A Modified GOLUMB Encoder and Decoder for Test Vector Compression

Nicky H. Bellani
Research Scholar: Dept. of Electronics Engineering
G. H. Raisoni College of Engineering, Digdoh Hills, Nagpur, India 440016.

Payal Ghutke
Professor: Dept. of Electronics Engineering
G. H. Raisoni College of Engineering, Digdoh Hills, Nagpur, India 440016.

ABSTRACT
This paper describes the detailed study and analysis of Golumb codes used for the test vector compression in VLSI testing. The Golumb Codes are widely used for lossless Data compression due to its lower complexity in encoding & Decoding methods. Furthermore, a comparative study of various compression techniques is also presented in this paper. This paper also gives out an idea about the design of Golumb Encoder & Decoder using VHDL for the test vectors thus achieving a good height of compression ratio. In order to prove its validity, the developed algorithm is simulated using the Xilinx 9.2i and MATLAB software.

KEYWORDS
Golumb, Encoder, Decoder and Compression

1. INTRODUCTION
The rapid widespread growth of digital technologies such as digital television, internet access, huge amount of data storage and video calls have increased the demand for high storage data and transmission capacity in order to fit the growing needs[1]. Data compression involves encoding information using fewer bits than the original representation. Compression of data in large-sized files reduces storage space and transmission time in a network and hence reduces the cost of storage and transmission. Most of the files has lot of redundancy which can be removed using Compression.

Compression can be achieved through a lossy or lossless mechanism. The selection of compression method depends on the application. Lossy compression of a tolerable limit can be used for video transmission, where the loss of data in the reconstruction cannot be easily noticed by human visual system. However, compression of data of crucial information such as the database of a bank needs to be lossless. There have been extensive research efforts in this field since the last 50 years.

Golomb coding is a lossless data compression method using a family of data compression codes invented by Solomon W. Golomb in the 1960s. Alphabets following a geometric distribution will have a Golomb code as an optimal prefix code, making Golomb coding highly suitable for situations in which the occurrence of small values in the input stream is significantly more likely than large values.

It is competent of compressing larger sized data into a smaller sized data while still allowing the original data to be reconstructed back after decompression[1]. Due to its lower design complexity and computational load, Golumb codes are more preferred over other lossless data compression codes.

Exp-Golomb VLC for entropy coding in H.264, which is mostly used for video coding standard is the most efficient application of golumb code. Golumb codes are also used for colour image [4], [6] and audio compression. Other application of golumb codes is for embedded cores in a system-on-a-chip (SOC) [7]. The key features of Golomb coding for test data include very elevated compression, analytically predictable compression results, and a stumpy cost and scalable on-chip decoder.

In this paper, a detail study of Golomb coding algorithms for test vector compression and decompression is presented. In order to have simplicity in development and testing, the Golomb coding parameter m is set to 4 [2].The goal of this work was to increase the compression ratio as high as possible without any loss in the original data.

The remainder of the paper is organised as follows. Section 1 presents the details of Golomb Coding and the basic compression and decompression method. Section 2 presents the review of past work carried out on golumb codes. Section 3 presents the future scope in the field of Golumb Codes. Lastly, Section 4 concludes the paper.

2. BACKGROUND OF GOLUMB CODES

2.1 Golumb Encoder and Decoder Algorithms

The details regarding Golomb Coding basic background information is described. Golomb coding uses a tunable parameter *m* to divide an input value into two parts: q, the result of a division by *m*, and r, the remainder. The quotientsent in unary coding, followed by the remainder in truncated binary encoding. When m=1, Golomb coding is equivalent to unary coding. In Golomb Coding, 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 [2].

After finalization of parameter m, a table which maps the runs of zeros or ones is created. A Run length of multiples of *m* are grouped into *Ak* and given the same prefix, which is (k – 1) number of one's followed by a zero, which can also termed as quotient and can be represented in the form of unary codes. A tail is given for each member of the group, which is the binary representation of $\log_2 m$ bits. The other term for tail is Remainder of the division of run length by m. The codeword is then produced by combining the prefix and the tail. An example of the analytically encoding the data stream using previous Golumb Encoder is given in table 2.

The simple golomb Encoder and Decoder models can be well explained with the help of flowcharts given in figure 1 and figure 2 [2]. The design of the algorithm is made with the assumption that the input data string will be terminated with a "1". This will not always be the case because the input data string may also be terminated with a "0".To overcome this problem, a modified version of Golumb Compression and decompression methods are used.

In order to avoid this problem, the algorithm must be capable of detecting the end of data and if the last bit is a '0' then additional '1' must be added during the encoding process and at the time of decompressing the encoded data, this extra appended 1 should be removed by the decoder.

