A New Collaborative Scheme of Test Vector Compression Based on Equal-Run-Length Coding (ERLC)

Wenfa Zhan[1], Aiman El-Maleh[2]
1. Dept. of Educational Technology, Anqing Normal College, Anhui Province, P. R. China
2. King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
E-mail: zhanwenfa@gmail.com; aimane@kfupm.edu.sa

Abstract

A new scheme of test data compression, namely equal-run-length coding (ERLC) scheme is presented, which is based on run-length. It first considers both types of runs of 0's and 1's, then it further explores the relationship between two consecutive runs on the basis of the traditional characteristic of run coding which uses shorter codeword to represent longer symbol (run-length). This scheme uses two shorter codewords to represent the whole second run of two consecutive runs, the lengths of which are the same. Compared with other already known schemes this scheme has some characteristics, such as high compression ratio, easy control and implementation. The performance of the algorithm is experimentally confirmed on the larger examples of the ISCAS89 benchmark circuits.

Keywords: Test Vector Compression, VLSI Test, Coding, Bulit-in Self-Test.

1. Introduction

The complex functionality and size caused by increasing integration levels of VLSI chips makes testing for these chips more and more difficult. Testing remains a dominant cost factor in VLSI design and reducing the testing cost is one of the objectives which VLSI producers are devoted to achieve vigorously. The most two important sources of the test cost are test data volume and test power.

Researches about reducing the test data volume fall into three aspects: Built-in Self Test (BIST)[1], test set compact[2] , and test data compression[3-14].

Test data compression can reduce test data effectively and increases the transferring speed from ATE to chip, which makes full use of limited bandwidth and memory of ATE. Moreover, the hardware overhead is very low and acceptable. Thus this technology is widely used.

A new scheme, namely equal-run-length coding (ERLC), is proposed. This scheme first considers both types of runs of 0's and 1's, so the total number of runs and transitions during scan-in will both decrease, which result in better compression effect and lower test power. Then it further explores the relationship between two consecutive runs on the basis of traditional coding characteristic which uses shorter codeword to represent longer symbol. This presented scheme uses two shorter codewords to represent the whole second run of two consecutive runs, the lengths of which are the same. Theoretical analysis and experimental results show that the compression effect of this scheme is higher than that of previously proposed schemes. Four major properties make our approach collaborative and advantageous:

(1) It is data-independent. (2)Both types of runs are considered together, which result in better compression effect and lower test power. (3)The decompression algorithm and corresponding decoder are very simple. The hardware overhead of the decoder is very low and can be accepted. (4) The relationship between runs is considered. It uses two shorter codewords to represent the whole second run of two consecutive runs, the lengths of which are the same. It results in better compression performance than Golomb and FDR for most benchmark circuits.

The rest of this paper is organized as follows. Section 2 explains the base of the algorithm of this scheme and analyzes the compression ratio theoretically. Decompression architecture of the proposed method is presented in section 3. Section 4 reports the experimental results, and section 5 concludes the paper.

2. Algorithm of ERLC

By considering both types of runs, the total number of runs will decrease, which could result in higher test data compression[11]. The coding table of ERLC is shown in Table 1.

The coding table of ERLC is a variable-to-variable-length coding way. The size of the groups is increasing as the number of groups increasing, which is in accordance with the distribution of the run-length in actual test data and this characteristic is similar to FDR[9].

This work is supported in part by the the Natural Science Foundation of Education Agency of Anhui Province of China in 2009 titled research about dynamic reducing test data technology and in part by the Science and Technology Plan Item of Education Agency of Anqing City in 2008 titled large volume data converter of compression and recovery.

It is found that the tails of those two codewords are very similar from Table 1 and EFDR coding. (1) the size of the groups is increasing as the number of groups increasing. (2) the tails are the same if the run-length of 0's and 1's are the same. The only difference between them is that of the run-length of EFDR is longer by 1 than that of ERLC for the same tail of the codeword.

