Information Technology Journal 12 (1): 204-208, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.204.208
© 2013 Asian Network for Scientific Information

Test Data Compression Scheme Based on Compatible Data Block Coding

[1,2]Jin Shang and [1]Liyong Zhang
[1]College of Measure-Control Technology and Communication Engineering,
Harbin University of Science and Technology, Harbin 150080, China
[2]School of Electrical and Information Engineering, Heilongjiang Institute of Technology,
Harbin 150050, China

Abstract: In order to reduce the storage requirements for the test pattern, a test data compression scheme based on compatible data block coding is presented. In the scheme, binary code is used to express the test data which are compatible or inversely compatible with reference data which are improved compression radio. The circuit structure of decompression with a Finite State Machine (FSM) and a Cyclical Scan Register (CSR) was proposed. Experimental results for the large ISCAS 89 benchmark circuits show that this scheme is a very efficient compression method than other compression schemes and the average compression ratio of up to 63.89%.

Key words: SoC, test data compression, coding, compatible data block, decompression

INTRODUCTION

With the complexity of VLSI continues to grow, more and more transistors are integrated on a single silicon die and test data volume have drastically increased. The channel bandwidth and memory capacity of Automatic Test Equipment (ATE) have their limits. Such it is difficult to transmit huge test data from ATE to System-on-a-Chip (SoC) (Rajski *et al*., 2004; Chang *et al*., 2010a; Liu *et al*., 2011; Gu *et al*., 2011). To solve this problem, various test data compression techniques have been proposed. There are three kinds of test compression techniques: linear-decompression scheme, broadcast-scan scheme and coding scheme (Touba, 2006; Chang *et al*., 2010b; Bala and Perinbam, 2006). Among them, coding scheme is a very popular-used one and it doesn't require any structural information of IP cores. In this scheme, a given test input T is compressed into T' by data compression techniques and stored in a VLSI tester storage. While a CUT on a chip is tested, the compressed test input set T' is transported to a decompression on the chip and then it is restored to T (Touba, 2006). The compressed test set can achieve the reduction of the time for test transportation, not just the size of the test storage device. Many coding schemes have been proposed for code-based scheme. Iyengar *et al*. (1998) have proposed Huffman coding, however, this method suffered from high area overhead. To cure this problem, the selective Huffman encoding technique was proposed (Jas *et al*., 2003). It only encodes the symbols with higher occurrence frequencies. The size of the decompression circuit can be reduced substantially. The optimal selective Huffman coding technique was proposed (Kavousianos *et al*., 2007), it was further reduced the test data. A 9C technique uses exactly nine code words aiming at pre-computed data of intellectual property cores in SoC. It is flexible in utilizing both fixed-length and variable-length blocks (Tehranipour *et al*., 2005). In addition, run-length methods can make a good trade-off between test data compression ratio and area overhead. Chandra and Chakrabarty (2001, 2003) improved using variable-to-variable encoding techniques: Golomb coding and FDR coding, El-Maleh (2008a) proposed an enhanced coding scheme named Extended Frequency-Directed Run-length (EFDR). This method took advantage of both runs of 0's and runs of 1's and outperformed the other coding techniques that are based on only runs of 0's. PRL is also an efficient approach. The compression is data-independent and the program for decompression is very small and simple, thereby allowing fast and high throughput to minimize test time (Ruan and Katti, 2006). El-Maleh (2008b) have proposed a new technique: a block merging technique, in which good compression effect was achieved by encoding runs of fixed-length blocks, only the merged block and number of block merged are recorded.

This study presents a new compatible data block techniques that reduce test data during test operation. The basic idea of this technique is take advantage of

Corresponding Author: Jin Shang, School of Electrical and Information Engineering,
Heilongjiang Institute of Technology Harbin 150050, China

exists a large number of don't-care bit in test data which random assigned values (0 or 1) according to the need and test data blocks after assigned values have relationship of compatible and inversely compatible. Compatibility or inversely compatibility with the data blocks are combined into one group, binary code is used to express the test data which are compatible or inversely compatible with reference data, by this way test data is compressed. The method considers combinational circuits or full scan sequential circuits and has no impact on the initial defect coverage. Compared with existing test data compression methods, this technique is the most efficient solution proposed up to now. This feature is illustrated with the experimental results gathered on the benchmarks.

