

# TEST DATA COMPRESSION BASED ON GOLOMB CODING AND TWO-VALUE GOLOMB CODING

Priyanka Kalode<sup>1</sup> and Mrs. Richa Khandelwal<sup>2</sup>

<sup>1</sup>Department of Electronics Engineering, Ramdeobaba college of Engg and Mgt, Nagpur  
kalode.priyanka@gmail.com

<sup>2</sup>Department of Electronics Engineering, Ramdeobaba college of Engg and Mgt, Nagpur  
richareema@rediffmail.com

## **ABSTRACT:**

*In this paper, we discuss test data compression and decompression method based on variable length Golomb codes and 2-V Golomb Codes for test data. The method is targeted to minimize the amount of test data, which reduces the size of memory required in ATE for test data and also time required to transfer test data to specific device on SOC. We completed MATLAB coding for both methods and applied test vectors of some standard ISCAS benchmark circuits and compared results for same. Experimental results on ISCAS benchmark circuits show that the compressed data produced by 2-V Golomb coding is better than Golomb Coding method.*

## **KEYWORDS:**

*Automatic test equipment (ATE), precomputed test sets, variable-to-variable-length codes, Golomb coding, RLE, SOC, Golomb Coding, 2-V Golomb Coding.*

## **1. INTRODUCTION**

Embedded cores are becoming commonplace in large system-on-a-chip (SOC) designs [1]. Along with the benefits of higher integration and shorter time to market, intellectual property (IP) cores pose several difficult test challenges. The volume of test data for an SOC is growing rapidly as IP cores become more complex and an increasing number of these cores are integrated in a chip. The volume of test data for an SOC is growing rapidly as IP cores become more complex and an increasing number of these cores are integrated in a chip. In order to effectively test these systems, each core must be adequately exercised with a set of precomputed test patterns provided by the core vendor. However, the input/output (I/O) channel capacity, speed and accuracy, and data memory of automatic test equipment (ATE) are severely limited.

The testing time for an SOC depends on the test data volume, the time required to transfer the data to the cores, and the rate at which it is transferred (measured by the cores test data bandwidth and ATE channel capacity). Lower testing time increases production capacity as well as reduces

test cost and time to market for an SOC. New techniques are therefore needed for decreasing test data volume in order to overcome memory bottlenecks and to reduce testing time.

An attractive approach for reducing test data volume for SOCs is based on the use of data compression techniques [2]–[4]. In this approach, the precomputed test set  $T_D$  provided by the core-vendor is compressed (encoded) to a much smaller test set  $T_E$  and stored in the ATE memory. An on-chip decoder is used for pattern decompression to generate  $T_D$  from  $T_E$  during pattern application. Test data compression using statistical coding of test sequences for synchronous sequential (nonscan) circuits was presented in [2] and for full-scan circuits in [3]. While the compression method in [2] is restricted to sequential circuits with a large number of flip-flops and relatively few primary inputs, the work presented in [3] does not conclusively demonstrate that statistical coding provides greater compression than standard ATPG compaction methods for full-scan circuits [5], [6].

Test data can be more efficiently compressed by taking advantage of the fact that the number of bits changing between successive test patterns in a test sequence is generally very small. This observation was used in [4], where a “difference vector” sequence  $T_{diff}$  determined from TD was compressed using run-length coding. A drawback of the compression method described in [14] is that it relies on variable-to fixed- length codes, which are less efficient than more general variable-to-variable-length codes [1], [9]. Furthermore, it is inefficient for cores with internal scan chains that are used to capture test responses; in these circuits, separate CSRs must be added to the SOC, thereby increasing hardware overhead. A more efficient compression and decompression method was used in [6], where  $T_{diff}$  was compressed using Variable-to-variable-length Golomb codes. However, this approach requires separate CSRs and is therefore also inefficient for cores that use the same internal scan chains for applying test patterns and capturing test responses.

The proposed compression approach for reducing test data volume is especially suitable for system-on-a-chip containing IP cores since it does not require gate-level models for the embedded cores. Precomputed test sets can be directly encoded without any fault simulation or subsequent test generation. This is in contrast to other recent techniques, such as LFSR-based reseeding for BIST [17] and scan broadcast [16], which require structural models for fault simulation and test generation.