Table 1. ERLC

Run-length	Group	Prefix	Tail	Codeword Runfs of 0's	Codeword Runfs of 1's
1	A_1	0	1	001	101
2	A_2	10	00	01000	11000
3			01	01001	11001
4			10	01010	11010
5			11	01011	11011
6	A_3	110	000	0110000	1110000
7			001	0110001	1110001
……			……	……	……
12			110	0110110	1110110
13			111	0110111	1110111
……	……	……	……	……	……

It needs to be pointed out that for all runs of 0's or 1's, but the last member in every group, this coding approach will not increase the length of codeword compared with the codeword of EFDR if the original run-length and later run-length obtained from original run-length plus 1 belongs to the same group of EFDR coding table. Therefore, for the runs of 0's or 1's this coding approach will get sililar compression effect compared with EFDR. The characteristic of this approach is that ERLC scheme explores the relationship between the two consecutive runs. If the length of a consecutive run is the same as that of the former, then the whole second run can be represented by a shorter codeword. Two codewords (000 and 100) are used to represent a repeated run. The run-length of encoded run is the same as the former run. Codeword 000 represent the encoded run is runs of 0's and 100 represent the encoded run is runs of 1's. This further improves the compression ratio.

From the above analysis, it can be concluded that the scheme of ERLC uses two different coding ways: (1) Type I. This coding way uses coding table of ERLC scheme to code the run (symbol) in the original test data. This coding way is similar to that of the traditional run coding. (2) Type II. This coding way uses two shorter codewords (000 and 100) to represent the second run of two consecutive runs, the lengths of which are the same. That is to say, if the lengths of two consecutive runs are the same, then the later run is represented by the codeword 000 or codeword 100, or else the later run should be coded by using coding table.

Original test data typically have only 1-10% of the bits specified, while the rest are don't-cares [3]. Filling those don't-care bits appropriately can increase the same length probability of two consecutive runs. If the two consecutive run lengths are equal, then the later can be represented by one of the two shorter codewords. Thus, the whole later run can be represent by 3 bits, which further improves the compression ratio.

In order to improve the effectiveness of proposed scheme, the coding way of Type II can be further explored. Namely, the don't-care bits should be appropriately filled to make the length of consecutive two runs the same. Then the first run is encoded by the way of Type I and the second run is encoded by the way of Type II. In this way compression effect can be improved further. For example, let take test vector slice of 000000X000000X000000XXXX1X111XXX111XXX11 10 (X represents don't care bit) into account. The don't-care bits can be filled by 0 and 1 appropriately and two consecutive runs 000000000000 000000001 and 11111111111111111110 can be gotten. Then the second run can be encoded by 101. The length of original test vector slice is 42 and that of corresponding compressed data is 12. If the don't-care bit is filled into 000000000000000000 00000011111111111 1111110 and Type I coding way is used, then the compressed data is 011101010 111100010, whose length is 18. In addition, if the don't-care bit is filled into 000000000000000000001 11111111111111111110 and Type I and Type II coding way can both be used, then the compressed data is 011100100 100, whose length is 12. From this, conclusion can be made that appropriately filling the don't-care bits can decrease the volume of test data.

In a word, type I coding approach can encode the runs of 0's and runs of 1's, which gets a good compression effect. And type II coding way will further compress the test data. Hence, ERLC scheme can get a better compression effect than some already known schemes, such as Golomb and FDR.

An example of coding using ERLC scheme is shown in Figure 1. This example codes the length of runs 0s. Let's take the test vector slice of 000000XX XXXXX110 into account. The lengths of those two runs equal to 7. So the later run can be encoded by 100. The results encoded of the slice is 0110001 100. The length coded is shorter by 6 than that of the original slice.

Original test vector slice: 000000XX XXXXX110 000000000001 000000000001 1111X1XXXXXX 000000000001
Encoded test vector: T_E=0110001 100 0110101 000 100 000
Length of original test vector: 64 bits; T_E: 26 bits; reduce: 38 bits.

Figure 1. An Example for ERLC Encoding

3. Design of decoder

In this section, the design of the decoder of the ERLC coding scheme is illustrated.

