A Unified Approach to Reduce SOC Test Data Volume, Scan Power and Testing Time

Anshuman Chandra and Krishnendu Chakrabarty

***Abstract*—We present a test resource partitioning (TRP) technique that simultaneously reduces test data volume, test application time, and scan power. The proposed approach is based on the use of alternating run-length codes for test data compression. We present a formal analysis of the amount of data compression obtained using alternating run-length codes. We show that a careful mapping of the don't-cares in precomputed test sets to 1's and 0's leads to significant savings in peak and average power, without requiring either a slower scan clock or blocking logic in the scan cells. We present a rigorous analysis to show that the proposed TRP technique reduces testing time compared to a conventional scan-based scheme. We also improve upon prior work on run-length coding by showing that test sets that minimize switching activity during scan shifting can be more efficiently compressed using alternating run-length codes. Experimental results for the larger ISCAS89 benchmarks and an IBM production circuit show that reduced test data volume, test application time, and low power-scan testing can indeed be achieved in all cases.**

***Index Terms*—Alternating run-length code, embedded core testing, scan testing, switching activity, system-on-a-chip test, test data compression, test resource partitioning.**

I. Introduction

Intellectual property (IP) cores are now commonly used in large system-on-a-chip (SOC) designs. Although IP cores help reduce design cycle time, they pose several difficult test challenges. The precomputed test patterns provided by the core vendors must be applied to each core without exceeding the power constraints of the SOC. The system integrator is confronted with the problems of test data volume, test application time, and power consumption during test. New techniques based on test resource partitioning (TRP) that reduce test data volume, testing time, and power during testing are, therefore, necessary to facilitate plug-and-play SOC test automation.

It is well known that power consumption in test mode is considerably higher than during normal mode [1]. Therefore, special care must be taken to ensure that the power rating of the SOC is not exceeded during test application. Test data volume and test application time are two additional problems faced in SOC test integration. As SOCs grow in size and complexity, the volume of test data and the test application time are also increasing rapidly. Current techniques do not provide a unified solution to the three problems of test data volume, test application time, and test power. These techniques typically provide point solutions that target at most two out of the above three objectives. For example, a number of techniques target only test data volume and test application time.

Structural methods for reducing test data volume and testing time typically require design modifications. For example, the Illinois scan architecture (ILS) presented in [2] and [3] offers an alternative to conventional scan design. In the ILS architecture, a single scan-in pin is used to simultaneously feed the multiple scan chains of cores during broadcast mode. However, a drawback of the above method is that it can lead to higher power dissipation in test mode; the ILS architecture does not address the problem of reducing power consumption during scan testing.

In a different approach, which can be described as an algorithmic strategy, the precomputed test set T_{D} provided by the core vendor is compressed (encoded) to a much smaller test set T_{E} and stored in automatic test equipment (ATE) memory. An on-chip decoder is used for pattern decompression to generate T_{D} from T_{E} during pattern application [4]–[8]. These techniques are typically based on statistical codes, run-length codes, and their variants, e.g., Golomb and frequency-directed run-length (FDR) codes. These codes can also form the basis for TRP [4]. A particularly attractive feature of TRP based on compression methods is that it does not require any redesign of IP cores.

Test data volume reduction techniques based on on-chip pattern decompression are also presented in [9]–[13]. The compression scheme presented in [11] utilizes a linear mapping network to drive a large number of internal scan chains through a small number of external pins. The RESPIN method proposed in [9] uses the scan chains of one embedded core to decode test patterns for another core or interconnection. The technique based on geometric shapes can be used for compressing test vectors if an embedded processor is available for pattern decompression [10]. The method presented in [12] achieves compression by filling don't-care bits in the test vectors such that these bits are not stored on ATE and are not transferred to the chip if decompression is done on-chip. Another method based on tester-based stimulus and response compression (OPMISR) has been shown to be very effective for testing large scan-based circuits with limited input–outputs (I/Os) [13].

Another way to reduce test data volume and testing time is to use built-in self-test (BIST) [14]. However, BIST can only be applied to SOCs if the IP cores in them are BIST-ready. BIST also imposes an area penalty and it often requires additional design changes. Since most currently available IP cores are not BIST-ready, the incorporation of BIST in them requires considerable redesign.

A number of techniques to control power consumption in test mode have also been presented in the literature. These can be broadly classified as a) structural, b) algorithmic, and c) tester based.

1) *Structural methods*: These methods, which do not address test data volume or testing time, are based on the following design techniques.
   - Gated scan chains: These refer to schemes that use gating techniques to clock portions of the scan chain during scan operation [15]–[17]. In [15], shift registers are used to gate portions of the scan chain during shifting while counters are used for gating in [16]. A decoder/multiplexer-based architecture for gating scan chains has also been proposed in [17].
   - Modified test pattern generator (TPG): Test generation circuits can be tailored to yield low-power vectors without significantly affecting the fault coverage and testing time [18], [19]. The method presented in [18] is based on gated clock scheme for the TPG. The TPG is divided into two groups of flip-flops and each group is activated by a clock running at half the speed of the normal clock. Another TPG based on cellular automata is presented in [19].
   - Modified scan latch and vector inhibition: Scan power can also be reduced by modifying the scan cell and adding gating logic to mask the scan path activity during shifting [20]. This approach coupled with random pattern suppression provides significant power savings during BIST [21]. The vector inhibiting technique presented in [21] provides

Manuscript received May 17, 2002; revised July 8, 2002. This work was supported in part by the National Science Foundation under Grant CCR-9875324, and in part by an Equipment Grant from Intel Corporation. An abridged version of this paper appeared in *Proc. IEEE/ACM Design Automation Conf.*, New Orleans, LA, June 2002, pp. 673–678. This paper was recommended by Associate Editor N. K. Jha.

The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: krish@ee.duke.edu).

Digital Object Identifier 10.1109/TCAD.2002.807895

a hardware solution to the power minimization problem and is shown to significantly decrease power consumption during BIST sessions. The method decreases the switching activity in the internal nodes of the circuit under test during scan-in and scan-out by holding the output of the scan cell to a fixed value.

- Scan chain organization: The switching activity in the scan chain can be reduced by shortening and reorganizing the scan chains. The scan array solution presented in [22] reduces power dissipation by using two-dimensional scan arrays, which reduce switching activity and allow the use of a slower scan clock.

2) *Algorithmic methods*: These include automatic test pattern generation (ATPG) under power constraints, techniques based on test data compression, and test scheduling algorithms.
   - ATPG techniques: ATPG techniques for generating vectors that lead to low power testing are described in [23] and [26]. However, while these techniques provide reduction in power consumption, they do not lead to any appreciable decrease in test data volume.
   - Test data compression: Test generation for low-power scan testing usually leads to an increase in the number of test vectors [23]. On the other hand, static compaction of scan vectors causes significant increase in power consumption during testing [26]. While compacted vectors are useless if they exceed power constraints, uncompacted vectors cannot be used as they require excessive tester memory. Power minimization based on test data compression was first presented in [34].
   - Test scheduling: Test scheduling techniques for system integration attempt to reduce testing time by applying scan/BIST vectors to several cores simultaneously [27]–[32]. Test scheduling is typically carried out under power constraints since multiple cores are tested in parallel.
3) *Tester frequency*: Reduction in power dissipation can be archieved by running the tester at a slower frequency. Although this method offers the simplest way to reduce power consumption, it leads to unacceptable testing times and is, therefore, impractical.

We note that structural methods for reducing test power in SOCs require modification to the embedded cores, e.g., via scan latch reordering (SLR) [24], scan chain, and scan cell redesign. This is usually not feasible for IP cores. ATPG techniques are also infeasible for IP cores since they require gate-level structural models [25]. Moreover, ATPG techniques that address test power do not directly consider test data volume and testing time issues. We, therefore, focus on test data compression for reducing test power, test data volume, and testing time simultaneously.

It was shown in [34] that test data volume and test power can be reduced simultaneously using Golomb coding. The key idea is to map the don't-cares in the test vectors to zero. This results in long runs of zeros that can be efficiently compressed using Golomb code [6]. The resulting fully specified test set also reduces switching activity during scan shifting. Hence, significant reduction in scan power is accompanied with test data compression. However, the switching activity can be reduced further if a more efficient procedure is used to map the don't-cares to binary values. In particular, further savings in power can be achieved if we map the don't-cares to derive test sets that minimize switching activity. Unfortunately, these test sets are not amenable for compression by run-length codes.

