Nine-Coded Compression Technique with Application to Reduced Pin-Count Testing and Flexible On-Chip Decompression

Mohammad Tehranipour, Mehrdad Nourani
Center for Integrated Circuits & Systems
The Univ. of Texas at Dallas
Richardson, TX 75083
{mht021000,nourani}@utdallas.edu

Krishnendu Chakrabarty
Dept. of Electrical and Computer Engineering
Duke University
Durham, NC 27708
krish@ee.duke.edu

Abstract— **This paper presents a new test data compression technique based on a compression code that uses exactly nine codewords. In spite of its simplicity, it provides significant reduction in test data volume and test application time. In addition, the decompression logic is very small and independent of the precomputed test data set. Our technique leaves many don't-care bits unchanged in the compressed test set, and these bits can be filled randomly to detect non-modeled faults. The proposed technique can be efficiently adopted for single- or multiple-scan chain designs to reduce test application time and pin requirement. Experimental results for ISCAS'89 benchmarks illustrate the flexibility and efficiency of the proposed technique.**

I. INTRODUCTION

Testing today's system-on-chip (SoC) circuits is difficult due to large test data volume, long test application time, high power consumption during test, and limited bandwidth of automatic test equipment (ATE). New techniques are needed to test embedded cores in an SoC without exceeding limits power, ATE memory and bandwidth. These techniques are often based on test resource partitioning (TRP) to reduce ATE memory and test power and tackle ATE bandwidth limitation.

There are several techniques to reduce SoC test data volume and test application time. Built-in self-test (BIST) methodology reduces the need for an expensive ATE [1]. In BIST, on-chip pseudo-random pattern generators and signature compaction are used. In practice, BIST cannot replace other test methods especially for large chips due to the long time needed to detect random pattern resistant faults. To overcome these difficulties, deterministic test patterns need to be transferred from the ATE to the SoC under test to shorten overall test time and improve fault coverage. While the fault coverage in BIST can be improved using techniques such as reseeding [2], bit flipping [3] and bit-fixing [4], these techniques need structural information for fault simulation and test generation.

Test data compression techniques are used to speed up the ATE-SoC interaction during test. These techniques are used to compress the precomputed test data set T_D provided by core vendor to a smaller test set T_E ($|T_E| < |T_D|$) which is then stored in the ATE's memory. An on-chip decoder decompresses T_E to T_D to be applied to the core under test. Many compression techniques have been proposed in recent years, however none has become standardized or is universally applicable. Many of the proposed solutions in the past few years are proprietary, this reduces flexibility and interoperability among tools.

Compression methods such as statistical coding [5] [6], selective Huffman coding [7], Golomb coding [8], FDR coding [9], alternating run-length coding using FDR [10], EFDR coding [11], MTC coding [12] and VIHC coding [13] have been proposed to reduce test data volume. These techniques compress T_D without requiring any structural information for the test application from the core vendor. There are some structural methods for reducing test volume and time which requires design modification. The proposed Illinois scan architecture (ILS) [14] needs fault simulation and test generation as post-processing steps to get high fault coverage.

Other techniques are based on on-chip pattern decompression, such as scan-chain concealment [15], geometric-primitive based compression [16], mutation encoding [17], deterministic embedded test (reusing the scan chain of one core in a SoC to compress the patterns for another core) [18], packet-based compression [19] and LFSR reseeding [2] [20] [21]. An embedded deterministic test technology for low cost test to reduce the scan test data volume and scan test time is presented in [22].

Several dictionary-based compression methods have recently been proposed to reduce test data volume. In [6], frequently occurring blocks are encoded into variable-length indices using Huffman coding. A dictionary with fixed-length indices is used to generate all the distinct output vectors in [23]. Test data compression techniques based on LZ77 and LZW methods are proposed in [24] and [25], respectively. The former uses a dynamic dictionary and the latter uses a memory in the on-chip decoder. The method proposed in [26] is a compression technique using dictionary with fixed-length indices.

There are usually a large number of don't-care bits in a precomputed test data set T_D. These don't-cares are used to help compression techniques achieve higher compression. Generally, ATPG fills don't-cares randomly to have more chance of detecting non-modeled faults to increase defect coverage. Some of the above proposed techniques replace existing don't-cares in T_D with 0 or 1 to generate a smaller T_E and the on-chip decoder generates T_D without don't-care. Such compression may adversely affect the fault coverage of non-modeled faults. Therefore, new techniques are required to leave at least a portion of don't-cares unchanged, such that they can be replaced randomly to help detect non-modeled faults.

