第26卷 第11期 半 导 体 学 报 Vol.26 No.11
2005年11月 CHINESE JOURNAL OF SEMICONDUCTORS Nov.,2005

An Efficient Test Data Compression Technique Based on Codes*

Fang Jianping, Hao Yue, Liu Hongxia, and Li Kang

(Key Laboratory of Ministry of Education for Wide Band-Gap Semiconductor Materials and Devices, School of Microelectronic, Xidian University, Xi'an 710071, China)

Abstract: This paper presents a new test data compression/decompression method for SoC testing, called hybrid run length codes. The method makes a full analysis of the factors which influence test parameters: compression ratio, test application time, and area overhead. To improve the compression ratio, the new method is based on variable-to-variable run length codes, and a novel algorithm is proposed to reorder the test vectors and fill the unspecified bits in the pre-processing step. With a novel on-chip decoder, low test application time and low area overhead are obtained by hybrid run length codes. Finally, an experimental comparison on ISCAS 89 benchmark circuits validates the proposed method.

Key words: test data compression; unspecified bits assignment; system-on-a-chip test; hybrid run-length codes

EEACC: 1200; 1265A; 1130B

CLC number: TP391 **Document code:** A **Article ID:** 0253-4177(2005)11-2062-07

1 Introduction

With the steady increase in complexity of system-on-a-chip (SoC) designs, testing is an important factor that determines the cost of the design. The test data volume is a major problem encountered in the testing of SoC designs. The increasingly large volume of SoC test data is exceeding the memory and I/O channel capacity of commercial automatic test equipment (ATE). For closing the ever-increasing gap between ATE speed, channel, and memory requirements versus SoC test data volume, there are two main potential solutions. The first solution is the built-in self-test (BIST)[9], and it has emerged as an alternative to ATE-based external testing. But the BIST is now extensively used for memory testing, and not as common for logic testing. The BIST also introduces performance penalties. The second solution is based on the use of test data compression/decompression techniques[1~8]. This solution does not introduce performance penalties and guarantees full reuse of the existing embedded cores. Recently, several efficient test data compression techniques have been proposed, such as statistical codes[1], Golomb codes[2], frequency-directed run-length (FDR) codes[3], selective Huffman codes[5], and alternating run-length codes[8]. In this paper, a new class of variable-to-variable run length compression codes is presented, which is referred to as hybrid run-length codes. The method is based on analyzing the factors that influence test parameters: compression ratio, test application time, and area overhead. Unlike other previous approaches[1~3,5,8], which improve some test parameters at the expense of the others,

* Project supported by the National High Technology Research and Development Program of China (No. 2003AA1Z1630) and the National Natural Science Foundation of China (No. 60206006)

Fang Jianping male, was born in 1978, PhD candidate. His research interests are in VLSI design and testing.

Hao Yue male, was born in 1958, professor. His research interests include reliability of ultra-deep submicron devices, novel wide band-gap semiconductor materials, and VLSI design.

Received 15 April 2005, revised manuscript received 10 June 2005 ©2005 Chinese Institute of Electronics

the proposed method is capable of improving all the three parameters simultaneously.

2 Hybrid run-length codes

2.1 Compression algorithm

In this section, the hybrid run-length codes algorithm is described. It is a variable-to-variable length coding, which maps variable-length runs of 0s to variable-length code words. The hybrid code is constructed as follows: the runs of 0s are divided into groups $A_1, A_2, A_3, \cdots, A_k$, respectively, and k is determined by the run-length L of test vector and the bits width of tail words (L_t). The block size of prefix L_p is determined by L_t ($L_p = L_t + 1$). The encoding procedure of $L_t = 1$ is shown in Fig. 1. The hybrid code has the following properties: (1) The tail section consists of two parts: the flag bit and the tail words. The flag bit is used to obtain lower area overhead, which is a constant "zero"; (2) The prefix is constituted of blocks, and the first bit of each block is the constant "one"; (3) For $L_t = 1$, the size of the kth group is equal to 2^k, i.e., A_k contains $2k$ members; (4) The codeword contains prefix, flag, and tail words. In order to show the compression efficiency of the new coding strategy, a probability analysis for the test data source is presented and hybrid run length coding is compared with FDR codes, Golomb codes, and entropy bounds. Defining a test set that produces 0s and 1s with probabilities $p(0 < p < 1)$ and $1 - p$, respectively. The smallest and the longest run lengths that belong to group A_k are $2^k - 2$ and $2^{k+1} - 3$ at $L_t = 1$. Whereas, if $L_t = 2$, the smallest and the longest run lengths are $(2^{2k} - 7)/3$ and $(2^{2k+2} - 7)/3$, respectively. So the compression gain of hybrid run length codes at $L_t = 1$ is given by

