

A HYBRID CODING STRATEGY FOR OPTIMIZED TEST DATA COMPRESSION

Armin Würtenberger, Christofer S. Tautermann, Sybille Hellebrand
University of Innsbruck, Austria

Abstract

Store-and-generate techniques encode a given test set and regenerate the original test set during the test with the help of a decoder. Previous research has shown that run-length coding, particularly alternating run-length coding, can provide high compression ratios for the test data. However, experimental data show that longer run-lengths are distributed sparsely in the code space and often occur only once, which implies an inefficient encoding. In this study a hybrid encoding strategy is presented which overcomes this problem by combining both the advantages of run-length and dictionary-based encoding. The compression ratios strongly depend on the strategy of mapping don't cares in the original test set to zeros or ones. To find the best assignment an algorithm is proposed which minimizes the total size of the test data consisting of the encoded test set and the dictionary. Experimental results show that the proposed approach works particularly well for larger examples yielding a significant reduction of the total test data storage compared to pure alternating run-length coding.

1 Introduction

Today's systems-on-a-chip usually integrate a variety of different cores. This design style allows a very flexible and efficient product development, but on the other hand, testing these systems becomes increasingly challenging [36]. Most of the problems, resulting from inaccessible cores, the need for at-speed testing, the individual test requirements of different cores, etc., can be tackled by built-in self-test (BIST) approaches. In particular, embedded test pattern generators (TPG) and test response evaluators (TRE) can provide a high bandwidth for handling test data while only a low-bandwidth connection to a lightweight ATE is required (see Figure 1).

Known strategies for on-chip test pattern generation comprise pseudo-random and weighted random patterns, pseudo-exhaustive patterns as well as mixed-mode techniques combining pseudo-random and deterministic patterns [1]. The latter can guarantee a high fault coverage for random pattern resistant circuits and work particularly

well, if structural information about the device under test is available. Then the BIST scheme can be specifically designed for the circuit as in bit-flipping and bit-fixing approaches or the BIST resources can be optimized using fault simulation and special ATPG procedures [15, 16, 22, 24, 26, 29, 31, 34].

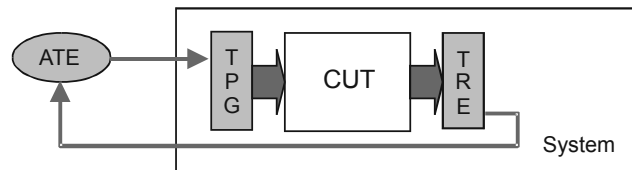


Figure 1: Reducing bandwidth for external access with BIST.

However, if the system contains IP blocks coming only with a set T of test patterns, then techniques are required to store the given test set T in a compact representation and regenerate it during test. For this task compression codes offer a two-fold solution:

- (1) The test data volume can be greatly reduced by simple encoding and decoding procedures.
- (2) As illustrated in Figure 2, there is a natural option for test resource partitioning, i.e. it is sufficient to implement the decoder on chip, and the encoded test data can be stored either on or off chip.

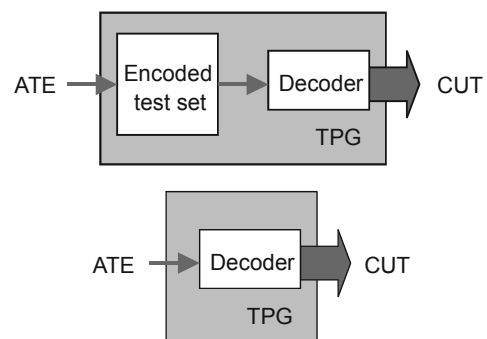


Figure 2: Store-and-generate approach for test pattern generation.

Recent coding strategies for test data compression are based on classical techniques such as statistical coding, run-length coding or dictionary-based coding [4 - 9, 11,

13, 14, 18 - 21, 27, 30, 32, 33, 35]. In particular, run-length coding has been studied intensively ranging from the application of Golomb codes to more sophisticated schemes specifically adapted to the situation of test data compression. For most run-length codes the decoder is independent of the test set, which allows to share it among multiple cores and supports an efficient implementation. With respect to the encoding efficiency, it has been shown that frequency directed run-length (FDR) coding efficiently exploits the fact that typical test data contain shorter runs of zeros or ones with higher frequencies than longer runs [6, 7, 11]. As shown in Figure 3, FDR coding maps a run of length r to a prefix of length k ($k-1$ ones terminated by zero) and a tail of length k (all combinations of k bits), if and only if the equation $2^k - 2 \leq r < 2^{k+1} - 2$ holds.

Run-length r	k	Prefix	Tail	Codeword
0	1	0	0	00
1			1	01
2	2	10	00	1000
3			01	1001
4			10	1010
5			11	1011
6	3	110	000	110000
7			001	110001
...				