In this paper, we present a new compression technique based on a compression code with only nine codewords. Our method achieves significant compression for precomputed test set, reduces test application time, and requires a very small decoder. Since we use fixed-length blocks, the proposed method offers ease of synchronization between the decoder and ATE. The

1530-1591/04 $20.00 (c) 2004 IEEE

technique is quite flexible in reducing test time and ATE's test-pin requirement for single- and multiple-scan chain designs, respectively. Another advantage of the proposed technique is having leftover don't-care bits in T_E, which can be filled randomly to detect non-modeled faults.

The rest of this paper is organized as follows. Section II describes the proposed test data compression technique. Section III discusses the single- and multiple-scan chain decoder architectures, and provide test time analysis. The experimental results are shown in Section IV. Finally, the paper presents the concluding remarks in Section V.

II. 9C Compression Technique

The proposed technique is based on fixed-length-input test data compression. Assume that K is the size of the fixed-length input blocks. The input test vectors are partitioned into groups of K bits and encoding is performed on each K-bit block. Table I shows the proposed coding for K=8. As shown in the table, each K-bit block is divided into two $K/2$ halves. As shown, there are overall nine codewords. For the rows 1 to 4, each half K-bit is matched with all 0's or 1's. Rows 5-8 show the mismatched bits, which have to be sent along with the codeword (UUUU is called a mismatch meaning this half has mix of 0 and 1 and perhaps don't-care X). Row 9 shows that each half is a mismatch and the entire K-bit block has to be sent along with the codeword. The third column shows a symbol for each case. The sixth column shows the decoder input which can be only a codeword (in the first four rows) or codeword plus the mismatch portion (in the last five rows). Finally, the last column shows the final code size for each case.

In this technique, regardless of K, we use only nine unique codewords shown in the fifth column in Table I. Hence, we call our method nine-coded compression, or 9C for short. In the 9C technique, the input test vectors are divided into K-bit blocks and each K-bit block is divided into two parts. The various cases that can arise are matched with one of the above symbols. For example, assuming X represents don't-cares in one half of a block, blocks of 00, 0X, X0 and XX are all matched with symbol 00. Blocks of 11, 1X and X1 are matched with symbol 11. Blocks of XU, 0U and UX, U0 are matched with symbols 0U and U0, respectively.

We acknowledge that more number of uniform K-bit blocks (e.g. 00110011, 11001100, etc.) can be added to Table I. However, a systematic coding in such cases requires $2^4+1=17$ codewords. This may slightly improve the compression ratio but results in a more complicated and expensive decoder. In this paper we focus on having nine codes since our experimentation indicates that it provides the best tradeoff between compression and decoder cost.

III. 9C Decompression Architecture

The 9C technique is flexible to be used for compressing input test data for single-scan chain or multiple-scan chain designs. Here, we propose a small and flexible decompression architecture for each case.

A. Single-Scan Chain Decoder

Figure 1 shows the decoder architecture for a single-scan chain. Decoder decodes only nine prefix-free codeword independent of K value and precomputed test data set. The decoder consists of a finite-state machine (FSM), a $log_2(K/2)$-bit counter and an $K/2$-bit shifter. The FSM takes data from *Data_in* to find out which codeword has been entered. If the input codeword is C_1, C_2, C_3 or C_4, all K bits are generated only with 0's or 1's accordingly. When the FSM receives the codewords C_5, C_6, C_7, C_8 or C_9, it expects to receive $K/2$-bit or K-bit exact data from *Data_in*.

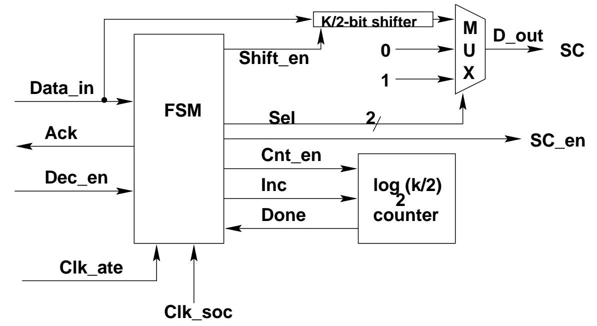

Figure 1. 9C decoder architecture for single-scan chain.