The compression approach [2] for reducing test data volume is especially suitable for system-on-a-chip containing IP cores since it does not require gate-level models for the embedded cores. Precomputed test sets can be directly encoded without any fault simulation or subsequent test generation. This is in contrast to other recent techniques, such as LFSR-based reseeding for BIST [7] and scan broadcast [3], which require structural models for fault simulation and test generation. The mixed-mode BIST technique in [14] relies on fault simulation for identifying hard faults and test generation to determine test cubes for these faults. The scan broadcast technique in [15] also requires test generation.

In this companion paper to [18], we found [13] an improved test data compression and decompression method for IP cores in an SOC. The proposed approach makes effective use of Golomb codes and the internal scan chain(s) of the core under test. No separate CSR is required for pattern decompression. The difference sequence  $T_{diff}$  is derived from the given precomputed test set  $T_D$  using the fault-free responses  $R$  of the core under test to  $T_D$ . Golomb coding is then applied to  $T_{diff}$ . The resulting encoded test set  $T_E$  is much smaller than the original precomputed

test set  $T_D$ . We apply our compression approach to test sets for the ISCAS 89 benchmark circuits and show that  $T_E$  is not only considerably smaller than the smallest test sets obtained using ATPG compaction [15], but it is also significantly smaller than the compressed test sets obtained using Golomb coding in [16].

We implemented first original Golomb coding algorithm and obtained results for different test patterns. In that original algorithm we found that compression ratio is very less and also many times getting larger data than compressed one. So we modified algorithm and implemented 2-V Golomb coding which is verified for different test vectors, we found that compression ratio is much higher than original algorithm.

## 2. GOLOMB CODING ALGORITHM

This section covers the details regarding golomb coding algorithm. Golomb coding is lossless data compression algorithm. It is a practical and powerful implementation of Run-Length Encoding of binary streams

Golomb coding algorithm contains tunable parameter  $M$ , run length  $N$  means count of continuous number of 0's followed by 1.

Golomb coding is implemented using following 3 steps.

1. Fix parameter  $M$  to an integer value.
2. For  $N$ , run length to be encoded, find  
Quotient,  $q = \text{int}[N/M]$   
Remainder,  $r = N \text{ modulo } M$
3. Codeword generation:  
Code format = <Quotient code><remainder code>

Where,

Quotient code: Quotient is represented in unary coding.

In this we get unary code by representing  $q$  strings of 0's followed by 1.

Remainder code: Remainder is represented in truncated binary code.

Remainder code: Remainder is represented in truncated binary code.

If  $M$  is power of 2 then code remainder as binary format using  $\log_2 M$  bits.

If  $M$  is not a power of 2, set  $b = \lceil \log_2(M) \rceil$  If  $r < 2b - M$  code  $r$  as plain binary using  $b-1$  bits.

If  $r \geq 2b - M$  code the number  $(r + 2b - M)$  in binary representation using  $b$  bits.

We chosen first case of remainder code for our implementation and which is special case of golomb coding known as Golomb Rice algorithm. Since the Golomb-Rice algorithm uses only a Power-of-two divider, the remainder and quotient can be calculated easily without complex hardware.

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 this we have considered the natural number is non-negative.

Table 1 unary coding

$n$ (non-negative)	$n$ (strictly positive)	Unary code	Alternative
0	1	0	1
1	2	10	01
2	3	110	001
3	4	1110	0001
4	5	11110	00001
5	6	111110	000001
6	7	1111110	0000001
7	8	11111110	00000001
8	9	111111110	000000001
9	10	1111111110	0000000001

Truncated binary coding:

In order to have simplicity in development and testing, the Golomb coding parameter  $M$  is set to 4. We simply represent the remainder in binary form using  $\log_2 M$  bits i.e  $\log_2(4) = 2$  bits.

Table 2. Truncated binary coding

Remainder (r)	Golomb parameter (M)	$\log_2(M)$	Truncated binary code
0	2	1	0
1	4	2	01
2	8	3	010
3	16	4	0011

Table 3. An example of golomb coding for  $M=4$ .

