Combining Dictionary Coding And LFSR Reseeding For Test Data Compression

Xiaoyun Sun*, Larry Kinney and Bapiraju Vinnakota
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN, 55455
* Tel. No: 612-625-5006, Fax. No: 612-625-4583, e-mail: xsun@ece.umn.edu

ABSTRACT

In this paper we describe a method to combine dictionary coding and partial LFSR reseeding to improve the compression efficiency for test data compression. We also present a fast matrix calculation method which significantly reduces the computation time to find a solution for partial LFSR reseeding. Experimental results on ISCAS89 benchmark circuits show that our approach is better than either dictionary coding or LFSR reseeding, and outperforms several test data compression methods proposed recently.

Categories and Subject Descriptors: B.8.1 [Performance and Reliability]: Reliability, Testing and Fault Tolerance.

General Terms: Algorithm, Design.

Keywords: VLSI test, Built-In Self Test.

1. INTRODUCTION

As technology advances, the amount of test data increases dramatically, which requires increased test time, test storage on the tester, and test data bandwidth between the tester and the chip. Consequently, there is a need for test data compression. One attractive approach for test data compression is to use Linear Feedback Shift Register (LFSR) reseeding [11]. Several methods based on LFSR reseeding have been proposed [6] [12] [13] [18] [20] [26]. Commercial tools based on LFSR reseeding have also been developed recently which include Mentor Graphics' TestKompress [21] and Synopsys' DBIST [24].

Also, a lot of research effort has been done on using lossless source coding to compress test data. Many coding techniques have been proposed for test data compression, which include run-length coding [9], Golomb coding [3], FDR coding [2], EFDR coding [16], statistical coding [10], variable-length index Huffman coding (VIHC) [5] and dictionary coding [14] [25]. Other test data compression techniques such as Illinois scan chain [8], folding counter [7], scan chain concealment(XOR network) [1], mutation encoding [22], Packet-based coding [17] and RESPIN++ [23] were also proposed.

LFSR reseeding based techniques are not efficient when the number of specified bits in each vector is very large. Source coding techniques can handle test data with a large number of specified bits. This suggests combining source coding and LFSR reseeding to improve the efficiency of test data compression. In this paper, we describe a method that combines dictionary coding and partial LFSR reseeding for test data compression. In the rest of the paper, the proposed method is called CDCR (Combining Dictionary Coding and Reseeding) for convenience. The rest of the paper is organized as follows: Section 2 introduces the methodology of dictionary coding and LFSR reseeding. Section 3 presents our approach which combines dictionary coding and LFSR reseeding for test data compression. Experimental results on ISCAS89 benchmark circuits are shown in Section 4.

2. PREVIOUS WORK

We describe a dictionary-based test data compression approach proposed by Li and Chakrabarty [14]. Let's assume the precomputed test data consists of d test vectors, and there are m scan chains in CUT with each k bits long. A word is defined as data loaded into the m scan chains per clock cycle. So the length of the word is m bits and there are k words in each test vector. The total number of words is t where $t = dk$. If the total number of entries in the dictionary is w, then the length of the dictionary index is q bits where $q = log_2 w$. In Li and Chakrabarty's method, a codeword consists of a prefix and a content. The prefix has only 1 bit and serves as an identifier that indicates whether the content is an index in the dictionary or a word not encoded into the dictionary. If the prefix is "1", the content is viewed as a dictionary index, which is q bits long. If the prefix is "0", the content is an uncompressed word of m bits. For the dictionary coding problem, vertices are test words and connecting edges indicate no conflicting specified bits. The dictionary encoding problem can be converted to a clique partitioning problem. The clique partitioning problem is described as follows: An undirected graph G consists of a set of vertices V and a set of edges E, where each edge connects two vertices. Given $G = (V, E)$, a clique of the graph is defined as a subset, each pair of which is connected by an edge in E. A heuristic algorithm for clique partitioning problem is described in [14]. The computational complexity of this algorithm is $O(t^3)$, where t is the number of vertices in G and $t = dk$.