2.2 ALGORITHM

1. Fix the parameter *M* to an integer value.

2. For *N*, the number to be encoded, find
   i. quotient = q = int[N/m]
   ii. remainder = $r = N$ modulo m

3. Generate Codeword

The Code format : <Quotient Code><Remainder Code>, Where

a. Quotient Code (in unary coding)

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

b. Remainder Code (in truncated binary encoding)

I. If M is power of 2, code remainder as binary format. So $\log_2(m)$ bits are needed.

II. If M is not a power of 2, set b= $\log_2(m)$.

i. If $r < 2^b - m$ code r as plain binary using b-1 bits.

ii. If r>= 2b - m code the number r + 2b–m in plain binary representation using b bits.

Table 1. Golomb Encoding example with parameter *m*

Group	Run-length	Group prefix (Quotient)	Tail (Remainder)	Codeword (Pefix+tail)
A1	0	0	00	000
	1		01	001
	2		10	010
	3		11	011
A2	4	10	00	1000
	5		01	1001
	6		10	1010
	7		11	1011
A3	8	110	00	11000
	9		01	11001
	10		10	11010
	11		11	11011

Table 2. Golomb Encoding example with parameter *m* = 4

Data Set	00100001000001000000100000101000101						
Subset	001	000 010	000 001	0000 001	0000 01	00 01	01
Encoded Output	010	100 0	100 1	1010	1001	01 1	00 1

3. Modified Golumb Encoder

In Simple Golumb codes, the data is divided into subsets which maps the runs of zeros until the code is ended with a one[2]. In future work of Golumb coding, a modified Golumb Coder is designed to serve as a good application for the test vector Compression. In that case, the input bit stream is divided into two subsets in which first subset will contain the test vectors which maps the runs of zeros and the second subset will contain the vectors which maps the runs of one's. The drawback of basic Golumb Encoder can be easily removed with the help of Modified Golumb Encoder. As there is no need of terminating a stream of zero's by one or stream of one's by zero or appending a zero or one in modified Golumb Coder. For first subset, the data is divided by the divisor m and in the second subset, the data is divided by the divisor n. For lower error probability, the group size m and n for both the subsets is kept equal. The optimum value of m and n is set to 4. The first bit of Input is copied as it is in the encoded output for better decoding purpose. An example of the analytically encoding the data stream using previous Golumb Encoder is given in table 3 and table 4.

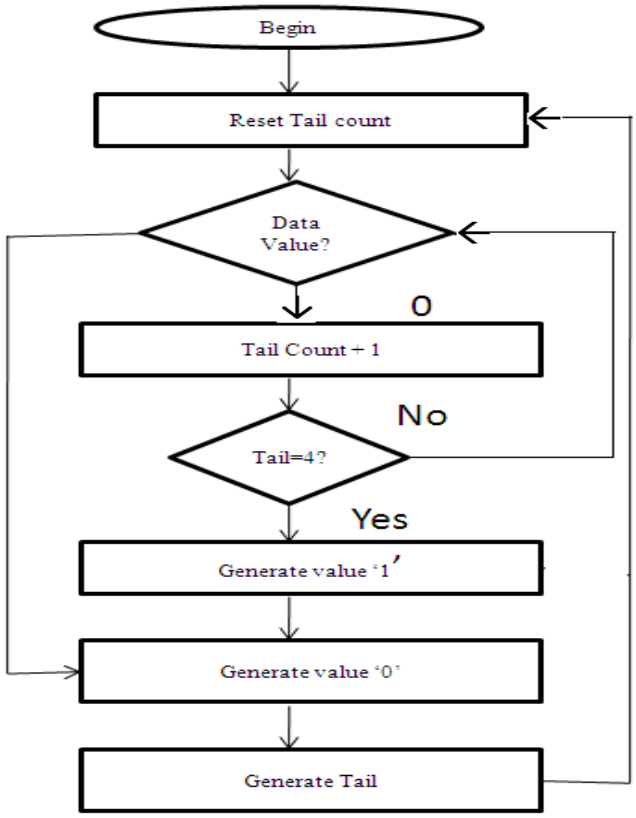

Figure 1. Golomb Encoder Flowchart with parameter *m* =4

Table 3. Modified Golomb Encoding example with parameter *m* = 4 & n=4

Data Set	11110000	
Subset	1111	0000
Encoded Output	1 0100	0100

Figure 3 gives the MATLAB simulation results for the data stream of 11110000. The Golumb Coded Ouput stream gives the same result as obtained by manual encoding of data stream.

The given data stream is 111100011111111. Initially the data is divided into subsets of zero's & one's. Thus by applying the Golumb Algorithm, the encoded output is obtained as shown in table 4. First subset contains 1111 which gives the runlength of 4. As m parameter is already set to 4,so when we divide the runlength of 4 by m. the quotient which we get is 1 and remainder is 0. The quotient can be written in unary code of "10" i.e 1 followed by number of zeros and the remainder in $\log_2$m bits i.e "00". The same algorithm can be applied to remaining subsets to get the encoded output.

