Enhanced Compression Code for SOC Test Data Volume Reduction

Shruti Chadha[1] and Harpreet Vohra[2]

[1,2]ECED, Thapar University,
Patiala, Punjab 147004, India

Abstract
Test data reduction is an important issue for the system-on-a-chip designs. A number of coding techniques have been developed in the past to compress the test data to achieve the best compression. In this paper we have merged two run length based codes i.e. Golomb code and AVR code to develop a new code called integrated compression code (ICC). The enhanced compression code (ECC) compresses the data achieved after compression form ICC by 9C coding technique to enhance the compression ratio further. We have also compared the proposed compression technique with previously developed compression codes. We have applied column bit stuffing, column bit stuffing with difference vector and ZERO filling for unspecified bits of the test set to reduce the average and peak power. These were calculated using WTM. Four ISCAS'89 benchmark test sets have been used to prove the effectiveness of the proposed ECC.
Keywords**—***SOC; integrated coding; test data compression; efficiency*

1. INTRODUCTION

As the technology is improving, the density of transistors on a single chip is increasing and the manufacturing cost per transistors is decreasing. But the silicon industry is not able to reduce test cost per transistor [1]. This cost includes a number of parameters, but the major one is the cost of Automatic Test Equipment (ATE) [1]. The speed of testing depends on number of channels, clock rate of the channel and required test data [2]. The commercial ATE's have limited memory, bandwidths and I/O channel capacity. The test application time depends on the amount of test data stored on ATE, the time required to transfer the test data from ATE to the core and length of the scan chain.

$$\text{TEST TIME} \geq \frac{amt\ of\ test\ data\ on\ tester}{no.of\ tester\ channelsXtester\ clk\ rates} \quad (1)$$

As we can see from the above equation that the test time is directly proportional to the test data hence we can reduce this test data to reduce testing time. Built-in self test (BIST)[3-5] can be a useful approach for alleviating the discussed problems [6]. BIST reduces dependencies on expensive ATE. But the problem with BIST is that, it can only be applied directly to SOC designs only if the embedded cores are BIST ready, considerable redesign is required or implementing BIST. Test data compression offers a promising solution to the problem of reducing the test data volume, special when the cores are not BIST ready.

The test vector reduction consists of compressing the original test data off line, storing the compressed data in the ATE, and then decompressing them for restoring the original test vectors. There are three basic methods for test data compression [7]. : Code-based schemes, Linear-decompression-based schemes and Broadcast-scan-based schemes. In code based schemes the internal architecture of the IP cores is not changed hence they are most easy to use.
The code based schemes are further classified as 1.Run length based, 2.Dictionary based, 3.Statistical codes and
4. Constructive codes [7]. The dictionary based codes are bit cumbersome to deal with as it requires in built dictionary which would store the code words. Talking about statistical codes they are although easy to create but are not flexible enough to include in them variable run length vectors [7]. In this paper we will concentrate on run length based codes. The proposed work has been compared with the Golomb [8], FDR [9], EFDR [10], AFDR [11], CPRL [12] and with AVR [13]. All these are variable to variable run length code. The Golomb code and AVR codes are discussed in details so that there functionality is clear as the proposed work is based on these codes.

The organization of the rest of the paper is as follows: Section 2 contains the discussion of Golomb and AVR codes. Section 3 proposes the enhanced compression code (ECC). Section 4 describes the decompression architecture of ECC. Section 5 contains power analysis of the test vectors. Section 6 contains experimental results for the data compression of ISCAS circuits. Finally section 7 draws the conclusion.

2. Run length based codes

The run length based codes encodes the test data by looking at the run's of either zero's or one's or both. Various coding schemes have been discussed in section 1. Two of which are discussed below

2.1 Golomb code

It is a variable to variable length code. It is used to encode the run length of zeros. A group parameter '*m'* is optimally chosen. Different values of *m* result in different compression. As discussed in [8] if the input data is random with zero probability *p*, then *m* must be chosen such that

$$p^m = 0.5 \quad (2)$$

The group parameter also decides the number of members in each group. The code consists of two parts: a) group prefix and b) tail. The prefix is fixed for a particular group and tail varies as the run length varies. The number of bits in tail are decided by *m* and its length is $log_2 m$. The coding was done for run of zeros till a'1' arrived.