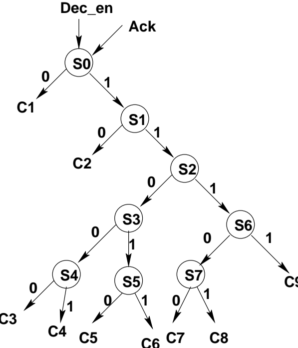

Figure 2. FSM diagram of 9C decoder.

As shown in this figure, MUX has three inputs 0, 1 and *Data_in* which is shifted through $K/2$-bit shifter. Two bits select (*Sel*) come from FSM to MUX. The $log_2(K/2)$-bit counter is used to control sending $K/2$-bit into scan chain. The FSM sends signals *Cnt_en* and *Inc* to activate and increment the counter, respectively. At the same time, it activates signal *SC_en* to enable the scan chain, hence *D_out* is shifted into the scan chain (*SC*). When counter finishes counting, it sends signal *Done* to the FSM to send the next *Sel* and *Cnt_en*. After sending the second signal *Done* by counter, FSM sends *Ack* signal to the ATE to send the next codeword and deactivates signal *SC_en*.

Figure 2 shows the state diagram of the 9C decoder. Note carefully that it is totally independent of K. It starts by checking *Dec_en*. When *Dec_en* is active, it receives the *Data_in* which is the codeword from ATE. Maximum of five cycles are required for the longest codeword. Since a K-bit block is divided into two halves, the FSM sends one *Sel* for each half. After detecting a codeword the required signals are sent to the counter, MUX and scan chain. When the job is done for one received codeword, the state controller starts again from state S_0. This is done when *Ack* signal becomes active, then ATE sends the next codeword.

B. Multiple-Scan Chain Decoder

Let m denote the number of scan chains in a circuit under test. The 9C technique can be used to compress the input test data

TABLE I
9C CODING FOR K=8.

Case (i)	Input Block	Symbol	Description	Codeword (C_i)	Decoder Input	Size (bits)
1	0000 0000	00	All 0's	0	0	1
2	1111 1111	11	All 1's	10	10	2
3	0000 1111	01	Left half 0, right half 1	11000	11000	5
4	1111 0000	10	Left half 1, right half 0	11001	11001	5
5	1111 UUUU	1U	Left half 1, right half mismatch	11010	11010UUUU	5+K/2=9
6	UUUU 1111	U1	Left half mismatch, right half 1	11011	11011UUUU	5+K/2=9
7	0000 UUUU	0U	Left half 0, right half mismatch	11100	11100UUUU	5+K/2=9
8	UUUU 0000	U0	Left half mismatch, right half 0	11101	11101UUUU	5+K/2=9
9	UUUU UUUU	UU	All mismatch	1111	1111UUUUUUUU	4+K=12

vertically i.e., with respect to chain. In this case, the m-bit data are divided into groups of K-bit blocks. The decoder is shown in Figure 3. The FSM sends Cnt_en to enable the $log_2(K/2)$ and $log_2(2m/K)$ counters and $Shift_en$ to let the D_out to be shifted into the m-bit shifter. The FSM used in multiple-scan chain decoder is the same as the one used for single-scan chain decoder. A $log_2(2m/K)$ counter is used to control shifting m bits into the m-bit shifter. Any time that $log_2(K/2)$ counter sends $K/2$ bit into the m-bit shifter, signal $Done$ is sent to the FSM to send the next Sel and Cnt_en and $log_2(2m/K)$ counter to be incremented by one. When the $log_2(2m/K)$ counter is equal to $2m/K$, meaning m bits have been shifted into m-bit shifter, it sends the signal $Load$ to m-bit shifter to load its content into the scan chains SC_1 to SC_m. This architecture reduces the input test pins which only one input test pin is required. In this case, only one decoder is used for m scan chains.

Assume that the length of scan chain is l. Figure 4(a) shows the single-scan chain decoder, used to decode and send the test patterns into a l-bit scan chain. In this case, l clock cycles are required to shift l bits into the scan chain. Assume that the l bit scan chain is rearranged into groups of m-bit scan chains. In this case, the length of each scan chain is l/m. Figure 4(b) shows a multiple-scan chain decoder with one input pin and one decoder. The ATE sends codewords through $Data_in$ and the decoder drives the detected bits into the m-bit shifter. This case does not increase test time compared to single-scan chain, because each detected K-bit is shifted into the m-bit shifter and after completing m-bit, the content of m-bit shifter is sent into m scan chains which uses l clock cycles. This architecture reduces test pins to 1 while keeping the test application time unchanged.