The first bit in red is used for decoding purpose. The first bit of dataset should be same as the first bit of encoded output. Figure 5 gives the simulation results of table 4 data stream using Xilinx. The 15 bit input stream takes the input of 111100011111111. The output stream gives the 13 bit compressed output of 15 bit Input. The upper two bits of output are undefined. The Done signal will give the output 1 as soon as the encoding process is completed.

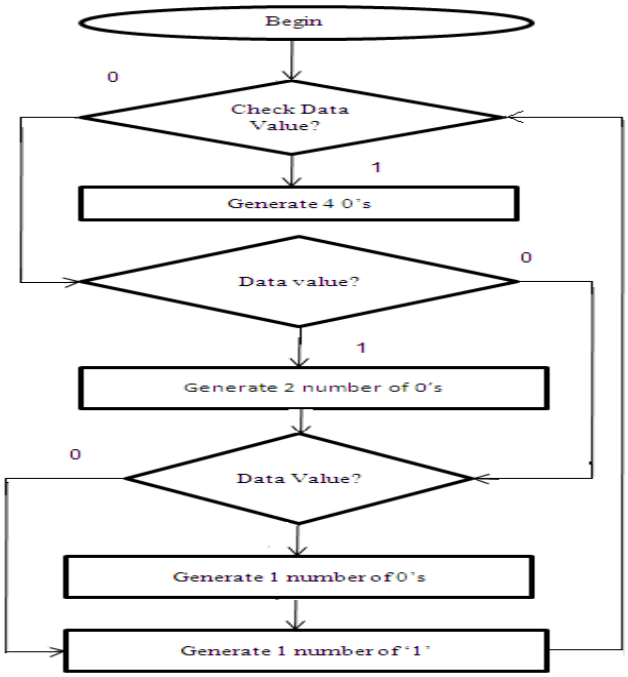

Figure 2. Golomb Decoder Flowchart with parameter *m* = 4

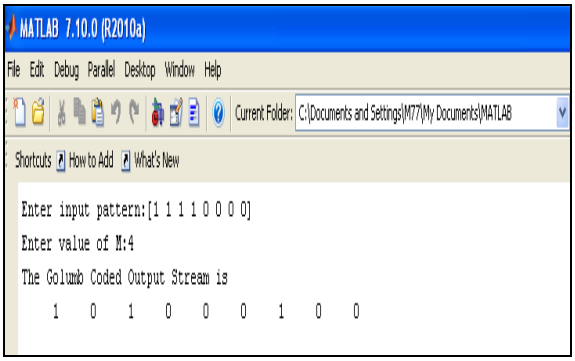

Figure 3. Simulation Waveform of Golumb Encoder using MATLAB

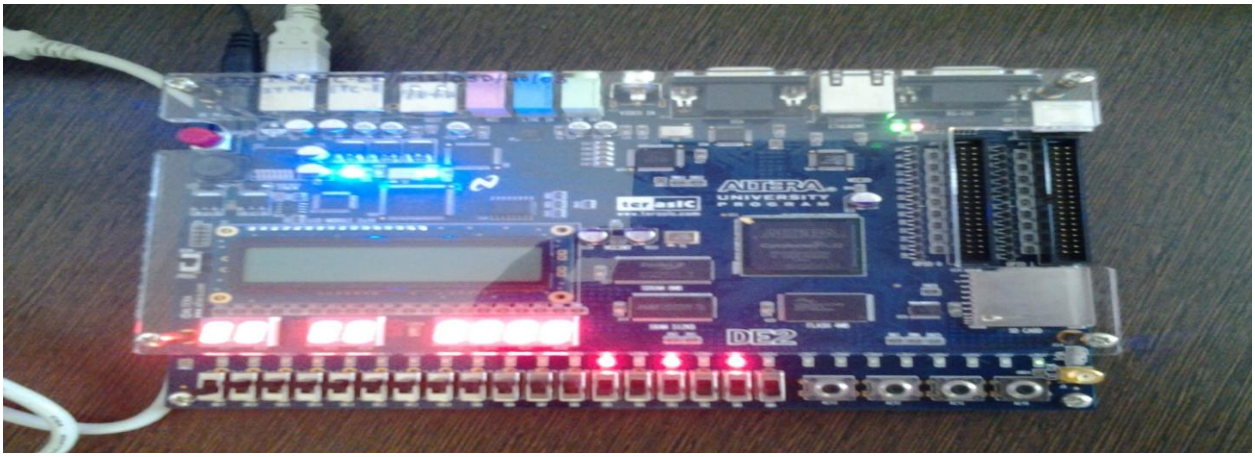

Figure4. FPGA Implementation of Golumb Encoder using Quartus

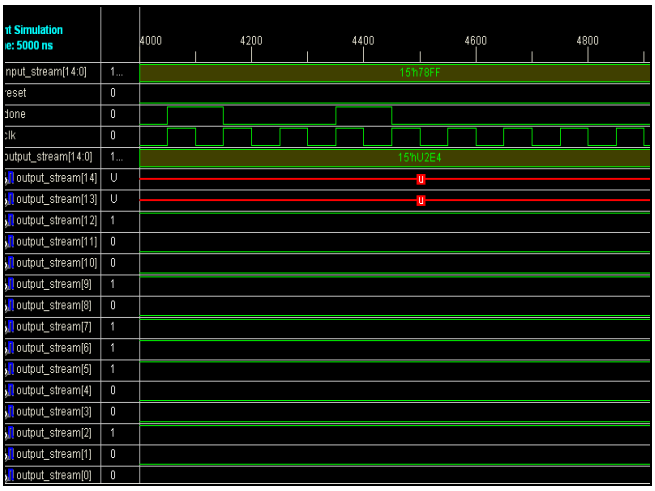

Figure 5. Simulation Result of Golumb Encoder using Xilinx

Table 4. Modified Golomb Encoding example with parameter m = 4 & n=4

Data Set	111100011111111		
Subset	1111	000	11111111
Encoded Output	1 0010	111	00100

Figure 4 gives the FPGA Implementation of Golumb Encoder The simulation part is carried on Quartus Software. The input stream of 1111000000 is given on the switches and the compressed output can be shown on the LED's. The compressed output is 000101010.

4. CONCLUSION

In this paper, algorithms of Golumb encoder and decoder for test vector compression has been studied. The successful working of these algorithms can be proved by comparing the results generated using simulation of the Golomb Encoder with the expected analytical results which should be identical. The use of encoded data as the input data for Golomb Decoder managed to generate back the original data can prove the success of Golomb Decoder System.

REFERENCES

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

[2] G. H. H'ng, M. F. M. Salleh and Z. A. Halim," Golomb Coding Implementation in FPGA", School of Electrical and Electronics Engineering, Universiti Sains Malaysia,Seri Ampangan,14300 Nibon Tebal, Pulau Pinag, Malaysia. VOL. 10, NO. 2, 2008, 36-40.