Figure 3: Coding table for FDR coding

To encode a test set T , Chandra, Chakrabarty and other authors propose to decompose it into runs of zeros followed by a single one and encode the lengths of the runs using the table of Figure 3 [5, 6, 7, 8, 9, 11, 19, 21]. The advantage of this encoding strategy (“uni-phase” coding) is that it doesn’t require any extra information to distinguish between runs of zeros and runs of ones. But this also means that runs of ones are encoded very inefficiently, because only the first one is produced automatically as the final one of the preceding run of zeros. Each of the remaining ones must be represented as a run of zeros of length zero, and overall $2(r-1)$ bits are required to encode a run of r ones.

One strategy to overcome this problem is to consider the differences between successive test vectors rather than the test vectors themselves. Then the data to be encoded usually contain only a very low percentage of ones [5, 6, 7, 8, 9, 11, 19]. A more efficient solution, referred to as “alternating run-length coding” has been independently proposed in [10] and [17]. It is based on the observation that a fully specified test set T is composed of alternating runs of zeros and runs of ones. Consequently, a test set T

can simply be encoded as a sequence of run-lengths. To decode such a sequence it is only necessary to know whether the first run-length corresponds to a run of zeros or ones. For any other run-length in the sequence it is then obvious which type of run it corresponds to. It has been shown that alternating run-length coding combined with the FDR code of Figure 3 efficiently encodes both runs of zeros and runs of ones and, in general, achieves better compression ratios than conventional uni-phase coding.

However, a more thorough analysis of the experimental data shows that for alternating run-length coding the assignment of don’t cares in the test set to zeros or ones provides a high optimization potential, which has not yet been fully exploited by the heuristics used so far. Secondly, the coding table for the FDR code contains many code words, in particular for the longer run-lengths, which are used only once or which are “useless”, because not all possible run-lengths really occur as run-lengths in the test set. As an example Figure 4 shows the distribution of run-lengths for the benchmark circuit s13207 (test set for the hard faults after 10000 random patterns in [17], frequencies greater than 10 are cut off to allow a larger scaling factor).

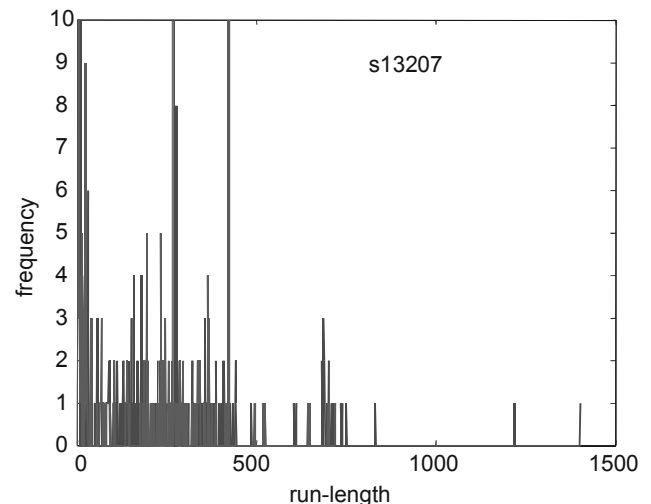


Figure 4: Distribution of run-lengths for s13207.

The maximum run-length is 1401, but overall there are only 270 different run-lengths. Run-length 1401 is for example encoded by 20 bits, although there are unused code words with only 8 bits. Obviously, a much better compression ratio could be expected, if a smaller code space were designed in such a way, that it contained only “useful” code words.

Taking advantage of this observation a hybrid coding strategy is proposed in this paper, which combines alternating run-length coding with a dictionary for the run-lengths. The strategy is supported by an optimization

algorithm which maps don't cares in the test set to specified values, such that the overall test data storage is minimized. The experimental results show, that the proposed encoding scheme combined with the minimization algorithm can achieve significantly better compression ratios than pure run-length coding.

The rest of this paper is organized as follows: In section 2 the basic concepts of alternating run-length coding will be briefly reviewed. Section 3 explains the proposed encoding strategy in detail, and subsequently in section 4 the optimization technique is described in detail. Finally, experimental results and conclusions are provided in sections 5 and 6.

2 Review of Alternating Run-Length Coding

Alternating run-length coding exploits the fact that a fully specified test set T is composed of alternating runs of zeros and runs of ones. As shown in Figure 5 a test set T can therefore simply be encoded as a sequence of run-lengths. In principle, this approach can be combined with any known run-length code, and in the example the FDR code of Figure 3 is used.

Original test data	0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1
Run-lengths	2 6 4 6
Encoded test data	0 1000 110000 1010 110000

Figure 5: Alternating run-length coding combined with the FDR code of Figure 3.

The original test data are decomposed into runs of zeros and runs of ones. The encoded data start with a zero indicating that the first run contains zeros, then the codes for the run-lengths follow. Overall, 21 bits are needed to store the encoded test data. Further reduction is possible, since the code for the run-length zero is not needed for alternating run-length coding, and the coding table of Figure 3 can be shifted, such that a run of length r is mapped to a prefix of length k and a tail of length k , if and only if $2^k - 1 \leq r < 2^{k+1} - 1$ holds. The resulting coding table is shown in Figure 6. Figure 7 shows that this way the example data can be encoded with only 15 bits.