Group	Run length	Group prefix	Tail	Codeword
A1	0	0	00	000
	1		01	001
	2		10	010
	3		11	011
A2	4	01	00	0100
	5		01	0101
	6		10	0110
	7		11	0111
A3	8	001	00	00100
	9		01	00101
	10		10	00110
	11		11	00111
...	...	...	...	...

The first step in the encoding procedure is to select the Golomb code parameter  $M$ . The value of  $M$  is taken in which the compression ratio highest. Once the group size  $M$  is determined, the runs of zeros in precomputed test set are mapped to groups of size  $M$  (each group corresponding to a run length). The number of such groups is determined by the length of the longest run of zeros in the precomputed test set. The set of run lengths  $\{0, 1, 2 \dots, m-1\}$  forms group  $A_1$ ; the set  $\{m, m+1, m+2 \dots, 2m-1\}$ , group  $A_2$ ; etc. In general, the set of run lengths  $\{(k-1)m, (k-1)m+1, (k-1)m+2 \dots, km-1\}$  comprises group  $A_k$  [1]. To each group  $A_k$ , we assign a group prefix of  $(k-1)$  1s followed by a zero. We denote this by  $0^{(k-1)}1$ . If  $M$  is chosen to be a power of two, i.e.,  $M = 2N$ , each group contains  $2N$  members and a  $\log_2 2M$ -bit sequence (tail) uniquely identifies each member within the group. Thus, the final code word for a run length  $L$  that belongs to group  $A_k$  is composed of two parts, a group prefix and a tail. The prefix is  $0^{(k-1)}1$  and the tail is a sequence of  $\log_2 2M$  bits. It can be easily shown that  $(k-1) = (N \bmod M)$  i.e.,  $k = (N \bmod M) + 1$ . The encoding process is illustrated in table 3 for  $M = 4$ .

Most of the research results on test data compression focus on reducing the amount of encoded test data ( $T_E$ ). However, this approach may actually lead to a larger volume of applied test data ( $T_D$ ), and thus increases test time [9]. In order to reduce both  $T_D$  and  $T_E$ , the proposed scheme starts from a compact test set, in which the number of test vectors is near minimum.

There are two possible types of test set: either every bit in a test vector is fully specified, or some bit are not specified. A test vector with unspecified bits is usually referred to as a test cube. If the initial test set is not fully specified, it will be easier to compress. However, the size of recovered test data  $T_D$  will be larger, and thus the test application time is longer. An example of test set with test cubes is shown in Fig.1. Since the don't care bits in test cubes can be randomly assigned to 0 or 1, it is possible to produce longer runs of 0's and 1's. As a result, the compression rate can be greatly increased.

We consider example of fully specified test vector for this method which is shown in following fig 1.

Fig.1. fully specified test vectors.

```

0001000
0011000
0100001
0000001
0010000
0001001
    
```

Table 4. Golomb coding algorithm.

<b><math>T_D</math></b>	0001	000001	1	00001	00001	0000001	001	00000001	001
<b>Run length</b>	N1=8	N2=11	N3=4	N4=14	N5=8	N6=5	N7=1	N8=8	N9=0
<b><math>T_E</math></b>	00100	00111	0100	000110	00100	0101	101	00100	100

### 3. TWO-VALUE GOLOMB CODING ALGORITHM

This method is same as Golomb Coding with difference that we calculate run length as count of continuous 0's or continuous 1's.

This method tries to change each vector so that the new vector has longer runs of 0's and 1's but the fault coverage is not sacrificed. Instead this method tries to exploit consecutive 0's and 1's inside test patterns for compression. Consider the test cubes for compression as shown in fig 2.

Fig 2 Test cubes.

```

x x x 0 0 x x x 1 x x x x x x x
x x x 1 0 x x x 1 x x x x x x x
x x x x 0 0 1 0 0 x x x x x 1 x
x x x x 1 0 1 0 0 x x x x x 1 x
    
```

we adopt a greedy approach to assign don't-care bits from left to right, in which either '0' or '1' is selected to make the current run longer. According to this heuristic, the test cubes in Fig. 2 are assigned to the test vectors as shown.