$$\beta_{H(L_t=1)} = \lambda / \overline{H_{L_t=1}} = 1/2(1-p)\sum_{k=1}^{\infty} p^{2^k - 2} \quad (1)$$

where λ is the average number of bits in any run generated by the data source and $\overline{H}$ is the average codeword length for hybrid run length. If $L_t = 2$, the compression gain also can be obtained, which is given by

$$\beta_{H(L_t=2)} = \lambda / \overline{H_{L_t=2}} = 1/(1-p) \times \left(\sum_{n=1}^{\infty} 3n\left(p^{(2^{2n}-7)/3}(1-p^{2k})\right)\right) \quad (2)$$

We now compare the compression gain obtained by hybrid run length codes with the different codes in Refs. [2, 3].

Run length	Group	Prefix ($L_p = 2$)	Tail Flag	Tail L_t	Code words
0	A_1	No	0	0	00
1		No	0	1	01
2	A_2	10	0	0	1000
3		10	0	1	1001
4		11	0	0	1100
5		11	0	1	1101
6	A_3	10_10	0	0	101000
7		10_10	0	1	101001
8		10_11	0	0	101100
9		10_11	0	1	101101
……	……	……	……	……	……

Fig. 1 Hybrid run-length coding ($L_t = 1$)

Figure 2 shows that the compression gain of hybrid codes is always higher than that of Golomb codes for all values with $p > 0.91$. At $L_t = 1$, the compression gain is the same as FDR codes. Whereas, there is a significant difference between β_H and β_F, when $L_t = 2$ and $p > 0.95$.

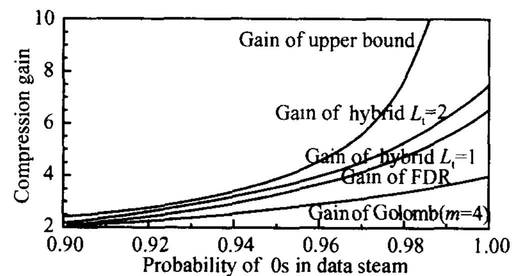

Fig. 2 Comparison of compression gain obtained with different codes

2.2 Pre-processing of test data

To improve the compression ratio of hybrid run length codes, before encoding the test sets, an efficient test vectors reordering algorithm and a dynamic do not cares bits filling procedure used in the pre-processing step are presented, called the itera-

tive sort filling strategy (ISFS). It assigns the donot cares bits in the original test set to zeros or ones. At the first step, it finds an unsorted vector from the original test set (array_input) randomly. Then it calculates the minimal and maximal hamming distances between the vector and the last vector in the array_output. According to the hamming distance, it fills do not care bits of the vector. Hamm_D is the threshold value of the hamming distance defined by user. The algorithm of the proposed procedure is shown in the following program.

Sort(Hamm_D, Hamm_L)
sort the first line: Array_Output[0] <= Array_Input[x];
while (sort process is not finish)
label1: Find an unsorted line;
label2: Calculate Hamming distance between Array_Input[i] and Array_Output[j];
if (Min_D > Hamm_D) //current Array_Output[j] is not good
  Find another unsorted line and goto label2;
else (Array_Output[j] has not changed) //change the filling strategy of Array_Output[j]
  change the x value using different bit index for Array_Output[j] and goto label1;
else //assort current Array_Output[j]
  j <= j - 1, Array_Output[j] <= Array_Input[x], Hamm_L <= (Hamm_L - 1) and goto label1;
else
  Fill X bit value for Array_Input[i] and Array_Output[j]; //let hamming distance equals to Max_D
endif
endwhile

3 Test application time analysis

Test application time (TAT) is firstly influenced by the compression ratio, and then by the type of the on-chip decoder. TAT is determined by the time needed to transport the code words into the on-chip decoder and the time needed to decode the compressed test set for internal scan chains. The upper bound of TAT can be obtained on the assumption that decoding begins after complete code words are transferred from the ATE. And if the decoder has a parallel structure, the lower bound of TAT can be obtained, which is determined by the larger one of T_{shift} and T_{decode}. If the T_{shift} is the time required to transport the code words and T_{decode} is the time required to decode the compressed test sets, the TAT can be given by

$$\max(T_{shift}, T_{decode}) \leq TAT \leq TAT_{up} = T_{shift} + T_{decode} = \frac{L_{cw}}{f_{ATE}} + \frac{L_{src}}{f_{scan}} = \frac{1}{f_{ATE}}\left(L_{cw} + \frac{L_{src}}{a}\right) \quad (3)$$

where $a = f_{scan}/f_{ATE}$ is the ratio between the on-chip test frequency (f_{scan}) and the ATE operating frequency (f_{ATE}), L_{cw} is the total length of the code words, and L_{src} is the length of the data source. The experimental results of TAT of hybrid run-length codes and the comparison with other codes methods will show in Sec. 5.

