International Journal of Engineering & Technology, 7 (4.10) (2018) 1089-1094

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Test Data Compression with Alternating Equal-Run-Length Coding

Sivanantham S[1]*, Aravind Babu S[2], Babu Ramki S,[2] Mallick P.S[3]

[1]*School of Electronics Engineering, Vellore Institute of Technology, Vellore, India*
[2]*Intel India Pvt Ltd, Bangalore, India*
[3]*School of Electrical Engineering, Vellore Institute of Technology, Vellore, India*
**Corresponding author E-mail: ssivanantham@vit.ac.in*

Abstract

This paper presents a new X-filling algorithm for test power reduction and a novel encoding technique for test data compression in scan-based VLSI testing. The proposed encoding technique focuses on replacing redundant runs of the equal-run-length vector with a shorter codeword. The effectiveness of this compression method depends on a number of repeated runs occur in the fully specified test set. In order to maximize the repeated runs with equal run length, the unspecified bits in the test cubes are filled with the proposed technique called alternating equal-run-length (AERL) filling. The resultant test data are compressed using the proposed alternating equal-run-length coding to reduce the test data volume. Efficient decompression architecture is also presented to decode the original data with lesser area overhead and power. Experimental results obtained from larger ISCAS'89 benchmark circuits show the efficiency of the proposed work. The AERL achieves up to 82.05 % of compression ratio as well as up to 39.81% and 93.20 % of peak and average-power transitions in scan-in mode during IC testing.

Keywords: *Test data compression; design for testability; low-power testing; run-length encoding; decompression; X-filling.*

1. Introduction

The amount of data required to test the integrated circuits (ICs) are increasing rapidly with the developments of technology. Also, the design of low-power high-performance portable computing devices has become a major objective for the design engineers. However, reduction of power dissipation is not only a critical parameter for design engineers, but also for design for testability (DFT) engineers as the system consumes much more power during the test than during normal operation [1]. Thus, low-power test data compression for digital VLSI systems has become a major concern for engineers and scientists of these areas in recent years. Due to the increase in the test data volume and higher test power, this area has always been actively researched on and a number of test data compression and power reduction techniques are introduced. Test data compression techniques can be broadly categorized into linear decompression based compression and code-based compression techniques. Reduction of test data volume using test compaction was described in [2, 3]. Test compaction techniques reduce also the test application time. However, the compacted test sets limit the detection of many non-modeled physical defects.

Linear compression schemes are very efficient in exploiting unspecified bits in the test cubes to achieve a large amount of compression. Several techniques were proposed based on LFSR reseeding to reduce the test data volume [4, 5, 6]. The LFSR reseeding techniques make use of the many unspecified bits in deterministic test patterns. The basic idea of LFSR reseeding techniques is to compute a set of seeds for LFSR that can be used to obtain the deterministic test cubes. The seed for each deterministic test cube can be computed by solving a set of linear equations based on feedback polynomial of LFSR. These seed values are expanded into actual test vector in the scan-chains with LFSR. Ward et al. [7] describe a compression scheme which combines linear decompressor with a non-linear decoder to provide a very high level of compression for test data. A technique for simultaneous reduction of both test data volume and test power named linear decompressor based test compression were presented in [8]. This scheme divides the test cubes into two blocks, test cube with low toggles and high toggles which feeds the scan-chain with novel DFT architecture to reduce the scan-in transitions. Kinsman et al. [9] present a time-multiplexing based test data compression, where the compressed seeds are passed to every embedded core by sharing the data channels. A scan architecture called reconfigured scan forest was proposed to reduce test data volume and test application cost by [10]. A new scan architecture called virtual chain partition (VCP) [11], which is useful for embedded cores to reduce the test application time, test data volume and test power. This architecture determines the maximum reduction in test cycles obtainable with the architecture and selects the most suitable configuration for each circuit.

Several other techniques such as embedded deterministic test (EDT) [12], smartBIST [13], reconfigurable serial multiplier [14] and reconfigurable interconnection network (RIN) [15] were also proposed to reduce the test data volume. Many commercial tools adopt LFSR reseeding based test data compression and combinational linear expansion networks which includes TestKompress from Mentor Graphics [12], DBIST from Synopsys, SmartBIST from IBM/Cadence [13], and ELT-Comp from LogicVision. However, these schemes require large area overhead. Also, all these methods are not suitable to test the embedded cores since

structural information of the circuits is required for test generation and fault simulation.
Another approach for test compression is to use data compression codes such as statistical coding [16, 17], Golomb coding [18] and run-length coding [19-26] to encode the test cubes. In these approaches, the original data are partitioned into symbols, and then each symbol is assigned with a codeword to form the encoded data. Each codeword is converted into the corresponding symbol with on-chip decompression hardware. These data compression codes can be further classified into four groups depending on whether the size of the symbols and codewords are fixed or variable lengths [27]. These are fixed-input to fixed-output (FIFO), fixed-input to variable-output (FIVO), and variable-input to fixed-output (VIFO) and variable-input to variable-output (VIVO) coding techniques. These methods do not require structural information about the CUTs and more suitable for intellectual property (IP) core based system-on-a-chips (SoCs).

