Test Data Compression for System-on-a-Chip using Extended Frequency-Directed Run-Length (EFDR) Code

Aiman H. El-Maleh

Department of Computer Engineering

King Fahd University of Petroleum & Minerals

P.O. Box 1063, Dhahran 31261

Saudi Arabia

Email: aimane@kfupm.edu.sa

Abstract

One of the major challenges in testing a System-on-a-Chip (SOC) is dealing with the large test data size. To reduce the volume of test data, several test data compression techniques have been proposed. Frequency-directed run-length (FDR) code is a variable-to-variable run length code based on encoding runs of 0's. In this work, we demonstrate that higher test data compression can be achieved based on encoding both runs of 0's and 1's. We propose an extension to the FDR code and demonstrate by experimental results its effectiveness in achieving higher compression ratio.

I. INTRODUCTION

Advances in process technology have resulted in a very large density of integration allowing the design of systems containing millions of transistors on a single chip. It is predicted that the density of integrated circuits will be in the order of few billions in few years [1]. This increase in integration density has resulted in a tremendous increase in the test data volume needed to test those chips. In stored pattern testing, the test patterns need to be stored in the tester and then transferred to the circuit under test. The large test data volume not only increases the testing time but may also exceed the tester's memory capacity. This may result in the need of multiple reloads of the test data to the tester memory which is inefficient as it is in the order of tens of minutes to hours [2]. The cost of automatic test equipment (ATE) increases significantly with the increase in their speed, channel capacity, and memory. Thus, to reduce the testing time and cost, it is necessary to reduce the test data volume.

Test data volume reduction can be achieved based on utilizing Built-In Self Test (BIST) techniques [3-6], test compaction [7-10] and test compression techniques [13-32]. BIST techniques often combine the use of on-chip test pattern generation and test response compaction and some stored test patterns on the tester to achieve high fault coverage. While BIST techniques may result in significant test data volume reduction, they require longer test application time and they impose design constrains to make the design BIST ready. Test compaction techniques have the advantage of reducing the number of test patterns that need to be applied to the chip without reducing the fault coverage. While test compaction techniques reduce the test application time, recent results [11-12] have shown that compacted test sets may achieve less detection of non-modeled physical defects. However, test compaction is often applied to generated test sets. Test compression involves

storing the test data in the tester in a compressed form and having an on-chip decompression circuitry to decompress the test set before applying it directly to the chip. Thus, test compression techniques have the advantage of reducing the amount of data stored in the tester memory.

Some test compression techniques are based on utilizing structural information of the circuit as they require fault simulation and/or test pattern generation. Examples of these techniques are linear decompression-based schemes [13-15] and broadcast-scan-based schemes [16-18]. Other test compression techniques do not require circuit structural information and are more suitable for Intellectual Property (IP) cores. Examples of these techniques include statistical coding [19-20], selective Huffman coding [21], run-length coding [22], mixed run-length and Huffman coding [23], Golomb coding [24], frequency-directed run-length (FDR) coding [25], alternating run-length coding using FDR [26], geometric-primitive-based compression [27], MTC coding [28], variable-input Huffman coding (VIHC) [29], 9-coded compression [30] and dictionary-based coding [31-32]. Test data compression techniques in this class can be further classified as being either test-dependent or test-independent. The advantage of test-independent compression techniques is that the decompression circuitry does not need to be changed due to changes in the test set. Examples of test-independent test compression techniques include Golomb coding [24], frequency-directed run-length (FDR) coding [25], alternating run-length coding using FDR [26], geometric-primitive-based compression [27], MTC coding [28] and 9-coded compression [30].

In [22], a code word is used to encode a block of data based on the number of zeros followed by a one in that block. This technique is used for compressing fully specified test data that feeds a cyclical scan chain. A cyclical scan chain is used to decompress this data and transfer it to the “test scan chain”. Golomb code is a variable-to-variable run-length code that is used in [24] to enhance the scheme described above. It divides the runs into groups each is of size m. The number of groups is determined by the length of the longest run, and the group size m is dependent on the distribution of test data. Another enhancement to the work in [22] and [24] was proposed in [25]. It uses frequency-directed run-length (FDR) code, which is another variable-to-variable coding technique. It is designed based on the observation that the frequency of runs decreases with the increase in their lengths. Hence, assigning smaller code words to runs with small length and larger code words to those with larger length could result in higher test data compression.

The techniques in [22, 24, 25] are all based on encoding only runs of 0's. This was motivated based on the idea that encoding the difference vectors instead of the actual test vectors may reduce the number of 1's in the encoded data. However, the use of cyclical scan chain adds an extra design overhead as the cyclical scan chain length is equal to the test vector width. Hence, encoding the actual test data is preferred.

Based on test data analysis, we have observed that the frequency of runs of 1's is as significant as runs of 0's, for many of the circuits. This suggests that encoding both runs of 0's and 1's could result in higher test data compression as it could significantly reduce the number of encoded runs. In this work, we propose an extension to the FDR code to encode the test data based on encoding both types of runs.

The rest of this paper is organized as follows. In Section II, we briefly describe the frequency-directed run-length (FDR) code. In Section III, we show test data analysis that demonstrates the motivation of the proposed extended frequency-directed run-length (EFDR) code. EFDR code is then described in Section IV. The EFDR decoder design is described in Section V. In Section VI, experimental results are presented to demonstrate the effectiveness of the proposed EFDR code. Finally, Section VII concludes the paper.