The decoding procedure is very simple: The first bit of an encoded test data stream determines whether the test set starts with a run of zeros or with a run of ones. If the first bit is zero as in the example, then the following code word corresponds to a run of zeros, the next code word to a run of ones, etc. As mentioned above alternating run-length coding can be combined, with any known run-

length code. If a decoder for the basic run-length code is available, then only small modifications are required to re-use it for the alternating run-length code. To keep track of the alternating types of runs a toggle flip-flop can be added, eg. [10, 17].

Run-length r	k	Prefix	Tail	Codeword
1	1	0	0	00
2			1	01
3	2	10	00	1000
4			01	1001
5			10	1010
6			11	1011
7	3	110	000	110000
9			001	110001
...				

Figure 6: Coding table for the shifted FDR code.

Original test data	0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1
Run-lengths	2 6 4 6
Encoded test data	0 01 1011 1001 1011

Figure 7: Alternating run-length coding combined with the shifted FDR code.

3 A Hybrid Coding Strategy

3.1 Basic Principles

As pointed out in the introduction alternating run-length coding can efficiently represent both runs of zeros and runs of ones. But the fact that in general the set of actually needed run-lengths does not correspond to a contiguous coding table leads to unnecessarily long code words, in particular for the longer run-lengths. A classical strategy to deal with such a situation is to work with a dictionary [30]. Usually dictionary-based data compression schemes are directly applied to the original data source and recurring patterns are stored in a dictionary [27, 33]. The encoded data then consist of a list of references to the dictionary. To encode the test sequence of Figure 5, a dictionary with entries for '00', and '111111' can for example be used. Instead of these patterns only the respective indices 1 or 2 in the dictionary are needed to represent the sequence. The dictionary itself can again be encoded as a sequence of patterns followed by an escape signal to indicate the end of the dictionary.

Using the FDR code of Figure 3, the code for the run-length zero is an appropriate escape signal. Figure 8 shows the result.

Original test data	0 0 1 1 1 1 1 0 0 0 1 1 1 1 1
Dictionary Index	1 2 1 1 2
Encoded test data	01 1000 01 01 1000

Dictionary:	
Index	Pattern
1	00
2	111111

Figure 8: Dictionary-based encoding.

In the example the test sequence can be decomposed into a sequence of only 2 different patterns. However, for a general test sequence, such a regular structure cannot be expected, and the required dictionary may get very large. Since the dictionary has to be stored together with the encoded test data, a pure dictionary-based approach may thus lead to a relatively high volume of test data storage.

To overcome this problem, the proposed approach combines the advantages of both run-length coding and dictionary-based encoding by the following four step encoding process:

- (1) First the original test set is decomposed into a sequence of alternating runs of zeros and ones.
- (2) Then a dictionary is built listing the run-lengths according to their frequencies of occurrence. I. e. the most frequent run-length has the lowest index in the dictionary.
- (3) The sequence of run-lengths obtained in step (1) is transformed into a sequence of dictionary indices.
- (4) The dictionary indices are encoded using the shifted FDR code of Figure 6.

Figure 9 illustrates this approach and compares it to the basic alternating run-length coding described in section 2. The example shows that the proposed hybrid approach reduces the size of the encoded test data from 15 to 11 bits. However, to reproduce the original test set the dictionary must also be delivered with the encoded test data. In the example the dictionary then corresponds to the sequence (6, 2, 4, 0) which is encoded as 1011 1000 1010 00. Thus 14 bits are required to encode the dictionary, and overall 25 bits are needed to characterize the test data. Consequently, for this small example, the purpose of which is only to demonstrate the encoding principle, the proposed approach cannot improve the compression ratio. But it will be shown in section 5 that the use of a dictionary

leads to significant improvements of the compression ratios for larger examples.

Original test data	0 0 1 1 1 1 1 0 0 0 1 1 1 1 1
Run-lengths	2 6 4 6
<i>a) Basic alternating run-length code (15 bits)</i>	
Encoded test data	0 01 1011 1001 1011
<i>b) Hybrid approach (11 bits + dictionary)</i>	
Dictionary indices	2 1 3 1
Encoded test data	0 01 00 1000 00

Dictionary:	
Index	Run-length
1	6
2	2
3	4

Figure 9: Hybrid encoding strategy.

3.2 Decoding

As explained above the original test data are represented by an encoded dictionary and the encoded test data obtained using the hybrid run-length and dictionary-based encoding. To decode this information, first the dictionary has to be decoded and stored on chip. Then the encoded test data have to be decoded using a decoder for alternating run-length coding and the dictionary. Since the dictionary is also encoded as a sequence of run-lengths, the core of the decoding hardware consists of a small RAM for the dictionary and a decoder for run-length codes. The decoder must be able to provide the run-lengths in binary representation and to switch between the FDR code used for the dictionary and the shifted FDR code used for the test data.