Fig 3 Test vectors.

```

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1
1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1
    
```

However advantage of such modification in algorithm is that as both runs of 0's and 1's are considered hence we don't have to insert 1 at the end of sequence like we have to insert in Golomb coding. This modified algorithm is known as 2-v Golomb coding as both runs of 0's and 1's are considering.

Table 5. 2-v Golomb coding

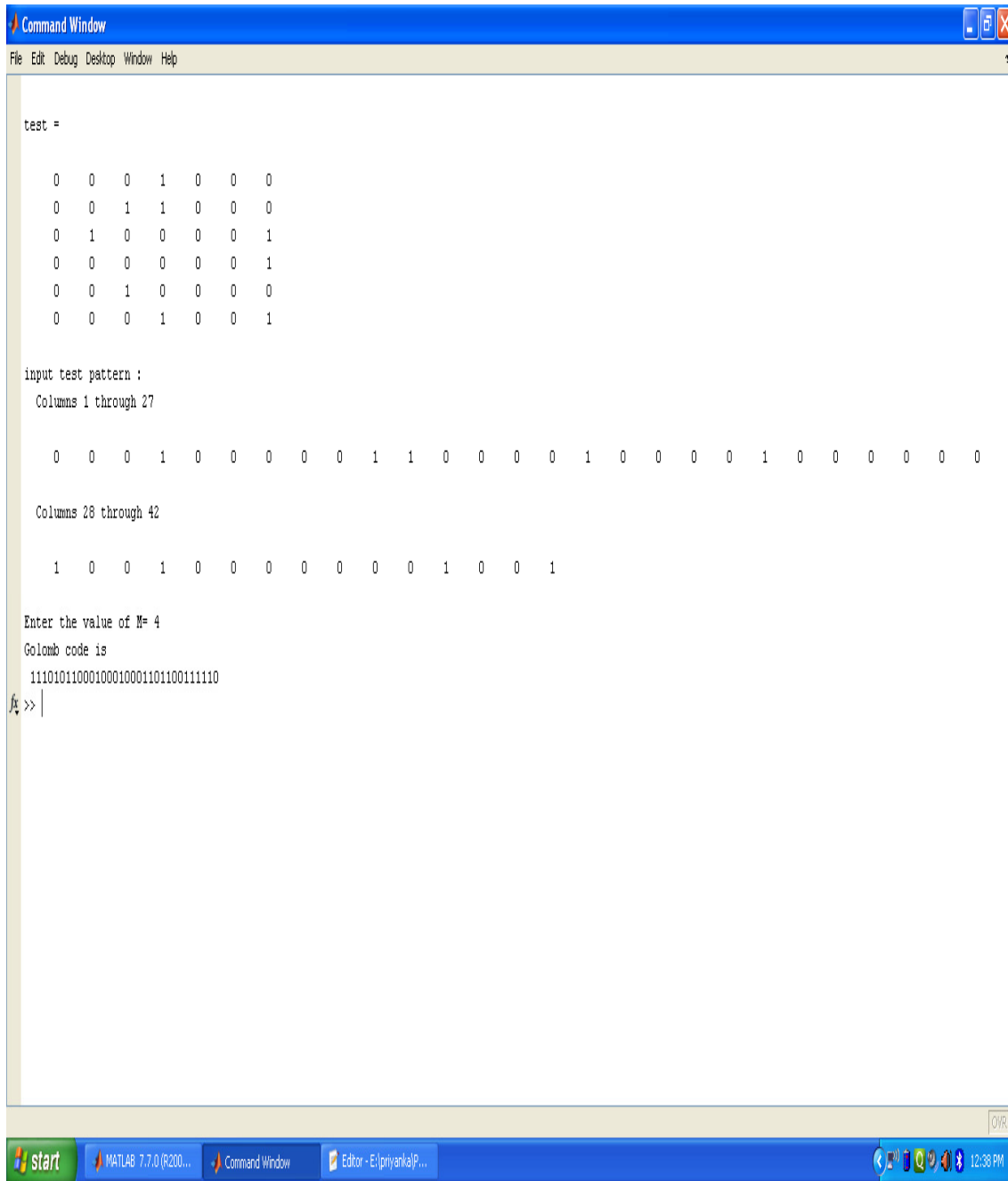
<b>T<sub>D</sub></b>	000000 00	1	000000 00000	1	000 0	1111111111111111 111	000000 00	11111 11	0	1	000000 00	11
<b>Run length</b>	8	1	11	1	4	16	8	7	1	1	8	2
<b>T<sub>E</sub></b>	00100	101	00111	101	010 0	0000100	00100	0111	101	101	00100	110