II. FREQUENCY-DIRECTED RUN-LENGTH (FDR) CODE

Many of the test data compression techniques are based on run-length coding. A run is a consecutive sequence of equal symbols. A sequence of symbols can be encoded using two elements for each run: the repeating symbol and the number of times it appears in the run. Frequency-directed run-length (FDR) coding is a variable-to-variable coding technique based on encoding runs of 0's. In FDR code, the prefix and the tail of any codeword are of equal size. In any group A_i, the prefix is of size i bits. The prefix of a group is the binary representation of the run length of the first member of that group. When moving from group A_i to group A_{i+1}, the length of the code words increases by two bits, one for the prefix and one for the tail. Runs of length j are mapped to group A_i, where $i = \lceil \log_2(j+3) \rceil - 1$. The size of the i'th group is equal to 2^i, i.e., group A_i contains 2^i members. The FDR code for the first three groups is shown in Table I.

III. TEST DATA ANALYSIS

Based on test data analysis, it has been observed that test sets contain a large number of runs of 1's in addition to runs of 0's. By considering both types of runs, the total number of runs will decrease, which could result in higher test data compression.

To support this observation, we have analyzed test data for the largest ISCAS 85 and full-scanned versions of ISCAS 89 circuits. We have used test sets generated by MinTest [8], using both static and dynamic compaction. Test sets generated by dynamic compaction option have the letter *d* appended in their name. All the test sets used achieve 100% fault coverage of the detectable faults in each circuit. Test sets generated based on static compaction were relaxed, as this has the advantage of keeping unnecessary assignments as X's, which enables higher compression. The majority of test compression techniques rely on the fact that test sets contain a large number of X's. Efficient test relaxation techniques for combinational and sequential circuits have been recently proposed in [33-35].

Given a relaxed test set, techniques based on encoding only runs of 0's fill all the X's by 0's to reduce the number of runs that need to be encoded. However, to encode both runs of 0's and 1's in a test set, X's are filled as described in section IV to minimize the total number of runs that need to be encoded. This results in increasing test compression and reducing the switching activity during test scan which results in test power reduction.

Table II shows the analysis of the number of runs on the used test sets. The first column indicates the test set name. The second column shows the number of runs of 0's in the test set assuming that only runs of 0's will be encoded. The third, fourth, and fifth columns indicate the number of runs of 0's, runs of 1's, and the total number of runs, respectively, assuming that both types of runs will be encoded. As can be seen from the table, for most of the test sets, the number of runs of 1's is as significant as the number of runs of 0's. In 9 out of 16 test sets, the number of runs of 1's is higher than the number of runs of 0's. On average, the number of runs of 0's is slightly higher. For all the test sets, the total number of encoded runs is reduced and for some test sets the reduction is significant. The reduction in the total number of runs ranges from 28% (for test sets s13207 and s15850) to 94% (for test set s35932d). The average reduction in the total number of runs is nearly

54%. Reducing the number of encoded runs not only results in higher test compression but also in reducing the circuit switching activity during test and hence reduces the test power dissipation.

Figures 1, 2, and 3 show the frequency of both runs of 0's and runs of 1's for the first 200 run lengths for the following test sets: s15850d, s38417d, and s35932d, respectively. As can be seen from the figures, the frequency of runs of 1's follows a similar shape to that of runs of 0's with either a smaller magnitude or larger magnitude. For the test set s15850d, shown in Figure 1, it can be observed that the frequency of runs of 0's is higher than that of runs of 1's for most run lengths. However, for the test set s38417d, shown in Figure 2, we can see that the frequency of runs of 1's is more than that of runs of 0's for most run lengths. For the test set s35932d, shown in Figure 3, it can be observed that both runs of 1's and runs of 0's are scattered with higher frequency of runs of 1's for most run lengths.

IV. EXTENDED FDR (EFDR) CODES

To encode both runs of 0's and runs of 1's, we extend the FDR code based on adding an extra bit to the beginning of a code word to indicate the type of run. If the bit is 0, this indicates that the code word is encoding a run of type 0, otherwise it encodes a run of type 1. This code, called Extended FDR (EFDR), is shown in Table III. This code is a direct extention to the FDR code shown in Table I. However, in this code we do not have run length of size 0. This is because we are encoding both runs of 0's and runs of 1's. Note that runs of 0's are strings of 0's followed by a 1, while runs of 1's are strings of 1's followed by a 0, i.e. runs of 1's of length i are the complement of runs of 0's of the same length, and vice versa. As with FDR code, in this code when moving from group A_i to group A_{i+1}, the length of code words increases by two bits, one for the prefix and one for the tail. Runs of length j are mapped to group A_i, where $i = \lceil \log_2(j+2) \rceil - 1$. The size of the i'th group is equal to 2^i, i.e., group A_i contains 2^i members.

To illustrate the use of EFDR code, let us consider the following example. Consider the test T={0110001111111000000001}, of size 22 bits. The number of 0 runs in this test is 10. However, the number of both 0 and 1 runs is 5. Encoding this test using FDR code results in the encoded test T_{FDR}={01 00 1001 00 00 00 00 00 00 110010} of size 26 bits. Thus, for this example the number of bits needed to encode the test data using FDR code is more than the actual size of the original test data. However, encoding this test

using EFDR code, we obtain the encoded test T_{EFDR}={000 100 001 11011 0110000}, of size 21 bits. Obviously, for this example EFDR code outperforms FDR code. It should be observed that FDR coding suffers whenever we have runs of 1's, as each 1 bit will be encoded by a separate 0 run of length 0.