DESCRIPTION OF THE CODING SCHEME

Two patterns are recognized as compatible if every bit pair at the same position has the same value or any of them is a don't-care bit(x bit). For example, test data "01xx" and "0x11" can be known as compatible; on the contrary, two patterns are recognized as inversely compatible if every bit pair at the same position has the inverse value or any of them is a don't-care (x bit). For example, test data "01xx" and "0x11" can be known as inversely compatible.

In general, test patterns generated by ATPG tools contain massive don't-care bit (x bit) which can be assigned with 1's and 0's in a way to skew the frequency distribution. Such as in the ISCAS 89 benchmark circuit test set, the proportion of 0 and 1 is quite small, usually no more than 15%, the remaining parts are don't-care bits. This provides the possibility for some compression algorithms application. The basic idea of our proposed is Compatibility or inversely compatibility with the data blocks are combined into one group, binary code is used to express the test data which are compatible or inversely compatible with reference data, by this way test data is compressed.

For example, test data 01xx0x11x1xx can be divided into three sub-segments: 01xx, x1xx and 0x11, 01xx as the reference test data and last 2 sub-segments will be found compatible with the first sub-segment. In this case, the last 2 sub-segments can be coded 001, The first bit "0" means last 2 sub-segments and reference test data is compatible ("1" for inverse compatible), "01" represents the quantity of compatible data block. Table 1 shows the relationship of uncompressed data and compressed data.

In Table 1, sign "0" represents that data blocks are compatible and sign "1" represents that data blocks are inversely compatible, m bit binary code represents number of compatible or inversely compatible data block, here we take m = 3. About relationship of the value of m and

Table 1: Coding scheme

Data block No.	Signed bit	Codeword	Signed bit	Codeword
1	0	001	1	001
2	0	010	1	010
3	0	011	1	011
4	0	100	1	100
5	0	101	1	101
6	0	110	1	110
7	0	111	1	111
8	0	000	1	000

Table 2: Compression results

Group	Data block	Coding results	Compression results
S1	11x11111		11111111
S2	x111x111	100 100	100100
S3	xxx11111		01101011 10011
S4	01x01xxx		10110
S5	xx10xxx1	10011	
S6	xxxxx0xx		
S7	0xxxxxxx		
S8	10xx0xxx	10110	
S9	1xxxxxx0		
S10	x00xxxxx		

compression ratio, we will later analysis. In Table 1, 0 behind reference data block indicated data and reference data is not compatible or inverse compatible (1 for compatible or inverse compatible).

The following steps describe the proposed scheme:

- Given parameter k and m, the original data is divided into blocks of length of the data block
- From the initial position to choose the length of k data blocks as reference data block, if the followed $s = 2^m$ data block and reference data block is compatible (inversely compatible), then add 1 behind the reference data block and according to Table 1 coding, jump step 4 (6), else jump step 3
- If s>0, then s = s-1 and jump step 2, else jump step 4
- If the followed $j = 2^m$ data block and reference data block is inversely compatible, then according to Table 1 coding, jump step 6, else jump step 5
- If j>0, then j = j-1 and jump step 4, else jump step 6
- Add the sign of the end "0", a reference data block coding end

In the following, an example is given to illustrate the above encoding process. Assume a test data is 11x11111x111x111xxx1111101x01xxxxx10xxx1xxxxx0xx0x xxxxxx10xx0xxx1xxxxxx0x00xxxxx, in this example, test data is divided into ten sub-segments, Coding results as shown in Table 2, the original 80 bits test data is compressed 32 bits.

THE DECOMPRESSION ARCHITECTURE

Here, the decoder circuits will be introduced. The main components include a FSM, three