### 4. MATLAB RESULTS:

MATLAB Coding for both Golomb coding and 2-V Golomb coding is done for encoding and decoding. In this input considered for compression is test patterns for different ISCAS benchmark circuits of sequential and combinational type.

Both algorithms are applied to test patterns of different ISCAS benchmark circuits and results are shown in table 6 and table 7. Results obtained by both are compared and comparison is shown in table 8. So it is observed from comparison that compression ratio by 2-V Golomb coding is much higher than Golomb Coding.

Encoding and decoding results for both methods are shown below.



```
Command Window
File Edit Debug Desktop Window Help

test =

    0    0    0    1    0    0    0
    0    0    1    1    0    0    0
    0    1    0    0    0    0    1
    0    0    0    0    0    0    1
    0    0    1    0    0    0    0
    0    0    0    1    0    0    1

input test pattern :
Columns 1 through 27

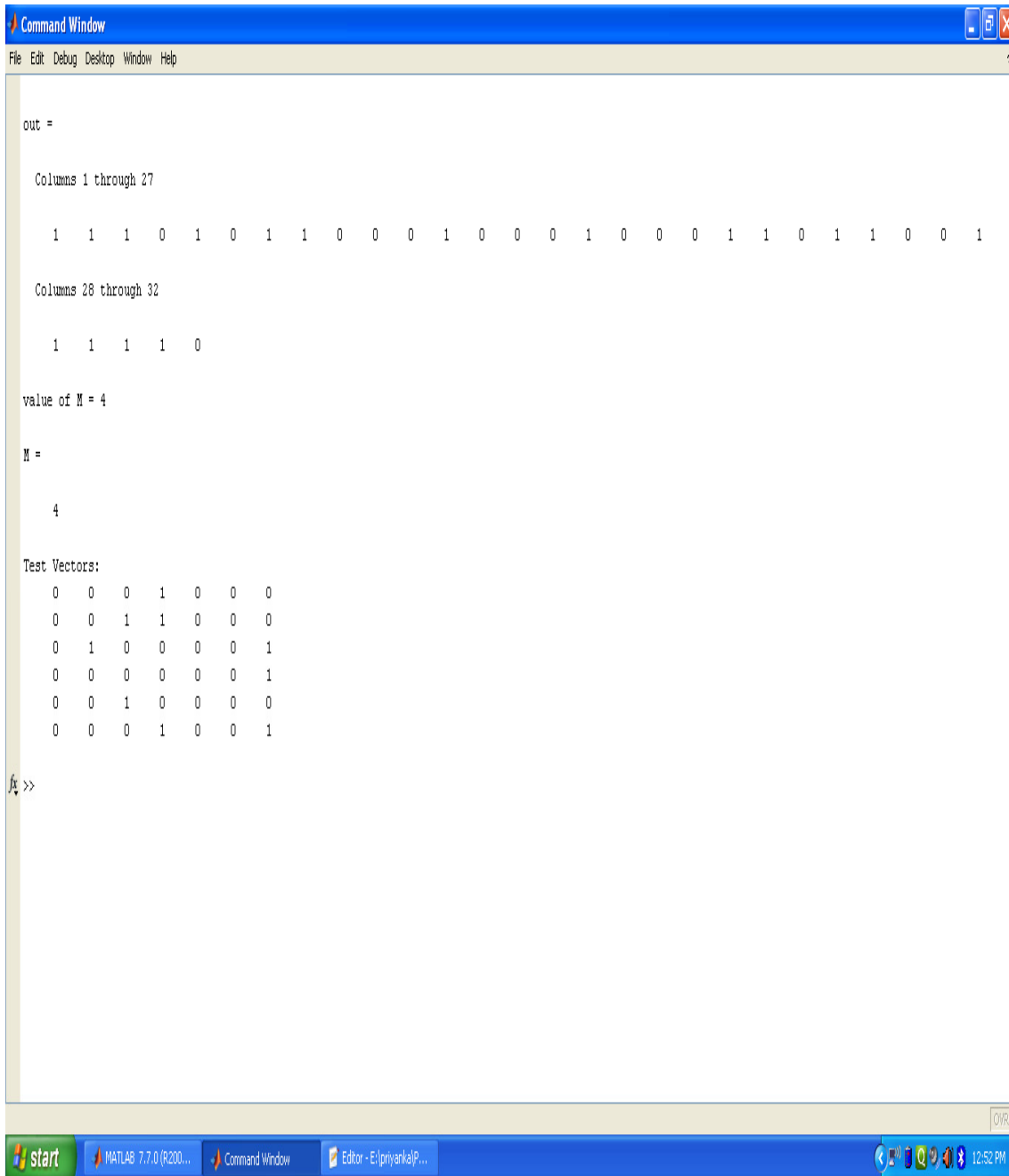
    0    0    0    1    0    0    0    0    0    0    1    1    0    0    0    0    1    0    0    0    0    0    1    0    0    0    0    0    0

Columns 28 through 42

    1    0    0    1    0    0    0    0    0    0    0    0    1    0    0    1

Enter the value of M= 4
Golomb code is
11101011000100010001101100111110
>>
```