Filling don't care bits in EFDR plays a crucial rule in reducing the number of encoded runs and increasing test compression. In [36], a simple technique was used by filling X's by 1's if they are bounded by 1's from both sides; otherwise they are filled by 0's. In this work, another heuristic is used that results in more reduction in the number of encoded runs and higher test compression. In EFDR, runs could be either runs of 0's followed by a 1 or runs of 1's followed by a 0. Thus, the first known bit encountered in a run (i.e. 0 or 1) determines how all the X's encountered in the run will be filled. If that bit is 0, then all the X's in the run will be filled by 0's, otherwise they will be filled by 1's. The following example illustrates how X's are filled in EFDR vs. FDR and its impact on test compression. Consider the test T={1XXX10X1X1X101XXX00XX1}, of size 22 bits. FDR compression technique replaces all the X's by 0's and then encodes the test set. Thus, the test set before FDR encoding becomes T={1000100101010100000001} and results in the following FDR encoded test set T_{FDR}={00 1001 1000 01 01 01 110000} of size 22 bits. However, the test set before EFDR encoding becomes T={1111101111110111100001} and results in the following EFDR encoded test set T_{EFDR}={11010 11011 11001 01000}, of size 20 bits.

V. EFDR DECODER DESIGN

Since the EFDR compression technique is based on the FDR compression technique, the design of both decoders is similar with some additional components and different behavior in the finite state machine (FSM) of the EFDR decoder. An important difference is that in EFDR code, the minimum run length is 1 and that the runs could be of either type of runs of 1's or runs of 0's. Figure 4 shows the main components of the EFDR decoder design, which are described below. The encoded data is read through the *bit_in* signal and the *en* signal is used by the decoder to indicate its readiness to input a single bit. The signal *out* is generated by the FSM for outputting the decoded runs and the signal *v* indicates when the output is valid.

K-bit counter: This counter is used to store the codeword prefix and tail. Each run is composed of the total number of bits specified in the codeword prefix and tail in addition to an extra bit. It gets its input serially

from *counter_in* input and it will shift it to the left. The signals *shift*, *dec1*, and *rst1* are used to shift the data in, to decrement, and to indicate the reset state of the counter, respectively.

log_2 K-bit counter: The function of this counter is to count the number of bits of the codeword tail that need to be shifted to the *K-bit counter*. For each input shifted to the *K-bit counter* in the codeword prefix, the counter will increment. This is because the number of bits in the prefix and tail of a codeword are equal. For each shifted input of the codeword tail, the counter will decrement. The signals *inc*, *dec2*, and *rst2* are used to increment, to decrement, and to indicate the reset state of the counter, respectively.

SR-Latch: This latch will store the type of run to be decoded and stores a 1 if the run type is one, otherwise it stores a 0.

XOR gate: This gate will control the type of output generated depending on the run type. The FSM is designed to produce runs of zeros, so if the run type is one, the XOR gate acts as an inverter and the zeros will be converted to ones.

EFDR FSM: The EFDR FSM is composed of nine states with 3 inputs and 10 outputs as shown in Figure 5. The behavior of the EFDR FSM is summarized as follows. *S0* sets the *en* signal to the tester indicating its readiness for receiving the next encoded bit. In *S1*, the first bit indicating the run type is read and the *SR-latch* is set accordingly. In states *S2-S3*, the prefix is read and stored in the *K-bit counter* and the number of bits read is stored in the *log_2 K-bit counter*. In *S4*, a run of 0's is generated according to the prefix count stored in the *K-bit counter* plus an extra bit. In states *S5-S8*, the prefix code is shifted in to the *K-bit counter* and its size is determined by the *log_2 K-bit counter*. In *S9*, a run of 0's is generated according to the tail count stored in the *K-bit counter.* This is followed by generating a 1.

Similar to other compression techniques based on coding, the EFDR decompression circuitry requires synchronization with the tester. Techniques for handling synchronization with the tester can be employed as has been proposed in [37].

VI. EXPERIMENTAL RESULTS

In this section, we present results to compare the test compression ratio of the proposed EFDR technique to other test independent compression techniques including FDR. We also compare the required test application

time and decoder area between FDR and EFDR compression techniques. For both FDR and EFDR, a maximum run length of 2000 was assumed and was sufficient for all benchmark test sets. Comparison is made based on the test data for the largest ISCAS 85 and full-scanned versions of ISCAS 89 circuits generated by MinTest [8], using both static and dynamic compaction.