Such a decoder can be built in a similar way as the decoder for alternating run-length coding [10, 17]. The design is based on the observation that for the shifted FDR code the run-length r corresponding to a codeword (p, t) with prefix p and tail t can be determined as $r = n(p) + n(t) + 1$, where $n(p)$ and $n(t)$ denote the integers obtained by interpreting p and t as binary numbers. The number $n(p)$ is always of the form $2^k - 2$, where k is the length of both the prefix and the tail. Therefore a simple strategy to get the binary representation of $r = n(p) + n(t) + 1$ works as follows:

While the prefix is read, a counter (*counter1* in Figure 10) is incremented to determine the value of k . When the end of the prefix is reached (indicated by the terminating zero), a one is shifted into a second counter (*counter2* in Figure 10) followed by the tail bits. After each tail bit the first counter is decremented, and the procedure stops when the counter reaches zero. The initial one shifted into the counter has then reached the position corresponding to 2^k , i.e. the counter contains the value $2^k + n(t) = n(p) + 2 + n(t) = r + 1$. Decrementing the counter once provides the binary representation of r . For the original FDR code of Figure 3 basically the same procedure can be used. Here $r = n(p) + n(t)$ holds, and the second counter has to be decremented twice after the tail has been shifted in.

An implementation of the complete decoder for the hybrid coding scheme is sketched in Figure 10. Please note that the counters are not standard counters, but have extra features as described above. To simplify the presentation, they are however just labeled as counters.

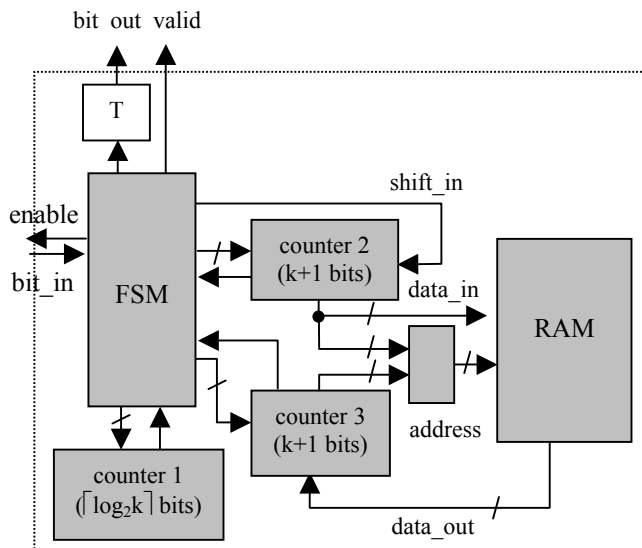


Figure 10: Block diagram for the decoding hardware

The finite state machine controls the decoding process in the following way: In the beginning *counter3* is reset to zero, and the decoding of the dictionary starts. Each binary representation of a run-length generated as described above in *counter2* is loaded into the RAM at the address selected by *counter3*. After each decoded run-length the counter is incremented, and the first decoding phase stops when the run-length zero is decoded and the dictionary is complete. During the second decoding phase the toggle flip-flop changes its value after each run produced at the output *bit_out*. For each code word received the binary representation produced in *counter2* is used as an address of the RAM and the contents at this address is loaded into *counter3*. While *counter3* is decremented the signal *valid*

is set to one, which accomplishes the synchronization with the CUT, e.g. by controlling the scan clock in a scan-based BIST environment. This way a run of zeros or ones is provided at the output depending on the contents of the toggle flip-flop. The communication with the ATE is established by the signals *enable* and *bit_in*.

It should be noted that the design of the decoder is independent of the specific test set. If the parameter k and the size of the RAM are selected according to the maximum run-length and the maximum number of different run-lengths to be expected it can easily be used to generate tests for multiple cores in a system.

4 Optimizing the Code

The efficiency of the proposed encoding strategy depends both on the size of dictionary and the size of the encoded test data. The size of the dictionary is mainly determined by the number of different run lengths while the size of the test data depends on the distribution of runs. Both parameters can be controlled to a large extent by the assignment of the don't cares in the original test set to zeros or ones. Already for the small example in Figure 11 the impact of the assignment becomes very clear.

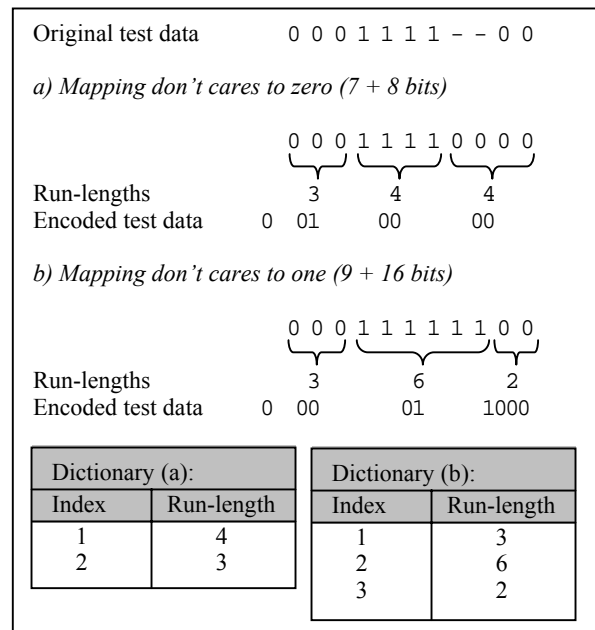


Figure 11: Impact of don't care assignment.