In FIVO coding, the original test cubes are partitioned into n-bit blocks to form the symbols. These symbols are then encoded using variable-length codewords. One form of fixed-to-variable coding is statistical coding, where the idea is to calculate the frequency of occurrence of the different symbols in the original test cubes and make the codewords that occur most frequently have fewer bits and those that occur least frequently more bits. This minimizes the average length of a codeword. A Huffman code is obtained by constructing a Huffman tree. Huffman coding technique with fixed-length of blocks to reduce the test data volume is described in [16]. However, it requires complex decoder architecture to decode a large number of distinct blocks. That is, full Huffman code that encodes all *n*-bit symbols requires a decoder with $2^n -1$ state. This issue was addressed in a selective Huffman code [28], in which only the k most frequently occurring symbols are encoded. In selective Huffman coding, an extra bit is added at the beginning of each codeword to indicate whether or not it is coded. The on-chip decoder requires only $n+k$ states since this approach selectively encodes only k symbols. Also, it was shown that a selective Huffman code achieves only slightly less compression than a full Huffman code for the same symbol size while using a much smaller decoder. Because the decoder size grows only linearly with selective Huffman encoding, it is possible to use a much larger symbol size, which significantly improves the effectiveness of the code thereby achieving much more overall compression. Apart from statistical coding, there are many fixed-to-variable coding which exploits the fact that most scan slices have a relatively small number of specified bits as described in [29]. If there are b channels coming from the tester, these techniques use a variable number of b-bit codewords to decode the specified bits in each scan slice.

In FIFO coding, the test cubes are partitioned into n- bit blocks to form the symbols. These symbols are then encoded with codewords that each have b-bits, where $b < n$. Dictionary-based compression techniques are an example of FIFO coding. In these techniques, each symbol and codeword respectively can be considered as an entry in a dictionary and as an index into the dictionary that points to the corresponding symbol. There are $2n$ possible symbols and $2b$ possible codewords, so not necessarily all possible symbols can be in the dictionary.

In VIFO coding, the original test cubes are partitioned into variable length symbols, and the codewords are each with b-bits long. The test data compression based on a run-length coding with cyclical scan architecture was described in [30], to enhance the effectiveness of the basic run-length coding. In this approach, the current data to be shifted is XORed with the previous test vector with the proposed cyclical scan architecture. That is, instead of applying the original test set (TD), a different test vector set is applied. The main advantage of this scheme is that the test vectors can be reordered in such a way that more similar test vectors come after each other which will increase the number of 0s in the difference vectors.
In VIVO coding, both the symbols and codewords have a variable length. In Golomb coding [31], the codewords are divided into groups of equal size m. The value of m is given as m= 2 b, where b is the block size. The codeword is divided into two parts: the prefix and the tail. The size of the group prefix is variable while the number of bits in the tail is fixed. This run-length based Golomb code provides inefficient compression in many cases since each group contains the same number of run-lengths. Later, Chandra et al. [19] proposed a frequency-directed run-length (FDR) code that has variable-length tails based on the group index. It can be constructed such that a shorter run-length can be encoded into a shorter codeword to give better compression. The FDR code is a variable-to-variable-length code which maps variable-length runs of 0s to variable-length codewords. In FDR coding, the codewords have two parts: the prefix and the tail and they are of equal length. Thus the test data compression can be more efficient if the runs of 0s with shorter run-length are mapped to shorter codewords. The FDR is similar to Golomb code but the difference is the variable group size. For a run-length k, mapping of k is done to a group A_j where *j= [log2 (k+)-1]*. The FDR code provides an efficient test compression for the test data which has long runs of 0s and fewer 1s. However, the test data composed of both runs of 0s and 1s, so FDR coding was inefficient in achieving good compression.

The compression methods such as alternating FDR (ARL) coding [22], extended FDR (EFDR) coding [21,32], alternating variable-length (AVR) coding [24], equal-run-length coding (ERLC) [22], alternating frequency-directed equal-run-length coding (AFDER) [33], low-power selective pattern compression (LP-SPC), [34] and Shifted Alternating FDR coding [35] consider both runs of 0s as well as 1s to form the codewords. The ARL code [22] is also a variable-to-variable length code. Here, the test set T is composed of alternating runs of 0s and 1s. This coding technique considers an alternating binary variable a, and encoding for each run-length is dependent on this parameter value. If a=0, the run-length is treated as a run of 0s. On the other hand, if a=1, the run-length is treated as a run of 1s. Then *a* is inverted after each run is encoded and it keeps alternating between 0 and 1 thereafter. The default initial value of a=0, that is the input data stream starts with a run of 0s.