Table IV compares the compression results using FDR and EFDR codes. The first column shows the test set name and the second column shows the size of the original test set in bits. The third and fourth columns show the number of compressed bits using FDR and EFDR codes, respectively. The last two columns indicate the respective compression ratios. The *compression ratio* is computed as:

$$Comp.\,Ratio = \frac{\#\,Original\ Bits - \#\,Compressed\ Bits}{\#\,Original\ Bits}\ X\,100$$

As can be seen from the table, the EFDR compression technique improves test compression from an average compression of 45.17% to 57.87%. For all the test sets, EFDR achieves higher test compression than FDR. Significant improvements in the compression ratio are obtained for some of the test sets. For example, for the test set s35932, the compression ratio is improved from 3.99% using FDR to 47.04% using EFDR. For the test set s35932d of the same circuit, the compression ratio is increased from 19.36% using FDR to 82.49% using EFDR. This significant increase in test compression is not surprising as based on the statistics for these test sets given in Table II, the total number of runs reduces significantly when both types of runs are used versus using only 0 runs. For test set s35932, the number of encoded runs is reduced from 7554 to 2187. Similarly for test set s35932d, it is reduced from 10018 to 563. This is mainly due to the presence of a large number of strings of 1's in these test sets which are encoded as runs of 0's of length 0 in FDR while in EFDR they are encoded as a single run.

Table V compares the proposed technique to a number of test-independent compression techniques including Golumb [24], FDR [25], ALT-FDR [26], MTC [28] and 9C [30]. For all the compared test sets, EFDR compression technique achieves higher compression than Golumb [24], FDR [25], ALT-FDR [26] and MTC [28]. It also achieves higher compression than 9C [30] for four out of the six compared test sets. On average, it achieves higher test compression.

Table VI compares FDR and EFDR techniques in terms of the number of clock cycles needed to decompress the test sets, which corresponds to the test application time, and the estimated area of the synthesized decompression circuitry. The number of clock cycles and the estimated area are obtained based on VHDL models of the decompression circuitry of the two techniques. As can be observed, for all the test sets the number of clock cycles needed for decompression is less than FDR, correlating with the compression ratios achieved. This indicates that the EFDR compression technique will result in less test application time.

The decompression circuitry for the two techniques were synthesized using Xilinx Spartan2 XC2S100-6tq144 FPGA optimized for area. For both FDR and EFDR, the circuits were synthesized assuming a maximum run length of 2000. The estimated gate count for the EFDR decompression circuitry is slightly higher than that for FDR. However, the area overhead for the two techniques is considered small in comparison with the circuit size.

VII. CONCLUSION

In this work, we have proposed an extension to the recently proposed FDR code, namely Extended FDR (EFDR) code. The proposed technique is based on encoding both runs of 0's and 1's as opposed to encoding only runs of 0's. Based on experimental results on ISCAS benchmark circuits, it has been demonstrated that the proposed EFDR code outperformed FDR code and resulted in significant increase in test data compression ratio for all considered test cases, improving the compression ratio from 19.36% to 80.8% for one of the benchmark circuits. The proposed technique has also the additional advantage of reducing switching activity in the circuit resulting in reducing power during testing. This is evident by the reduction in the total number of encoded runs.

ACKNOWLEDGMENT

This work is supported by King Fahd University of Petroleum & Minerals under project FT2000/07. The author would like to thank Mr. Raslan Al-Abaji and Mr. Mohammad Kalkattawi for their help in this work,

REFERENCES

[1] Semiconductor Industry Association, International Technology Roadmap for Semiconductors, 2001 Edition. http://public.itrs.net/Files/2001ITRS/Home.htm

[2] Vranken, H., Hapke, F., Rogge, S., Chindamo, D., and Volkerink, E. : "ATPG padding and ATE vector repeat per port for reducing test data volume," Proc. Int. Test Conf., Charlotte, NC, Sep. 2003, pp. 1069–1078.

[3] Zorian, Y., Marinissen, E., and Dey, S.: "Testing embedded-core-based system chips," Comput. Mag., 1999, 32, (6), pp. 52–60.

[4] Hellebrand, S., Linag, H., and Wunderlich, H.-J.: "A mixed-mode BIST scheme based on reseeding of folding counters," Proc. Int. Test Conf., Atlantic City, NJ, October 2000, pp. 778–784.

[5] Wunderlich, H.-J., and Kiefer, G.: "Bit-flipping BIST," Proc. Int. Test Conf., Washington, DC, October 1996, pp. 337–343.

[6] Touba, N., and McCluskey, E.: "Altering a pseudo-random bit sequence for scan based BIST," Proc. Int. Test Conf., Washington, DC, October, 1996, pp. 167–175.

[7] Pomeranz, I., Reddy, L., and Reddy, S.: "COMPACTEST: A method to generate compact test sets for combinational circuits," Proc. Int. Test Conf., Nashville, TN, October 1991, pp. 194-203.

[8] Hamzaoglu, I., and Patel, J. H.: "Test set compaction algorithms for combinational circuits'', Proc. Int. Conf. Computer-Aided Design, San Jose, California, November 1998, pp. 283-289.

[9] Chang, J., and Lin, C.: "Test set compaction for combinational circuits," IEEE Trans. Computer Aided Design, 1995, 14, (11), pp. 1370-1378.

[10] El-Maleh, A., and Osais, Y.: "Test vector decomposition based static compaction algorithms for combinational circuits", ACM Transactions on Design Automation of Electronic Systems, 2003, 8, (4), pp. 430-459.

[11] Ma, S.C., Franco, P., and McCluskey, E.J.: "An experimental chip to evaluate test techniques Experimental results," Proc. Int. Test Conf., Washington, DC, October 1995, pp. 663-672.