Table 1 shows coding through Golomb code for *m=4.*

Table 1: Golomb code [8]

Group	Run length	Group prefix	Tail	Codeword
A_1	0 1 2 3	0	00 01 10 11	000 001 010 011
A_2	4 5 6 7	10	00 01 10 11	1000 1001 1010 1011
A_3	8 9 10 11	110	00 01 10 11	11000 11001 11010 11011

Example :

Coding through Golomb Code:
$T_{D=}$0000001111111111110000100010000000111110000 1
No. of bits= 45
After applying Golomb code with m=4
$T_{E=}$1010 000 000 000 000 000 000 000 000 000 000 000 000 1000 011 1011 000 000 000 000 1000
No of bits =67

Due to runs of one Golomb code is not effective. Hence we have to use a code which should efficiently encode runs of one's as well.

2.2 Alternate variable code

The alternating variable run-length code is also a variable to variable length code, consisting of two parts – the group prefix and the tail. The prefix determines the group in which the present run length lies and the tail tells us about the number of ones or zeros (runlength) within the group. The test data can b classified into runs of zero's ending with a one and runs of one's ending with a zero. Extra parameter 'a' was used to whether the run length being decoded was of '0'or '1'. This parameter was also used during decoding to produce the correct run length.

Run length	Code to be used	No. of bits in codeword
0	Bypass mode	0
1	Bypass mode	1
2-4	AVR 0/1	3
5-15	Golomb for 0/ AVR for 1	5-7
16-31	Golomb for 0/ AVR for 1	6-7

Group	Run length	Group prefix	Tail	Codeword
A_1	1 2 3 4	01 10 	0 1 0 1	010 011 100 101
A_2	5 6 7 8 9 10 11 12	001 110 	00 01 10 11 00 01 10 11	00100 00101 00110 00111 11000 11001 11010 11011
A_3	13 14 .. 20 21 22	0001 1110 	000 001 111 000 001 	0001000 0001001 0001111 1110000 1110001

Table 2: AVR code [13]

Example:

Coding through AVR:
$T_{D=}$ 0000001111111111110000100010000000 1 1111 00001
No. of bits= 40
$T_{D=}$ 00101 11011 011 011 00110 101 100
a=0 a=1 a=0 a=0 a=0 a=1 a=0
No. of bits =27

compared by number of bits compressed by the two, then AVR is better option than Golomb code.

3. Enhanced Compression Code

The drawback of the Golomb code is that it is beneficial only for the run's of zeros. This was evident from the experimental results. Hence it was needed to encode runs of ones also. Although AVR was efficient enough in encoding ones but when moved from one group to another it increments two bits in the code word. The codes are used depending on the run length.

Now the two codes are merged. The code providing minimum number of bits to a particular run length is used. For runs of ones, only AVR code is used and for runs of zeros, the code providing least bits in the codeword is used. Also it is observed that if the run length 0 and 1 of whether one or zero are not encoded the bits are further reduced. Hence a *bypass mode* is used to bypass the coding of run length 0 and 1. This scheme is named as ICC.

Table 3: Run length based utilization

The Golomb code for m=16 is used above.
Considering the previous example

$T_{D=}$ 0000001
11111111111000010001000000001111100001
No. of bits =45
$T_{E=}$ 00110 11011 011 011 0111 101 100
No. of bits =26

Hence, on using Integrated Compression code (ICC), better results are obtained. Although here only one bit difference can be observed from AVR but when the test data will be huge then the difference will b remarkable. Also there is no area overhead in the on-chip decoder.

3.3 Enhanced compression code

Even after compressing the bits with ICC there are still more bits available which can be compressed. Nine coded compression (9c) [18] can be used on the already compressed data to improve the compression further. Taking the above example again,

$T_{E=}$ 00110 11011 011 011 0111 101 100 No. of bits =26
$T_{E1=}$ 1111 1111 11110
No. of bits =13

Therefore the compression has improved by 50%

4. Decompression Architecture

The ICC decoder is explained in the next section, then the Enhanced compression decoder is implemented using the ICC decoder.