As mentioned in [4], the real run-length (i) of ERLC is the sum of prefix (*a*) and tail (*b*), namely i=a+b. For example, $(13)_{10} = (110)_2+(111)_2$. Further analyzing the prefix, we find that it will appear in the form of 111……1110. By transforming the equation $i=a+b$, we can get

$$i=(a+2+b)-2=(((\underbrace{11\cdots\cdots110}_{k})_2+(\underbrace{10}_{2})_2)+(\underbrace{b'}_{k})_2)-2$$

$$=((\underbrace{100\cdots\cdots00}_{k+1})_2+(\underbrace{b'}_{k})_2)-2=(\underbrace{1b'}_{k+1})_2-2$$

That is to say, if we increase 1 bit to the tail of ERLC codeword and put the value of 1 in this bit, then we get a new code which represents the run-length that is longer by 2 than that of actual run. That is to say, the length information of ERLC is hided in the tail of codeword. So we can use this discipline to decompress. This discipline can be implemented by a special counter. The special counter has two characteristics: (1) setting the lowest bit of the counter to number 1. When the data needing to be decoded is shifted into the counter, the number 1 and other data are together shifted to high bits of the counter. (2) using non-0 low bound. Traditional counter counts down to 0, but this counter counts down to 2 (the content of counter is "10"). This function can be implemented easily by some combinational circuits, which will not introduce high hardware overhead.

The design of this decoder is similar to that of the EFDR using the FSM-based design. This decoder is embedded in chip, which has some characteristic of small size and simple structure. This decoder does not introduce hardware overhead obviously, and is independent of not only DUT, but precomputed test data as well. The block diagram of the decoder is shown in Figure 2. The decoder decodes the precomputed test data T_E to output T_D. The structure of this decoder is simple, which only needs a FSM, a special k+1 bit counter, a $\log_2$k bit counter and a k+1 bit register.

The signal bit_in is primary input, the coded test data is sent via bit_in to the FSM and the signal en is high, and the signal out outputting decoded data is valid while the signal v is high. The counter_in is a path which sends the tail section of codeword to the k+1-bit counter, and the signal shift is a signal that takes control of the counter_in. The dec1 is used to notify k+1-bit counter to start counting down, and the signal rst1 is high as the counting finishes. The signal dec2 indicates to the $\log_2$k-bit counter to start counting down, on the other hand, signal inc notifies it to start counting up. The signal rst2 indicates that the $\log_2$k-bit counter finished counting. The signal load tells when the content of k+1 bits register should replace the content of k+1 bits counter. The block diagram of the proposed ERLC decoder is given in Figure 2.

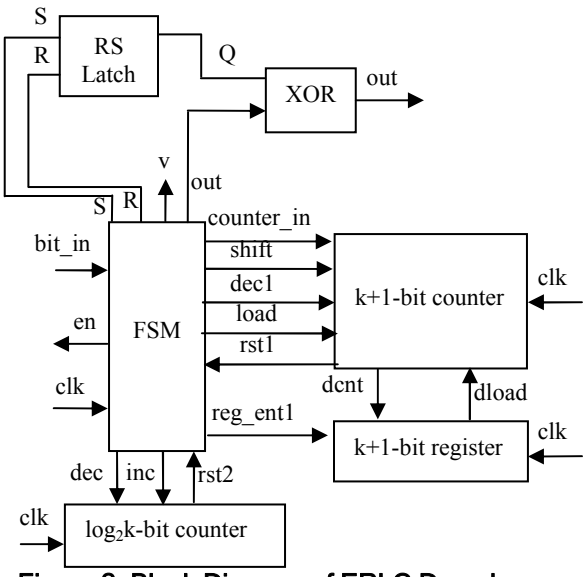

Figure 2. Block Diagram of ERLC Decoder

4. Experimental results

In this section we will verify the effectiveness of the proposed ERLC scheme by using experiment results. For comparison with other schemes, we adopted first the MinTest test sets[15], which are the same as [4][7-11][13-14]. Experiments were performed on the several largest ISCAS 89 benchmark circuits.

The compression ratio of this scheme compared with other schemes is shown in Table 2.