In ERLC encoding [22], both types of runs are considered like in EFDR scheme. The novel characteristic of this approach is that ERLC scheme explores the relationship between two consecutive runs. If there similar run-lengths occurring consecutively, shorter codewords like *000* and *100* are assigned indicating the repetition. A technique based on merging consecutive compatible blocks of the test data is presented in [36]. Other variable-to-variable codes that are not based on run-length coding include packet-based codes and nine-coded compression technique. Several Huffman based compression techniques such as variable-input Huffman coding, variable-to-variable Huffman coding, optimal selective Huffman coding, complementary Huffman coding and run-length based Huffman coding (RLHC) available to improve the compression efficiency, area overhead and the test application time[37]. Many low-power compression techniques are available in the literature for the minimization of test power, test data volume and test time [8, 33-35]. In observation-oriented test pattern generation with the scan-chain disabling technique, the test pattern generation process is assisted by testability analysis to generate the observation-oriented test patterns. A weighted compatibility analysis is performed to densely cluster the frequently-used scan cells into scan-chains. Consequently, the number of scan-chains is disabled during the capture cycle.

Many code based compression techniques have the objective of only reducing the test data volume without emphasis on test power reductions. For example, the compression techniques described in [21] focus mainly to reduce the test data volume. Several test independent compression techniques were used to reduce the test power and test data volume [21, 22, 24]. The zero-fill algorithm was used to maximize the 0-runs to reduce scan-in test power in [31]. The zero-fill algorithm fills the unspecified bits with 0's. The X-bits were filled with 0s or 1s in order to improve the skewing of the occurrence frequencies of the distinct blocks [28]. Minimum transition count (MTC) filling is used for the simultaneous reduction of test data volume and power dissipation [33]. In alternating FDR (AFDR) coding, all unspecified bits were filled to minimize the weighted transition metric (WTM). The AFDR also reports a significant reduction in the scan-out phase but achieves less compression ratio since the unspecified bits are filled to reduce the test power.

This paper presents a new X-filling algorithm to reduce the scan-power as well as to assist the test data compression by maximizing the number of consecutive equal-runs of codes. Then the proposed alternating-equal-run-length (AERL) coding encodes the long repeated runs of codes into shorter codewords.

2. Alternating equal-run-length filling for power minimization

The objective of the proposed X-filling algorithm is to increase the number of repeated alternating runs with equal length, thereby to minimize the test power as well as to enhance the compression ratio. In a test vector, a sequence of consecutive *d* bits, where $d \in \{0, 1\}$ is called a run of type *d*. Run-length is the number of bits in a run. If two consecutive runs have the same number of bits, irrespective of the run type then the runs are said to be of equal run-length. For example, *0000000* is a run of type 0 with run-length of *7*, while *1111111* is a run of type *1* with run-length of *7*. The steps involved in the alternating equal-run-length filling (AERL filling) algorithm is described as follows:

Input: Test sets with do not care (*X*) bits.
Output: Filled test sets with maximum alternating equal-run-lengths.
Assumptions: *Let startpos* be the first bit of the test set, *t* - first specified bit and subsequent *t'* and t be *midpos* and *lastpos* respectively. The portion of the test set between *startpos* and *midpos* is termed as a test slice.

1) Let *L1=(midpos-startpos) and L2=(lastpos-midpos+1)* .
2) If test slice is even go to 3 else if lastpos = X then remove the last bit.
3) If L1=L2 , then go to 5, else go to 4.
4) If L1>L2 and if midpos-1 =X then midpos =(midpos-1) and go to 3. Else if lastpos =X then lastpos=lastpos-1 and go to 3.
5) If L1<L2 and if midpos+1 =X then midpos =(midpos+1) and go to 3. Else if lastpos =X then lastpos=lastpos-1 and go to 3.
6) Fill all X-bits between startpos and midpos with t and midpos and lastpos with t' . Now set bit position next to lastpos as next startpos and restart algorithm.
7) Fill all X-bits between startpos and midpos with t and set new startpos as current midpos and restart algorithm.
8) Repeat the algorithm until all X-bits in the test cube are filled.
9) end.