4.1 Decoder of ICC

The decoder architecture is similar to the on chip decoder architecture of Golomb code[8] and AVR code[13] with minute additional signals. The decoder decodes the encoded test data to produce the original test data (Td). The testing time has also been reduced as test pattern compression could be carried out on chip at higher clock frequencies. l_{max} is the longest run of zeros or 1s in Td, and k=$\lceil log_2(l_{max}+4)-2 \rceil$. The decoder is made up of a k+1- bit counter, a $log_2(k+1)$ bit counter, a T filp flop and an exclusive or gate. Fig.1 shows the view of decoder architecture. It has following signals.

- The select signal tells us which code was used for encoding. If it is 0 then AVR has been used and if it is 1 then Golomb has been used
- Bypass signal reflects the bypass mode if it is high
- Bit_in as the name suggest feeds the FSM with the input bits and an enable signal en has been used to control the input when the decoder was ready
- Shift signal controlled the prefix and the tail of the codeword to shift in the k+1 bit counter. Rs1 and RS2 told about the reset stats of the two counters
- Another $log_2(k+1)$ counter has been used to count prefix and tail, which helped in identifying the group.
- The out of the FSM controlled the toggle of the T f/f, and indicated that it has decoded the runs of 0's and 1's according to the binary parameter 'a'. Single v signifies the validity of the output.

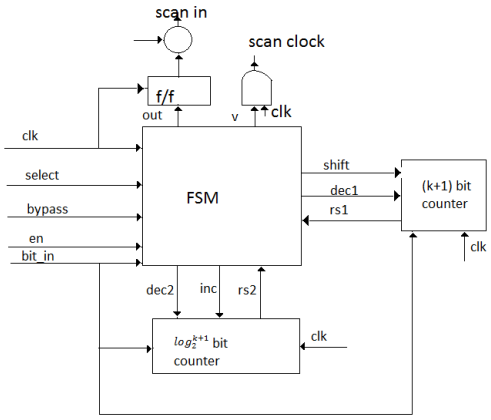

Fig.1 decoder architecture

The operation of the decoder can be understood as follows
Firstly the T f/f was reset and v was set to 0. Signal en if is low means that it is busy counting number of zeros and if it is high means that it has finished counting. So initially it has been set to 1. Now the FSM is ready to receive the data.
The function of the k+1 bit counter was to identify the prefix with the help of separator .this is valid for both Golomb code and AVR. The signals en, shift and inc were kept high until 0 or 1 was received.
The FSM outputs, 0s and 1s, decremented the k+1 bit counter and made the signal dec1 high. It continued it until rs1 is high. The v signal showed whether the output was high or not.
The tail part was again shifted to k+1 bit counter, but was under the control of $log_2(k+1)$ bit counter. It controlled the length of the control word.
Fig.2 shows FSM. It has 6 states. The S0 state is starting state as well as bypass state. For AVR code S0-S1-S3 states are used for decoding prefix starting with 0 and S0-S2-S4 states are used for decoding prefix starting with 1. For Golomb code the prefix decoding is done by S0-S3-S4. The tail decoding is done by states S4-S4.