Note that the multiple-scan chain architecture can be extended to reduce the ATE pin count requirement to any desired fraction of m as shown in Figure 4(c). For example, assume that we use one decoder for each K-bit. Instead of having m-bit shifter as shown in Figure 3, we need K-bit shifter such that for each K number of scan chains only one input pin is required. Therefore, for m number of scan chains, m/K input pins and decoders are needed which increases hardware overhead but decreases the test time by a factor of m/K compared to having only one input test pin (see Figure 4(c)) because m/K decoders work in parallel. In this case, Kl/m clock cycles are required.

C. Test Application Time Reduction

Reducing the overall test application time is one of the important goals of any test data compression method. In general, the amount of time reduction depends on the compression ratio and decompression method. In this section, we estimate the overall test application time reduction ratio (TAT) for single-scan chain. Since 9C is a fixed-block compression technique, the test time analysis is simpler compared to the variable-block compression techniques. Suppose the ATE and SoC scan frequencies are f_{ate} and f_{scan} ($f_{ate} \leq f_{scan}$), respectively. TAT is given by

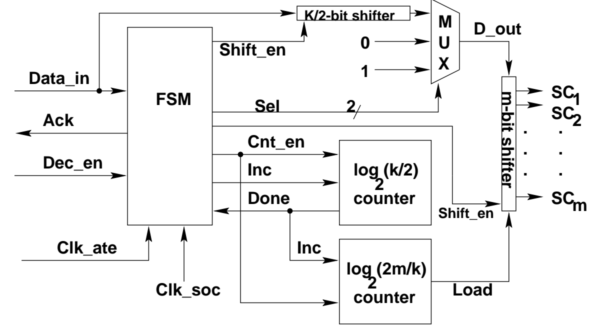

Figure 3. 9C decoder architecture for multiple-scan chains.

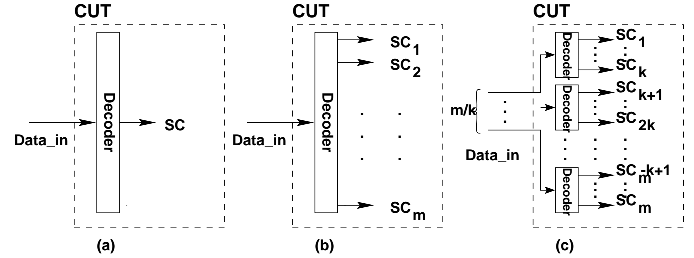

Figure 4. Different scan architectures: (a) Single-scan single-pin, (b) Multiple-scan single-pin (c) Multiple-scan m/K pins.

$$TAT\% = \frac{t_{no_comp} - t_{comp}}{t_{no_comp}} \times 100$$

where t_{no_comp} and t_{comp} are the test application times for applying uncompressed and compressed test data, respectively. Assume that the scan clock frequency of the system under test is p times that of the ATE's clock frequency.

$$\frac{f_{ate}}{f_{scan}} = \frac{1}{p}$$

The test application time for the case of no compression (t_{no_comp}) depends on the total number of input bits (T_D) and ATE's working frequency.

$$t_{no_comp} = \frac{T_D}{f_{ate}} = \frac{p \cdot T_D}{f_{scan}}$$

After compressing the test data, the test application time t_{comp} depends on the occurrence frequency (repetitions) of each

codeword C_i. Let N_i denote occurrence frequency of C_i. Therefore, in our method t_{comp} is given by:

$$t_{comp} = t_1 + t_2 + t_{3-4} + t_{5-8} + t_9$$

where t_1, t_2, t_{3-4}, t_{5-8} and t_9 are the application time of codewords $\{C_1\}$, $\{C_2\}$, $\{C_3, C_4\}$, $\{C_5, C_6, C_7, C_8\}$ and $\{C_9\}$, respectively. Note that t_{3-4}=t_3+t_4 and t_{5-8}=t_5+t_6+t_7+t_8. When codeword C_1 is entered into the FSM, only one ATE's clock is required (because C_1 is one bit), K system's clocks are needed for applying K 0s into the scan chain. Hence, $t_1 = \frac{1}{f_{ate}} + \frac{K}{f_{scan}}$, where $f_{scan}=p \cdot f_{ate}$. Due to the lack of space, the details of the estimation approach for all t's are not discussed here. However, the time parameters will be:

$$\begin{cases} t_1 & = \frac{K+|C_1| \cdot p}{f_{scan}} \cdot N_1 \\ t_2 & = \frac{K+|C_2| \cdot p}{f_{scan}} \cdot N_2 \\ t_{3-4} & = \frac{K+|C_3| \cdot p}{f_{scan}} \cdot N_{3-4} \\ t_{5-8} & = \frac{K+(K/2+|C_5|) \cdot p}{f_{scan}} \cdot N_{5-8} \\ t_9 & = \frac{K+(K+|C_9|) \cdot p}{f_{scan}} \cdot N_9 \end{cases}$$

where N_1, N_2, N_{3-4}, N_{5-8} and N_9 are the occurrence frequency of codewords $\{C_1\}$, $\{C_2\}$, $\{C_3, C_4\}$, $\{C_5, C_6, C_7, C_8\}$ and $\{C_9\}$, respectively. Note that N_{3-4}=N_3+N_4 and N_{5-8}=N_5+N_6+N_7+N_8. $|C_i|$ is the size of codeword C_i, that are $|C_1|$=1, $|C_2|$=2, $|C_3|$=$|C_4|$=$|C_5|$=$|C_6|$=$|C_7|$=$|C_8|$=5 and $|C_9|$=4.

IV. Experimental Results

Table II shows the compression results of ISCAS89's benchmarks for different K's using 9C technique for single-scan chain. The compression ratio (CR) is computed by:

$$\begin{cases} CR\% & = \frac{T_D - T_E}{T_D} \times 100 \\ T_E & = |C_1| \cdot N_1 + |C_2| \cdot N_2 + |C_3| \cdot N_{3-4} + \\ & \quad (K/2 + |C_5|) \cdot N_{5-8} + (K + |C_9|) \cdot N_9 \end{cases}$$

As seen in Table II, the maximum compression ratio for these benchmarks happens in K=8, 12 or 16. When K increases the compression ratio increases to reach a peak at K=8, 12, or 16 and then it starts to decrease in most of the cases. As shown, K=32 generates less compression ratio compared to other K's. The last row shows the average compression ratio for each K on all benchmarks and it indicates that K=8 shows more average compression ratio compared to other K's for these benchmarks. It means that having K=8 for all these benchmarks, the decoder gives very high compression ratio which is independent of the benchmarks and precomputed test sets. The FSM has been implemented by Synopsys Design Compiler [27] and it includes 140 gates.

Table III shows the leftover don't-care (LX) percentage for each benchmark for different K's. These don't-care bits can be replaced randomly to detect some of the non-modeled faults. The second column in the table shows the existing don't-care percentage (X%) of each benchmark. As shown, when K increases the LX increase in which the maximum LXs for all benchmark happen in K=32. Note that for $K = 4$ each half is a very small 2-bit block where all don't-cares need to be replaced according to our technique. The last row shows the average LX for all benchmarks for each K. Of course, the leftover don't-care bits can be also used to reduce the total scan-in power. Some researchers replace the don't-cares with 0 or 1 to minimize the transitions between the consecutive bits. Power issue is beyond the scope of this paper.

Based on Tables II and III, we are able to tradeoff between the leftover don't-cares (LX) and compression ratio. If user asks for a specific amount of don't-cares to possibly detect some of the non-modeled faults. K is obtained from Table III and the compression ratio is obtained from Table II.

Comparing the 9C technique with the other techniques (FDR [10], VIHC [13], MTC [12] and Selective Huffman [7]) is shown in Table IV. As the last row shows the average compression ratio of the 9C technique is more than others.

Table V shows the test application time reduction for ISCAS89 benchmarks for different working frequencies of the system under test. The next three columns show the result for p=8, p=6 and p=4, respectively. TAT is bounded by CR, meaning as $p = f_{scan}/f_{ate}$ increases, the test application time approaches compression ratio. For example, assume that f_{ate}=20MHz (a very slow ATE) and SoC frequency to shift test data into scan chain f_{scan}=160MHz (p=8). The average $TAT\%$=50.59%, indicates good reduction in test application time for such a low speed ATE.

Table VI shows the statistics of occurrence frequencies of each codeword (N_i) in the benchmarks. As the last row in the table shows, codeword C_1 always happens more frequently than the other codewords which it has the minimum codeword size (one bit). C_2 is the second frequent codeword which has the second minimum codeword length (two bits). And C_9 is in third place with codeword size four bits. The other codeword (C_3, C_4, C_5, C_6, C_7 and C_8) have the same codeword size (five bits) which occur less frequently compared to C_1, C_2 and C_9. This statistics show that the proposed coding is the best for these benchmarks and indicates the coding efficiency for these benchmarks and we expect it to be the same specially for large circuits. When the occurrences are different, the 9C technique can be employed using a frequency-directed strategy. Such statistics for a circuit can be obtained *a priori*. And for a special architecture, codewords can be rearranged according to the real occurrence frequencies of codewords in the test pattern set. Although for majority of circuits we expect the order proposed in Table I be the best.