The AERL filling algorithm focuses on maximizing the runs with equal run-length to achieve better compression ratio as well as fewer transitions to reduce test power. The process of filling the X-bits in the test cube with AERL-filling is demonstrated in Fig 1. The weighted transition metric (WTM) is used for measuring scan power transitions as described in [34]. A transition occurs if any two bits are different, i.e. of the type *d* followed by *d'* or vice versa. In scan mode, the bit applied at the beginning of scan chain has to

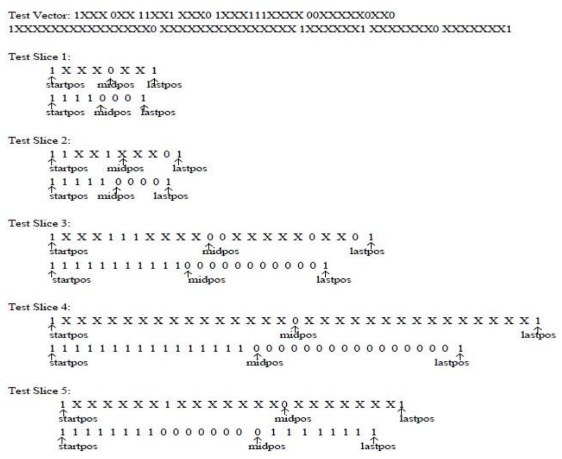


Fig.1: Illustration of filling the X-bits in the test cube with AERL-filling

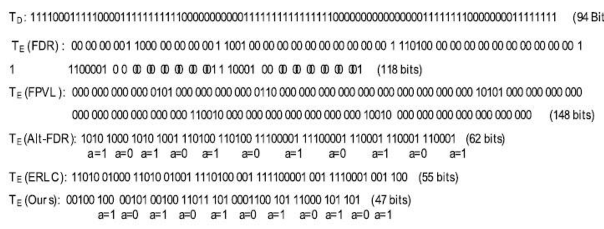
reach the last register in scan-chain passing through all registers in scan-chain. If there is a transition present between the first two bits, then the transition passes through all registers in the chain, causing more power consumption as compared to no transition in the first two bits. The effect of transition in the second and third bit will significantly lower impact compared to the first two bits as this transition will propagate the full scan chain. The average of the power consumed while loading all the test patterns is average power. The pattern for which has WTM result in high peak-power consumption.

2.1. Alternating equal-run-length (aerl) coding for test data compression

Once the given test set is filled with AERL-filling, compression can be performed with the proposed alternating equal run-length (AERL) coding. Table 1 presents the AERL encoding scheme for test data compression. The code group of a run *n* is determined using $n = [\log_2(L+5) - 2]$, where *L* is the run-length. The codeword can be formed by combining both prefix and tail of the concerned run. It is worth to note that in the given encoding scheme, the repetitive equal-run-length is coded with shorter codeword instead of the actual codeword to improve the compression. This work uses the codeword *101* for the repeated runs.

Table 1: AERL encoding scheme

Group	Run-length	Prefix	Tail	Codeword	Length
	-	10	1	101	3
A1	1	01	0	010	
	2		1	011	3
	3	10	0	100	
A2	4		00	00100	
	5	001	01	00101	
	6		10	00110	
	7		11	00111	5
	8		00	11000	
	9	110	01	11001	
	10		10	11010	
	11		11	11011	
A3	12		000	0001000	
	13	0001	001	0001001	
	14		010	0001010	
	15		011	0001011	
	16		100	0001100	
	…	…	…	…	
	20		000	1110000	
	21	1110	001	1110001	
	22		010	1110010	
	23		011	1110011	
	24		100	1110100	
		…			



```
TD: 1111000111110000111111111110000000000011111111111111100000000000000001111111100000000011111111    (94 Bits)

TE (FDR) :  00 00 00 001 1000 00 00 00 001 1001 00 00 00 00 00 00 00 00 00 00 1 110100 00 00 00 00 00 00 00 00 00 00 1
1               1100001  0 0  00  00  00  00  00  001 1 10001  00  00  00  00  00  00  001    (118 bits)
TE (FPVL ):  000 000 000 000 0101 000 000 000 000 0110 000 000 000 000 000 000 000 000 000 000 10101 000 000 000 000
             000 000 000 000 000 000 110010 000 000 000 000 000 000 000 10010  000 000 000 000 000 000 000    (148 bits)
TE (Alt-FDR): 1010 1000 1010 1001 110100 110100 11100001 11100001 110001 110001 110001   (62 bits)
               a=1  a=0  a=1  a=0   a=1     a=0     a=1      a=0     a=1    a=0    a=1
TE (ERLC): 11010 01000 11010 01001 1110100 001 111100001 001 1110001 001 100   (55 bits)
TE (Ours):  00100 100  00101 00100 11011 101 0001100 101 11000 101 101   (47 bits)
               a=1 a=0  a=1  a=0  a=1  a=0  a=1  a=0 a=1 a=0 a=1
```

Fig.2: Illustration of various encoding scheme including AERL

Figure 2 shows the example for encoding the test set with proposed AERL coding and other run-length based codes. Where TD represents the size of uncompressed test data and TE represents the size of the encoded bits. It is shown that the AERL encodes the 94-bits data into 47-bits whereas the FDR, FPVL, ALT-FDR, and ERLC schemes are encoded into 118-bits, 148-bits, 62-bits and 55-bits respectively. Better compression can be achieved as the number of repeated run-length is increased. An on-chip decoder is used to decompress the test data and provide the decompressed test data to the scan chain of the circuit under test (CUT). This work considers only the runs of different bit type occur alternatively so as to simplify the decompression architecture.