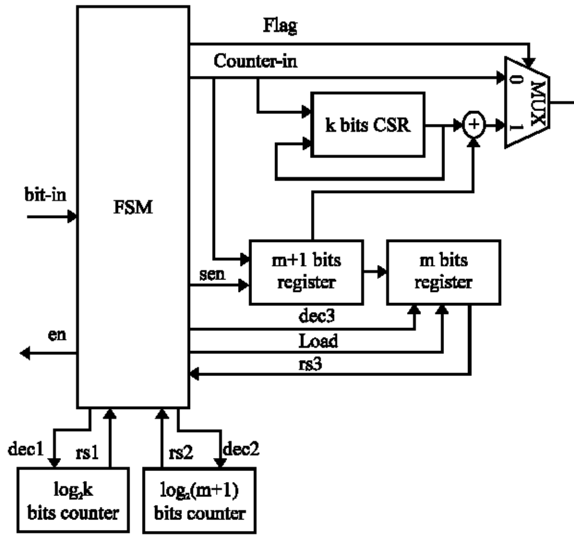

Fig. 1: Decompression architecture

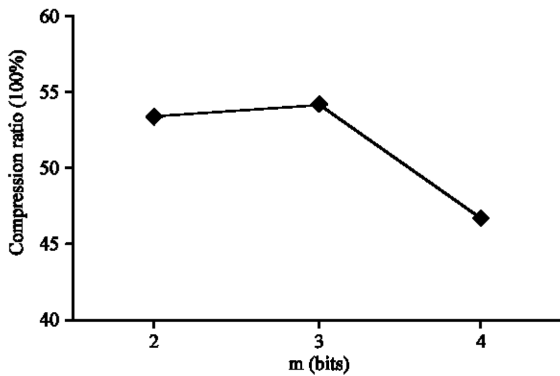

Fig. 2: The impact of different m on compression effect for circuit s5378

counters, two buffers, a XOR gate and a MUX. As shown in Fig. 1. The decompression circuits can be briefly illustrated as follows:

- FSM launches the EN = 1 signal to ATE from counter=in, simultaneously, dec1 = 1, $log_2k = log_2k-1$, the k bit blocks of reference data by MUX 0 channel directly into the scan chain and CSR, till rs1 is valid
- If input data from bit-in is 1, then jump step 3, else jump step 1
- dec = 1, sen is valid, $log_2(m+1) = log_2(m+1)-1$, the m+1 bit blocks of reference data by counter-in directly into the m+1 bit counter, till rs1 is valid and sen is invalid
- Load is valid, later m bit data in the m+1 bit buffer is read into m bit counter.

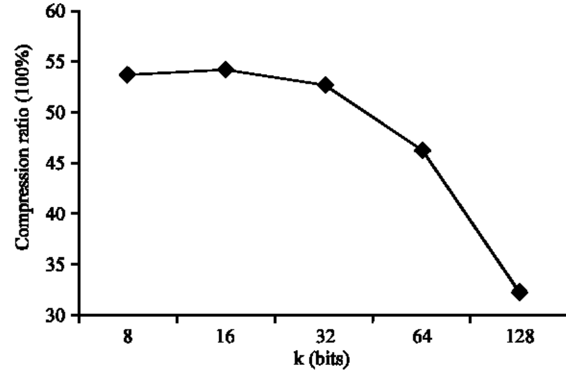

Fig. 3: The impact of different k on compression effect for circuit s5378

- dec1 = 1, $log_2k = log_2k-1$, till rs1 is valid and dec3 = 1, m = m-1
- If rs3 = 0, then jump step 5, else jump step 7
- If input data from bit-in is 0, then jump step 1, else jump step 3

EXPERIMENTAL RESULTS

We have implemented experiments on six large ISCAS 89 benchmark circuits, the compression effect is evaluated by the compression ratio which is defined as $CR = (|T_D|-|T_E|)/||T_D$, where $|T_D|$ is the size of the test set and $|T_E|$ is the size of compressed test set.

For a compressed sequence, the value of m is very important, compression effect is obviously different for different values of m. In Fig. 2, with the circuit s5378 as an example, the impact of varying m on the compression effect is obtained. As shown in Fig. 2, the compression ratio increases with the increase in m. It reaches the peak at m = 3 and then decreases as m continues to grow. For the compression types adopting a smaller m, due to fewer matches in compression types, only low compression is obtained. While for those adopting larger m, compression effect is negatively affected by the long exponent in each codeword.

In this scheme, for a different k value, the experimental results are also different. In Fig. 3, with the circuit s5378 as an example, the impact of varying k on the compression effect is obtained, where continue the above experiment, m is assumed 3 and k varies form 8 to 128. As shown, the best compression ratio occurs at k = 16.

In Table 3, the impact of k value on compression effect for the six circuits is obtained and reports the best results in the last column. As shown, four out of the six reach the best compression ratios at k = 8 while the other two at k = 16. Table 4 reports the compression rations

Table 3: The impact of different k on compression effect for six circuits

	Different k				
Circuit	8	16	32	64	Proposed
s5378	53.09	54.16	52.53	46.35	54.16
s9234	53.47	52.78	50.32	36.54	53.47
s13207	83.58	84.69	82.15	81.41	84.69
s15850	68.12	66.53	64.18	60.03	68.12
s38417	56.74	54.31	51.58	44.79	56.74
s38584	66.18	65.01	64.29	60.57	66.18
AVG	63.53	62.91	60.84	54.95	63.89