4 Test data decompression

In this section, we illustrate the design of the decoder of the hybrid run length codes. The design is different from the FSM-based decoder in Refs. [2,3,8]. The block diagram of the decoder is shown in Fig. 3. Due to the tail flag bit in the code words, the controller of the decoder is really simple and can be efficiently implemented. The tail flag is used to distinguish between the prefix section and the tail section of the code words, and then put them into different shift registers.

The ctl is the input signal used to control the shift of prefix or tail bits. The shift registers store the input code words, and are divided into two parts, the even bits and the odd bits. These two groups of registers, the adder and the selector C are used to count the length of the even and the odd prefix, respectively. The B[1:0] is one of the input signals of the adder. The signal Dout is the decoder output. For group size k, the decoders of FDR

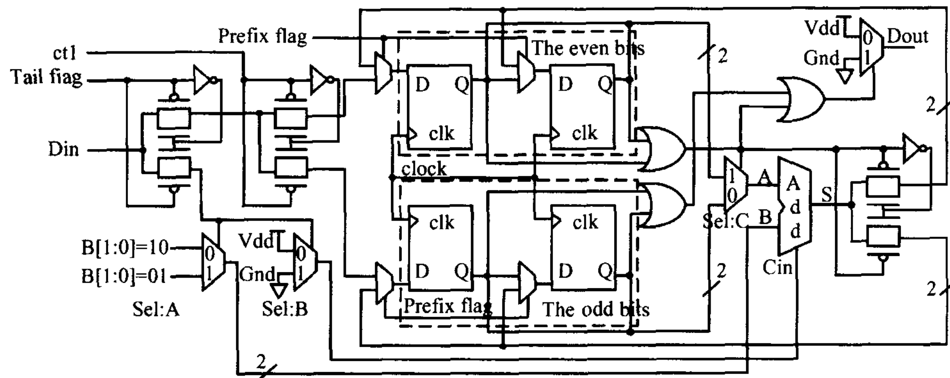

Fig. 3 Decoder used for on-chip pattern decompression

and Golomb are formed from a FSM ,at the least 9 states ,and a $k + \log_2 k$ bits counter. Compared with the decoders of FDR and Golomb ,the proposed decoder has only an internal buffer ,a shift register of size $2k$,and an adder. Thus ,for the same group size k ,a visible reduction in area overhead is obtained. For all of these on-chip decoders ,with the different group size of 4 ,8 ,and 16 ,the statistical and comparative data of the area overhead are illustrated in Table 1. All of these circuits are synthesized with the synopsys design compiler and mapped to the 2-input nand cell of the TSMC35 library. As shown in Table 1 ,the area overhead of Golomb is up to larger one time than the hybrid at $L_t = 1$. And the area overhead of the FDR is larger than that of the hybrid at the different values of L_t and group sizes.

Table 1 Area overhead comparison

Compression method	Area overhead (map to 2-input nand gate)		
	Group size		
	4	8	16
Golomb	125	227	307
FDR	320		
Hybrid ($L_t = 1$)	76	138	212
Hybrid ($L_t = 2$)	118	182	267

On the other hand, the simple and efficient structure of the decoder is also beneficial to reduce the TAT. As described in Sec. 3, TAT is determined by the T_{shift}, T_{decode} ,and type of decoder. Because the decoder of the hybrid is made up of the shift registers and adder ,it almost does not need any wait cycles between the transfer and decoding processes. For example, when the code words transfer into odd bits of the prefix shift register at the ATE operating frequency f_{ATE} ,the decoder can generate patterns at on-chip test frequency f_{scan} and output from Dout immediately. On the next stage, the ATE can transfer the next code words into the even bits of the prefix shift register ,and the decoder will begin to generate the second pattern after the first pattern has fed into the internal scan chains. Therefore ,the decoder of the hybrid almost works in parallel characteristic as long as $f_{scan} > f_{ATE}$.