3. Decompression architecture

An on-chip decoder is used to decompress the test data and provide the decompressed test data to be applied to the scan chain. Test time can be reduced by using on-chip decoder as the chip works at a frequency higher than the ATE frequency in most cases. Figure 4 is the module level block diagram of the decompression architecture. The decompression architecture consists of a FSM, a (n+1)-bit counter, a *n=log*$_2$*(n+1)*-bit counter, a *(n+1)*-bit register, a *n=log*$_2$*(n+1)* -bit register, a T-flip flop and associated gates as shown in Fig. 3. If *Lmax* is the longest run in the test volume, then *n=log*$_2$*[(L*$_{max}$*+5)-2](n+1)* gives the bit size of the prefix of the longest codeword and *log*$_2$*(n+1* gives the size of the tail. These counters find the number of bits to be read for decompression. A T-flip flop is used to flip output for each run length that was produced by flipping a-bit in compression. Figure 4 shows the state diagram of the FSM used in decompression architecture.

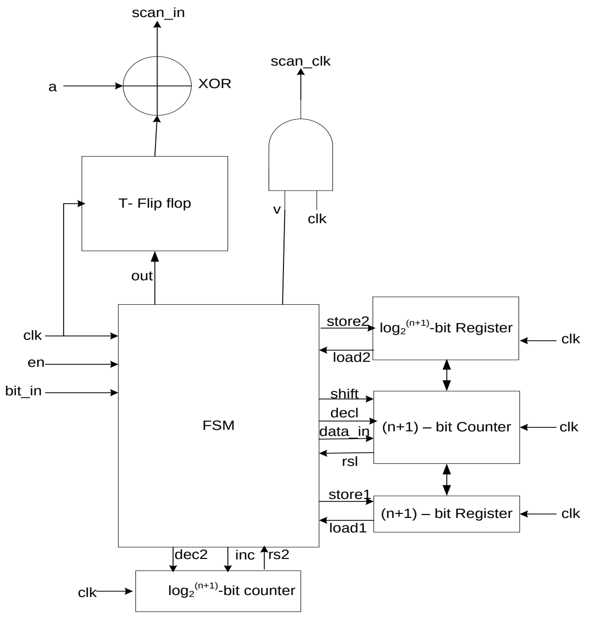


Fig. 3: Decoder Architecture

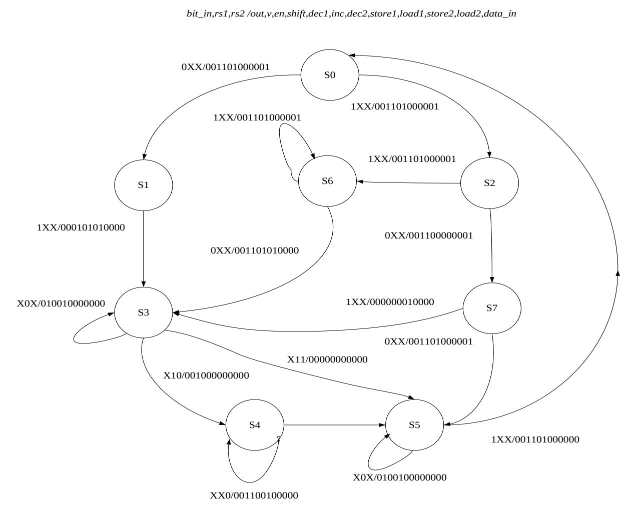


Fig. 4: State diagram for the FSM used in decompression architecture

The working of a finite state machine (FSM) used in the decompression architecture is described as follows:

- bit_in is the compressed test vector signal from ATE, that acts as input to FSM and data_in is the input to the *(n+1)*-bit counter.
- *en* signifies whether the circuit can receive *bit_in*.
- *shift* signal is used to indicate the counter to shift right, and data from *bit_in* is placed in the LSB of the *(n+1)*-bit counter.
- *dec1* and *dec2* signal are used to control the decrement of the *(n+1)*-bit counter and the $\log_2(n+1)$-bit counter respectively.
- *inc* increment the content of $n = \log_2(n+1)$-bit counter.
- *save1* and *save2* signals are used to copy content of *(n+1)*-bit counter into the *(n+1)*-bit register and the content of *log*$_2$*(n+1)*-bit counter into *log*$_2$*(n+1*-bit register respectively.
- *load1* and *load2* signals are used to copy content back from *(n+1)*-bit register into *(n+1)*-bit counter and the content of $\log_2(n+1)$-bit register into $\log_2(n+1)$-bit counter respectively.
- *rs1* signal is used to reset the FSM to *S0* state.
- *out* is the output signal from the FSM. This signal is fed to T-flip flop to produce the same output till a *1* is received, upon which the output flips. This output is received as *scan_in* output that can be connected directly to the scan-chain.
- *scan_in* is valid only at the positive edge of *scan_clk* which will be used as a scan-mode clock.
- *clk* is the system clock which controls the operation of the entire circuit.

