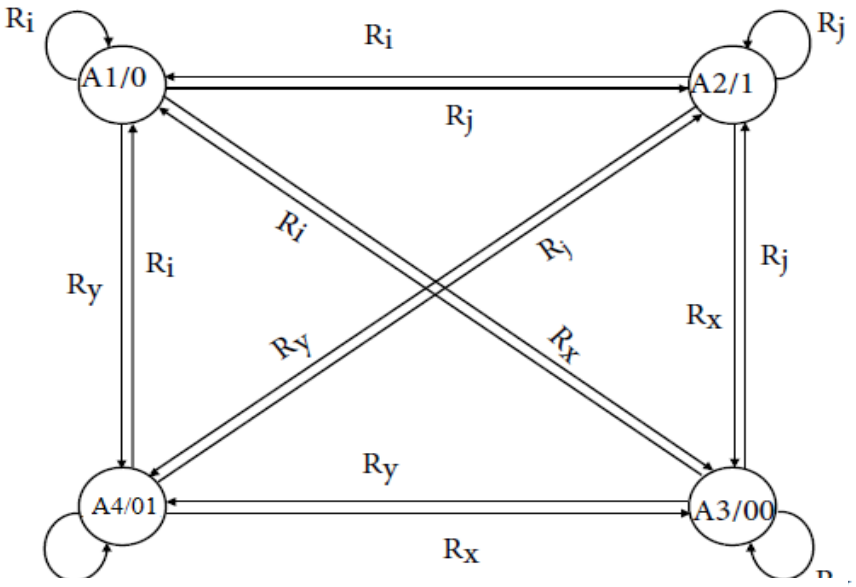
i. Write a q-length string of 1 bits ii. Write a 0 bit

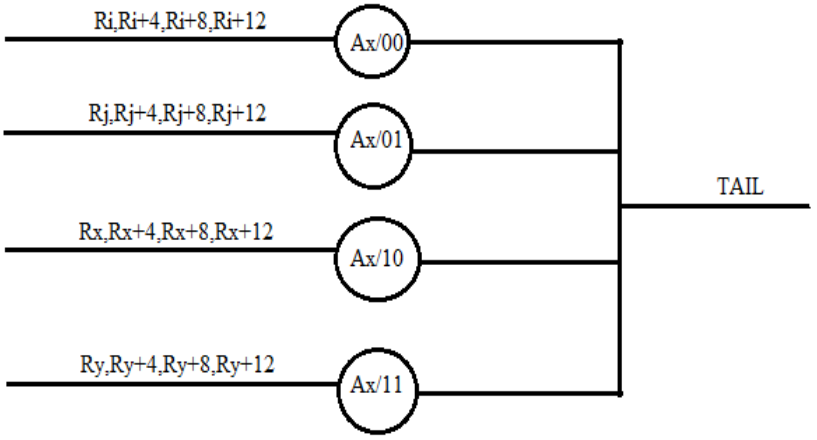
In AGC optimal prefix still compresses the bits so that compression is still increased. Substrings are generated by dividing the original strings until 1 is encountered. Binary strings can be divided into subsets of binary strings and replacing the subsets with the equivalent code word as shown in Fig.2.

Group	Run-length	Holebit	Group prefix	Tail	Codeword
A1	0	1	0	00	000
	1	1		01	001
	2	1		10	010
	3	1		11	011
A2	4	1	1	00	100
	5	1		01	101
	6	1		10	110
	7	1		11	111
A3	8	1	00	00	0000
	9	1		01	0001
	10	1		10	0010
	11	1		11	0011
A4	12	1	01	00	0100
	13	1		01	0101
	14	1		10	0110
	15	1		11	0111

Fig. 2. Amended Golomb coding example with parameter M= 4

A hole bit is appended before the optimal prefix which acts as the distinction between the code word which is discarded at the decoder stage. Generation of prefix is stated in state diagram Fig.3.When Runlength is between 0- 3 prefix is generated as „0"and the group is A1.As long as the runlength is 0-3 it remains in the same state A1 and when Runlength 4-7 is encountered it goes to state A2 that is prefix „1"is generated and when Runlength is 8-11 it goes to state A3 and prefix „00" is generated. When Runlength is 12-15 it goes to state A4 and prefix „01"is generated and so on. Whatever may be the present state when runlength changes out of that particular group it changes its state and corresponding optimal prefix is generated.





s generated (refer
ich is the binary
ıced by combining

Fig. 4.Tail generation for M=4 (here Ax-Group=1 or 2 or 3 or 4

Ri,Rj,Rx,Ry-Run Length.wherei=0,j=1,x=2,y=3)Whatever may be the group the tail is generated as per the remainder of N/M. In Fig.4 for i, the tail is „00" as remainder is 0.For j, the remainder is 1 so the tail is „01".As the remainder is 2 the tail is „10" for x. For y the tail is„11"because the remainder is 3.The code word is generated by appending the hole bit, optimal prefix and tail. And at the decompression stage hole bit which act as distinction is now isolated and discarded and the original data is retrieved as it is. Hence become a lossless compression. The proposed AGC still reduces the bits as compared to the existing GOLOMB codes.