Fig 4 Golomb Encoding for test pattern



```
Command Window
File Edit Debug Desktop Window Help

out =

Columns 1 through 27

    1    1    1    0    1    0    1    1    0    0    0    1    0    0    0    1    0    0    0    1    1    0    1    1    0    0    1

Columns 28 through 32

    1    1    1    1    0

value of M = 4

M =

    4

Test Vectors:

    0    0    0    1    0    0    0
    0    0    1    1    0    0    0
    0    1    0    0    0    0    1
    0    0    0    0    0    0    1
    0    0    1    0    0    0    0
    0    0    0    1    0    0    1

fx >>
```

Fig 5 Golomb Decoding for test pattern



```

Command Window
File Edit Debug Desktop Window Help

test =

    0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1
    1  1  1  1  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1
    0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  1
    1  1  1  1  1  0  1  0  0  0  0  0  0  0  0  0  0  1  1

input test pattern
Columns 1 through 27

    0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0  1  1

Columns 28 through 54

    1  1  1  1  1  1  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1  1  1  1  1

Columns 55 through 68

    1  1  0  1  0  0  0  0  0  0  0  0  1  1

Enter value of M:4
The Golomb Coded Output Stream is
Columns 1 through 27

    0  0  1  0  0  0  0  0  1  0  1  0  1  0  0  0  0  1  0  1  0  1  1  0  1  0  1

Columns 28 through 50

    0  0  1  0  0  0  1  1  1  1  0  1  1  0  1  0  0  1  0  0  1  1  0

fx >>
    
```

Fig 6 2-V Golomb Encoding for test pattern

```

Command Window
File Edit Debug Desktop Window Help

out =

Columns 1 through 27

    0    0    1    0    0    0    0    0    0    1    0    1    0    1    0    0    0    0    1    0    1    0    1    1    0    1    0    1

Columns 28 through 50

    0    0    1    0    0    0    1    1    1    1    0    1    1    0    1    0    0    1    0    0    1    1    0

run_bits =

    0    1    0    1    0    1    0    1    0    1    0    1

value of M = 4

M =

    4

Test Vectors

    0    0    0    0    0    0    0    0    1    1    1    1    1    1    1    1    1
    1    1    1    1    0    0    0    0    1    1    1    1    1    1    1    1    1
    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0    1    1
    1    1    1    1    1    0    1    0    0    0    0    0    0    0    0    1    1

fx >>
    
```

Fig 7 2-V Golomb Decoding for test vectors

## 5. EXPERIMENTAL RESULTS

In this section, we experimentally evaluate the test data compression/ Decompression method for ISCAS benchmark circuits. We considered test patterns of both sequential and combinational circuits for compression and calculated the amount of compression obtained by using formula below:

Compression ratio (%), CR =  $\frac{\text{Total number of bits in } T_D - \text{Total number of bits in } T_E}{\text{Total number of bits in } T_D} \times 100$

$$\frac{(T_D - T_E) \times 100}{T_D}$$

Here  $T_D$  is total number of test bits for given benchmark circuit.  $T_E$  is test bits after encoding test  $T_E$ . We consider test vectors of some benchmark circuits and applied both methods on it for different values of M.