Working of entire decompression architecture which is part of the manufactured IC used to get back the actual test vector used to test the IC are as follows: Initially FSM, counters, registers, signal *v*, signal out and T- flip flop are reset to zero. The signal is made *en=1* the circuit is ready to receive input. input from *bit_in* is received and shifted into *(n+1)*-bit register till a complement bit is received. If *1…0* type prefix is received the input is loaded directly through *data_in* and if of type *0…1* is received then the *data_in* is the complement of the bit received and it is shifted into the counter. Each bit loaded into counter sends *inc* signal to *log*$_2$*(n+1*-bit counter. When the complement bit is received, *load1*, *dec1* are set high and *inc*, *en* is made low. If the input received is *101* then *load1* and *load2* are made high. FSM outputs strings of 0's till *(n+1)*-bit counter is reset (ie. till it becomes 1). Signal *v* is made high to indicate that the output is valid.

Once *(n+1)*-bit counter is reset, the content of *log*$_2$*(n+1*-bit counter is decremented by setting *dec1* is low and *dec2*, *en* and *bit_in* signals high. This *log*$_2$*(n+1)*-bit counter determines the number of tail bits to be shifted into the *(n+1)*-bit counter. When *log*$_2$*(n+1*-bit counter becomes zero, *dec2*, *en* and *bit_in* signals are made low and dec1 is made high. FSM outputs strings of 0's till *(n+1)*-bit counter is reset (ie. till it becomes 1). Signal *v* is made high to indicate that the output is valid. On reset, *rst1* is made high and FSM is reset to *S0* state. Signal v made low to indicate the output is invalid.

4. Experimental results and analysis

The experiments were conducted on six larger ISCAS'89 benchmark circuits to validate the effectiveness of the proposed work. For comparison of this work with others, we have used the test sets generated using Mintest ATPG. The proposed AERL compression technique is implemented using C-language and compiled using Dev C++ tool (version 5.3.0.4) on a 64-bit machine with Intel i5-3210 processor operating at 2.50 GHz having 4 GB RAM capacity.

Table 2: Compression ratio of AERL against Mintest test set

Circuit	#SFFs	Original	% of X-bits	AERL	(α) in %
s5378	214	23754	72.62	11822	50.23
s9234	247	39273	73.01	21029	**46.45**
s13207	700	165200	93.15	29648	**82.05**
s15850	611	76986	83.56	23962	**68.87**
s38417	1664	164736	68.08	61814	**62.48**
s38584	1464	199104	82.28	71798	**63.94**

Table 2 shows the compression achieved using AERL compression on Mintest test cubes. Column 1 is the name of the ISCAS'89 benchmark circuit and column 2 and 3 provide the number of scan flip-flops in the circuit and Mintest ATPG generated original test volume (TD) respectively. Percentage of X-bits in the test cube for each circuit is provided in column 4. Columns 5 and 6 are the compressed test volume (TE) and compression ratio (α) obtained from AERL methodology.

The compression ratio is computed using the Eq. (1).

$$\text{Compression ratio}(\alpha \text{ in } \%) = \frac{|T_D| - |T_E|}{|T_D|} \times 100 \qquad (1)$$

The comparison of compression results obtained from AERL scheme with various coding based compression methods like Golomb, FDR EFDR, ALT-FDR, and ERLC are presented in Table 3. Columns 3 through 7 describe the compressed test data volume (TE)) obtained for Golomb, FDR EFDR, ALT-FDR and ERLC respectively. The last column presents the compression obtained from AERL method. Table 3 clearly indicates that the AERL technique provides a compression ratio of up to 82.05% and also better reduction in test data volume as compared to other methodologies.

The peak and average power transitions during scan-in test mode are presented in Table 4. The weighted transition metric (WTM) is used to compute the average and peak-power in scan-in test applications. The WTM of the given test vector is computed using equation 2.

$$\text{WTM} = \sum_{j=1}^{N-1} (S_j \oplus S_{j+1}) \times j \qquad (2)$$

Where *N* is the number of scan cells in the scan chain, *Sj* denotes the logic value ('0' or '1') of *j*th scan-cell in the test vector.