IV. RESULTS

The proposed AGC is simulated and compared with expected output and is shown in Fig.5.It is shown that proposed AGC reduces the bits effectively. Compression efficiency is calculated as below:

$$\text{Compression efficiency} = \frac{\text{Uncompressed bits}}{\text{Compressed bits}} * 100$$

INPUT	00010000	00101010	00100000	00000110	00001000
	10101011	00010000	01010100	00000001	00010001
	00000000	00000001			

INPUT SUBSET	**RUN LENGTH**	**EXPECTED AGC OUTPUT**
0001	3	011
0000001	6	1010
01	1	001
01	1	001
0001	3	011

00000000001	10	11010
1	0	000
000001	5	1001
0001	3	011
01	1	001
01	1	001
01	1	001
1	0	000
0001	3	011
000001	5	1001
01	1	001
01	1	001
0000000001	9	11001
0001	3	011
0001	3	011
00000000 00000001	15	0111
OUTPUT	01110100 01001011 11010000 10010110 01001001 00001110 01001001 11001011 0110111	

Fig. 5.Expected output of simulation

The compression efficiency is calculated for the proposed AGC and compared with existing lossless compression techniques Bitmasking (BM), Runlength encoding (RLE) Dictionary (DIC) and old Golomb and is compared in Fig 6. Length before and after compression is indicated as LB and LA respectively.Table1gives the comparison of compression efficiency of various techniques. It is observed that AGC compresses the data more effectively than other techniques.

V.CONCLUSION

In this paper, AGC, An AMENDED GOLOMB CODING (AGC) algorithm a lower complex and less computational load method for data compression and decompression has been proposed.Thus AGC codes are simulated and the result is compared with existing Golomb codes and is proven that Golomb codes are enhanced in the proposed AGC.Amended Golomb coding is highly suitable for situations in which the occurrence of small values in the input stream is significantly more likely than large .

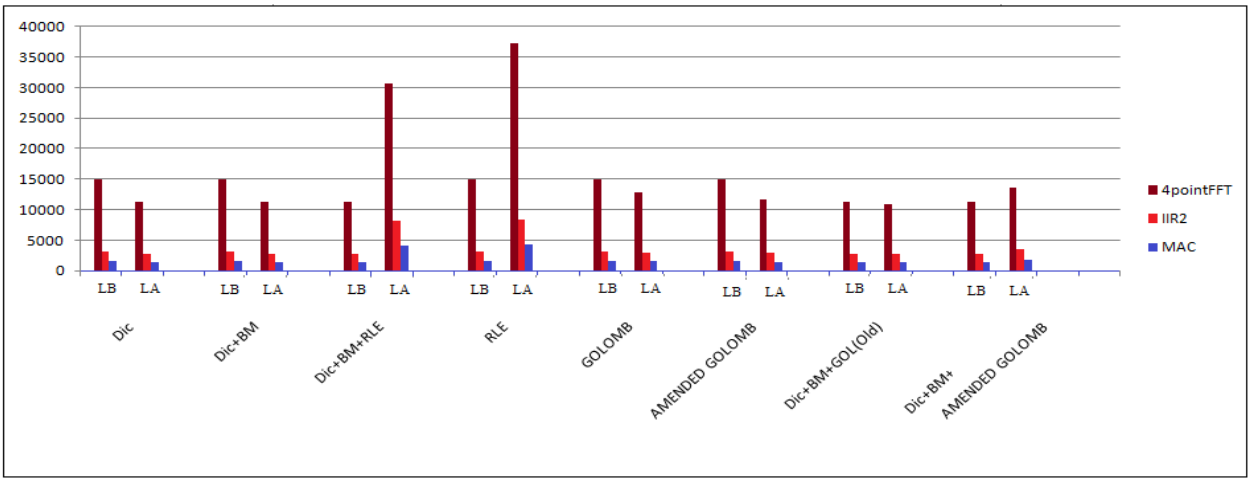


Fig. 6. Comparison of AGC with existing systems

TABLE.1. COMPARISION OF COMPRESSION EFFICIENCY OF VARIOUS TECHNIQUES IN PERCENTAGE

	DIC	DIC+BM	DIC+BM + RLE	RLE	OLD GOLOMB	AMENDED GOLOMB	DIC+BM + GOL(OL D)	DIC+BM+ AMENDED GOLOMB
4POINT FFT	1.32	1.33	0.36	0.40	1.17	1.21	1.02	0.82
IIR2	1.16	1.17	0.32	0.36	1.02	1.03	0.95	0.74
MAC	1.06	1.11	0.32	0.33	0.96	1.13	0.95	0.74