Table 6 illustrates the compressed test vectors ( $T_E$ ) for different ISCAS circuits and its CR using Golomb Coding.

In table 6 test patterns of some ISCAS benchmark circuits are considered and it is compressed using Golomb coding Algorithm. Then after getting compressed test bits ( $T_E$ ) calculation of compression ratio is done using formula explained earlier. It is observed that maximum times compression ratio is negative.

Table 6 Experimental results: ISCAS benchmark circuit's test patterns using Golomb coding.

Circuit	$T_D$	M	$T_E$	CR (%)
C432	1152	2	2034	-71.21
	1152	4	2972	-15.01
	1152	8	3936	-231.31
C499	2132	2	2241	-5.11
	2132	4	2707	-26.96
	2132	8	3355	-57.36
C1355	3444	2	4518	-31.2
	3444	4	6043	-75.46
S298	420	2	260	38.09
	420	4	182	56.66
	420	8	176	58.09
	420	16	189	55

Table 7 illustrates the compressed test bits for different ISCAS circuits and its CR using 2-V Golomb Coding method.

In table 7 test patterns of some ISCAS benchmark circuits are considered and it is compressed using 2-V Golomb coding Algorithm. Then after getting compressed test bits ( $T_E$ ) calculation of compression ratio is done using formula explained earlier. It is observed that compression ratio is maximum than earlier algorithm.

Table 7 Experimental results: ISCAS benchmark circuit's test patterns using 2-V Golomb coding.

Circuit	T <sub>D</sub>	M	T <sub>E</sub>	CR (%)
C432	1152	2	846	26.56
	1152	4	720	37.5
	1152	8	741	35.67
C499	2132	2	1454	31.8
	2132	4	1163	45.45
	2132	8	1155	45.82
C1355	3444	2	2482	27.93
	3444	4	2077	39.69
	3444	8	2137	37.95
S298	417	2	324	22.30
	417	4	280	32.85
	417	8	307	26.37
	417	16	353	15.34

Table 8 illustrates the compressed test bits for different ISCAS circuits and its CR using Golomb coding and 2-V Golomb Coding and its comparison.

After calculating compression ratio (CR) for some ISCAS benchmark circuits by both methods. We compared results for both methods which is shown in table 8.

Table 8 comparison of compression ratio (CR) in both methods.

Golomb Coding					2-V Golomb Coding				
circuit	T <sub>D</sub>	M	T <sub>E</sub>	CR	circuit	T <sub>D</sub>	M	T <sub>E</sub>	CR
C432	1152	2	2034	-71.21	C432	1152	2	846	26.56
	1152	4	2972	-15.01		1152	4	720	37.5
	1152	8	3936	-231.31		1152	8	741	35.67
C499	2132	2	2241	-5.11	C499	2132	2	1454	31.8
	2132	4	2707	-26.96		2132	4	1163	45.45
	2132	8	3355	-57.36		2132	8	1155	45.82
C1355	3444	2	4518	-31.2	C1355	3444	2	2482	27.93
	3444	4	6043	-75.46		3444	4	2077	39.69
S298	420	2	260	38.09	S298	417	2	324	22.30
	420	4	182	56.66		417	4	280	32.85
	420	8	176	58.09		417	8	307	26.37
	420	16	189	55		417	16	353	15.34

Based on compared results in table 8 we found best method for each ISCAS benchmark circuit shown in table 9.