Table 3: Comparison: Compression ratio of various techniques with AERL coding

Circuit	T_D (bits)	FDR [19]	EFDR [21]	ALT-FDR [20]	ERLC [22]	AERL
s5378	23754	12346	**11419**	11694	12389	11822
s9234	39273	22152	21250	21612	22210	**21029**
s13207	165200	30880	29992	32648	32044	**29648**
s15850	76986	26000	24643	26306	25844	**23962**
s38417	164736	93466	64962	64976	67990	**61814**
s38584	199104	77812	73853	77372	76473	**71798**
Avg.	111509	43776	37687	39101	39492	**36679**

Table 4: Comparison: Reduction of peak and average power transitions

Circuit	Mintest [24]		AERL		% of reduction	
	P_{peak}	P_{Avg}	P_{Peak}	P_{Avg}	P_{Peak}	P_{Avg}
S5378	13423	11081	9537	2526	28.95	77.20
s9234	17494	14630	12087	3665	30.91	74.94
s13207	135607	122031	94893	8299	30.02	93.20
s15850	100228	90899	63536	14239	36.61	84.34
s38417	683765	601840	411564	119156	39.81	80.20
s38584	572618	535875	480792	90368	16.04	83.14
Avg.	253856	229393	178735	39709	29.59	82.69

In Table 4, columns 2 and 3 are the peak and average power transitions for the Mintest ATPG generated test vectors. Columns 4 and 5 are the peak and average power transitions obtained from proposed AERL technique. Column 6 and 7 are the percentage of reduction of peak and average power for the proposed approach against [24]. Columns 8 and 9 are the peak and average power obtained with a result of AERL filling. The AERL scheme saves up to 39.81% and 93.20% of peak and average power transitions, and overall 29.59% and 82.69% of less peak-power and average-power transitions in respectively compared to the actual power transitions obtained from Mintest test set.

The decompression architecture used for on-chip decompression is modeled using Verilog hardware description language (HDL) and simulated using NCsim from Cadence for functional verification. The design is synthesized using RTL compiler from Cadence with TSMC180nm CMOS standard cell library. The synthesis result of a finite state machine used in the decompression architecture takes only 43 logic cells with a chip area of 675 nm^2. Also, its total power consumption is 37719.14nW. This shows the proposed compression technique can be implemented for industrial circuits with almost negligible area overhead and power dissipation.

5. Conclusion

An efficient test data compression technique which employs on run-length based coding to reduce the test data volume and test power consumption is presented. A new compression algorithm replacing redundant equal-run-length with a shorter codeword and a new X-filling technique presented in this paper increases the number of equal-run-lengths. Experimental results show that the proposed AERL coding technique reduces the test data volume without increasing the peak and average scan-in test power. A decompression architecture presented in this paper occupies little silicon area in hardware but the advantage obtained is significantly higher. This compression technique can also be applied to any industrial circuits and System-on-a-Chips (SoCs) which contain much intellectual property (IP) cores since this method does not require the internal structure of CUTs.

References

[1] P. Girard, X. Wen & N. Touba, "Low power testing, in System On Chip Test Architectures", Morgan Kaufmann, (2008).

[2] R. Sankaralingam, R. R. Oruganti & N. A. Touba, "Static compaction techniques to control scan vector power dissipation," *Proceedings of the IEEE VLSI Test Symposium*, (2000), pp. 35–40.

[3] A. El-Maleh, S. Khursheed & S. Sait, (2006), "Efficient static compaction techniques for sequential circuits based on reverse-order restoration and test relaxation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2556–2564.

[4] H. Kim, S. Kang & M. S. Hsiao, (2008) "A new scan architecture for both low power testing and test volume compression under soc test environment," *Journal of Electronic Testing: Theory and Applications (JETTA),* vol. 24, no. 4, pp. 365–378.

[5] Z. Wang, H. Fang, K. Chakrabarty & M. Bienek, (2009), "Deviation-based lfsr reseeding for test-data compression," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 2, pp. 259–271.

[6] W. Lien, K. Lee & T. Hsieh, "A test-per-clock lfsr reseeding algorithm for concurrent reduction on test sequence length and test data volume," in *Proceedings of the Asian Test Symposium*, (2012), pp. 278–283.

[7] S. Ward, C. Schattauer & N. Touba, "Using statistical transformations to improve compression for linear decompressors," in *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, (2005), pp. 42–50.

[8] Sivanantham, S., Gopakumar, G., Pandey, A., & Paikada, M. J. (2013), 'Adaptive test clock scheme for low transition LFSR and external scan based testing," *2013 International Conference on Computer Communication and Informatics,*

[9] A. B. Kinsman & N. Nicolici, (2010) "Time-multiplexed compressed test of soc designs," *IEEE Tranactions Very Large Scale Integration, (VLSI) System*, vol. 18, no. 8, pp. 1159–1172.

[10] D. Xiang, Y. Zhao, K. Chakrabarty & H. Fujiwara, (2008), "A reconfigurable scan architecture with weighted scan-enable signals for deterministic bist," IEEE *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 6, pp. 999–1012.

[11] J. M. Solana, (2009), "Reducing test application time, test data volume and test power through virtual chain partition," Integration, the VLSI Journal, vol. 42, no. 3, pp. 385–399.

[12] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, (2004), "Embedded deterministic test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 776–792.

[13] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A smartbist variant with guaranteed encoding," in *Proceedings of the Asian Test Symposium*, (2001)pp. 325–330.