As shown in Table VI, some benchmarks (e.g. s5378, s9234 and s15850) do not comply with the order of occurrence frequencies shown in the last row. For example, for the benchmarks s9234 and s15850 some other codewords like C_8 (in s9234) and C_7 (in s15850) have more occurrence frequency than codeword C_9. To get the best compression ratio for these two benchmarks, we change the order of codewords as C_8 in s9234 and C_7 in s15850 to have four bit codeword size, respectively. Table VII shows the new compression results based on re-assigned codewords (frequency-directed assignment) according to the occurrence frequencies in these three benchmarks. As shown, there are slight improvements for each K compared to the original results shown in Table II.

TABLE II

COMPRESSION RATIO FOR DIFFERENT K'S.

Circuit	T_D	*CR%*							
		K=4	K=8	K=12	K=16	K=20	K=24	K=28	K=32
s5378	23754	43.98	50.69	**51.64**	49.41	46.61	44.77	42.18	39.37
s9234	39273	48.87	**50.91**	46.53	41.92	37.38	33.27	28.97	26.44
s13207	165200	69.63	79.81	81.86	**82.31**	80.65	80.93	79.80	78.96
s15850	76986	60.14	**66.38**	65.11	62.95	60.73	58.51	56.55	55.13
s38417	164736	52.63	**60.63**	59.37	57.54	54.40	51.78	49.31	47.44
s38584	199104	58.82	**65.53**	64.43	62.39	59.79	56.98	54.66	52.13
Avg.	–	55.68	**62.32**	61.49	59.42	56.60	54.37	51.91	49.91

TABLE III

LEFTOVER DON'T-CARES (LX) FOR DIFFERENT K'S.

Circuit	*X%*	*LX%*							
		K=4	K=8	K=12	K=16	K=20	K=24	K=28	K=32
s5378	71	0.00	3.02	6.80	10.54	14.57	17.87	20.80	24.38
s9234	72	0.00	5.36	12.71	19.31	25.35	30.68	35.56	38.95
s13207	92	0.00	1.03	2.65	4.15	6.44	7.30	8.99	10.12
s15850	83	0.00	3.21	7.41	11.60	15.04	18.19	20.66	22.69
s38417	68	0.00	2.71	6.09	9.01	11.87	14.45	16.96	18.92
s38584	82	0.00	2.95	7.09	10.92	14.64	18.12	21.24	24.33
Avg.	78	0.00	3.05	7.13	10.92	22.38	17.77	20.70	23.23

TABLE IV

COMPARISON BETWEEN DIFFERENT TECHNIQUES.

Circuit	*K*	*9C*	*FDR* [10]	*VIHC* [13]	*MTC* [12]	*Selec. Huff. [7]*
s5378	12	51.64	50.77	51.52	38.49	55.10
s9234	8	50.91	44.96	54.84	39.06	54.20
s13207	16	82.31	80.23	83.21	77.00	77.00
s15850	8	66.38	65.83	60.68	59.32	66.00
s38417	8	60.63	60.55	54.51	55.42	59.00
s38584	8	65.53	61.13	56.97	56.63	64.10
Avg.	–	**62.90**	60.57	60.28	54.32	62.57

TABLE V

TEST APPLICATION TIME REDUCTION (TAT%) USING 9C TECHNIQUE.

Circuit	*K*	*CR%*	*TAT%*		
			$\frac{f_{ate}}{f_{scan}} = \frac{1}{8}$	$\frac{f_{ate}}{f_{scan}} = \frac{1}{6}$	$\frac{f_{ate}}{f_{scan}} = \frac{1}{4}$
s5378	12	51.64	39.00	34.84	26.51
s9234	8	50.91	38.99	34.82	26.49
s13207	16	82.31	69.61	66.12	57.79
s15850	8	66.38	54.24	49.78	41.74
s38417	8	60.63	48.27	43.90	35.77
s38584	8	65.53	53.40	49.23	40.90
Avg.	–	62.90	50.59	46.45	38.20

TABLE VI

CODEWORD STATISTICS OF THE BENCHMARKS.