[12] Reddy, S.M., Pomeranz, I., and Kajihara, S.: "On the effects of test compaction on defect coverage," Proc. IEEE VLSI Test Symp., Princeton, NJ, April 1996, pp. 430-435.

[13] Bayraktaroglu, I., and Orailoglu, A.: "Concurrent application of compaction and compression for test time and data volume reduction in scan designs," IEEE Trans. Computers, 2003, 52, (11), pp. 1480-1489.

[14] Wohl, P., Waicukauski, J.A., Patel, S., DaSilva, F. , Williams, T.W., and Kapur, R. : "Efficient compression of deterministic patterns into multiple PRPG seeds," Proc. Int. Test Conf., Austin, Texas, November 2005, pp. 916-925.

[15] Rajski, J., Tyszer, J., Kassab, M., and Mukherjee, N.: "Embedded deterministic test," IEEE Trans. on Computer-Aided Design, 2004, 23, (5), pp. 776-792.

[16] Samaranayake, S., Gizdarski, E., Sitchinava, N., Neuveux, F., Kapur, R., Williams, T.W.: "A reconfigurable shared scan-in architecture," Proc. 21th VLSI Test Symp., Napa Valley, CA, April 2003, pp. 9-14.

[17] Sitchinava, N., Gizdarski, E., Samaranayake, S., Neuveux, F., Kapur, R., Williams, T.W.: "Changing the Scan Enable During Shift," Proc. 22nd VLSI Test Symp., Napa Valley, CA, April 2004, pp. 73-78.

[18] Wang, L.-T. , Xiaoqing, Wen, Furukawa, H., Fei-Sheng, Hsu, Shyh-Horng, Lin, Sen-Wei, Tsai, Abdel-Hafez, K.S., Shianling, Wu: "VirtualScan: A new compressed scan technology for test cost reduction," Proc. Int. Test Conf., Charlotte, NC, October 2004, pp. 916-925.

[19] Iyengar, V., Chakrabarty, K., and Murray, B.: "Built-in self testing of sequential circuits using precomputed test sets," Proc. VLSI Test Symp., Princeton, NJ, April 1998, pp. 418–423.

[20] Jas, A., Ghosh-Dastidar, J., andTouba, N.: "Scan vector compression/decompression using statistical coding," Proc. VLSI Test Symp., San Diego, CA, April 1999, pp. 114–120.

[21] Jas, A., Gosh-Dastidar, J., Ng, M., and Touba, N.: "An efficient test vector compression scheme using selective Huffman coding," IEEE Trans. Computer Aided Design, 2003, 22, (6), pp. 797–806.

[22] Jas, A., and Touba, N.: "Test vector decompression via cyclical scan chains and its application to testing core-based designs," Proc. Int. Test Conf., Washington, DC, October 1998, pp. 458-464.

[23] Nourani, M., and Tehranipour, M.: "RL-Huffman encoding for test compression and power reduction in scan application," ACM Transactions on Design Automation of Electronic Systems, 2005, 10, (1), pp. 91–115.

[24] Chandra, A., and Chakrabarty, K.: "System-on-a-chip data compression and decompression architecture based on Golomb codes," IEEE Trans. Computer Aided Design, 2001, 20, (3), pp. 355–368.

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

[26] Chandra, A., and Chakrabarty, K.: "A unified approach to reduce SoC test data volume, scan power, and testing time," IEEE Trans. Computer Aided Design, 2003, 22, (3), pp. 352–363.

[27] El-Maleh, A., Al Zahir, S., and Khan, E.: "A geometric-primitives-based compression scheme for testing system-on-chip," Proc. VLSI Test Symp., Marina' Del Rey, CA, April 2001, pp. 54–59.

[28] Rosinger, P., Gonciari, P., Al-Hashimi, B., and Nicolici, N.: "Simultaneous reduction in volume of test data and power dissipation for system-on-a chip," Electron. Lett., 2001, 37, (24), pp. 1434–1436.

[29] Gonciari, P., Al-Hashimi, B., and Nicolici, N.: "Improving compression ratio, area overhead, and test application time for system-on-a-chip test data compression/decompression," Proc. Design Automation Test in Europe, Paris, France, March 2002, pp. 604–611.

[30] Tehranipour, M., Nourani, M., and Chakrabarty, K.: "Nine-coded compression technique for testing embedded cores in SoCs", IEEE Trans. on VLSI Systems, 2005, 13, (6), pp. 719-731.

[31] Li, L., and Chakrabarty, K.: "Test data compression using dictionaries with fixed length indices," Proc. VLSI Test Symp., Napa Valley, CA, April 2003, pp. 219–224.

[32] Wurtenberger, A., Tautermann, C., and Hellebrand, S.: "A hybrid coding strategy for optimized test data compression," Proc. Int. Test Conf., Charlotte, NC, Sep. 2003, pp. 451–459.

[33] El-Maleh, A., and Al-Suwaiyan, A.: "An efficient test relaxation technique for combinational and full-scan sequential circuits" Proc. VLSI Test Symp., Monterey, CA, April 2002, pp. 53-59.

[34] Miyase, K., and Kajihara, S.: "Don't care identification of test patterns for combinational circuits," IEEE Trans. Computer Aided Design, 2004, 23, (2), pp. 321-326.

[35] El-Maleh, A., and Al-Utaibi, K.: "An efficient test relaxation technique for synchronous sequential circuits" Proc. VLSI Test Symp., Napa Valley, CA, April 2003, pp. 179-185.