A recent test data compression approach offers an elegant solution to the problem of simultaneous reduction of test data volume and scan power [35]. The key idea in this work is to use a minimum transition count (MTC) mapping of don't-cares in the test set and a variant of Golomb coding that can effectively handle runs of both 0's and 1's.

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

Fig. 1. Example of FDR coding.

In this paper, we present a new class of codes, called alternating run-length codes, for test data compression. These codes are particularly effective for compressing test sets that lead to minimum switching activity. They also reduce testing time due to the reduction in the amount of test data that needs to be transported from the tester to the SOC. We, therefore, demonstrate that we can reduce test data volume, test application time, and test power simultaneously by first using MTC mapping and then using alternating run-length codes for compressing the resulting runs of 0's and 1's. Compared to [35], we achieve the same reduction in scan power but significantly greater test data compression.

The organization of the paper is as follows. In Section II, we first review FDR coding. We then present the alternating run-length code, describe the data compression procedure and the decompression architecture, describe the power estimation model, and highlight the key differences from [34]. In Section III, we present a rigorous testing time analysis for TRP based on alternating run-length codes. In Section IV, we present experimental results for the large ISCAS89 benchmark circuits as well as for a real-life microprocessor circuit from IBM. Finally, in order to explain the experimental results, we present in Section V a formal analysis of the alternating run-length code for a memoryless binary data source and for deterministic sequences.

II. Alternating Run-Length Code

In this section, we present the alternating run-length code, and describe the compression procedure and decompression architecture. We also present the power estimation model used to estimate the power dissipation during scan testing.

We first review FDR coding and its application to test data compression [33]. The FDR code is a data compression code that maps variable-length runs of 0's to variable-length codewords. The encoding procedure is illustrated in Fig. 1. As an example, consider a run of six 0's (0000001) in the input stream. This run belongs to group A_3 and it is mapped to the codeword 110000. The reader is referred to [33] for a detailed discussion and motivation for the FDR code.

It was shown in [33] that the FDR code is very efficient for compressing data that has few 1's and long runs of 0's. However, for data streams that are composed of both runs of 0's and runs of 1's, the FDR code is rather inefficient. In fact, in our initial experiments, the sizes of encoded test sets obtained for such test sets were larger than the sizes of uncompressed test sets. This provides the motivation to develop a code that can efficiently compress both runs of 0's and 1's.

Group	a=0 Run-length of 0s	a=1 Run-length of 1s	Group prefix	Tail	Codeword
	0	0	0	0	00
A_1	1	1		1	01
	2	2		00	1000
A_2	3	3	10	01	1001
	4	4		10	1010
	5	5		11	1011
	6	6		000	110000
	7	7		001	110001
	8	8		010	110010
A_3	9	9	110	011	110011
	10	10		100	110100
	11	11		101	110101
	12	12		110	110110
	13	13		111	110111
...	...	...	...	...	...

Fig. 2. Example of the alternating run-length code.

Input data stream: 000000111111000001 (18-bits)
FDR encoded data: 1100000000000000011010 (22-bits)
Alternating run-length encoded data: 11000010111010 (14-bits)
$|a=0|a=1|a=0|$

Fig. 3. Example to compare FDR coding with alternating run-length coding.

Fig. 2 illustrates the encoding procedure for the new alternating run-length code. The alternating run-length code is also a variable-to-variable-length code and consists of two parts—group prefix and tail. The prefix identifies the group in which the run-length lies and the tail identifies the member within the group. An additional parameter associated with this code is the alternating binary variable a. The encoding produced by the alternating run-length code for a given run-length depends on the value of a. If $a = 0$, the run-length is treated as a run of 0's. On the other hand, if $a = 1$, the run-length is treated as a run of 1's. Note that the value of a for the different runs are not added to the encoded data stream.

Fig. 3 shows the encoded data obtained using the two codes for a data stream composed of interleaved runs of 0's and 1's. We observe that the size of the FDR-encoded data set (22 bits) is larger than the size of the input data set (18 bits); hence, the FDR code provides no compression for this case. On the other hand, the size of the alternating run-length-encoded data set (14 bits) is smaller than the size of the input data set. Therefore, we are able to achieve compression with the new code. We also note that $a = 0$ is used for compressing a runs of 0's, and $a = 1$ is used for compressing a runs of 1's, and $a = 0$ is then used for compressing the next run of 0's. Hence, a is inverted after each run is encoded and it keeps alternating between 0 and 1 thereafter. In this paper, we assume a default initial value of $a = 0$, i.e., we assume that the input data stream starts with a run of 0's.

A. Decompression Architecture

An on-chip decoder decompresses the encoded test set T_{E} and produces T_{D}. Even though T_{D} contains more patterns than test sets obtained after static compaction of ATPG vectors, the testing time is reduced since pattern decompression can be carried out on-chip at higher clock frequencies. Let $r_{\max}$ be the longest run of 0's in T_{D} and let $k = \lceil \log_2 r_{\max} \rceil$. As discussed in [33], the FDR decoder can be efficiently implemented by a k-bit counter, a $\log_2\ k$-bit counter and a finite-state machine (FSM), and it is independent of the precomputed test set and the circuit under test. The decoder for alternating run-length code can be implemented by making a small modification to the FDR decoder. The block diagram of the alternating run-length decoder is shown in Fig. 4. An additional toggle flip-flop and an exclusive-OR gate are required to switch between $a = 0$ and $a = 1$. For any circuit whose test set is compressed using alternating run-length code, the given logic is the only additional hardware required other than the two small counters.

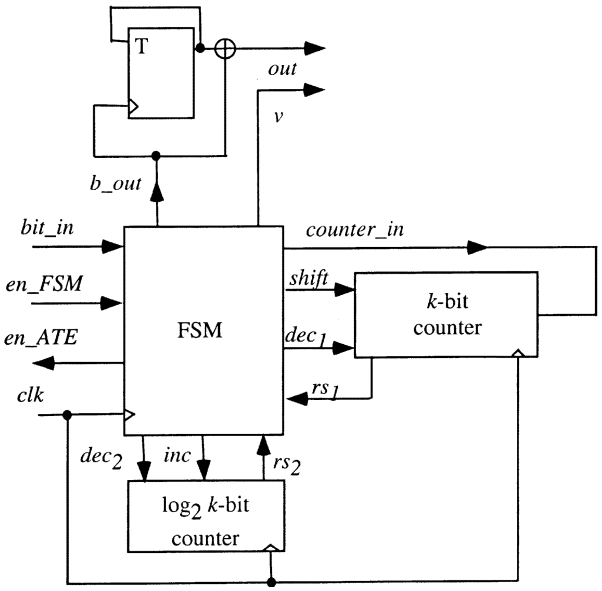

Fig. 4. Decoder block diagram for alternating run-length code.

The *bit_in* is the input to the FSM and an enable (*en*) signal is used to input encoded data when the decoder is ready. The FSM output *counter_in* is used to shift in the prefix or the tail into the k-bit counter and the signals *shift*, dec_1, and rs_1 are used to shift the data in, to decrement, and to indicate the reset state of the counter, respectively. The second counter of $\log_2\ k$-bits is used to count the length of the prefix and the tail so as to identify the group. The signals *inc* and dec_2 are used to increment and decrement the counter, respectively, and rs_2 indicates that the counter has finished counting. Finally, the signal *out* is the decoder output and v indicates when the output is valid. The operation of the decoder is as follows.

- The FSM feeds the k-bit counter with the prefix. The end of the prefix is identified by the separator zero. The *en*, *shift*, and *inc* signals are high till the 0 is received.
- The FSM outputs 0's, decrements the k-bit counter, and makes the signal dec_1 high. It continues to output 0's until rs_1 goes high. The signal v is used to indicate a valid output.
- The tail part is shifted in until the $\log_2 k$-bit counter resets to zero. The dec_2 signal then goes high, the counter is decremented, and the signal rs_2 indicates when it is in the zero state.
- The FSM output 0's corresponding to the tail followed by a zero at the end of tail decoding.

The state diagram for the FSM used for pattern decompression is shown in Fig. 5. We note that the state diagram consists of only nine states. We synthesized the FSM using Synopsys Design Compiler. The synthesized circuit contains only four flip-flops and 38 gates. Therefore, the additional hardware needed for the decoder is very small, and existing counters on the SOC can be reused for decompression. The decoder area is, therefore, comparable to the area of the decoder in [35].