Circuit	*K*	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9
s5378	12	929	220	24	38	73	81	144	144	326
s9234	8	1952	1143	243	249	164	159	355	366	279
s13207	16	7624	1158	209	138	114	100	298	279	406
s15850	8	6425	1067	294	278	178	158	471	450	303
s38417	8	11454	3779	500	506	816	797	832	895	1014
s38584	8	15473	3946	753	724	487	451	1017	976	1062
Avg.	–	**7309**	**1885**	337	322	305	291	519	518	**565**

TABLE VII

COMPRESSION RATIO FOR THREE BENCHMARKS AFTER RE-ASSIGNING THE CODEWORDS BASED ON OCCURRENCE FREQUENCIES.

Circuit	*CR%*							
	K=4	K=8	K=12	K=16	K=20	K=24	K=28	K=32
s5378	44.12	50.69	**51.79**	50.46	47.78	45.77	43.68	41.17
s9234	49.48	**51.09**	46.53	41.94	38.95	35.72	32.28	29.81
s15850	60.58	**66.52**	66.45	63.32	61.47	59.60	57.73	56.21

TABLE VIII

COMPRESSION RESULTS FOR TWO LARGE CIRCUITS FROM IBM.

Circuit	X%	T_D	K					
			8	12	16	32	48	64
CKT1	99.36	11613472	87.06	91.11	93.07	95.68	**96.21**	96.18
CKT2	97.90	4124288	85.81	89.69	91.43	**93.30**	93.19	92.57

We also evaluated the compression efficiency of the 9C technique for very large test sets from IBM previously reported in [26]. Circuit CKT1 contains 3.6 million gates, 726000 flip flops and requires about 11.6 Mbit test data. Circuit CKT2 has 1.2 million gates, 32200 flip flops and needs 4.1 Mbit test data. Table VIII shows the compression ratio for different K's. As shown, K=48 and K=32 show the maximum compression for CKT1 and CKT2, respectively.

The style, cost and flexibility of on-chip decompresser have become important factors in practicality of compression techniques. Specifically, some decoders such as those proposed in [5] [6] [7] [13] [23] are dependent on the precomputed test set and thus are customized for the circuit under test. The decompression logics in [9] [10] are independent of the precomputed test set. However, since the methods are based on variable-length coding, the number of codewords required to be decoded for the large circuits can be huge. Therefore, for having an independent decoder, they have to consider a possible maximum length happening in the test set. But, the 9C technique's decoder is totally independent of the circuit under test and precomputed test set. In other words, for the circuits using the same K, the decoder remains the same, no matter what the precomputed test data set is. This feature makes our 9C technique superior in terms of cost, flexibility and design reuse.

V. CONCLUSION

A new compression technique using only nine codewords called 9C has been proposed in this paper. It significantly reduces the test data volume and application time. The leftover don't-care bits form 10-50% of data volume and can be utilized for detecting some of the non-modeled faults. Using the 9C technique, it is possible to tradeoff between test volume-time, pin-count requirement and leftover don't-cares. Implementation of 9C technique on ISCAS benchmarks has shown up to 83% compression. The decompression logic is very small and flexible and independent of the precomputed test data set.

ACKNOWLEDGEMENTS

The work of M. Tehranipour and M. Nourani was supported in part by the National Science Foundation CAREER Award #CCR-0130513. The work of K. Chakrabarty was supported by the National Science Foundation under grants CCR-9875324 and CCR-0204077.

REFERENCES

[1] Y. Zorian, E. Marinissen and S. Dey, "Testing Embedded-Core-Based System Chips," *Computer Magazine*, 32(6), pp. 52-60, 1999.

[2] S. Hellebrand, H. Linag and H. -J. Wunderlich, "A Mixed-Mode BIST Scheme Based on Reseeding of Folding Counters," in Proc. *Int. Test Conf. (ITC'00)*, pp. 778-784, 2000.

[3] H. -J. Wunderlich and G. Kiefer, "Bit-Flipping BIST," in Proc. *Int. Test Conf. (ITC'00)*, pp. 337-343, 1996.

[4] N. Touba and E. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan Based BIST," in Proc. *Int. Test Conf. (ITC'00)*, pp. 167-175, 1996.

[5] V. Iyengar, K. Chakrabarty and B. Murray, "Built-In Self Testing of Sequential Circuits Using Precomputed Test Sets," in Proc. *VLSI Test Symp. (VTS'98)*, pp. 418-423, 1998.