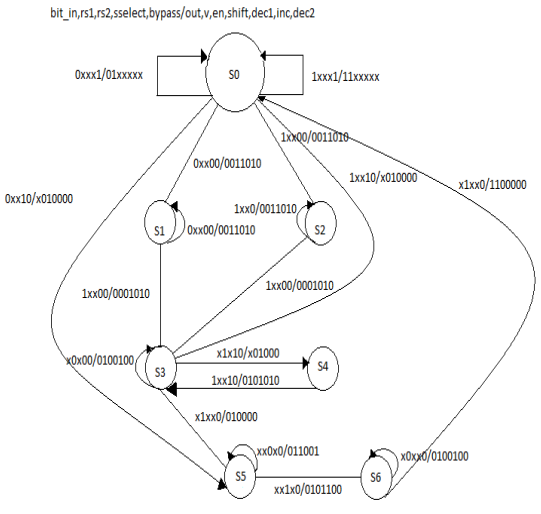

Fig.2 FSM of ICC

4.2 Decoder of ECC

The decoder contains two FSM, one counter, one synchronization circuit, a multiplexer and control signals. The decoder operates with two clocks, one external i.e. ATE_Clk and the internal clock SOC_Clk. FSM* is of 9C and FSM# is of ICC. FSM of 9C has been explained in [18]. FSM*

receives the data from DATA_IN. This data is compressed data. Firstly the compressed data is passed to FSM* which will remove the 9C coding and then this data is passed to FSM# through Data_in_1to decode it further. The signals have the same value as discussed for the ICC decoder architecture.

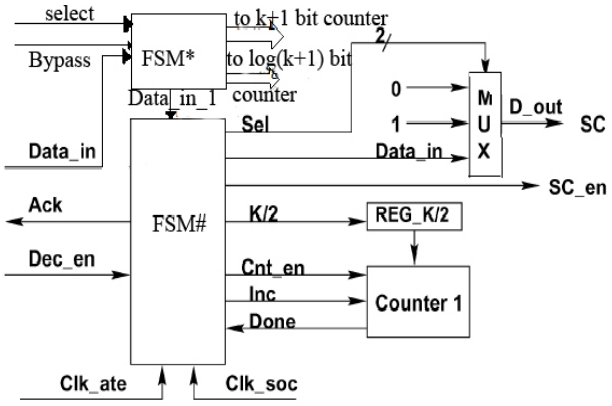

Fig.3 Decoder of ECC

5. Power Analysis

Power dissipation is an important problem for the circuit under test. A huge amount of power is dissipated when the circuit elements switched from logic 1 to 0 and vice versa. During testing this activity is dominantly controlled by test vectors [9].The power dissipation of the applied input and its response can be calculated using the weighted transitions in the vector. Scan vectors having the higher value of weighted transition metrics will dissipate more power in the circuit under test(CUT). The equation used for calculating weighted transitions is

Weighted_transions=Σ(size_of_scan_chainposition_of_transition)(3)

Table 4: Mapping of don't cares in test data to binary values.

Scan vector with don't care terms	0000110xxxx1001xxxx0
Filling with WTM	00001100000100111110 = 20bits Golomb code length = 29 bits, AVR code length= 17bits, ICC code length= 15, WTM= 53
Filling with zeros	00001100000100100000 =20 bits Golomb code length =21 bits, AVR code length= 15 bits, ICC code length= 13, WTM=58

Weighted transition metric has been discussed in details in [17] to estimate the power dissipation of the scan data. It told that power dissipation does not only depend on number of transitions but also on the relative positions of the vectors. If vector 'z' had more WTM than the vector 'y', then 'z' is said to dissipate more power than 'y'.

Let us say that a scan chain of length t is being dealt with and a scan vector $l_j = l_{j,1}l_{j,2}\ldots\ldots l_{j,t}$, and $l_{j,1}$ is scanned before l_{j2}. As shown in [15], we can calculate the value of the WTM for inputs as well as their responses by the following equation

$$WTM_j = \sum_{i=1}^{t-1}(t-i)(l_{j,i} \oplus l_{j,i+1}) \quad (4)$$

With the help of the above equation we can also calculate the peak(P_{peak}) and the average(P_{avg}) power as follows

$$P_{avg} = \frac{\sum_{j=1}^{n}\sum_{i=1}^{t-1}(t-i)(l_{j,i} \oplus l_{j,i+1})}{n} \quad (5)$$

$$P_{peak} = max_{j\in(1,2..n)}\left\{\sum_{i=1}^{t-1}(t-i)(l_{j,i} \oplus l_{j,i+1}\right\} \quad (6)$$

These equations were used to calculate the average and peak power for the four different test vector set for benchmark circuits

There are different techniques that are useful to reduce the switching power dissipation in the scan chain. These can be minimum transmission fill[16],column bit stuffing[14], zero filling, random filling[16]. These techniques combined with the proposed ICC provided excellent results for power dissipation and test data volume reduction.