Since the decoder needs to communicate with the tester, and both the codewords and the decompressed data can be of variable length, proper synchronization must be ensured through careful design. In particular, the decoder must communicate with the tester to signal the end of a block of variable-length decompressed data. These and other related decompression issues are discussed in detail in [33].

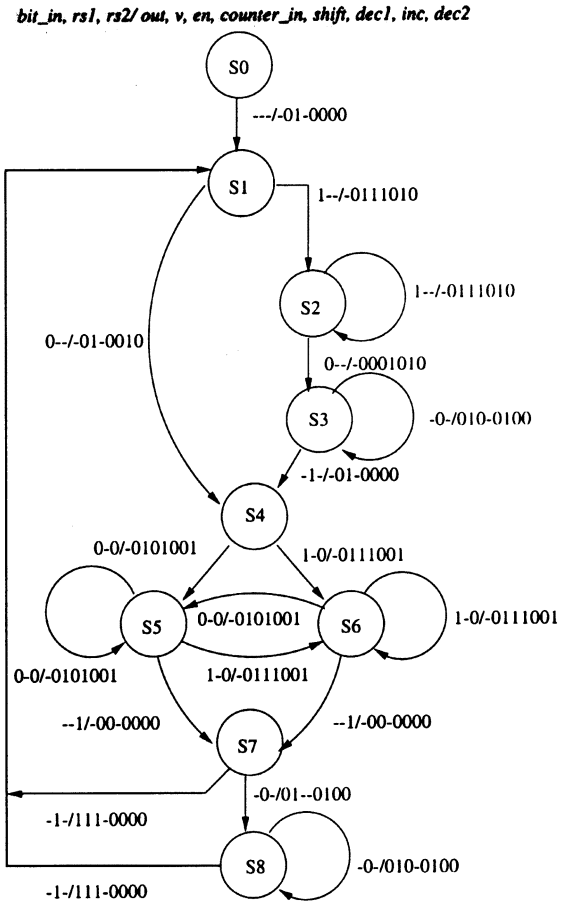

Fig. 5. State diagram of the FSM used for on-chip pattern decompression.

B. Power Estimation

We now examine the impact of test set encoding on power consumption during scan testing. We then show how power consumption can be minimized by appropriately assigning binary values to the don't-care bits in $T_{\rm D}$ and then applying the alternating run-length code for test data compression.

We use the weighted transitions metric (WTM) introduced in [26] to estimate the power consumption due to scan vectors. The WTM models the fact that the scan power for a given vector depends not only on the number of transitions in it but also on their relative positions. For example, consider a scan vector $v_1v_2v_3v_4v_5 = 01000$, where v_1 is first loaded into the scan chain. The 0-to-1 transition between v_1 and v_2 causes more switching activity in the scan chain than the 1-to-0 transition between v_2 and v_3. We use the same model to estimate the power consumption during the scan-in and scan-out operations.

The weighted transitions count metric is also strongly correlated to the switching activity in the internal nodes of the core under test during the scan in operation. It was shown experimentally in [26] that scan vectors with higher weighted transition metric dissipate more power in the core under test. In addition, experimental results for an industrial ASIC [17] confirm that the transition power in the scan chains is the dominant contributor to test power.

Consider a scan chain of length l and a scan vector $t_j = t^\star_{j,1}t^\star_{j,2}\ldots t^\star_{j,l}$, with $t^\star_{j,1}$ scanned in before $t^\star_{j,2}$, and so on. As shown in [34], the WTM for t_j, denoted WTM_j, is given by $\mathrm{WTM}_j = \sum_{i=1}^{l-1}(l-i)\cdot(t^\star_{j,i} \oplus t^\star_{j,i+1})$. If the test set $T_{\rm D}$ contains n vectors $t_1, t_2, \ldots, t_n$ then the average scan in power $P_{\rm avg}$ and peak scan in power $P_{\rm peak}$ are estimated as follows:

$$P_{\rm avg} = \frac{\sum_{j=1}^{n}\sum_{i=1}^{l-1}(l-i)\cdot\left(t^\star_{j,i} \oplus t^\star_{j,t+1}\right)}{n}$$

$$P_{\rm peak} = \max_{j\in\{1,2,\ldots,n\}}\left\{\sum_{i=1}^{l-1}(l-i)\cdot\left(t^\star_{j,i} \oplus t^\star_{j,i+1}\right)\right\}.$$

If the peak power exceeds a threshold value, it can cause structural damage to the silicon or to the package. Likewise, elevated average power can also cause structural damage to the silicon, bonding wires or the package. It also adds to the thermal load that must be transported away from the device under test.

It was shown in [34] that Golomb coding can be used to simultaneously reduce the volume of test data, $P_{\rm avg}$ and $P_{\rm peak}$. The don't-care bits were mapped to 0's and the resulting test data was compressed using the Golomb code. While this approach provides significant reductions in power consumption, and at the same time, decreases the test data volume considerably, it does not minimize either $P_{\rm avg}$ or $P_{\rm peak}$.

Here, we improve upon [34] by using the alternating run-length code. Table I shows a partially specified scan vector $t_i = 01XXX10XXX01$ with scan chain length $l = 12$, where X denotes a don't-care bit. If the don't-cares are mapped to appropriate binary values to minimize the weighted transition metric, then a sequence $dXXXXd'$, $d \in \{0,1\}$ must be mapped to $dddddd'$. Similarly, a sequence $dXXXX$ must be mapped to $ddddd$. This ensures that the few unavoidable transitions occur "late" during scan-in. This mapping of don't-cares is identical to the MTC mapping used in [35].

Table I shows the impact of don't-care mapping on the WTM_i for a given test vector t_i. The compression obtained using the FDR code and the alternating run-length code are also shown. Note that the Golomb code was used for compression in [34]. Since then, the FDR code has been shown to be more efficient than the Golomb code, hence, we consider only the FDR code here. The WTM value is clearly higher if the don't-cares are always mapped to zero. However, FDR coding is much more effective in reducing test data volume if this strategy is used. On the other hand, while FDR coding is ineffective for the fully specified test vector which minimizes WTM, the alternating run-length code provides the same compression as achieved with the FDR code with all don't-cares mapped to 0's. The above example demonstrates that a careful mapping of the don't-cares to 0's and 1's, followed by alternating run-length coding of the resulting test data, not only provides reduction in test data volume, but also minimizes the scan power dissipation.

III. Testing-Time Analysis

We now analyze the testing time when a single scan chain is fed by the alternating run-length decoder. Test data compression decreases testing time and allows the use of a low-cost ATE running at a lower frequency to test the core without imposing any penalties on the total testing time. Let the ATE frequency and the on-chip scan frequency be $f_{\rm ATE}$ and $f_{\rm scan}$, respectively, where $f_{\rm ATE} < f_{\rm scan}$. Since the ATE and the scan chain operate at two different frequencies, the decoder also consists of two parts—one operating at $f_{\rm ATE}$ and the other operating at $f_{\rm scan}$ such that $f_{\rm ATE} = f_{\rm scan}/\alpha$, $\alpha > 1$. The parameter α should ideally be a power of two since it is easier to synchronize the ATE clock with the scan clock for such values of α [37]. If the scan chain has multiple segments operating at different clock frequencies, each segment has a dedicated decoder for test data decompression.

Fig. 6 outlines the decoder partitioned into two frequency domains. The proposed TRP scheme, therefore, decouples the internal scan chain(s) from the ATE via the use of a decoder interface. This decoupling implies that the scan clock frequency is no longer constrained by the ATE clock frequency limitation. Thus, $f_{\rm scan}$ can now be made larger than $f_{\rm ATE}$.

TABLE I
MAPPING OF DON'T-CARES IN A TEST CUBE TO BINARY VALUES

Partially-specified scan vector (test cube)	Fully-specified vector (Minimum WTM)	Fully-specified vector (Don't-cares mapped to 0s)
$t_i = 01XXX10XXX01$ (12 bits)	011111000001 FDR code length: 14 bits Alternating run-length code length: 10 bits $WTM_i = 18$	010001000001 FDR code length: 10 bits $WTM_i = 25$

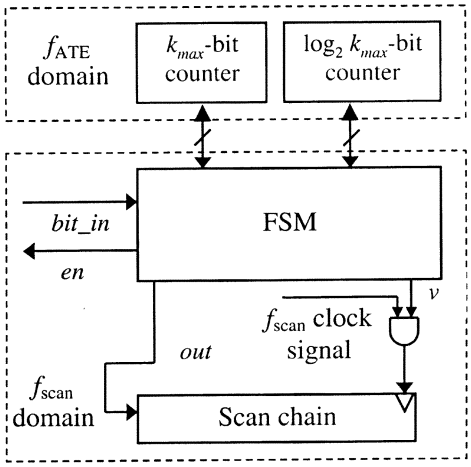

Fig. 6. Block diagram of the alternating run-length decoder partitioned into two frequency domains for TRP.