[36] El-Maleh, A., and Al-Abaji, R.: "Extended Frequency-Directed Run Length Code with Improved Application to System-on-a-chip Test Data Compression" Proc. of the 9th IEEE

International Conference on Electronics, Circuits and Systems, Dubrovnik, Croatia, September 2002, pp. 449-452.

[37] Gonciari, P. T., Al-Hashimi, B. and Nicolici, N.: "Synchronization overhead in SoC compressed test", IEEE Transactions on VLSI Systems, 2005, 13, (1), pp. 140-153.

Table I FDR code.

Group	Run Length	Group Prefix	Tail	Code Word
A_1	0	0	0	00
	1		1	01
A_2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
A_3	6	110	000	110000
	7		001	110001
	8		010	110010
	9		011	110011
	10		100	110100
	11		101	110101
	12		110	110110
	13		111	110111

Table II Analysis of number of runs in test data.

Test Set	Original Bits	Encoding 0 Runs	Encoding 0 and 1 Runs		
		#0 Runs	#0 Runs	#1 Runs	Total Runs
c2670	10252	1677	375	460	835
c5315	6586	1628	410	527	1015
c7552	15111	2695	975	715	1690
s5378	20758	2915	820	915	1735
s9234	25939	3843	1115	1280	2395
s13207	163100	4804	1937	1509	3446
s15850	57434	4635	1910	1415	3325
s35932	21156	7554	1066	1121	2187
s38417	113152	20970	3797	4318	8115
s5378d	23754	3537	958	1136	2049
s9234d	39273	4816	1661	1555	3216
s13207d	165200	5021	2005	1545	3550
s15850d	76986	5329	2222	1425	3647
s35932d	28208	10018	215	348	563
s38417d	164736	29473	4379	5169	9548
s38584d	199104	16814	5678	5093	10771
AVG		7858	1845	1783	3630

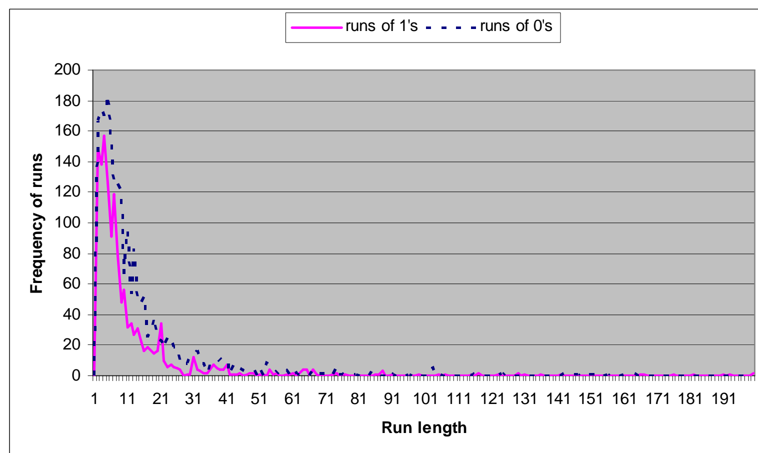

Figure 1 Distribution of runs of 0's and 1's for test set s15850d.

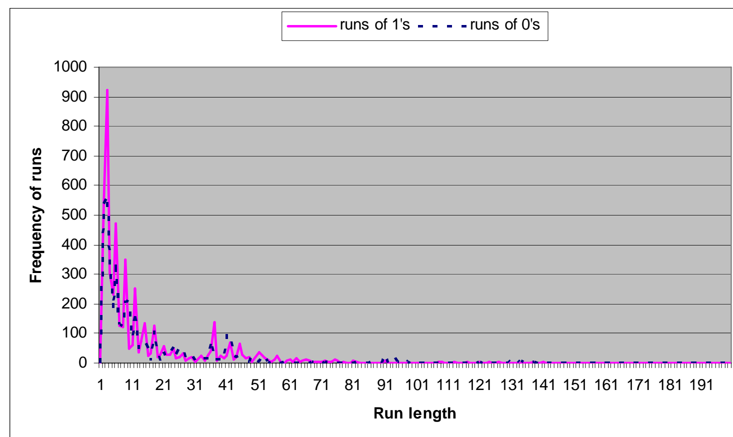

Figure 2 Distribution of runs of 0's and 1's for test set s38417d.

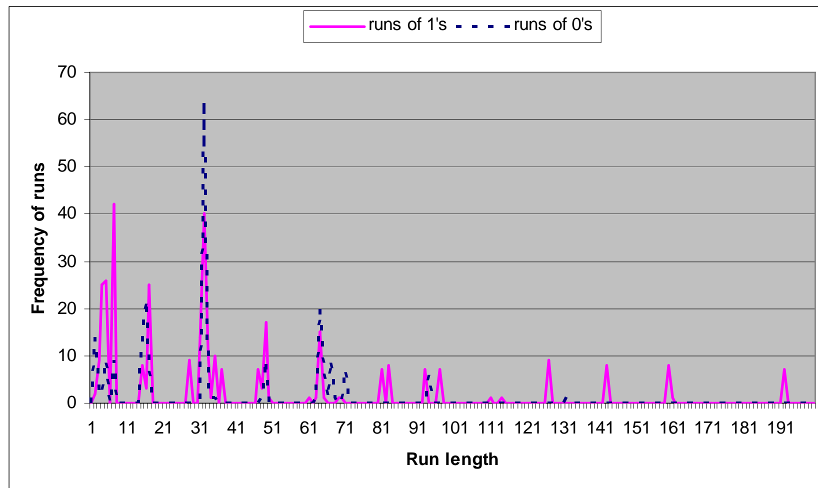

Figure 3 Distribution of runs of 0's and 1's for test set s35932d.