In Figure 11 a) the don't cares are mapped to zeros which results in a dictionary with only two words and an encoded test data stream of only 7 bits. The dictionary can be encoded with 8 bits, such that a total of 15 bits is needed to represent the test data. The complementary assignment in Figure 11 b) leads to a dictionary with 3

words requiring 16 bits for encoding. The encoded test data stream contains 9 bits, and overall 25 bits are needed. In general, it is a complex optimization problem to find the best assignment of don't cares. The search space consists of all possible assignments and grows exponentially with the number of don't cares in the test set. To reduce the search space, in the following only assignments extending existing runs are considered. As illustrated in Figure 12, the assignment problem is then reduced to finding the best boundary positions between runs of complementary values.

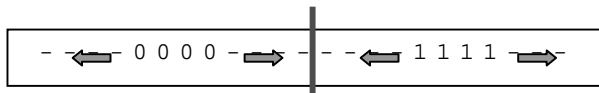


Figure 12: Assignment of don't cares.

To solve this reduced, but still very complex optimization problem any general purpose optimization technique could be employed, as for example a genetic algorithm or simulated annealing [12, 23, 25, 28]. In this project a simulated annealing procedure has been implemented.

For a set of states S and a cost function $c: S \rightarrow \mathbb{N}$ simulated annealing randomly generates a sequence of state transitions and converges to the state with minimal cost function. If a transition from a state s to a state s' leads to a cost improvement it is always accepted, otherwise it is only accepted with probability

$$e^{-\frac{c(s)-c(s')}{K \cdot t}}$$

where K is a constant and the "temperature" t is a control parameter which decreases in the course of the algorithm.

For the reduced assignment problem the states represent the boundary positions between runs of complementary values and the cost function is the total test data storage in bits, i.e. the size of the dictionary plus the size of the encoded test data. The constant K is adjusted to the specific problem size and is obtained as $K = c_0 / t_0$, where c_0 denotes the initial value of the cost function and t_0 the initial temperature.

5 Experimental Results

To evaluate the proposed encoding scheme a series of experiments has been performed for the ISCAS 85 and the full-scan versions of the ISCAS89 benchmark circuits [2, 3]. For each circuit test sets have been generated using a commercial ATPG tool to generate test cubes for the hard faults remaining after 10000 pseudo-random patterns. The test sets have been encoded using alternating run-length coding combined with the shifted FDR code of Figure 6 [17], and the proposed hybrid encoding strategy. For the first approach don't cares following a run of zeros were mapped to zeros, and don't cares following a run of ones

were mapped to one. The don't care assignment for the hybrid coding scheme was optimized with the simulated annealing procedure described in the previous section using the following cooling schedule:

Overall, 450,000 state transitions have been generated with an initial temperature of $t_0 = 0.3$. For each circuit $K = c_0 / t_0$ is then determined individually by the initial value of the cost function c_0 . The temperature t has been reduced every 150 steps by multiplying t with a factor $1/s$, where s denotes the total number of transitions performed so far. After 300,000 transitions the temperature has been heated up again, and the final 150,000 transitions have been started with an initial temperature of $t_1 = 0.00007$.

Table 1 shows the results for the proposed hybrid encoding strategy. The first two columns show the names of the circuit and the size of the original test set in bits. Then the sizes of the encoded test sets and of the required dictionaries are shown. The last two columns report the total test data storage (encoded test set + dictionary) and the compression ratio (total test data storage for hybrid encoding divided by the size of the original test set). It can be observed that in particular the larger circuits are compressed very efficiently.

Circuit	Original test set (bits)	Hybrid encoding			
		Encoded test set (bits)	Dictionary (bits)	Total test data storage (bits)	Compression ratio
c2670	26096	3229	356	3585	0,14
c7552	26703	6059	534	6593	0,25
s420	1088	397	130	527	0,48
s838	10692	2445	350	2795	0,26
s1196	320	201	52	253	0,79
s1238	448	273	66	339	0,76
s5378	5992	687	210	897	0,15
s9234	73112	13287	640	13927	0,19
s13207	220500	7678	896	8574	0,04
s15850	163748	8514	964	9478	0,06
s38417	2201472	68359	5482	73841	0,03
s38584	456768	9185	1602	10787	0,02

Table 1: Results for the hybrid encoding strategy.

As explained in section 3, the binary representations of the run-lengths are stored in a RAM to have the dictionary available on chip during the decoding of the test data. Since binary representations are used, the RAM size only depends on the number of different run-lengths. Assuming a word-oriented RAM with 16-bit words, the RAM sizes for the example circuits are determined by the num-

bers shown in Table 2 (number of different run-lengths multiplied by 16).

Circuit	RAM (bits)	Circuit	RAM (bits)
c2670	592	s5378	352
c7552	864	s9234	992
s420	304	s13207	1072
s838	688	s15850	1216
s1196	144	s38417	5328
s1238	176	s38584	1792

Table 2: Size of the RAM for the on-chip dictionary.