Table 2. Compression Obtained by Using Different Codes Methods (No Difference)

Circuit	Size of T_D bit	Scheme presented		Golomb		FDR α %
		Size /bit	α %	m	α %	
s5378	23754	12389	47.84	4	40.70	**48.02**
s9234	39273	22210	43.45	4	43.34	**43.59**
s13207	165200	32044	80.60	16	74.78	**81.30**
s15850	76986	25844	**66.43**	4	47.11	66.22
s35932	28208	5400	**80.83**	4	4.69	10.19
s38417	164736	67990	**58.73**	4	44.12	43.26
s38584	199104	76473	**61.59**	4	47.71	60.91
Avg.			**62.79**		43.21	50.50

The first column of Table 2 is the circuit name. The second column is test size of original test data. The compression effect is shown in third column of Table 2. The other columns are data size of compressed data and compression ratio of Golomb, and FDR coding. The compression ratio can be determined by:

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

T_D is original length of test data and T_E is the length of compressed data. In Table 2 the original test data did not do the operation of difference. The filling of don't-care bits is to make the length of consecutive runs the same as possible. For Example , for the test data slice 000000000001X00000000001X111X1XXXXXXX0000000000 1, it should be filled to 000000000001000000000001 111111111111 000000000001. If there are any don't care bits left, then they are filled randomly. From Table 2, as can be seen, the proposed scheme achieves higher compression than the Golomb, and FDR techniques for some of the circuits. The scheme achieves higher compression ratio than Golomb for all of the compared test sets. It also achieves higher compression ratio than FDR for four out of the seven compared test sets. On average, the percentage compressions of the proposed scheme are 19.58%, and 12.29% higher than those of Golomb, and FDR, techniques.

Next we present the experimental results on the peak and average power consumption during the scan-in operation. These results show that test data compression can also lead to significant savings in power consumption. We estimate power using the weighted transitions metric. Let P^C_{peak} (P^C_{avg}) be the peak (average) power with compacted test sets obtained using Mintest. Similarly, let P^G_{peak} (P^G_{avg}) be the peak (average) power when Golomb coding is used by mapping the don't cares in T_D to zeros and let P^{ERLC}_{peak} (P^{ERLC}_{avg}) be the peak (average) power when ERLC coding is used by mapping the don't cares in T_D to zeros or ones according to our program. Table 3 compares the average and peak power consumption for Mintest test sets for Golomb coding and our proposed scheme.

Table 3. Experimental Results on Peak and Average Scan-in Power consumption

Circuit	Mintest		Golomb		Proposed	
	Peak power	Average Power	Peak power	Average Power	Peak power	Average Power
s9234	17494	14630	12994	5692	13720	4534
s13207	135607	122031	101127	12416	94886	8190
s15850	100228	90899	81832	20742	70908	13815
s35932	707280	583639	172834	73080	107226	40395
s38417	683765	601840	505295	172665	438005	118628
s38584	572618	535875	531321	136634	481185	86415
Avg.	369499	324819	234233	70205	200988	45329

The first column of Table 3 is the circuit name. The second and third columns are the peak power and average power if original Mintest test sets are used. The fourth and fifth columns are the peak power and average power if Golomb coding is used. The last two columns are peak power and average power if our proposed scheme is used. Table 3 shows that the peak power and average power are significantly less if proposed scheme is used for test data compression and the decompressed patterns are applied during testing. On average, the peak (power) is 33.41% (12.43%) less in our proposed scheme than for the Golomb coding.

For a test application time (TAT) comparison, a simulator was implemented based on the TAT analysis for Golomb, FDR and ERLC. For Golomb and FDR decoders, it was assumed that the data is fed into the decoder at the ATE operating frequency and the internal FSM reaches a stable state after one internal clock cycle. In order to provide an accurate comparison, the same group size is used. The results are reported in Table 4. The first column of Table 4 is the circuit name. The second column of Table 4 is the frequency of ATE working. The last three columns are TAT of different codes schemes. From Table 4, it can be seen that our proposed scheme reduces the test application time compared with that of Golomb and FDR. From Table 6, it is found that the test application time of Golomb and FDR is respectively 1.96 times and 1.72 times of that of proposed scheme.

Table 4. TAT Comparison for Different Codes Schemes

Circuit	f_{ATE} (MHz)	Test Application Time (ms)		
		Proposed	Golomb	FDR
s5378	50	0.237	0.333	0.294
s9234	50	0.413	0.514	0.483
s13207	50	0.619	1.328	1.260
s15850	50	0.510	0.943	0.793
s38417	50	1.282	2.196	1.869
s38584	50	1.520	2.550	2.219
Avg.		0.668	1.311	1.153

In a word, what can be concluded from the experimental results is that our proposed scheme can achieve better compression result, lower power, and shorter application time at lower hardware overhead, which is acceptable.