6. Experimental results

Columnwise bit stuffing with difference vector(CBSTDIFF) [14], column bit stuffing(CBS) [14] and zero filling[16], are used to fill the don't care terms of the test vectors generated by the Mintest ATPG program. The ISCAS'89 benchmark circuits used are s298, s400t, s1494, s1196t. The efficiency is used to compare the different codes with each other, where

$$Efficiency = \frac{T_D - T_E}{T_D} \times 100 \quad (2)$$

Where T_D is the original test data and T_E is the data achieved after compression. The coding and bit stuffing was done in the C programming language on the work station of corei3 and 4GB memory. The area overhead is very small and even less than the AVR code as the proposed FSM contains one less state.

7. Conclusion

The paper focuses on the problem of test data volume and power consumption of scan vectors for system-on-chip testing. The test data volume was efficiently reduced by the proposed compression code i.e. ECC. The experimental results from the ISCAS'89 benchmark circuits have presented to show that ECC is better at producing compression than run length based codes like Golomb, FDR, EFDR, AFDR, CPRL and AVR. It is effective for both, runs of zero's and runs of one's. WTM has been used to calculate the peak and average power. To reduce the power consumption column wise bit stuffing has come out to be the best technique.

Table 5: Compression results for circuit s298

	Avg.power	Peak pwr	Golomb	FDR	EFDR	AFDR	CPRL	AVR	ECC
CBSTD	57.4419	96	-4.56	-16.37	-34.93	-14.64	5.96	20.88	25.15
CBSTDDIFF	20.9612	57	46.97	43	32.92	34.34	60.82	68.23	70.67
ZERO FILL	25.5039	53	-14.09	-11.81	28.09	48.02	62.02	70.12	71.82

Table 6: Compression results for circuit s1494

	Avg.power	Peak pwr	Golomb	FDR	EFDR	AFDR	CPRL	AVR	ECC
CBSTD	42.904	88	-24.15	-32.6	-32	-9.13	3.86	11.96	19.34
CBSTDDIFF	21.4233	62	19.32	19.76	3.55	9.74	38.92	53.42	59.11
ZERO FILL	16.67733	52	-12.54	-16.2	0	23.13	42.06	63.89	64.00

Table 7: Compression results for circuit s400t

	Avg.power	Peak pwr	Golomb	FDR	EFDR	AFDR	CPRL	AVR	ECC
CBSTD	14.3566	28	-32.82	-42.27	-29.99	-3.62	20.17	31.59	35.18
CBSTDDIFF	2.8005	28	60.54	56.3	49.22	45.82	65.29	66.05	66.89
ZERO FILL	2.8775	28	-59.26	-63.28	45.85	67.89	34.62	68.03	75.90

Table 8: Compression results for circuit s1196t

	Avg.power	Peak pwr	Golomb	FDR	EFDR	AFDR	CPRL	AVR	ECC
CBSTD	87.8046	151	-20.2	-29.94	-40.33	-20.69	11.02	20.32	22.77
CBSTDDIFF	43.2622	118	36.19	35.33	23.84	26.64	34.57	36.83	40.01
ZERO FILL	48.1671	130	-10.05	-14.73	2.35	24.1	10.08	25.67	28.91

References

[1] Pranab K. Nag, Anne Gattiker and Sichao Wei, "Modelling the Economics of Testing: A DFT Perspective", in IEEE Design & Test of Computers,2002

[2] B. T. Murray and J.P. Hayes, "Testing ICs: Getting to the core of the problem," Computer, Vol.29, pp.32-38, Nov. 1996.