For the alternating run-length code, let $t(k,i)$ be the total time required to decompress a codeword that is the ith member of the kth group, and $t_{\rm shift}(k,i)$ and $t_{\rm decode}(k,i)$ be the time required to transfer the data from the ATE to the chip and to decode the codeword, respectively. An upper bound on $t(k,i)$ can be obtained by assuming that decoding begins after the complete codeword is transferred from the ATE. This implies that

$$t(k,i) \le t_{\rm shift}(k,i) + t_{\rm decode}(k,i).$$

For the alternating run-length code, the prefix length and the tail length of the codeword belonging to the kth group are each equal to k bits; see Fig. 2. Since data is transferred from the ATE to the chip at the tester frequency, the time required to transfer any codeword of the kth group is given by

$$t_{\rm shift}(k,i) = \frac{2k}{f_{\rm ATE}}.$$

For any codeword, the prefix is identical to the binary representation of the run-length corresponding to the group's first element. As shown in Fig. 2, the number of 0's in the prefix of a codeword belonging to the kth group is equal to $2^k - 2$. The decoder has to output $(2^k - 2)$ 0's before the tail decoding starts. The time $t_{\rm prefix}(k)$ required to decompress the prefix of any codeword from the kth group is, therefore, given by

$$t_{\rm prefix}(k) = \frac{2^k - 2}{f_{\rm scan}}.$$

Similarly, the time $t_{\rm tail}(k,i)$ required to decompress the tail of the ith member of the kth group is equal to the sum of time required to output $(i-1)$ 0's and a single one. Hence

$$t_{\rm tail}(k,i) = \frac{(i-1)+1}{f_{\rm scan}} = \frac{i}{f_{\rm scan}}.$$

The total decoding time $t_{\rm decode}(k,i)$ is given by

$$\begin{aligned} t_{\rm decode}(k,i) &= t_{\rm prefix}(k) + t_{\rm tail}(k,i) \\ &= \frac{2^k - 2}{f_{\rm scan}} + \frac{i}{f_{\rm scan}}. \end{aligned}$$

The total time needed to decompress the codeword is given by

$$\begin{aligned} t(k,i) &\le t_{\rm shift}(k,i) + t_{\rm decode}(k,i) \\ &= \frac{1}{f_{\rm ATE}}\left(2k + \frac{2^k - 2 + i}{\alpha}\right) \end{aligned} \quad (1)$$

where $f_{\rm scan} = \alpha f_{\rm ATE}$.

Let $q(k,1), q(k,2), q(k,3), \ldots, q(k,2^k)$ be the absolute frequencies of the members of the kth group. Therefore, the decompression time $\tau(k)$ for the runs belonging to the kth group is given by

$$\begin{aligned} \tau(k) &= \frac{1}{f_{\rm ATE}} \sum_{i=1}^{2^k} \left(2k + \frac{2^k - 2 + i}{\alpha}\right) q(k,i) \\ &= \frac{1}{f_{\rm ATE}} \left(2k \sum_{i=1}^{2^k} q(k,i) + \frac{1}{\alpha} \sum_{i=1}^{2^k} (2^k - 2 + i) q(k,i)\right). \end{aligned}$$

Let us assume that $k_{\max}$ is the largest group. The test application time $TAT^{\rm A}_{\rm SSC}$ (A in the superscript denotes the alternating run-length code) for the entire test set with a single scan chain (SSC) is given by

$$\begin{aligned} TAT^{\rm A}_{\rm SSC} &\le \sum_{k=1}^{k_{\max}} \tau(k) \\ &= \frac{1}{f_{\rm ATE}} \left(|T_{\rm E}| + \frac{1}{\alpha} \sum_{k=1}^{k_{\max}} \sum_{i=1}^{2^k} (2^k - 2 + i) q(k,i)\right) \end{aligned} \quad (2)$$

where $|T_{\rm E}|$ is the size of the encoded test set. Next, to derive a lower bound on the testing time, suppose the tail bits are shifted in while the prefix is being decompressed. Since the tail bits are now shifted from the ATE while the prefix bits are decoded, the time required to shift any codeword of the kth group is given by

$$t_{\rm shift}(k,i) = \frac{k}{f_{\rm ATE}}.$$

Therefore, a lower bound on decoding time is given by

$$\begin{aligned} t(k,i) &\ge t_{\rm shift}(k,i) + t_{\rm decode}(k,i) \\ &= \frac{1}{f_{\rm ATE}} \left(k + \frac{2^k - 2 + i}{\alpha}\right). \end{aligned}$$

Therefore

$$TAT^{\rm A}_{\rm SSC} \ge \frac{1}{f_{\rm ATE}} \left(\frac{|T_{\rm E}|}{2} + \frac{1}{\alpha} \sum_{k=1}^{k_{\max}} \sum_{i=1}^{2^k} (2^k - 2 + i) q(k,i)\right). \quad (3)$$

We next compare the testing time using the proposed TRP scheme with that for an ATPG-compacted test set with p patterns and an external tester operating at frequency $f^{\star}_{\rm ATE}$. Let the length of the scan chain be n bits. The size of the ATPG-compacted test set is pn bits and

TABLE II
EXPERIMENTAL RESULTS ON TEST DATA COMPRESSION USING FDR CODE AND ALTERNATING RUN-LENGTH CODE

Circuit	No. of bits in T_D	Percentage of don't-care bits in T_D	Size of Mintest test in T_D	FDR coding using T_D (don't-cares mapped to 0s)		Alternating run-length coding using T_D (don't-cares mapped to minimize power)			Results from [35] (Compression, percent)
				Compression (percent)	No. of bits in T_E	Compression (percent)	No. of bits in T_E	Improvement over Mintest (percent)	
s5378	23754	72.62	20758	48.02	12346	50.77	11694	43.66	38.49
s9234	39273	73.01	25935	43.59	22152	44.96	21612	16.66	39.06
s13207	165200	93.15	163100	81.30	30880	80.23	32648	79.98	77
s15850	76986	83.56	57434	66.22	26000	65.83	26306	54.19	59.32
s38417	164736	68.08	113152	43.26	93466	60.55	64976	42.57	55.42
s38584	199104	82.28	161040	60.91	77812	61.13	77372	51.95	56.63
Average	—	78.79	—	57.21	—	60.57	—	48.16	54.32

the test application time $\mathrm{TAT}_{\mathrm{SSC}}^{\mathrm{ATPG}}$ equals $pn/f_{\mathrm{ATE}}^{\star}$. Experimental results presented in Section IV show that the testing time is reduced considerably using the proposed method if $f_{\mathrm{ATE}}^{\star} = f_{\mathrm{ATE}}$. Hence, alternating run-length coding allows us to decrease the volume of test data, test power, and testing time.

To conclude the analysis, we note that the above bounds allow us to evaluate the testing time without a detailed analysis of the asynchronous handshaking protocol between the tester and the decoder. The exact testing time, which lies between the two bounds, can be determined through a bit-by-bit analysis of the encoded test data. The formulation, based on upper and lower bounds, allows us to demonstrate the effectiveness of the proposed TRP scheme without resorting to such detailed analysis.

IV. EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of alternating run-length coding for reducing test data volume, testing time, and power consumption during scan testing. We carried out experiments for the ISCAS89 benchmark circuits and a production circuit from IBM. The experiments were conducted on a Sun Ultra 10 workstation with a 333-MHz processor and 256 MB of memory. We only considered the large ISCAS89 full-scan circuits with a single scan chain each.