To compare the results for the proposed hybrid encoding strategy to pure alternating run-length coding Table 3 lists the total test data storage in bits and the number of different run-lengths for both approaches. Additionally, the last column shows the total test data storage for the proposed approach as a fraction of the total test data storage for pure alternating run-length coding.

Circuit	Alternating run-length coding [17]		Hybrid encoding		$\frac{II}{I}$
	I: Total test data storage (bits)	# run-lengths	II: Total test data storage (bits)	# run-lengths	
c2670	4903	95	3585	37	0,73
c7552	7483	99	6593	54	0,88
s420	533	27	527	19	0,99
s838	3443	61	2795	43	0,81
s1196	231	13	253	9	1,10
s1238	319	15	339	11	1,06
s5378	1091	26	897	22	0,82
s9234	20429	134	13927	62	0,68
s13207	15061	270	8574	67	0,57
s15850	12861	220	9478	76	0,74
s38417	90397	538	73841	333	0,82
s38584	15083	314	10787	112	0,72

Table 3: Comparing pure alternating run-length coding to the hybrid encoding strategy.

The results show that the proposed technique doesn't lead to a further improvement for circuits s420, s1196, and s1296. Here the original test set is already rather small, and an encoding may not be necessary at all. However, for the larger examples the proposed approach leads to a significant reduction of the total test data storage. The storage requirements for pure alternating run-length

coding have been tremendously reduced: In the best case 57% of the test data volume are sufficient to characterize the complete test data. Overall the reduced test data storage ranges between 57% and 88% of the storage requirements for pure alternating run-length coding.

The efficiency of the employed optimization procedure for the don't care assignment becomes also clear by comparing the number of different run-lengths. For all examples the heuristic assignment proposed in [17] leads to a much higher number of different run-lengths than the proposed optimization technique. Figure 13 shows the distribution of run-lengths for s13207 before and after optimization in more detail.

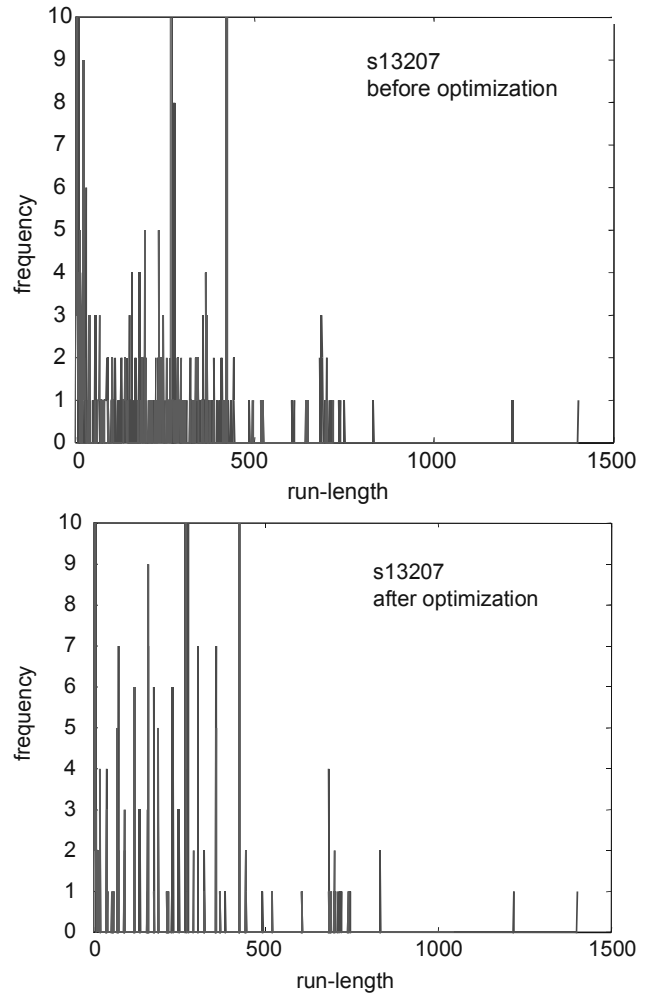


Figure 13: Distribution of run-lengths before and after optimizing the don't care assignment for s13207.

The histograms (frequencies larger than 10 are cut off to allow a better scaling factor) show that after the optimization there are less run-lengths occurring with increased frequencies. The evolution of the cost function during the simulated procedure is shown in Figure 14.

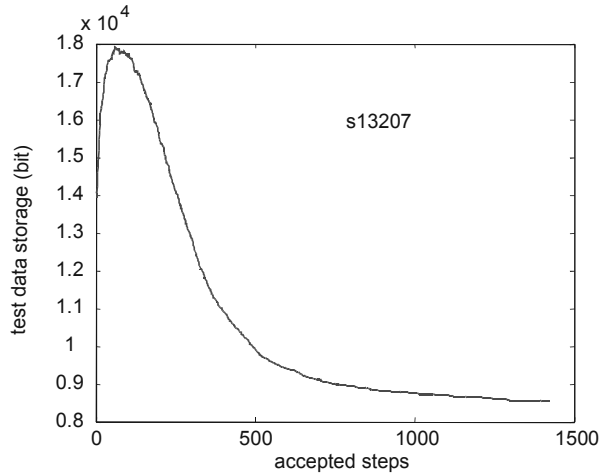


Figure 14: Evolution of the cost function during the simulated annealing procedure.