5 Results

To validate the efficiency of the new method, experiments were performed on the full scan version of the large ISCAS 89 benchmark circuits. These benchmark circuits are sequential, synchronous ,and use only D-type flip-flops. For example, S9234 ,S13207 ,S15850 ,S38417 ,and S38584 are real-chip based and rely on partial scans. Especially, S35932 ,S38417 ,and S38584 are the largest circuits in the ISCAS 89 and the scales of combinational gates are 16065, 22179, and 19253, respectively. Test sets used in these experiments were obtained by MinTest dynamic compaction, which is also used in Refs. [2 ,3 ,8]. MinTest is an advanced ATPG system presented by Hamzaoglu and Patel in 1998 ,and it is considered as one of the best test set compaction tools. The proposed hybrid run length

codes were implemented in C++.

In the first set of experimental data, as shown in Table 2, the compression percentage (cp) of different codes methods were compared. Before the test sets are encoded by hybrid run length codes, the iterative sort filling strategy (ISFS) is used to reorder the test vectors and fill the unspecified bits in the test vectors. Table 2 clearly shows that the proposed method leads to better compression ratios, especially at $L_t = 2$. TD is the original test data bits of the source data and TE is the code words bits after encoding. The cp is $(1 - TE/TD) \times 100\%$. On average, the percentage compressions of hybrid run length codes at $L_t = 1$ are 16.74%, 11.04%, and 5.81% higher than that of Golomb, FDR and alternating run length codes. These values are increased to 17.88, 12.18, and 6.87, respectively at $L_t = 2$, and is better the values than at $L_t = 1$. All of these validate the efficiency of the proposed method.

Table 3 shows that the ISFS is not only efficient for use with hybrid run length codes, but also suitable for other compression codes to improve the compression ratios. Before the test sets are encoded by Golomb or FDR codes, we can reorder and fill the unspecified bits in them by ISFS to increase the percentage of zero, namely 0s probability. According to Section 2.1, the 0s probability is directly proportional to the compression ratio. Table 3 shows that the percentage of zero increases up to 90.31% on average after using ISFS. Table 3 also describes that Golomb and FDR codes obtain an obvious increase of the compression percentage by the ISFS in the pre-processing step of encoding. As table shows, there is an increase about 7.49% for the Golomb codes and 13.20% for the FDR codes by the ISFS.

Table 2 Compression obtained using different codes methods

Circuit	Size of TD/bit	Golomb[2]		FDR[3]	Alt run length[8]	Hybrid run length $L_t=1$		Hybrid run length $L_t=2$	
		m	cp/%	cp/%	cp/%	Size of TE	cp/%	Size of TE	cp/%
S5378	23754	4	40.70	48.19	N/A	10347	56.44	9987	57.96
S9234	39273	4	43.34	44.88	44.97	15376	60.85	16068	59.09
S13207	165200	16	74.78	78.67	80.23	25093	84.81	21847	86.78
S15850	76986	4	47.11	52.87	65.83	22371	70.94	21201	72.46
S35932	28208	N/A	N/A	10.19	N/A	14739	47.75	15928	43.53
S38417	164736	4	44.12	54.53	60.56	60187	63.47	58072	64.75
S38584	199104	4	47.71	52.85	61.14	76208	61.73	71641	64.02

Table 3 Compression ratios obtained after using iterative sort filling strategy

Circuit	Size of TD/bit	Max of run length	Ave of run length	Percentage of zero/%	Golomb coding		FDR coding	
					Size of TE	cp/%	Size of TE	cp/%
S5378	23754	153	10.11	90.10	12085	49.13	10090	57.52
S9234	39273	226	12.25	91.83	18009	54.14	16826	57.16
S13207	165200	443	36.11	97.63	51387	68.89	26516	83.95
S15850	76986	573	17.19	94.18	30596	60.26	22744	70.46
S35932	28208	1601	3.91	74.39	26687	5.40	18024	36.10
S38417	164736	1366	12.32	91.88	74727	54.64	57500	65.10
S38584	199104	746	12.91	92.25	88255	55.67	71168	64.26

For a TAT comparison, a simulator was implemented based on the TAT analysis for Golomb[2], FDR[3], and the hybrid run length codes. 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 other words, they need a wait cycle. However, as described in Sec. 4, the decoder of the hybrid does not need wait cycles between the transfer and the

decoding processes, so TAT is determined by the larger one of T_{shift} and T_{decode}. In order to provide an accurate comparison, we use the same group size as in Table 2. The results are reported in Table 4.