Table II presents the experimental results for the ISCAS benchmarks for test sets obtained from the Mintest ATPG program [36]. We compare the compression obtained using the FDR code and the alternating run-length code. In order to compare with [34], we first assigned all don't-cares to 0's and compressed T_{D} using the FDR code. We then carefully mapped the don't-cares to minimize WTM and compressed the resulting T_{D} using the alternating run-length code. Table II shows the sizes of T_{D}, the size of the smallest encoded test set obtained after static compaction using Mintest, the size of compressed test set obtained with all don't-cares mapped to zero ($|T_{\mathrm{E1}}|$) and size of compressed test set obtained with an optimal mapping of don't-cares to minimize WTM ($|T_{\mathrm{E2}}|$). We also compare our compression results with the results published in [35]. Note that we do not include results for s35932 here since the test cubes that are available to us were found to provide incomplete fault coverage.

As is evident from Table II, the alternating run-length code yields better compression than the FDR code for four out of the six benchmark circuits. This is particularly remarkable since, as we show later in this section, high compression with the alternating run-length code is also accompanied with significant reduction in testing time and test power. The results also show that ATPG compaction is not always necessary for saving memory and reducing testing time. In all cases, the size of the encoded test set is less than the smallest ATPG-compacted test sets known for these circuits. This comparison is essential in order to show that storing T_{E} in ATE memory is more efficient than simply applying static compaction to test cubes and storing the resulting compact test sets. On average, the size of T_{E} is 48.16% less than that of the compacted test sets obtained using Mintest.

TABLE III
COMPARISON OF TESTING TIME USING THE PROPOSED TRP METHOD WITH TRADITIONAL SCAN-BASED EXTERNAL TESTING

Circuit	f_{ATE} (MHz)	α	Lower bound on TAT_{SSC}^{A} (ms)	Upper bound on TAT_{SSC}^{A} (ms)	TAT_{SSC}^{ATPG} (ms)
s5378	20	4	0.589	0.881	1.037
		8	0.440	0.733	1.037
		16	0.366	0.658	1.037
s9234	20	4	1.031	1.571	1.296
		8	0.785	1.326	1.296
		16	0.663	1.203	1.296
s13207	20	4	2.881	3.697	8.155
		8	1.848	2.664	8.155
		16	1.332	2.148	8.155
s15850	20	4	1.619	2.277	2.871
		8	1.138	1.796	2.871
		16	0.898	1.555	2.871
s38417	20	4	3.683	5.308	5.657
		8	2.654	4.278	5.657
		16	2.139	3.763	5.657
s38584	20	4	4.423	6.357	8.052
		8	3.178	5.113	8.052
		16	2.556	4.490	8.052

The compression results are also significantly better (over 16% on average) than in [35]. This is not entirely unexpected, since the alternating run-length code, which uses an underlying FDR code, is better tailored than the Golomb code to exploit the properties of test sets. Note that the compression is considerably higher in [35] if SLR is allowed; however, we assume in this work that SLR is not possible for the IP cores in an SOC.

Table III presents test application times for the proposed method using the alternating run-length code and for traditional scan-based testing with $f_{\mathrm{ATE}}^{\star} = f_{\mathrm{ATE}}$. We note that in all the cases the upper bound on test application time using the proposed scheme is lower than that for scan-based external testing. The actual test application time for the proposed TRP scheme lies between the lower and upper bounds. For example, the test application time for s13207 with $\alpha = 8$, and $f_{\mathrm{ATE}}^{\star} = f_{\mathrm{ATE}}$ =20 MHz lies between 1.848 and 2.664 ms, which is lower than the time of 8.155 ms required for conventional scan testing using ATPG-compacted patterns derived using Mintest. Furthermore, let us assume that the desired testing time for s38417 is the average of lower and upper bounds i.e., 2.256 ms. In this case, an external tester operating at $f_{\mathrm{ATE}}^{\star}$ =72.29 MHz will be required, as opposed to a 20–MHz tester with the proposed TRP scheme.

We next present results on the peak and average power consumption during the scan-in operation. These results show that using test data

TABLE IV
IMPACT OF THE MAPPING OF DON'T-CARES TO BINARY VALUES ON SCAN-IN POWER CONSUMPTION

Circuit	Uncompacted test sets with don't-cares mapped to 0s				Uncompacted test sets with don't-cares mapped to minimize WTM			
	Peak power P^F_{peak}	Peak power reduction (percent)	Average power P^F_{avg}	Average power reduction (percent)	Peak power P^A_{peak}	Peak power reduction (percent)	Average power P^A_{avg}	Average power reduction (percent)
s5378	10127	24.55	3336	69.89	9531	28.99	2435	78.02
s9234	12994	25.72	5692	61.09	12060	31.06	3466	76.30
s13207	101127	25.42	12416	89.82	97606	28.02	7703	93.68
s15850	81832	18.35	20742	77.18	63478	36.66	13381	85.27
s38417	505295	26.10	172665	71.31	404617	40.82	112198	81.35
s38584	531321	7.21	136634	74.50	479530	16.25	88298	83.52
Average	—	28.98	—	75.89	—	37.72	—	84.32

TABLE V
EXPERIMENTAL RESULTS ON PEAK AND AVERAGE SCAN-OUT POWER CONSUMPTION

Circuit	Mintest test sets after static compaction		Uncompacted test sets with don't-cares mapped to minimize WTM			
	Peak power P^C_{peak}	Average power P^C_{avg}	Peak power P^A_{peak}	Peak power reduction (percent)	Average power P^A_{avg}	Average power reduction (percent)
s9234	16777	14555	16927	-0.89	12957	10.97
s13207	132129	116695	120992	8.42	97004	16.87
s15850	99647	88385	88445	11.24	67064	24.12
s38417	619929	553491	549950	11.28	311072	43.79
s38584	577365	521882	529315	8.32	437339	16.19
Average	—	—	—	20.25	—	33.17

compression with a careful mapping of don't-cares to 0's and 1's can also lead to significant savings in power consumption. As discussed in Section II, we estimate power using the WTM. Let $P^{\mathrm{C}}_{\mathrm{peak}}$ ($P^{\mathrm{C}}_{\mathrm{avg}}$) be the peak (average) power with compacted test sets obtained using Mintest, and let $P^{\mathrm{F}}_{\mathrm{peak}}$ ($P^{\mathrm{F}}_{\mathrm{avg}}$) be the peak (average) power when the FDR code is applied to T_{D} after mapping the don't-cares to 0's. Similarly, let $P^{\mathrm{A}}_{\mathrm{peak}}$ ($P^{\mathrm{A}}_{\mathrm{avg}}$) be the peak (average) power when the alternating run-length code is applied to T_{D} after mapping the don't-cares to minimize the WTM. Table IV compares the average and peak power consumption when FDR coding and alternating run-length coding are used. The percentage reduction in power was computed as follows:

$$\text{Reduction in peak power} = \frac{P^{\mathrm{C}}_{\mathrm{peak}} - P^{\mathrm{F}}_{\mathrm{peak}}}{P^{\mathrm{C}}_{\mathrm{peak}}} \times 100$$

$$\text{Reduction in average power} = \frac{P^{\mathrm{C}}_{\mathrm{avg}} - P^{\mathrm{F}}_{\mathrm{avg}}}{P^{\mathrm{C}}_{\mathrm{avg}}} \times 100.$$

Table IV shows that the peak power and average power are significantly less if the alternating run-length code is used for test data compression and the decompressed patterns are applied during testing. On average, the peak (average) power is 37.72% (84.32%) less in this case than for the Mintest test sets. (Note that the power values for the Mintest test sets are shown in Table V.) Thus, our results demonstrate that substantial reduction in test data volume and testing time are also accompanied by significant reduction in power consumption during scan testing. Note that the power savings obtained in this paper are identical to that in [35] since the don't-cares are optimally mapped to 0's and 1's in both papers. Additional power reduction was obtained in [35] using SLR, a structural modification that we do not consider in this paper due to the underlying assumption of IP cores.

We next present results on the peak and average power consumption during the scan-out operation. The scan-out power depends, to a large extent, on the responses of the core under test to the test patterns. Since our approach is aimed at minimizing the scan-in power, we conducted a set of experiments to evaluate the impact of scan-in power minimization on the scan-out power. Table V shows that the peak power and average power are significantly less in five out of six cases if alternating run-length coding is used for test data compression. The peak power during scan-out operation was only slightly higher for the s9234 benchmark circuit. On average, the peak (average) power is 20.25% (33.17%) less than for the Mintest test sets. Thus, our results demonstrate that the substantial reduction in test data volume is also accompanied by significant reduction in power consumption during scan testing. The reduction in scan-out power is an important added advantage since we do not directly target scan-out power in our compression scheme.