5. Conclusion

Test data compression is an effective solution to reduce the increasing volume of test data. A new test data compression scheme (ERLC) is proposed. This scheme first considers both types of runs of 0's and 1's, then further explores the relationship between two consecutive runs on the basis of traditional coding characteristic which uses shorter codeword to represent longer symbol in the original test data. This proposed scheme uses two short codewords to represent the whole second run of two consecutive runs, the lengths of which are the same. It is shown that the presented scheme decreases the ATE memory and channel capacity requirements by obtaining

good compression ratios. Moreover, this scheme can reduce power dissipation. Thus, it is an effective solution for test data compression/decompression of IC design.

6. References

[1] X. Chen, M. S. Hsiao, "Testing embedded sequential cores in parallel using spectrum-based BIST", *IEEE Transactions on Computers*, 2006, 55(2), pp. 150-162.

[2] Aiman El-Maleh, Saqib Khursheed, and Sadiq Sait, "Efficient Static Compaction Techniques for Sequential Circuits Based on Reverse Order Restoration and Test Relaxation", *IEEE Transactions on Computer-Aided Design Of Integrated Circuits And Systems*, 2006, 25(11), pp. 2556-2564.

[3] N. A. Touba, "Survey of Test Vector Compression Techniques", *IEEE Design & Test of Computers*, 2006, 23(4), pp. 294- 303.

[4] Wenfa. Zhan, Huaguo. Liang, Feng. Shi et al., "Test data compression scheme based on variable-to-fixed-plus-variable-length coding", *Journal of Systems Architecture*, 2007, 53 (11), pp. 877- 887.

[5] L. Lei, , and K. Chakrabarty, "Test data compression using dictionaries with fixed-length indices", *Proc. VLSI Test Symposium*, pp. 219-224, 2003.

[6] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, "An Efficient Test Vector Compression Scheme Using Selective Huffman Coding", *IEEE Trans. Computer-Aided Design*, 2003, 23 (6), pp. 797-806.

[7] T. Paul, B. Al-Hashimi, and N. Nicolici, "Variable-Length Input Huffman Coding for System-on-a-Chip Test", *IEEE Transactions on CAD of Integrated Circuits and Systems*, 2003, 22 (6), pp. 783-796.

[8] A. Chandra, and K. Chakrabarty, "System-on-a-Chip test data compression and decompression architectures based on Golomb codes", *IEEE Transitions on CAD of Integrated Circuits and System*, 2001, 20 (3), pp 355-368.

[9] A. Chandra, and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes", *IEEE Transactions on Computers*, 2003, 52 (8), pp. 1076-1088.

[10] A. Chandra, and K. Chakrabarty, "Reduction of SOC test data volume, scan power and testing time using alternating run-length codes", *Proc. IEEE/ACM Design Automation Conference*, pp. 673-678, 2002.

[11] El-Maleh A H., "Test data compression for system-on-a-chip using extended frequency-directed run-length code", *IET Computers & Digital Techniques*, 2008, 2 (3), pp. 155-163.

[12] M. Tehranipoor, M. Nourani, and K. Chakrabarty, "Nine-Coded Compression Technique for Testing Embedded Cores in SoCs", *IEEE Transactions on VLSI Systems*, 2005, 13(6), pp. 719-731.

[13] A. Chandra, K. Chakrabarty, "Test data compression for system-on-a-chip using Golomb codes", *Proceedings of the VLSI Test Symposium*, pp 113-120, 2000.

[14] Wenfa Zhan. "An Efficient Collaborative Test Data Compression Scheme Based on OLEL Coding", *Proceedings of the 12th Conference on Computer Supported Cooperative Work in Design(CSCWD 2008)*, pp. 1107-1111, 2008.

[15] Hamzaoglu, I., and Patel, J. H. "Test Set Compaction Algorithms for Combinational Circuits", *Proceedings of Int. Conf. Computer-Aided Design*, pp. 283-289, 1998.