[3] Walter D. Leon-Salas, Sina Balkir, Khalid Sayood, and Michael W. Hoffman, "An Analog-to-Digital Converter with Golomb–Rice Output Codes" IEEE transactions on circuits and systems—ii: express briefs, vol. 53, no. 4. APRIL 2006.

[4] Hong-Sik Kim, Joohong Lee, Hyunjin Kim, Sungh Kang, and Woo Chan Park, A Lossless Color Image Compression Architecture using a Parallel Golomb-Rice Hardware CODEC, IEEE transactions on circuits and systems for video Technology, vol. 21, no. 11, November 2011.

[5] K. Somasundaram and S. Domnic, "Extended Golomb Code for Integer Representation", IEEE transactions on multimedia, VOL. 9, NO. 2, FEBRUARY 2007**.**

[6] Chin-Chen Chang, "An Enhancement of JPEG Still Image Compression with Adaptive Linear Regression and Golomb-Rice coding, 2009 Ninth International Conference on Hybrid Intelligent Systems.

[7] Anshuman Chandra, Student Member, IEEE, and Krishnendu Chakrabarty, Senior Member, IEE " System-on-a-Chip Test-Data Compression and Decompression Architectures Based on Golomb Codes",IEEE transactions on computer aided design of integrated circuits and systems, VOL. 20, NO. 3, MARCH 2001.

[8] Henrique S. Malvar, "Lossless and Near-Lossless Audio Compression Using Integer- Reversible Modulated

Lapped Transforms, 2007 Data Compression Conference.

[9] X. Chen, N. Canagarajah, J. L. Nunez-Yanez, and R.Vitulli, "Hardware architecture for lossless image compression based on context-based modeling and arithmetic coding," in *Proc. IEEE Int. SoC Conf.*, Sep. 2007, pp. 251–254.

[10] T.-H. Tsai, Y.-H. Lee, and Y.-Y. Lee, "Design and analysis of high throughput lossless image compression engine using VLSI-oriented FELICS algorithm," *IEEE Trans. Very Large Scale Integr. Syst.*, vol.18, no. 1, pp. 39–52, Jan. 2010.

[11] L. Xiaowen, X. Chen, X. Xie, G. Li, L. Zhang, C. Zhang, and Z. Wang,"A low power, fully pipelined JPEG-LS encoder for lossless image compression," in *Proc. IEEE Int. Conf. Multimedia EXPO*, Jul. 2007,pp. 1906–1909.