Table VI presents the experimental results when the FDR code and the alternating run-length code are applied to scan vectors for a production circuit from IBM. This circuit contains 1.2 million gates, 32 200 latches and a total of 30 scan chains. We were provided with four sets of scan vectors for this circuit. Each vector set consists of 32 patterns (a total of 1 031 072 bits of test data per vector set). We find that the compression obtained using the alternating run-length code is comparable to that obtained using the FDR code. The slight (1–2%) decrease in compression is offset by the significant savings in test power as is shown next.

Table VII presents the average and peak power values for the IBM circuit when the FDR code and the new alternating run-length code are applied to the scan vectors. We find that the alternating run-length code is extremely efficient for reducing test power. Compared to the FDR code, as much as 43.36% greater reduction is obtained in peak power and as much as 42.38% greater reduction is obtained in average power.

Finally, we present results on test application time for the IBM circuit when the TRP scheme based on the new alternating run-length codes is applied to the scan vectors. We assume that the circuit consists of a single scan chain. The minimum (maximum) testing time for the four vectors using TRP and a tester with $f_{\mathrm{ATE}} = 50$ MHz is 12.757 ms

TABLE VI
EXPERIMENTAL RESULTS ON TEST DATA COMPRESSION USING THE FDR CODE AND THE ALTERNATING RUN-LENGTH CODE FOR AN IBM CIRCUIT

Scan vector set	No. of bits in T_D	FDR coding using T_D		Alternating run-length coding using T_D	
		Compression (percent)	No. of bits in T_E	Compression (percent)	No. of bits in T_E
1	1031073	95.08	50648	93.90	62862
2	1031073	94.71	54476	93.98	62030
3	1031073	95.02	51286	93.75	64354
4	1031073	95.63	45026	94.60	55596
Average	—	95.11	—	94.05	—

TABLE VII
POWER REDUCTION ASSOCIATED WITH THE COMPRESSION OF IBM TEST DATA

Circuit	Uncompacted test sets with don't-cares mapped to 0s		Uncompacted test sets with don't-cares mapped to minimize WTM			
	Peak power P^F_{peak}	Average power P^F_{avg}	Peak power P^A_{peak}	Peak power reduction (percent)	Average power P^A_{avg}	Average power reduction (percent)
1	9825669	3587758	6113059	37.78	2101363	41.42
2	10964254	3892459	6255619	42.94	2151415	44.72
3	8996102	3611662	5171926	42.50	2187382	39.43
4	7712574	3161122	3836810	50.25	1771358	43.96
Average	—	—	—	43.36	—	42.38

(15.205 ms). On the other hand, the test application time based on traditional scan-based testing using the same external tester is 61.863 ms.

V. ANALYSIS OF ALTERNATING RUN-LENGTH CODE

In this section, we provide analytical insights into the reasons underlying the high compression obtained using alternating run-length codes. We first analyze the effectiveness of the alternating run-length code for a memoryless data source. We then present an analysis for deterministic sequences. Let the probability of producing 0's be p and the probability of producing 1's be $p1 = 1 - p$. The entropy $H(p)$ of this memoryless source is given by the following equation:

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p).$$

From Fig. 2, the smallest and the longest run-lengths that belong to group A_k are given by $(2^k - 2)$ and $(2^{k+1} - 3)$, respectively. Therefore, the probability $P(i,k)$ that an arbitrarily chosen run of length i belongs to group A_k is given by

$$\begin{aligned} P(i,k) &= \sum_{i=2^k-2}^{2^{k+1}-3} \left[p^i(1-p) + p_1^i(1-p_1)\right] \\ &= p^{2^k-2}\left(1-p^{2^k}\right) + p_1^{2^k-2}\left(1-p_1^{2^k}\right). \end{aligned}$$

The codeword in group A_k consists of $2k$ bits. Therefore, the average codeword length $\bar{\mathrm{A}}$ for alternating run-length codes is given by

$$\begin{aligned} \bar{\mathrm{A}} &= \sum_{k=1}^{\infty} 2k\left[p^{2^k-2}\left(1-p^{2^k}\right) + p_1^{2^k-2}\left(1-p_1^{2^k}\right)\right] \\ &= 2\sum_{k=1}^{\infty}\left[p^{2^k-2} + (1-p)^{2^k-2}\right]. \end{aligned}$$

We next determine λ the average number of bits in any run generated by the data source. It can be easily shown that

$$\begin{aligned} \lambda &= 1 + \sum_{i=1}^{\infty} ip^i(1-p) + 1 + \sum_{i=1}^{\infty} ip_1^i(1-p_1) \\ &= \frac{1}{p(1-p)}. \end{aligned}$$

Even though we do not have a closed-form expression for $\bar{\mathrm{A}}$, the above equations can be used to evaluate the effectiveness of alternating run-length codes. The compression gain β_{A} for alternating run-length codes is given by

$$\beta_{\mathrm{A}} = \frac{1}{2p(1-p)\sum_{k=1}^{\infty}\left[p^{2^k-2} + (1-p)^{2^k-2}\right]}.$$

An upper bound on the compression gain is obtained from the entropy $H(p)$ of the source using the following equation:

$$\beta_{\max} = \frac{1}{H(p)}.$$

Fig. 7 shows a comparison between the compression gain β_{F} and β_{A}, where β_{F} is the compression gain corresponding to FDR codes. The upper bound $\beta_{\max}$ is also shown in the figure. The figure shows that compression gain for the FDR code is always lower than that for the alternating run-length code for all values of $p < 0.5$. This is expected because the FDR code is designed only to compress runs of 0's. For $p < 0.5$, the data source has more runs of 1's and since alternating run-length code is designed to compress runs of both 0's and 1's, $\beta_{\mathrm{A}} > \beta_{\mathrm{F}}$ for $p < 0.5$. Fig. 7 also shows that for values of $p > 0.5$, there is a not significant difference between β_{F} and β_{A}. The figure shows how closely the alternating run-length code gain curve follows the upper bound $\beta_{\max}$. Hence, these results show that alternating run-length codes are very efficient for compressing data that is composed of runs of 0's and 1's.

Next, we develop an analysis technique to determine the worst and best case compressions that can be achieved using alternating run-length codes for some generic parameters of precomputed test sets using this technique. Suppose T_{D} contains r_1 runs of 0's, r_2 runs of 1's, and a total of n bits. We first determine $C^{\mathrm{A}}_{\max}$, the number of bits in the encoded test set T_{E} in the worst case, i.e., when the compression is the least effective. In doing so, we also determine the distribution of the runs of 0's and runs of 1's that gives rise to this worst case compression.

Suppose T_{D} contains k_i runs of 0's and j_i runs of 1's of length i with maximum run-length $l_{\max}$. Let the size of the encoded test set T_{E} be A bits, and let $\delta = A - (n - r_1 - r_2)$ measure the amount

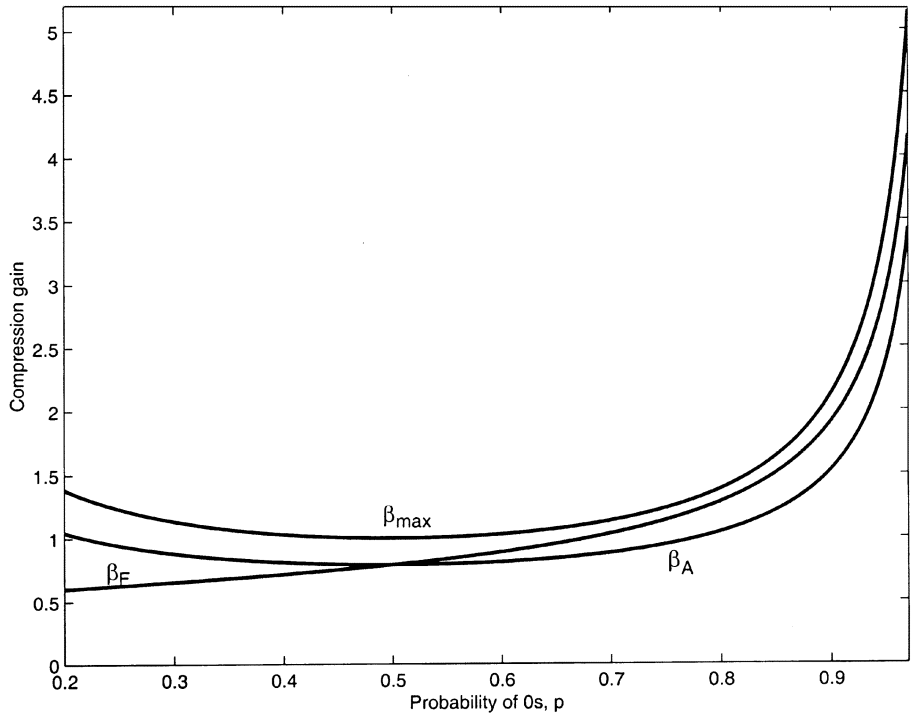

Fig. 7. Comparison of compression gain obtained with FDR codes and alternating run-length codes for $0.2 \leq p \leq 0.97$.