[3] L. M. Denq, Y. T. Hsing, C. W. Wu, "Hybrid BIST scheme for multiple heterogeneous embedded memories", IEEE Des. Test Comput.26(2) pp. 64-72,2009

[4] M. Nourini, M. Tehranipoor, N. Ahmed, "Low –transition test pattern generation for BIST application", IEEE Trans. Comput. 57(3) pp 303-315,2008

[5] S. Lei, X. Hou. Z. Shao, F. Liang, "A class of SIC circuits: Theory and application in BIST design", IEEE Trans. Circuit system II 55(2) pp.161-165,2008

[6] C.W. Wu, J.F. Li and C.T. Hung, "Core-Based System-on-Chip Testing: Challenges and Opportunities", in Journal of The Chinese Institute of Electrical Engineering, vol. 8, no. 4, pp. 335-353, 2001.

[7] N. A. Tauba, "Survey of Test Vector Compression Techniques", IEEE transaction Design & Test of Computers, 2006

[8] A. Chandra and K. Chakarbarty," System-on-a-Chip Test-Data Compression and Decompression Architecture Based on Golomb Codes,"IEEE Trans. Computer-Aided Design, vol.20, no. 3, pp. 355-368, 2001

[9] A. Chandra and K. Chakarbarty, "Test Data Compression and Test Resource Partitioning for System-on-Chip Using Frequency-Directed Run-Length(FDR) Codes," IEEE Trans. Computers, vol. 52, no. 8, pp.1076-1088, Aug 2003

[10] H. Aiman, El-Maleh, H. Raslan and Al-Abaji, "Extended Frequency-Directed Run-Length Code with Improved Application to System-on-a-Chip Test Data Compression" Int. Conf: on Electronics, Circuits and Systems, 2:449-452,Sep 2002

[11] A. Chandra and K. Chakarbarty, "Reduction of SOC Test Data Volume, Scan Power and Testing Time Using Alternating Runlength Codes", DAC'02: Proceeding of the 39th conference on Design automation, June 2002

[12] Haiying Yuan, Jiaping Mei, Hongying Song and Kun Guo, "Test data compression for System-on-a-Chip using Count Compatible Pattern Run length coding" J Electron Test 30:237–242, 2014

[13] Bo Ye, Qian Zhao, Duo Zhou, Xiaohua Wang, Min Luo, "Test data compression using alternate variable run length code", Integration, the VLSI journal 44, pp. 103-110, 2011

[14] Usha S. Mehta, Kankar S. Dasgupta and Niranjan M. Devashrayee, "Hamming distance based reordering and column wise bit stuffing with difference vector: A better scheme for test data compression with run length based codes", 23rd International Conference on VLSI design,2010

[15] A. Chandra, K. Chakarbarty, "A unified approach to reduce SOC test data volume, scan power and testing times", IEEE Trans. Comput.-aided des. Integer circuit system 22(3) 352-362, 2003

[16] M. Kassab, G. Murgalski , "Low power scan shift and capture in EDT environment", IEEE International Test Conference, 2008

[17] R. Sankaralingam, R. Oruganti, N.A. touba "Static Compaction techniques of control scan power dissipation", in proceedings of the 18th IEEE VLSI test Symposium pp. 35-40, 2000

[18] M. Tehronipoor, M. Nourani and K. Chakarbarty, "Nine-coded compression technique for testing embedded cores in SOCs" IEEE Trans. On very large scale integration (VLSI) systems, Vol. 13, No. 16, June 2005

[19] Deepika Sharma, Debbrat Ghosh and Harpreet Vohra, "Test data volume minimization using double hamming distance reordering with mixed Rl-huffman based compression scheme for system on chip" Nirma University International Conference on Engineering, Nuicone, 2012

Shruti Chadha received her B.Tech degree in Electronics and communication engineering from G.N.E. Ludhiana, Punjab, India in 2013. She is currently pursuing M.Tech degree in VLSI design from department of electronics and communication engineering, Thapar University, Patiala, India.

Harpreet Vohra had completed her B.E. in Electronics and communication engineering from SLIET, Longowal in 2004. In 2006 she received her M.Tech degree in VLSI design from CDAC, Mohali. Currently she is a faculty at Department of Electronics and communication, Thapar University, Patiala and pursuing PhD from IITKharagpur.