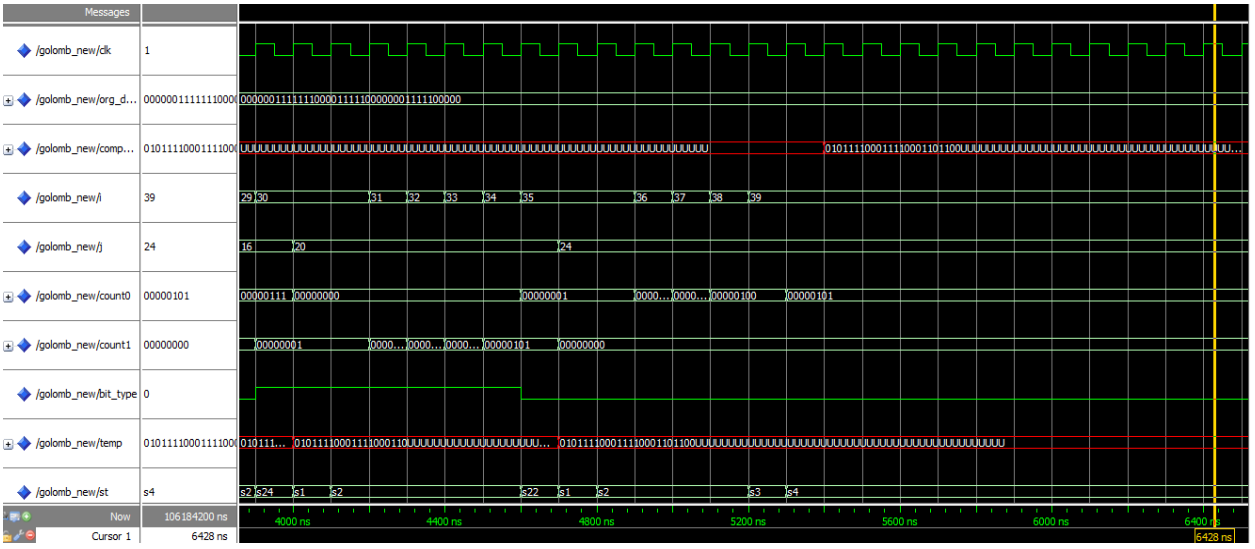


Fig. 7. Simulation output

==========

REFERENCES

[1] S. W. Golomb, "Run Length Encodings," IEEE Transactions on Information Theory, vol. 12, pp. 399-401, 1966.

[2] Chandra and K. Chakrabarty, "System on-a-chip test data compression and decompression architectures based on Golomb codes," IEEE Trans. Computer-Aided Design, vol. 20, pp.355-368, Mar. 2001.

[3] Wang S.J. and Chiou S.N. (2001), „Generating efficient tests for continuous scan", in Proceedings of Design Automation Conference, PP. 162-165.

[4] M. Ghanbari, Video Coding an Introduction to Standard Codecs. London: The Institution of Electrical Engineering, UK, 2003.

[5] T. Sikora, "Trends and perspectives in image and video coding," Proceedings of IEEE, vol. 93, no. 1, pp. 6-17, Jan 2005.

[6] T. Silva, et. al.," FPGA based design of CAVLC and Exp-Golomb coders for H.264/AVC baseline entropy coding," Proc 3rd IEEE Southern Conference on Programmable Logic, pp. 161-166, Feb 2007.

[7] G. H. H"ng, M. F. M. Salleh and Z. A. Halim,"Golomb Coding Implementation in FPGA", Faculty of Electrical Engineering Universiti Teknologi Malaysia VOL. 10, NO. 2, 2008, 36-40.

[8] F. Marcelloni and M. Vecchio. 2009. "An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks," Computer Journal, Vol. 52, No.8, pp.969 – 987.

[9] Tharini and P. Vanaja Ranjan. 2009. "Design of modified adaptive Huffman data compression algorithm for wireless sensor network," Journal of Computer Science, Vol. 5, No. 6, pp. 466– 470.

[10] Ayari, N., LIP2 LabTunis, Tunisia ; Thé Van Luong " hybrid genetic algorithm for Golomb ruler problem",Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference,pp.1 - 4 ,May 2010.

==========

Saranya G.
Assistant Professor
Saranyagr80@gmail.com

Loganya R., M.E. Student
Loganya.bharani@gmail.com

Department of Electronics & Communication Engineering
Sri Subramanya College of Engineering & Technology
NH - 209, Sukkamanaickenpatti
Palani 624615
Tamil Nadu
India