Table III Extended FDR (EFDR) code.

Group	Run Length	Group Prefix	Tail	Code Word Runs of 0's	Code Word Runs of 1's
A_1	1	0	0	000	100
	2		1	001	101
A_2	3	10	00	01000	11000
	4		01	01001	11001
	5		10	01010	11010
	6		11	01011	11011
A_3	7	110	000	0110000	1110000
	8		001	0110001	1110001
	9		010	0110010	1110010
	10		011	0110011	1110011
	11		100	0110100	1110100
	12		101	0110101	1110101
	13		110	0110110	1110110
	14		111	0110111	1110111

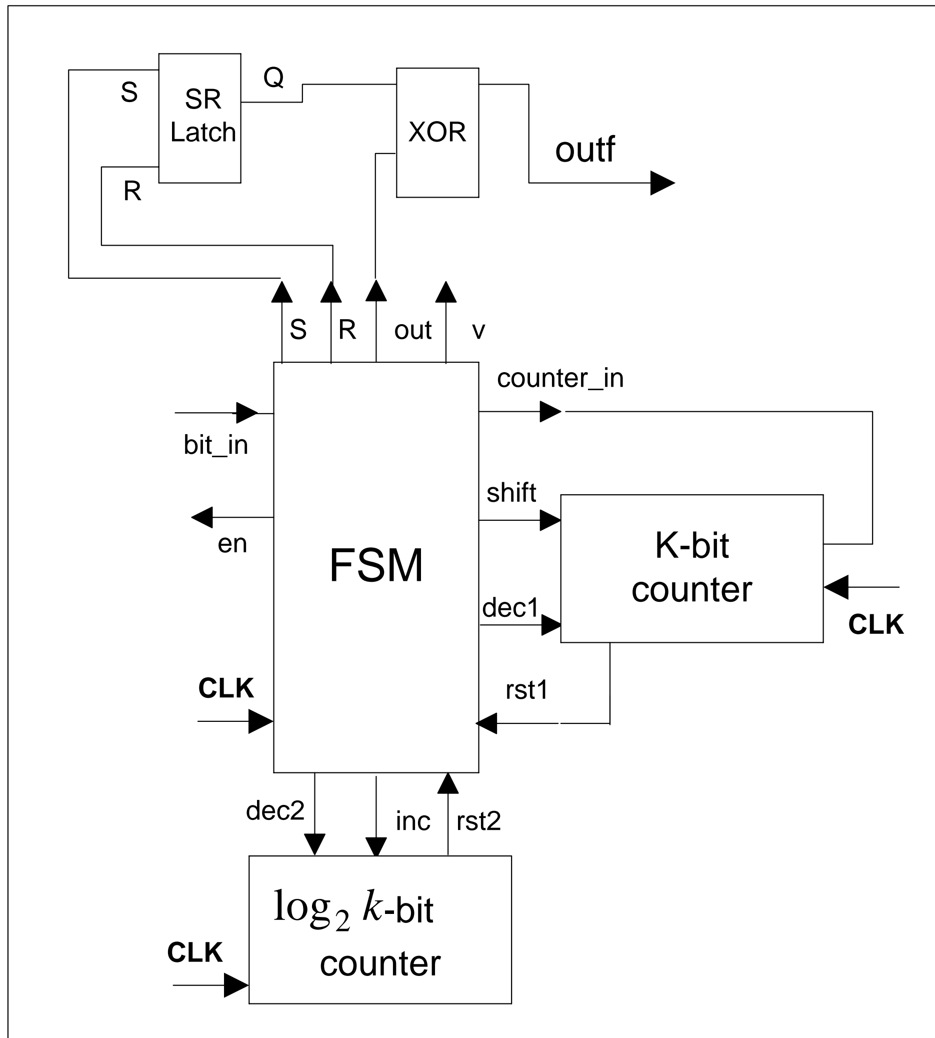

Figure 4 EFDR decoder design.

Table IV Compression results of FDR and EFDR codes.

Test Set	Original Bits	FDR Bits	EFDR Bits	FDR Comp. Ratio	EFDR Comp. Ratio
c2670	10252	5760	4559	43.82	**55.53**
c5315	6586	5238	4471	20.47	**32.11**
c7552	15111	9500	8610	37.13	**43.02**
s13207	163100	34608	32144	78.78	**80.29**
s15850	57434	24992	24263	56.49	**57.75**
s35932	21156	20312	11205	3.99	**47.04**
s38417	113152	70536	51357	37.66	**54.61**
s5378	20758	11032	9739	46.85	**53.08**
s9234	25939	16912	15057	34.80	**41.94**
s5378d	23754	12356	11006	47.98	**53.67**
s9234d	39273	22148	20162	43.61	**48.66**
s13207d	165200	30880	28930	81.31	**82.49**
s15850d	76986	26016	24127	66.21	**68.66**
s35932d	28208	22746	5415	19.36	**80.80**
s38417d	164736	93452	62568	43.27	**62.02**
s38584d	199104	77798	71121	60.93	**64.28**
AVG				45.17	**57.87**

Table V Compression ratio comparison with other test-independent techniques.