[6] A. Jas, J. Ghosh-Dastidar and N. Touba, "Scan Vector Compression/Decompression Using Statistical Coding," in Proc. *VLSI Test Symp. (VTS'99)*, pp. 114-120, 1999.

[7] A. Jas, J. Gosh-Dastidar, M. Ng and N. Touba, "An Effi cient Test Vector Compression Scheme Using Selective Huffman Coding," *IEEE Trans. on Computer-Aided Design (TCAD)*, vol. 22, pp. 797-806, 2003.

[8] A. Chandra and K. Chakrabarty, "System-on-a-Chip Data Compression and Decompression Architecture Based on Golomb Codes," *IEEE Trans. on Computer-Aided Design (TCAD)*, vol. 20, pp. 355-368, 2001.

[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 Trans. on Computers (TCOMP)*, pp. 1076-1088, 2003.

[10] A. Chandra and K. Chakrabarty, "A Unifi ed Approach to Reduce SOC Test Data Volume, Scan Power and Testing Time," *IEEE Trans. on Computer-Aided Design (TCAD)*, pp. 352-363, 2003.

[11] A. El. Maleh and R. Al-Abaji, "Extended Frequency-Directed Run-Length Codes with Improved Application to System-on-a-Chip Test Data Compression," in Proc. *Int. Conf. Elect. Circuits and Systems (ICECS'02)*, pp. 449-452, 2002.

[12] P. Rosinger, P. Gonciari, B. Al-Hashimi and N. Nicolici, "Simultaneous Reduction in Volume of Test Data and Power Dissipation for System-on-a-Chip," *Electronics Letters*, vol. 37, no. 24, pp. 1434-1436, 2001.

[13] P. Gonciari, B. Al-Hashimi and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-chip Test Data Compression/Decompression," *Design, Automation and Test in Europe (DATE'02)*, pp. 604-611, 2002.

[14] F. Hsu, K. Butler and J. Patel, "A Case Study on the Implementation of the Illinois Scan Architecture," in Proc. *Int. Test Conf. (ITC'01)*, pp. 538-547, 2001.

[15] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment," in Proc. *Design Automation Conf. (DAC'01)*, pp. 151-155, 2001.

[16] A. El-Maleh, S. Al Zahir and E. Khan, "A Geometric-Primitives-Based Compression Scheme for Testing System-on-Chip," in Proc. *VLSI Test Symp. (VTS'01)*, pp. 54-59, 2001.

[17] S. Reda and A. Orailoglu, "Reducing Test Application Time Through Test Data Mutation Encoding," in Proc. *Design, Automation and Test in Europe (DATE'02)*, pp. 387-393, 2002.

[18] L. Schafer, R. Dorsch and H. -J. Wunderlich, "RESPIN++-Deterministic Embedded Test," in Proc. *European Test Workshop (ETW'02)*, pp. 37-44, 2002.

[19] E. Volkerink, A. Khoche and S. Mitra, "Packet-Based Input Test Data Compression Technique," in Proc. *Int. Test Conf. (ITC'02)*, pp. 167-175, 2002.

[20] C. Krishna, A. Jas and N. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," in Proc. *Int. Test Conf. (ITC'01)*, pp. 885-893, 2001.

[21] E. Volkerink and S. Mitra, "Effi cient Seed Utilization for Reseeding Based Compression," in Proc. *VLSI Test Symp. (VTS'03)*, pp. 232-237, 2003.

[22] J. Rajski, et. al, "Embedded Deterministic Test for Low Cost Manufacturing Test," in Proc. *Int. Test Conf. (ITC'02)*, pp. 301-310, 2002.

[23] S. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs," in Proc. *VLSI Test Symp. (VTS'02)*, pp. 103-108, 2002.

[24] F. Wolff and C. Papachristou, "Multiscan-Based Test Compression and Hardware Decompression Using LZ77," in Proc. *Int. Test Conf. (ITC'02)*, pp. 331-339, 2002.

[25] M. Knieser, F. Wolff, C. Papachristou, D. Weyer and D. McIntyre, "A Technique for High Ratio LZW Compression," in Proc. *Design, Automation and Test in Europe (DATE'03)*, pp. 116-121, 2003.

[26] L. Li and K. Chakrabarty, "Test Data Compression Using Dictionaries with Fixed Length Indices," in Proc. *VLSI Test Symp. (VTS'03)*, pp. 219-224, 2003.

[27] Synopsys Inc., "User Manuals for SYNOPSYS Toolset Version 2002.05," Synopsys, Inc., 2002.