Table 4: Comparison of different compression schemes

Circuit	Golomb codes	Frequency-directed run-length (FDR) codes	9C codes	Proposed
s5378	40.7	48.02	51.64	54.16
s9234	43.34	43.59	50.91	53.47
s13207	74.78	81.3	82.31	84.69
s15850	47.11	66.21	66.38	68.12
s38417	44.12	43.26	60.63	56.74
s38584	47.71	60.91	65.53	66.18
AVG	49.63	57.22	62.9	63.89

compared with other methods such as Golomb, FDR and 9C. As can be seen from Table 4, the proposed compression techniques achieves higher compression ratio. The average compression ratio is increased by 14.26, 6.67 and 0.99% compared to Golomb, FDR and 9C.

CONCLUSIONS

As the increased complexity of SoC, much more faults are created accordingly. To detect them, a large amount of test data is essentially required. An actively researched area is to use the compression techniques to reduce the large test data volume. In this study, a new approach based on compatible data block coding is proposed for reduce test data volume. This approach taking advantage of the fact a large number of don't-care bit in test data which random assigned values (0 or 1) according to the need and test data blocks after assigned values have compatible and inversely compatible. Compatible data block coding can encode compatible and inversely compatible test data block ensures the growth of compression ratio. This method is suitable for various cores of unknown structures. The experiments carried out with six of the ISCAS 89 benchmark circuits show that the average compression ratio of up to 63.89%.

ACKNOWLEDGMENT

This study was supported by the National Science Foundation of China under Grant No. 61179024.

REFERENCES

Bala, G.J. and J.R.P. Perinbam, 2006. A novel low adiabatic data compressor. Inform. Technol. J., 5: 25-29.

Chandra, A. and K. Chakrabarty, 2001. System-on-a-Chip test data compression and decompression architectures based on Golomb codes. IEEE Trans. CAD of Integr. Circuits Syst., 20: 355-368.

Chandra, A. and K. Chakrabarty, 2003. Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes. IEEE Trans. Comput., 52: 1076-1088.

Chang, C., L. Gao, H. Kou, X. Liu, Z. Qin and G. Xu, 2010a. An integrity batch report scheme based on the waiting stack. Inform. Technol. J., 9: 79-88.

Chang, C.H., L.J. Lee, W.D. Tseng and R.B. Li, 2010b. 2^n pattern run-length for test data compression. Proceedings of the International Computer Symposium, December 16-18, 2010, Tainan City, Taiwan, pp: 562-567.

El-Maleh, A.H., 2008a. Efficient test compression technique based on block merging. IET Comput. Digital Tech., 2: 327-335.

El-Maleh, A.H., 2008b. Test data compression for system-on-a-chip using extended frequency-directed run-Length code. IET Comput. Digital Tech., 2: 155-163.

Gu, Y., Y. Li, J. Xu and Y. Liu, 2011. Novel model based on wavelet transform and GA-fuzzy neural network applied to short time traffic flow prediction. Inform. Technol. J., 10: 2105-2111.

Iyengar, V., K. Chakrabarty and B.T. Murray, 1998. Huffman encoding of test sets for sequential circuits. IEEE Trans. Instrum. Meas., 47: 21-25.

Jas, A., Ghosh-Dastidar, M.E. Ng and N.A. Touba, 2003. An efficient test vector compression scheme using selective Huffman coding. IEEE Trans. Comput. Aided Des. Inter. Circuits Syst., 22: 797-806.

Kavousianos, X., E. Kalligeros and D. Nikolos, 2007. Optimal selective huffman coding for test-data compression. IEEE Trans. Comput., 56: 1146-1152.

Liu, Z.C., X.F. Lin, Y.J. Shi and H.F. Teng, 2011. A micro genetic algorithm with Cauchy mutation for mechanical optimization design problems. Inform. Technol. J., 10: 1824-1829.

Rajski, J., J. Tyszer, M. Kassab and N. Mukherjee, 2004. Embedded deterministic test. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., 23: 776-792.

Ruan, X. and R. Katti, 2006. An efficient data-indepindent technique for compression test vectors in systems-on-a-chip. Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures, March 2-3, 2006, Karlsruhe, Germany.

Tehranipour, M., M. Nourani and K. Chakrabarty, 2005. Nine-coded compression technique for testing embedded cores in SoCs. IEEE Trans. Very Large Scale Integr. Syst., 13: 719-731.

Touba, N.A., 2006. Survey of test vector compression techniques. Des. Test Comput. IEEE, 23: 294-303.