The technique of LFSR reseeding to compress the test data was first proposed in [11]. At the beginning of each test vector, a r-bit seed is load into LFSR. If there is only one channel from ATE, then it will take r clock cycles for loading the seed. After the seed is loaded, the m scan chains are loaded by the LFSR for k clock cycles, where k is the length of each scan chain. For each specified bit in the test vector, there is a corresponding linear equation which is constructed based on the position of the specified bit, the structure of XOR network, and the number of clock cycles. The variables in the linear equation are the r bits in the seed loaded into LFSR. Solving all the linear equations for the corresponding specified bits in a test vector will determine the r-bit seed to be loaded into LFSR. If the number of specified bits for each test vector is s, then r should be larger than $s_{max}+20$ to ensure that there is a solution for the linear equations, where s_{max} is the maximum number of specified bits in one test vector [4] [11].

There are 3 issues limiting the performance of basic LFSR Reseeding [11]:

1. Compression limit: The best compression that reseeding based methods can achieve is the total number of specified bits in all the test vectors.

2. Solvability: To ensure there is a solution, the number of bits in the seeds and the size of LFSR should be large enough. In order to reduce the probability of not finding a solution to be less than 10^{-6}, r should be larger than $s_{max}+20$[4] [11] .

3. Variation of the specified bits: The number of specified bits in each test vector can vary significantly, while the size of LFSR and the number of seeds loaded into LFSR for each test vector is a fixed number (i.e., $r \sim s_{max}+20$). So the size of compressed data $dr \sim d(s_{max}+20)$ may be much larger than the total number of specified bits ds_{avg}, where s_{avg} is the average number of specified bits in one test vector.

To address the 3rd issue mentioned above, Krishna, Jas and Touba proposed a method to use partial reseeding to improve the efficiency of LFSR reseeding [12]. The basic idea of partial reseeding is: a n-bit seed is loaded into the LFSR at the beginning of each test vector, where $n \sim s_{avg}$. Instead of solving the linear equations for each test vector, all the linear equations for all the test vectors are solved together. To improve the solvability, the test vectors need to be reordered. The advantage of partial reseeding is that, the compression efficiency has been greatly improved. The disadvantage of partial reseeding is that, the linear equations for all the test vectors need to be solved together, which significantly increases the computational complexity. A solution could be to partition the test vectors into several groups and handle one group each time [12].

3. COMBINING CODING AND RESEEDING

LFSR reseeding based techniques are not efficient when the number of specified bits in each vector is very large. Source coding techniques such as dictionary coding can handle test data with large number of specified bits. This suggests combining dictionary coding and reseeding to improve the efficiency of test data compression. In this section, we describe a method combining dictionary coding and partial reseeding for test data compression.

3.1 Methodology

The basic idea for CDCR is that, for those words not encoded into dictionary, partial reseeding is used to compress them. Different from the original partial reseeding proposed in [12], a seed is loaded into LFSR for each word instead of for each test vector. Since reordering the words is impossible, it may be necessary to insert a "dummyword"(DW) between words to ensure the linear equations can be solved. A DW is equivalent to adding a n-bit seed into LFSR without loading the scan chain. Since more variables are added into the linear equations, the solvability of the linear equations is improved. An algorithm to determine where to add a DW is described below:

1. Initialize LFSR.

2. Choose the first m-bit word to be the current word.

3. Generate linear equations for the current word and see if the equations can be solved.

4. If yes (if pivoting is successful), choose the next m-bit word to be the current word, go back to step 3.

   Else, if the number of DWs added for the current word is larger than 10 (10 is a randomly chosen number), reconfigure the XOR network and go back to step 1. Else, add a DW and go back to step 3.