[14] S. Sivanantham, M. Padmavathy, S. Divyanga & P. V. Anitha Lincy, "System-on-a-chip test data compression and decompression with reconfigurable serial multiplier," International Journal of Engineering and Technology, vol. 5, no. 2, pp. 973–978, 2013.

[15] L. Li & K. Chakrabarty, (2004), "Test set embedding for deterministic bist using a reconfigurable interconnection network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, no. 9, pp. 1289–1305.

[16] A. Jas, G.-D. Jayabrata & N. A. Touba, "Scan vector compression/ decompression using statistical coding," in *Proc IEEE VLSI Test Symp.* , (1999), pp. 114–120.

[17] H. Ichihara, Y. Iwamoto, Y. Yoshikawa & T. Inoue, "Test compression based on lossy image encoding," in *Proceedings of the Asian Test Symposium*, (2011), pp. 273–278.

[18]] A. Chandra & K. Chakrabarty, (2002) "Test data compression and decompression based on internal scan chains and golomb coding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 6, pp. 715–722.

[19] A. Chandra & K. Chakrabarty, (2003), "Test data compression and test resource partitioning for system on-a-chip using frequency-directed run-length (fdr) codes," *IEEE Transaction on Computers*, vol. 52, no. 8, pp. 1076 – 1088.

[20] A. Chandra & K. Chakrabarty, (2003), "A unified approach to reduce soc test data volume, scan power and testing time," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 22, no. 3, pp. 352–362.

[21] A. H. El-Maleh, (2008), "Test data compression for system-on-a-chip using extended frequency-directed run-length code," *IET Computers and Digital Techniques,* vol. 2, no. 3, pp. 155–163.

[22] W. Zhan & A. El-Maleh, (2012),"A new scheme of test data compression based on equal-run-length coding(erlc)," *Integration, the VLSI Journal*, vol. 45, no. 1, pp. 91–98.

[23] P. Rosinger, P. Gonciari, B. Al-Hashimi & N. Nicolici, "Simultaneous reduction in volume of test data and power dissipation for systems-on-a-chip," Electronics Letters, vol. 37, no. 24, pp. 1434–1436, 2001.

[24] B. Ye, Q. Zhao, D. Zhou, X. Wang & M. Luo, (2011),"Test data compression using alternating variable run-length code," *Integration, the VLSI Journal,* vol. 44, no. 2, pp. 103–110.

[25] A. Chandra & K. Chakrabarty, "Combining low-power scan testing and test data compression for system-on-a-chip," in *Proceedings of Design Automation Conference,* (2001), pp. 166 – 169.

[26] J. Feng and G. Li, "A test data compression method for system-on-a-chip," in Proceedings - 4th *IEEE International Symposium on Electronic Design, Test and Applications,* (2008), pp. 270–273.

[27] L. T. Wang, C. W. Wu & X. Wen, VLSI Test Principles and Architectures: Design for Testability, 2006.

[28] A. Jas, J. Ghosh-Dastidar, M. Ng & N. Touba, (2003) "An efficient test vector compression scheme using selective huffman coding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 797–806.

[29] S. Reda & A. Orailoglu, "Reducing test application time through test data mutation encoding," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, (2002), pp. 387–393.

[30] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in *IEEE International Test Conference*, (1998), pp. 458–464.

[31] A. Chandra and K. Chakrabarty, (2001) ,"System-on-a-chip test-data compression and decompression architectures based on golomb codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 355–368.

[32] A. El-Maleh and R. Al-Abaji, "Extended frequency-directed run-lengthcode with improved application to system-on-a-chip test data compression," in *9th International Conference on Electronics, Circuits and Systems,* (2002), vol. 2, pp. 449 – 452.

[33] S. Sivanantham, M. Padmavathy, G. Gopakumar, P. S. Mallick, and J. R. P. Perinbam, (2014), "Enhancement of test data compression with multistage encoding," *Integration, the VLSI Journal,* Vol.47 No.4, pp. 499-509.

[34] S. Sivanantham, P. S. Mallick, and J. Raja Paul Perinbam, (2014), "Low power selective pattern compression for scan-based test applications," *Computers and Electrical Engineering,* Vol.40, No.4, pp. 1053-1063.

[35] S. Sivanantham, J. Manuel, K. Sarathkumar, P. S. Mallick, and J. R. P. Perinbam, "Reduction of test power and test data volume by power aware compression scheme," in International Conference on Advances in Computing and Communications, (2012), pp. 158–161.

[36] A. H. El-Maleh, "Efficient test compression technique based on block merging," IET Computers and Digital Techchiques, vol. 2, no. 5, pp. 327–335, 2008.

[37] Thilagavathi, K., Sivanantham, S. (2018), "Two-stage low power test data compression for digital VLSI circuits" *Computers and Electrical Engineering*, 71, pp. 309-320.