To relate the proposed approach also to other approaches based on run-length coding, the test sets have been encoded using uni-phase FDR coding [6] and alternating run-length coding combined with the FDR code of Figure 3 [10]. For uni-phase run-length coding the don't cares in the test sets were mapped to zeros, for the alternating run-length coding proposed in [10] the don't cares following a run of zeros were again mapped to zeros, and don't cares following a run of ones were mapped to one. The results are summarized in Table 4.

Circuit	Size of original test set (bits)	Total test data storage (bits)			
		Uni-phase FDR coding [6]	ARL coding & FDR code [10]	ARL coding & shifted FDR code [17]	Hybrid Approach
c2670	26096	7730	5016	4903	3585
c7552	26703	9606	8056	7483	6593
s420	1088	602	582	533	527
s838	10692	4540	3582	3443	2795
s1196	320	272	260	231	253
s1238	448	356	364	319	339
s5378	5992	1110	1160	1091	897
s9234	73112	22038	21222	20429	13927
s13207	220500	14080	15534	15061	8574
s15850	163748	14370	13286	12861	9478
s38417	2201472	103760	94682	90397	73841
s38584	456768	13840	15696	15083	10787

Table 4: Comparison of coding strategies.

For almost all circuits the same trend can be observed. Conventional uni-phase run-length coding has the largest

storage requirements. Pure alternating run-length coding reduces the total test data storage, and the shifted FDR code of Figure 6 leads to better results than the original coding table of Figure 3. Consequently, compared to other approaches based on run-length coding the proposed hybrid encoding strategy leads to even higher reductions in the total test storage as shown above for the pure alternating run-length coding combined with the shifted FDR code of Figure 6.

6 Conclusions

Store-and-generate approaches based on compression codes offer an efficient solution for testing IP blocks coming only with a set of pre-computed test patterns. In particular alternating FDR codes can provide an efficient and flexible solution with a simple and circuit independent decoding mechanism. The proposed hybrid encoding scheme overcomes inefficiencies due to a sparse distribution of run-lengths and low frequencies of occurrence by exploiting both the advantages of run-length coding and the advantages of dictionary based encoding. Combined with an algorithm for an optimal assignment of don't cares the proposed strategy leads to considerable reductions of the total test data storage. For the larger ISCAS85 and ISCAS89 examples the number of bits to be stored ranges between 57% to 88% of the storage requirements for the best previously published approaches.

7 References

- 1 M. Abramovici, M. Breuer, A. Friedman: Digital Systems Testing and Testable Design; New York: Computer Science Press (W. H. Freeman and Co.), 1990
- 2 F. Brglez et al.: Accelerated ATPG and fault grading via testability analysis; Proc. IEEE Int. Symp. on Circuits and Systems, Kyoto, Japan, 1985
- 3 F. Brglez, D. Bryan and K. Kozminski: Combinational Profiles of Sequential Benchmark Circuits; Proc. IEEE Int. Symp. on Circuits and Systems, 1989, Portland, OR, USA, pp. 1929-1934
- 4 J. Capon: A Probabilistic Model for Run-Length Coding of Pictures; IRE Trans. on Information Theory, December 1959, pp. 157-163
- 5 A. Chandra, K. Chakrabarty: System-on-a-chip test data compression and decompression architectures based on Golomb codes; IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 20, No. 3, March 2001, pp. 355-368
- 6 A. Chandra, K. Chakrabarty: Frequency-Directed Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression; Proc. 19th VLSI Test Symp., Marina Del Rey, CA, USA, 2001, pp. 42-43