Similar to dictionary coding, each codeword consists of a prefix and a content. For those words can be encoded into the dictionary, a 1-bit prefix is used and the content has q bits. For those words cannot be encoded into the dictionary, a 2-bit prefix is used and the content has n bits. The first bit of the codeword is checked by the decoder to determine if it is a codeword for the dictionary. If the result is true, the remaining q bits are used as a dictionary index. If the result is false, the second bit of the codeword is checked by the decoder to determine whether a DW is used. If a DW is not needed, the remaining n bits are considered as seeds and are loaded into LFSR in n clock cycles. the scan chains are loaded for one clock cycle. If a DW is needed, the scan chains are not loaded. For dictionary coding, $m+1$ bits are needed to encode those words not encoded into dictionary, while only $n+2$ bits are needed for CDCR. For most cases, $n \ll m$, so compared to dictionary coding, CDCR greatly improves the compression efficiency.

Fig. 1 shows the hardware for CDCR. Compared to dictionary coding [14], the m-bit shifter is replaced by a r-bit LFSR. A XOR network, which serves as the phase shifter, is added between LFSR and the multiplexer to eliminate the linear dependency. For many cases, $r < m$, so the size of LFSR is smaller than the m-bit shifter. It is shown in [14] that the size of the combinational logic for the dictionary is about 1000 logic gates for the ISCAS89 benchmark circuits, and is between 1000 to 2000 gates for two large size industrial circuits. The size of the XOR network is proportional to the number of scan chains.

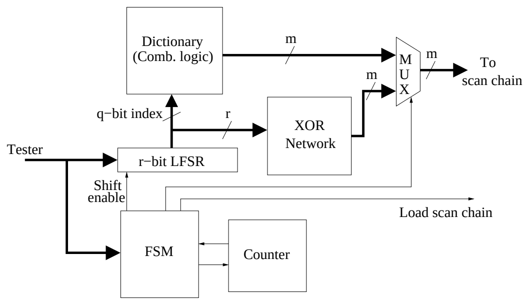

Figure 1: CDCR hardware

The limitation of CDCR is that, same as the dictionary coding, the decoding structure should be determined after the generation of the test patterns. The hardware overhead is also an issue for CDCR.

3.2 Fast Matrix Calculation

One disadvantage of partial reseeding is that the computation time is high. If the total number of words is t where $t = dk$, then a tn by ts_{avg} matrix must be solved, which is impractical for large size IC. However, the matrix structure in this case permits using the method of pivoting to do the computation, which reduces the computational complexity.

Fig. 2 shows an example matrix when there are only two words ($t = 2$) and the size of the seed is 4 ($n = 4$). There are 3 specified bits in the 1st word and 5 in the 2nd word.

Bench. circuits	test vec.	scan cells	Test data	m=64					m=128					m=200				
				n	r	t_p	DW	CDCR	n	r	t_p	DW	CDCR	n	r	t_p	DW	CDCR
s5378	111	214	23754	9	48	68	23	4009	13	48	65	10	2381	13	48	65	10	2381
s9234	159	247	39273	14	64	215	64	7832	16	64	70	64	4396	16	64	70	64	4396
s13207	236	700	165200	12	48	63	19	21412	20	48	4	1	11402	-	-	0	-	7552
s15850	126	611	76986	14	64	223	93	13352	18	64	95	53	7240	20	64	65	35	5712
s35932	16	1763	28208	20	64	92	73	6478	21	64	14	15	2347	-	-	0	-	1152
s38417	99	1664	164736	15	64	1294	1109	56671	27	128	1051	668	51739	29	128	728	798	48610
s38584	136	1464	199104	14	64	1104	569	42960	18	128	768	424	30752	21	128	529	321	24022

Table 1: The compression result by CDCR for different number of scan chains

So there are 3 linear equations for the 1st word and 5 for the 2nd. For partial reseeding, if Gaussian elimination is used, the computation complexity is 8^3.

$$\left(\begin{array}{cccc|cccc} 1&1&0&1&0&0&0&0\\ 0&1&1&1&0&0&0&0\\ 1&0&1&0&0&0&0&0\\ \hline 1&0&0&0&0&1&0&0\\ 1&1&1&0&1&0&1&0\\ 0&0&0&1&1&0&0&0\\ 1&0&0&1&1&0&0&1\\ 0&1&1&1&0&1&1&1 \end{array}\right)\begin{pmatrix} X1\\X2\\X3\\X4\\X5\\X6\\X7\\X8 \end{pmatrix}=\begin{pmatrix}1\\0\\1\\1\\0\\1\\0\\1\end{pmatrix}$$