Table 4　TAT comparison for different codes methods

Circuit	f_{ATE}/MHz	a	Test application time/ms			
			Golomb[2]	FDR[3]	Hybrid runlength $L_t=1$	Hybrid runlength $L_t=2$
S5378	50	4	0.333	0.294	0.247	0.239
S9234	50	4	0.514	0.483	0.343	0.359
S13207	50	4	1.328	1.260	0.897	0.782
S15850	50	4	0.943	0.793	0.489	0.463
S35932	50	4	0.658	0.412	0.240	0.259
S38417	50	4	2.196	1.869	1.502	1.449
S38584	50	4	2.550	2.219	1.802	1.694

6　Conclusion

According to the demonstration in the above sections and the experiments results, we can draw a conclusion that compression ratio is influenced by the filling and reordering algorithm and by the compression algorithm, that area overhead is influenced by the feature of the decoder, and that test application time is influenced by the compression ratio and the type of the on-chip decoder. This paper presents a new compression method called hybrid run length codes. Unlike previous approaches that reduce some test parameters at the expense of the others, the proposed compression method is capable of minimizing test parameters simultaneously. This is achieved by accounting for multiple interrelated factors that influence the results, such as pre-processing the test set to the coding algorithm with ISFS, and the type of the decoder. The results in Section 5 show that the proposed method obtains constantly better compression ratios compared with references. Furthermore, the decoder leads to saving more hardware overhead compared with decoders in the references. It is shown that the proposed method decreases the ATE memory and channel capacity requirements by obtaining good compression ratios. Thus, it is an effective solution for test data compression/decompression for SoC designs.

References

[1] Jas A, Ghosh-Dastidar J, Touba N A. Scan vector compression/decompression using statistical coding. Proceeding of 17th IEEE VLSI Test Symposium, Dana Point, California, 1999:114

[2] Chandra A, Chakrabarty K. System-on-a-chip test data compression and decompression architectures based on Golomb codes. IEEE Trans CAD of Integrated Circuits and System, 2001, 20(3):355

[3] Chandra A, Chakrabarty K. Frequency-Directed run length (FDR) codes with application to system-on-a-chip test data compression. Proceeding of 20th IEEE VLSI Test Symposium, Marina Del Rey, California, 2001:42

[4] Gonciari P T. Improving compression ratio, area overhead, and test application time for SoC test data compression. Proc Design and Test in Europe, Paris, 2002:604

[5] Jas A, Ghosh-Dastidar J. An efficient test vector compression scheme using selective Huffman coding. IEEE Trans Computer-Aided Design of Integrated Circuits and Systems, 2003, 22(6):797

[6] Li L, Chakrabarty K. Test data compression using dictionaries and fixed-length indices. Proceeding of IEEE VLSI Test Symposium, Napa Valley, California, 2003:219

[7] Han Yinhe, Xu Yongjun. Test resource partitioning based on efficient response compaction for test time and tester channels reduction. Proceeding of Asian Test Symposium, Xi'an, 2003:440

[8] Chandra A, Chakrabarty K. Reduction of SOC test data volume, scan power and testing time using alternating run-length codes. Proceeding of IEEE/ACM, Design Automation Conference, New Orleans, Louisiana, USA, 2002:673

[9] Gao Haixia, Dong Gang. Improving detectability of resistive shorts in FPGA interconnects. Chinese Journal of Semiconductors, 2005, 26(4):683

一种基于编码的低硬件开销的测试数据压缩方法*

方建平　郝　跃　刘红侠　李　康

(西安电子科技大学微电子学院 宽禁带半导体材料与器件教育部重点实验室, 西安　710071)

摘要: 提出了一种新的测试数据压缩/解压缩的算法,称为混合游程编码,它充分考虑了测试数据的压缩率、相应硬件解码电路的开销以及总的测试时间.该算法是基于变长-变长的编码方式,即把不同游程长度的字串映射成不同长度的代码字,可以得到一个很好的压缩率.同时为了进一步提高压缩率,还提出了一种不确定位填充方法和测试向量的排序算法,在编码压缩前对测试数据进行相应的预处理.另外,混合游程编码的研究过程中充分考虑到了硬件解码电路的设计,可以使硬件开销尽可能小,并减少总的测试时间.最后,ISCAS 89 benchmark 电路的实验结果证明了所提算法的有效性.

关键词: 测试数据压缩; 不确定位填充; SoC 测试; 混合游程编码

EEACC: 1200; 1265A; 1130B

中图分类号: TP391　　**文献标识码**: A　　**文章编号**: 0253-4177(2005)11-2062-07

* 国家高技术研究发展计划(批准号:2003AA1Z1630)和国家自然科学基金(批准号:60206006)资助项目

方建平　男,1978 年出生,博士研究生,主要从事数字集成电路设计和测试技术研究.

郝　跃　男,1958 年出生,教授,博士生导师,主要从事新型宽禁带半导体材料与器件和 VLSI 设计技术的研究.

2005-04-15 收到,2005-06-10 定稿　©2005 中国电子学会