of compression achieved using alternating run-length codes. To make the presentation simpler, we subtract a constant term from A for all distributions of runs, given a fixed n, r_1 and r_2. If the alternating run-length coding procedure of Fig. 2 is applied to $T_{\rm D}$, we have $\delta = 2k_0 + 2j_0 + k_1 + j_1 + 2k_2 + 2j_2 + k_3 + j_3 - k_5 - j_5 - k_7 - j_7 - 2k_8 - 2j_8 - 3k_9 - 3j_9 \pm \cdots (\text{up to } l_{\max})$. This can be explained as follows: run of one of length i, we compare the size of the run-length (i) with the size of the corresponding codeword. For example, the codeword corresponding to a run of length zero contains two bits (one more than the original run), the codeword for run-length one is of the same size as the original run-length, and so on. The difference between these two quantities contributes to δ, and it appears as the coefficient of the appropriate k_i and j_i term in the equation for δ.

We next use the following simple integer linear programming (ILP) model to determine the maximum value of δ. This yields the worst case compression $(C^{\rm A}_{\max})$ using alternating run-length codes.

Maximize:

$$\delta = 2k_0 + 2j_0 + k_1 + j_1 + 2k_2 + 2j_2 + k_3 + j_3 - k_5 - j_5 - k_7 - j_7 - 2k_8 - 2j_8 - 3k_9 - 3j_9 \pm \cdots (\text{up to } l_{\max})$$

subject to:

$$\sum_{i=1}^{l_{\max}} ik_i + \sum_{i=1}^{l_{\max}} ij_i = n - r_1 - r_2$$

$$\sum_{i=1}^{l_{\max}} k_i = r_1$$

$$\sum_{i=1}^{l_{\max}} j_i = r_2.$$

This ILP model can be easily solved, e.g., using a solver such as *lpsolve* [38], to obtain the worst case values for the k_is and j_is. Note that even though $l_{\max}$ appears in the above ILP model, we do not make any explicit use of it. Our goal here is to determine a worst case distribution of the runs of 0's and runs of 1's. Generally, short run lengths yield the worst case compression; however, if $l_{\max}$ must exceed a minimum value to satisfy the constraints listed above. We can use *lpsolve* to determine the minimum $l_{\max}$ by incrementally increasing $l_{\max}$ until the optimization problem becomes feasible.

Table VIII lists the size $C^A_{\max}$ of the encoded data set for worst case compression for various values of n, r_1, and r_2. The last two columns show the distribution of runs of 0's and 1's for which the worst case compression is achieved (a/b indicates a runs of length b). Note that this distribution is not unique since a number of run-lengths can yield the worst case distribution.

Next, we analyze the best case compression achieved using alternating run-length codes for any given n, r_1, and r_2. Since the compression is better for longer run-lengths, we also need to constrain the maximum run-length in this case. As before, we formulate this problem using ILP, and the following model can be solved using *lpsolve* to obtain a best case distribution of runs and $C^A_{\max}$, the number of bits in the encoded test set in the best case.

Minimize:

$$\delta = 2k_0 + 2j_0 + k_1 + j_1 + 2k_2 + 2j_2 + k_3 + j_3 - k_5 - j_5 - k_7 - j_7 - 2k_8 - 2j_8 - 3k_9 - 3j_9 \pm \cdots (\text{up to } l_{\max})$$

subject to:

$$\sum_{i=1}^{l_{\max}} ik_i + \sum_{i=1}^{l_{\max}} ij_i = n - r_1 - r_2$$

$$\sum_{i=1}^{l_{\max}} k_i = r_1$$

$$\sum_{i=1}^{l_{\max}} j_i = r_2.$$

Table IX lists the run-length distributions corresponding to the best case compression using alternating run-length codes. The corresponding percentage compression values are also listed. In Fig. 8, we plot the best case and worst case size of the encoded data as the

TABLE VIII
WORST CASE COMPRESSION USING THE ALTERNATING RUN-LENGTH CODE

n	r_1	r_2	C^A_{max}	Percentage compression $(1 - C^A_{max}/n) \times 100$	Worst-case distribution of runs of 0s	Worst-case distribution of runs of 1s
1000	20	20	320	68.0	9/14, 1/20, 10/28	1/20, 3/22, 16/28
1000	20	40	480	52.0	20/14	32/14, 1/16, 7/28
1000	20	60	590	41.0	20/6	5/6, 55/14
1000	40	20	480	52.0	40/14	12/14, 1/16, 7/28
1000	40	40	590	41.0	25/6, 15/14	40/14
1000	40	60	674	32.6	40/6	19/6, 4/7, 37/14
1000	60	20	590	41.0	25/6, 35/14	20/14
1000	60	40	674	32.6	21/6, 2/7, 37/14	38/6, 2/7
1000	60	60	760	24.0	60/6	40/6, 20/14

TABLE IX
BEST CASE COMPRESSION USING THE ALTERNATING RUN-LENGTH CODE

n	r_1	r_2	C^A_{min}	Percentage compression $(1 - C^A_{min}/n) \times 100$	Best-case distribution of runs of 0s	Best-case distribution of runs of 1s
1000	20	20	278	72.2	4/0, 16/29	3/1, 17/29
1000	20	40	310	69.0	1/13, 19/29	28/1, 12/29
1000	20	60	340	66.0	20/29	50/1, 10/29
1000	40	20	306	69.4	1/13, 8/1, 31/29	20/1
1000	40	40	340	66.0	10/1, 30/29	40/1
1000	40	60	374	62.6	1/0, 11/1, 28/29	1/0, 58/1, 1/19
1000	60	20	340	66.0	30/1, 30/29	20/1
1000	60	40	374	62.6	1/0, 31/1, 28/29	1/0, 38/1, 1/19
1000	60	60	404	59.6	32/1, 1/5, 27/29	60/1

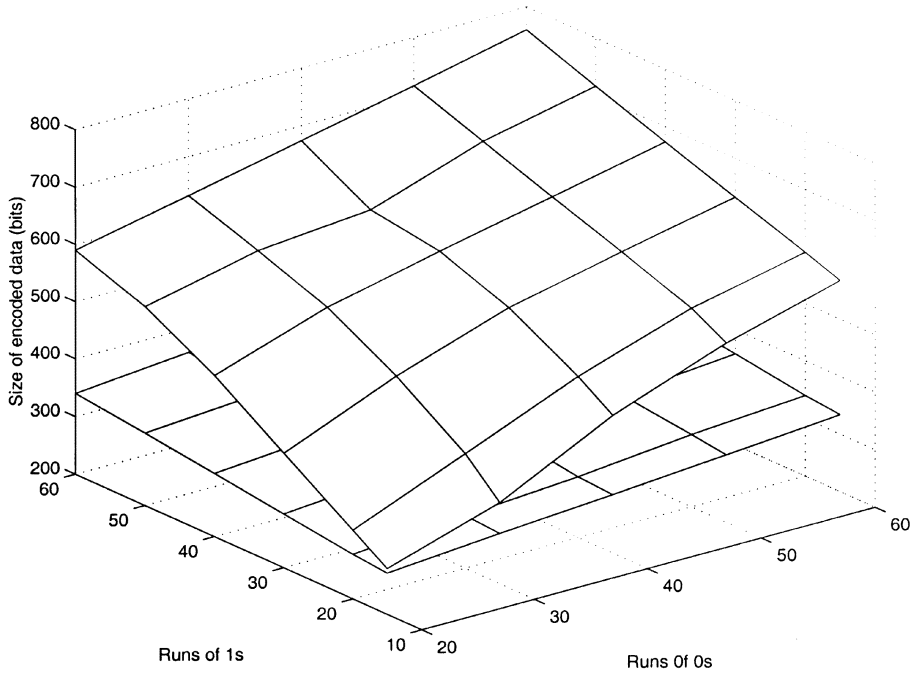

Fig. 8. Comparison between the upper and lower bounds on percentage compression for $n = 1000$.

number of runs r_1 and r_2 are varied (for $n = 1000$). The upper surface represents the worst case compression and the lower surface represents the best case compression that can be obtained using alternating run-length codes for different values of r_1 and r_2. The actual size of the encoded data will lie between these two bounds and is dependent on the distribution of the runs of 0's and runs of 1's. We note that for small values of r_1 and r_2, the bounds are very close to each other. For example, for $r_1 = 20$ and $r_2 = 15$, the difference between $C^{\mathrm{A}}_{\mathrm{max}}$ and $C^{\mathrm{A}}_{\mathrm{min}}$ is only eight bits; hence, the alternating run-length code is robust, i.e., its efficiency is relatively insensitive to variations in the distributions of the runs.