Figure 2: The matrix for all the linear equations

Note that the upper-right part of the matrix in Fig. 2 only contains '0's. This is because the second seed has not been loaded into LFSR at the end of the first word and there are only 4 variables X_1, X_2, X_3 and X_4 for the first three equations. Therefore, the method of pivoting can be used to do the computation. The basic idea is described as follows: Instead of computing the matrix at the end of all the words, a pivoting step can be applied at the end of each word. The purpose of the pivoting step is to get rid of some variables and reduce the complexity of the matrix. The number of variables that can be eliminated is equal to the number of linear equations involved. For the sample in Fig. 2, a pivoting step is shown in Fig. 3 which expresses three variables X_1, X_2, and X_3 by X_4. The computation complexity for pivoting is 3^3.

$$\begin{pmatrix}1&1&0&1\\0&1&1&1\\1&0&1&0\end{pmatrix}\begin{pmatrix}X1\\X2\\X3\\X4\end{pmatrix}=\begin{pmatrix}1\\0\\1\end{pmatrix}$$

$$\begin{pmatrix}1&0&0&1\\0&1&0&0\\0&0&1&1\end{pmatrix}\begin{pmatrix}X1\\X2\\X3\\X4\end{pmatrix}=\begin{pmatrix}0\\1\\1\end{pmatrix}$$

$$\begin{pmatrix}X1\\X2\\X3\end{pmatrix}=\begin{pmatrix}1\\0\\1\end{pmatrix}X4+\begin{pmatrix}0\\1\\1\end{pmatrix}$$

Figure 3: Pivoting to eliminate some variables

Fig. 4 shows the final matrix need to be solved. Since variables X_1, X_2, and X_3 have been eliminated, only five variables remain in the matrix. The computation complexity for solving the matrix is 5^3. The overall computation complexity for fast matrix calculation is $3^3+5^3+overhead$, which is much less than the original approach. If there are t words, then $t-1$ pivots and 1 matrix solution is required. The computational complexity for each pivoting is $O(n^3)$, so the computational complexity for fast matrix calculation is $O(tn^3)$, while the computational complexity for the original approach is $O(t^3n^3)$.

$$\begin{pmatrix}1&0&1&0&0\\0&1&0&1&0\\1&1&0&0&0\\0&1&0&0&1\\0&0&1&1&1\end{pmatrix}\begin{pmatrix}X4\\X5\\X6\\X7\\X8\end{pmatrix}=\begin{pmatrix}1\\0\\1\\0\\1\end{pmatrix}$$

$$\begin{pmatrix}X4\\X5\\X6\\X7\\X8\end{pmatrix}=\begin{pmatrix}0\\1\\1\\1\\1\end{pmatrix}$$

Figure 4: Final matrix equation after pivoting

4. EXPERIMENTAL RESULTS

The CDCR method was applied to test data for several large ISCAS89 benchmark circuits. The test vectors are generated by Mintest program[19]. Same as [14], The size of the dictionary was set to 128 ($w = 128$ and $q = 7$). Table 1 shows the compression results for 64, 128, and 200 scan chains. The size of the original test data (in bits) is shown in column 4. The remaining 15 columns show the size of the seed (n), the size of LFSR (r), the number of words not encoded into dictionary (t_p), the number of DWs, and the compressed test data by CDCR for 64, 128, and 200 scan chains. For benchmark s13207 and s35932 when 200 scan chains are used, all words were successfully encoded into dictionary.

The size of compressed test data is determined by the number of scan chains, the number of words encoded into dictionary, the size of the seed, and the number of "dummy-word" used. For example, for the benchmark circuit s38584, when the number of scan chains is 200 ($m = 200$), the total number of words is 1088 ($t = 1088$). Among the 1088 words, 559 words were encoded into dictionary while 529 words were encoded using partial LFSR reseeding. The number of dummy words used is 321, and the size of the seed is 21 ($n = 21$). So the size of compressed test data is $559 \times 8 + (529 + 321) \times (21 + 2) = 24022$ bits.