Table 9 Suitable method for ISCAS circuit's

Circuit	T <sub>D</sub>	T <sub>E</sub>	Best compression(M)	Method
C432	1152	720	4	2-V Golomb Coding
C499	2132	1155	8	2-V Golomb Coding
C1355	3444	2077	4	2-V Golomb Coding
S298	420	176	8	Golomb Coding

## 6. CONCLUSION

We presented Golomb coding and 2-V Golomb coding in MATLAB. Experimental results for both methods is calculated and compared. Experimental results for the ISCAS benchmark show that the compression technique is very efficient for combinational and full-scan circuits. Golomb coding gives good compression ratio in sequential circuits and 2-V Golomb coding in combinational circuits.

After verification of both algorithms in MATLAB, hardware implementation of 2-V Golomb coding have to carry out using VHDL code.

## References

- [1] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core based system chips," in Proc. Int. Test Conf., 1998, pp. 130–143.
- [2] V. Iyengar, K. Chakrabarty, and B. T. Murray, "Deterministic built-in pattern generation for sequential circuits," J. Electron. Testing: Theory and Applications (JETTA), vol. 15, pp. 97–115, Aug./Oct. 1999.
- [3] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, "Scan vector compression/decompression using statistical coding," In Proc. IEEE VLSI Test Symp., 1999, pp. 114–120.
- [4] Sybille Hellebrand, Armin Würtenberger, "Alternating Run-Length Coding -A Technique for Improved Test", Handouts 3rd IEEE International Workshop on Test Resource Partitioning, Baltimore, MD, USA, October 10 –11, 2002 Data Compression
- [5] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," in Proc. Int. est Conf., 1998, pp. 283–289.
- [6] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, "On compacting test sets by addition and removal of vectors," in Proc. VLSI Test Symp., 1994, pp. 202–207.
- [7] I. Hamzaoglu and J. H. Patel, "Reducing test application time for full scan embedded cores," in Proc. Int. Symp. Fault-Tolerant Computing, 1999, pp. 260–267.
- [8] H. Kobayashi and L. R. Bahl, "Image data compression by predictive coding, Part I: Prediction algorithm," IBM J. Res. Devel., vol. 18, p. 164, 1974.
- [9] Anshuman Chandra and Krishnendu Chakrabarty, "Test Data Compression for System-on-a-Chip Using Golomb Codes I", IEEE Trans. Computer-Aided Design, 2000.

- [10] Y. Zorian, S. Dey, and M. Rodgers, "Test of future system-on-chips," in Proceedings of International Conference Computer-Aided Design, 2000, pp. 392-398.
- [11] PO-CHANG TSAI, SYING-JYAN WANG, CHING-HUNG LIN AND TUNG-HUA YEH, "Test Data Compression for Minimum Test Application Time," JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 23, 1901-1909 (2007)
- [12] A. Chandra and K. Chakrabarty, "A unified approach to reduce SoC test data volume, scan power, and testing time," IEEE Transactions on Computer-Aided Design, Vol. 22, 2003, pp. 352-363.
- [13] Chandra, A.; Chakrabarty, K., "Test Data Compression and Decompression Based on Internal Scan Chains and Golomb Coding", IEEE Trans. Computer-Aided Design, Volume: 21 , Publication Year: 2002 , Page(s): 715 - 722
- [14] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core- based design," in Proc. Int. Test Conf., 1998, pp. 458-464.
- [15] S. J. Wang and S. N. Chiou, "Generating efficient tests for continuous scan," in Proceedings of Design Automation Conference, 2001, pp. 162-165.
- [16] A. 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.
- [17] S. Hellebrand, H.-G. Liang, and H.-J. Wunderlich, "A mixed-mode BIST scheme based on reseeding of folding counters," in Proc. Int. Test Conf., 2000, pp. 778-784.
- [18] A. 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, Sep. 2000.
- [19] Huizhuo Niu, Yuanyuan Shang, Xinhua Yang, Dawei Xu, Baoyuan Han, Chuan Chen, "Design and Research on the JPEG-LS Image Compression Algorithm", 2010 Second International Conference on Communication Systems, Networks and Applications
- [20] Tsung-Han Tsai, Member, IEEE, and Yu-Hsuan Lee, "A 6.4 Gbit/s Embedded Compression Codec for Memory-Efficient Applications on Advanced-HD Specification", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 20, NO. 10, OCTOBER 2010 1277