- 7 A. Chandra, K. Chakrabarty: Test Resource Partitioning and Reduced Pin-Count Testing Based on Test Data Compression; Proc. 2002 Design and Test in Europe, Paris, France, March 4-8, 2002, pp. 598-603
- 8 A. Chandra, K. Chakrabarty: Low Power Scan Testing and Test Data Compression for System-on-a-Chip; IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 21, No. 5, May 2002, pp. 597-604
- 9 A. Chandra, K. Chakrabarty: Test Data Compression and Decompression Based on Internal Scan Chains and Golomb Coding; IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 21, No. 6, June 2002, pp. 715-722
- 10 A. Chandra, K. Chakrabarty: Reduction of SoC Test Data Volume, Scan Power and Testing Time Using Alternating Run-length Codes; Proc. ACM/IEEE Int. Design Autom. Conf., 2002, New Orleans, LA, USA, pp. 673-678
- 11 A. Chandra, K. Chakrabarty, R. A. Medina: How Effective are Compression Codes for Reducing Test Data Volume ?; Proc. 20th IEEE VLSI Test Symp., Monterey, CA, USA, 2002, pp. 91-96
- 12 D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning; Reading, MA: Addison-Wesley, 1989
- 13 S. W. Golomb: Run-Length Encodings; IEEE Trans. on Information Theory, July 1966, pp. 399-401
- 14 P. T. Gonciari, B. M. Al-Hashimi, N. Nicolici: Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-Chip Test Data Compression / Decompression; Proc. 2002 Design and Test in Europe, Paris, France, March 4-8, 2002, pp. 604-611
- 15 S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois: Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers; IEEE Trans. on Comp., Vol. 44, No.2, February 1995, pp. 223-233
- 16 S. Hellebrand, B. Reeb, S. Tarnick, H.-J. Wunderlich: Pattern Generation for a Deterministic BIST Scheme; Proc. IEEE/ACM Int. Conf. on CAD-95, San Jose, CA, USA, November 1995, pp. 88-94
- 17 S. Hellebrand, A. Würtenberger: Alternating Run-Length Coding – A Technique for Improved Test Data Compression; IEEE Int. Workshop on Test Resource Partitioning, Baltimore, MD, USA, October 2002
- 18 M. Ishida, D. S. Ha, T. Yamaguchi: Compact: A Hybrid Method for Compressing Test Data; Proc. 16th IEEE VLSI Test Symp., Monterey, CA, USA, 1998, pp. 62-69
- 19 A. Jas, N. A. Touba: Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs; Proc. IEEE Int. Test Conf., Washington, DC, USA, 1998, pp. 458-464
- 20 A. Jas, J. Gosh-Dastidar, N. A. Touba: Scan Vector Compression/Decompression Using Statistical Coding; Proc. 17th IEEE VLSI Test Symp., Dana Point, CA, USA, 1999, pp. 114-120
- 21 D. Kay, S. Mourad: Compression Technique for Interactive BIST Application; Proc. 19th VLSI Test Symp., Marina Del Rey, CA, USA, 2001, pp. 9-14
- 22 G. Kiefer, H. Vranken, E. J. Marinissen, H.-J. Wunderlich: Application of Deterministic Logic BIST on Industrial Circuits; Proc. IEEE Int. Test Conf., ITC 2000, Atlantic City, NJ, USA, October 3 - 5, 2000
- 23 S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi: Optimization by Simulated Annealing; Science, Vol. 220, No. 4598, May 1983, pp. 671 - 680
- 24 B. Koenemann: LFSR-Coded Test Patterns for Scan Designs; Proc. Europ. Test Conf., Munich, Germany, 1991, pp. 237-242
- 25 P. J. M. Laarhoven, E. H. L. Aarts: Simulated Annealing: Theory and Applications; Dordrecht, Boston, Lancaster, Tokyo: Verlag D. Reidel Publishing Company, 1987
- 26 Hua-Guo Liang, S. Hellebrand, H.-J. Wunderlich: Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST; Proc. IEEE Int. Test Conf., Baltimore, MD, USA, November 2001
- 27 L. Li, K. Chakrabarty: Dictionary-Based Test Data Compression; Proc. 21st IEEE VLSI Test Symp. (VTS 2003), Napa Valley, CA, USA, April 2003
- 28 N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller: Equation of State Calculations by Fast Computing Machines; J. Chem. Phys., 21, 1953, pp. 1087 - 1092
- 29 J. Rajski, J. et al.: Embedded Deterministic Test for Low Cost Manufacturing Test; Proc. IEEE Int. Test Conf., Baltimore, MD, USA October 2002, pp. 301-310
- 30 K. Sayood: Introduction to Data Compression; London, San Francisco: Academic Press – The Morgan Kaufmann Series in Multimedia Information and Systems, 2nd Edition, 2000
- 31 N. A. Touba and E. J. McCluskey: Altering a Pseudo-Random Bit Sequence for Scan-Based BIST; Proc. IEEE Int. Test Conf., Washington, DC, USA, 1996, pp. 167-175
- 32 E. H. Volkerink, A. Khoche, S. Mitra: Packet-based Input Test Data Compression Techniques; Proc. IEEE Int. Test Conf.; Baltimore, MD, USA October 2002, pp. 154-163
- 33 F. G. Wolff, C. Papachristou: Multiscan-based Test Compression and Hardware Decompression Using LZ77; Proc. IEEE Int. Test Conf., Baltimore, MD, USA, October 2002, pp. 331-339
- 34 H.-J. Wunderlich, G. Kiefer: Bit-Flipping BIST; Proc. ACM/IEEE Int. Conf. on CAD-96 (ICCAD96), San Jose, CA, USA, November 1996, pp. 337-343
- 35 T. Yamaguchi, M. Tilgner, M. Ishida, D. S. Ha: An Efficient Method for Compressing Test Data; Proc. IEEE Int. Test Conf., Washington, DC, USA, 1997, pp. 79-88
- 36 Y. Zorian: Testing the monster chip; IEEE Spectrum, July 1999, pp. 54-60