Table 2 shows the comparison of compressed test data between dictionary coding and CDCR. The right 6 columns of Table 2 compares compression results between dictionary coding and CDCR for 64, 128, and 200 scan chains. For most cases, the performance of CDCR is better than dictionary coding. For benchmark s13207 and s35932 when 200 scan chains are used, CDCR and dictionary coding have the same results. This is because all words were successfully encoded into the dictionary. The 5th column lists the total number of specified bits for each circuit. Note that the total number of specified bits is the compression limit for LFSR reseeding. For most cases, the compression result by CDCR is better than the compression limit for LFSR reseeding.

Table 3 compares CDCR with several recently-proposed test data compression techniques such as EFDR [16], variable-length input Huffman coding (VIHC) [5], RESPIN++ [23], XOR network [1], mutation encoding + XOR network [22], and LFSR reseeding with seed compression [13]. Table 3 show that CDCR outperforms all the other techniques (ex-

Benchmark circuits	# test vectors	# scan cells	Test data	No. of spec. bits	m=64		m=128		m=200	
					dict.	CDCR	dict.	CDCR	dict.	CDCR
s5378	111	214	23754	6505	6345	4009	8794	2381	12970	2381
s9234	159	247	39273	10601	12783	7832	11498	4396	16826	4396
s13207	236	700	165200	11313	24074	21412	13990	11402	7552	7552
s15850	126	611	76986	12657	18573	13352	13873	7240	14840	5712
s35932	16	1763	28208	18251	7403	6478	3123	2347	1152	1152
s38417	99	1664	164736	52582	94350	56671	104192	51739	114243	48610
s38584	136	1464	199104	35287	58207	42960	58189	30752	53287	24022

Table 2: Compare dictionary coding and CDCR

Benchmark circuits	Test data	No. of spec. bits	EFDR	VIHC	RESPIN ++	XOR network	Mutation + XOR net.	[13]*	CDCR		
									m=64	m=128	m=200
s5378	23754	6505	11419	11516	17332	-	-	6180	4009	2381	2381
s9234	39273	10601	21250	17736	17198	-	-	12112	7832	4396	4396
s13207	165200	11313	29992	27737	26004	25344	15783	11285	21412	11402	7552
s15850	76986	12657	24643	30271	32226	22784	10798	12438	13352	7240	5712
s35932	28208	18251	5554	9458	-	7128	3972	-	6478	2347	1152
s38417	164736	52582	64962	74938	89132	89856	42264	34767	56671	51739	48610
s38584	199104	35287	73853	85674	63232	38976	22636	29397	42960	30752	24022

* Use ATPG other than Mintest

Table 3: Compare CDCR and several compression techniques

cept mutation encoding + XOR network [22] and LFSR reseeding with seed compression [13]) for all benchmark circuits when $m = 200$. CDCR outperforms mutation encoding + XOR network [22] for 3 of 5 benchmark circuits and LFSR reseeding with seed compression [13] for 5 of 6 benchmark circuits.

The hardware overhead for the combinational logic of the dictionary and comparison of hardware between dictionary coding and selective huffman coding [10] were described in detail in [15].

5. CONCLUSION

In this paper, we described a method that combines dictionary coding and partial LFSR reseeding to improve the efficiency of test data compression. We also presented a fast matrix calculation method which significantly reduces the computation complexity for solving the linear equations for partial LFSR reseeding. Experimental results for 7 large ISCAS89 benchmark circuits showed that our approach is better than either dictionary coding or LFSR reseeding, and outperforms several test data compression methods proposed recently.

Acknowledgment: We appreciate Prof. Nur Touba of University of Texas for valuable discussions and comments on this paper.

6. REFERENCES

[1] I.Bayraktaroglu and A.Orailoglu, "Test volume and application time reduction through scan chain concealment", *Proc. DAC*, pp. 151-155, 2001.

[2] A.Chandra and K.Chakrabarty, "Frequency-directed run-length(FDR) codes with application to system-on-a-chip test data compression", *Proc. VTS*, pp. 42-47, 2001.