Test Set	GOLUMB[24]	FDR[25]	ALT-FDR[26]	MTC[28]	9C[30]	EFDR
S5378d	37.11	47.98	50.77	38.49	51.64	**53.67**
S9234d	45.25	43.61	44.96	39.06	**50.91**	48.66
S13207d	79.74	81.3	80.23	77	82.31	**82.49**
S15850d	62.82	66.21	65.83	59.32	66.38	**68.66**
S38417d	28.37	43.37	60.55	55.42	60.63	**62.02**
S38584d	57.17	60.93	61.13	56.32	**65.53**	64.28
AVG	51.74	57.23	60.58	54.27	62.90	**63.30**

Table VI Test application time and area comparison with FDR.

Test Set	# CLK Cycles		Estimated Area (Gates)	
	FDR[25]	EFDR	FDR [25]	EFDR
S5378d	110081	102210	1380	1457
S9234d	184331	171961		
S13207d	691169	679540		
S15850d	337633	325607		
S35932d	135665	115489		
S38417d	765365	702936		
S38584d	885537	846395		
Average	444254	420591		

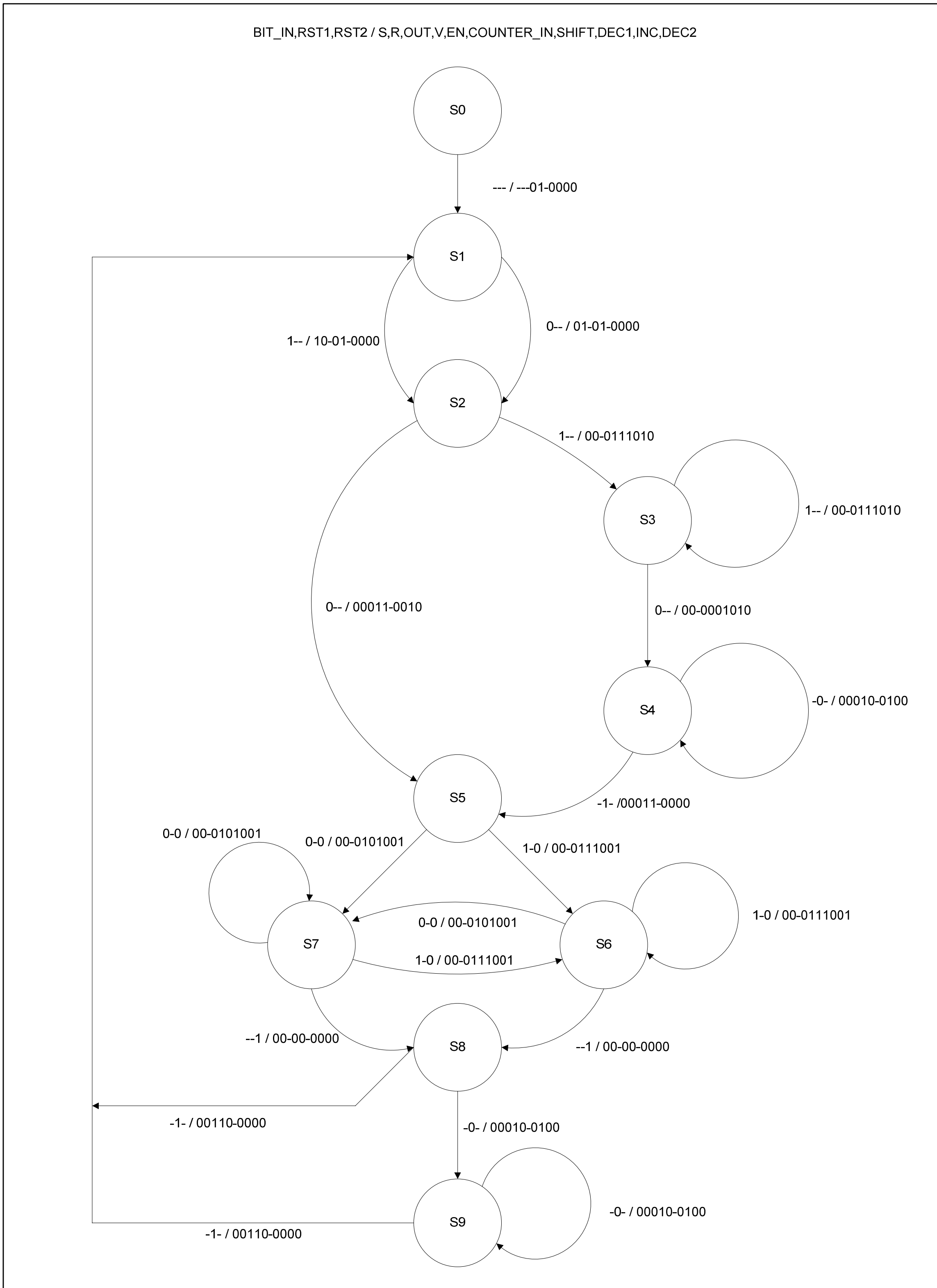

Figure 5 EFDR FSM design.