VI. CONCLUSION

We have shown that TRP based on test data compression can be used to reduce SOC test data volume, testing time, and test power simultaneously. The proposed TRP method is based on the use of a new code, which we call the alternating run-length code. We have shown that a

careful mapping of the don't-cares in precomputed test sets to 1's and 0's leads to significant savings in peak and average power, without requiring either a slower scan-clock or blocking logic in the scan cells. In addition, the on-chip decompression of test pattern decouples the internal scan chain(s) from the ATE, thereby allowing higher scan clock frequency. We have presented a rigorous testing-time analysis for compression/decompression based on alternating run-length codes. Experimental results for the ISCAS89 benchmark circuits and for an IBM production circuit show that a slower ATE can often be used with no adverse impact on testing time. Therefore, the proposed approach not only decreases test data volume and the amount of data that must be transferred from the ATE, but it also reduces test power and testing time, and facilitates the use of less expensive ATEs.

ACKNOWLEDGMENT

The authors thank B. Keller of Cadence Design Systems, Endicott, NY, (formerly with IBM Corporation, Endicott, NY), for providing scan vectors for the production circuit.

REFERENCES

[1] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," in *Proc. VLSI Test Symp.*, 1993, pp. 4–9.

[2] I. Hamzaoglu and J. H. Patel, "Reducing test application time for full scan embedded cores," in *Proc. Int. Symp. Fault-Tolerant Comput.*, 1999, pp. 260–267.

[3] F. F. Hsu, K. M. Butler, and J. H. Patel, "A case study on the implementation of Illinois scan architecture," in *Proc. Int. Test Conf.*, 2001, pp. 538–547.

[4] A. Chandra and K. Chakrabarty, "Test resource partitioning for SOCs," *IEEE Design Test Comput.*, vol. 18, pp. 80–91, Sept.–Oct. 2001.

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

[6] 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, Mar. 2001.

[7] P. T. 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," in *Proc. Design, Automation, Test Eur. Conf.*, 2002, pp. 604–611.

[8] S. Kajihara, K. Taniguchi, I. Pomeranz, and S. M. Reddy, "Test data compression using don't-care identification and statistical encoding," in *Proc. Int. Workshop Electron. Design, Test, Applicat.*, 2002, pp. 413–416.

[9] R. Dorsch and H.-J. Wunderlich, "Tailoring ATPG for embedded testing," in *Proc. Int. Test Conf.*, 2001, pp. 530–537.

[10] A. El-Maleh, S. al Zahir, and E. Khan, "A geometric-primitives-based compression scheme for testing systems-on-chip," in *Proc. VLSI Test Symp.*, 2001, pp. 54–59.

[11] I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction through scan chain concealment," in *Proc. ACM/IEEE Design Automation Conf.*, 2001, pp. 151–155.

[12] A. Khoche, E. Volkerink, J. Rivoir, and S. Mitra, "Test vector compression using EDA-ATE synergies," in *Proc. VLSI Test Symp.*, 2002, pp. 97–102.

[13] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, and B. Koenemann, "OPMISR: the foundation for compressed ATPG vectors," in *Proc. Int. Test Conf.*, 2001, pp. 748–757.

[14] 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. Comput.*, vol. 44, pp. 223–233, Feb. 1995.

[15] N. Nicolici and B. M. Al-Hashimi, "Scan latch partitioning into multiple scan chains for power minimization in full scan sequential circuits," in *Proc. IEEE/ACM Design, Automation, Test Eur. Conf.*, Mar. 2000, pp. 715–722.

[16] L. Whetsel, "Adapting scan architectures for low power operation," in *Proc. Int. Test Conf.*, 2000, pp. 863–872.

[17] J. Saxena, K. Butler, and L. Whetsel, "An analysis of power reduction techniques in scan testing," in *Proc. Int. Test Conf.*, 2001, pp. 670–677.

[18] P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, and H.-J. Wunderlich, "A modified clock scheme for a low power BIST test pattern generator," in *Proc. VLSI Test Symp.*, 2001, pp. 306–311.

[19] F. Corno, M. Rebaudengo, and M. S. Reorda, "Low power BIST via nonlinear hybrid cellular automata," in *Proc. VLSI Test Symp.*, 2000, pp. 29–34.

[20] S. Gerstendörfer and H.-J. Wunderlich, "Minimized power consumption for scan-based BIST," in *Proc. Int. Test Conf.*, 1999, pp. 77–84.

[21] P. Girard, L. Guiller, C. Landrault, and S. Pravossoudovitch, "A test vector inhibiting technique for low energy BIST design," in *Proc. VLSI Test Symp.*, 1999, pp. 407–412.

[22] L. Xu, Y. Sun, and H. Chen, "Scan solution for testing power and testing time," in *Proc. Int. Test Conf.*, 2001, pp. 652–659.

[23] S. Wang and S. K. Gupta, "ATPG for heat dissipation minimization during scan testing," in *Proc. ACM/IEEE Design Automation Conf.*, 1997, pp. 614–619.

[24] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. M. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 1325–1333, Dec. 1998.

[25] I. Pomeranz, S. M. Reddy, and R. Guo, "Static Test compaction for synchronous sequential circuits based on vector restoration," *IEEE Trans. Computer-Aided Design*, pp. 1040–1049, July 1999.

[26] R. Sankaralingam, R. R. Oruganti, and N. A. Touba, "Static compaction techniques to control scan vector power dissipation," in *Proc. VLSI Test Symp.*, 2000, pp. 35–40.

[27] V. Iyengar and K. Chakrabarty, "System-on-a-chip test scheduling with precedence relationships, preemption, and power constraints," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 1088–1094, Sept. 2002.

[28] K. Chakrabarty, "Test scheduling for core-based systems using mixed-integer linear programming," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1163–1174, Oct. 2000.

[29] M. Sugihara, H. Date, and H. Yasuura, "A novel test methodology for core-based system LSI's and a testing time minimization problem," in *Proc. Int. Test Conf.*, 1998, pp. 465–472.

[30] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S. M. Reddy, "Resource allocation and test scheduling for concurrent test of core-based SOC design," in *Proc. Asian Test Symp.*, 2001, pp. 265–270.

[31] Y. Huang, S. M. Reddy, W.-T. Cheng, P. Reuter, C.-C. Tsai, N. Mukherjee, O. Samman, and Y. Zaidan, "Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm," in *Proc. Eur. Test Workshop Dig. Papers* , 2002, pp. 35–41.

[32] P. M. Rosinger, B. M. Al-Hashimi, and N. Nicolici, "Power profile manipulation: a new approach for reducing test application time under power constraint," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 1217–1225, Oct. 2002.

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

[34] ——, "Low-power scan testing and test data compression for system-on-a-chip," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 597–604, May 2002.

[35] P. Rosinger, P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici, "Simultaneous reduction in volume of test data and power dissipation for system-on-a-chip," *Electron. Lett.*, vol. 37, no. 24, pp. 1434–1436, Nov. 2001.

[36] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 283–289.

[37] D. Heidel, S. Dhong, P. Hofstee, M. Immediato, K. Nowka, J. Silberman, and K. Stawiasz, "High-speed serializing/de-serializing design-for-test methods for evaluating a 1 GHz microprocessor," in *Proc. VLSI Test Symp.*, 1998, pp. 234–238.

[38] M. Berkelaar. (1999) lpsolve, Vers. 3.0. Eindhoven Univ. of Tech., Eindhoven, The Netherlands. [Online]ftp://ftp.ics.ele.tue.nl/pub/lp_solve