[3] A.Chandra and K.Chakrabarty, "System-on-a-chip test data compression and decompression architectures based on Golomb codes", *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 355-368, March 2001.

[4] C.L.Chen, "Linear dependencies in linear feedback shift registers", *IEEE trans. on Computers*, Vol. 35, No. 12, pp. 1086-1088, Dec. 1986.

[5] 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", *Proc. DATE*, pp. 604-611, 2002.

[6] S.Hellebrand et al, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers", *IEEE Trans, on Computers*, Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[7] S.Hellebrand, H.Liang, H.Wunderlich, "A mixed-mode BIST scheme based on reseeding of folding counters", *Proc. ITC*, pp. 778-784, 2000.

[8] F.Hsu, K.Butler, and J.Patel, "A case study on the implementation of Illinois scan architecture", *Proc. ITC*, pp. 538-547, 2001.

[9] A.Jas and N.A.Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based design", *Proc. ITC*, pp.458-464, 1998.

[10] A.Jas, J.Ghosh-Dastidar and N.A.Touba, "Scan vector compression/decompression using statistical coding", *Proc. VTS*, pp. 114-120, 1999.

[11] B.Konemann, "LFSR-coded test patterns for scan designs", *Proc. Euro. Test Conf.*, pp.237-242, 1991.

[12] C.V.Krishna, A.Jas, and N.A.Touba, "Test vector encoding using partial LFSR reseeding", *Proc. ITC*, pp. 885-893, 2001.

[13] C.V. Krishna and N.A. Touba, "Reducing test data volume using LFSR reseeding with seed compression", *Proc. ITC*, pp. 321-330, 2002.

[14] L.Li and K.Chakrabarty, "Test data compression using dictionaries with fixed-length indices", *Proc. VTS*, 2003.

[15] L.Li, K.Chakrabarty and N.A.Touba, "Test data compression using dictionaries with selective entries and fixed-length indices", *ACM Trans. on Design Automation of Electronic Systems*, pp. 470-490, 2003.

[16] A.El-maleh and R.Al-Abaji, "Extended frequency directed run length codes with improved application to system-on-a-chip test data compression", *Proc. Int. Conf. Elec. Cir. and Sys.*, pp. 449-452, 2002.

[17] E.Volkerink, A.Khoche and S.Mitra, "Packet-based input test data compression techniques", *Proc. ITC*, pp. 154-163, 2002.

[18] E.Volkerink and S.Mitra, "Efficient Seed Utilization for Reseeding based Compression", *Proc. VTS*, 2003.

[19] I.Hamzaoglu and J.H.Patel, "Test set compaction algorithms for combinational circuits", *Proc. ICCAD*, pp. 283-289, 1998.

[20] J.Rajski, J.Tyszer and N.Zacharia, "Test data decompression for multiple scan designs with boundary scan", *IEEE trans. on computers*, Vol. 47, No. 11, pp. 1188-1200, Nov. 1998.

[21] J.Rajski et al., "Embedded deterministic test for low-cost manufacturing test", *Proc. ITC*, pp. 301-310, 2002.

[22] S.Reda and A.Orailoglu, "Reducing test application time through test data mutation encoding", *Proc. DATE*, pp. 387-393, 2002.

[23] L.Schafer, R.Dorsch, and H.Wunderlich, "RESPIN++ - deterministic embedded test", *Proc. Euro. Test Workshop*, pp. 37-44, 2002.

[24] P.Wohl, J.Waicukauski, S.Patel, and M.Amin, "Efficient Compression and Application of Deterministic Patterns in a Logic BIST architecture", *Proc. DAC*, pp. 566-569, 2003.

[25] F.Wolff and C.Papachristou, "Multiscan-based test compression and hardware decompression using L77", *Proc. ITC*, pp. 331-339, 2002.

[26] N. Zacharia, J. Rajski, and J. Tyszer, "Decompression of test data using variable-length seed LFSRs", *Proc. of VTS*, pp. 426-433, 1995.