Short Papers

Test Data Compression and Decompression Based on Internal Scan Chains and Golomb Coding

Anshuman Chandra and Krishnendu Chakrabarty

Abstract—**We present a data compression method and decompression architecture for testing embedded cores in a system-on-a-chip (SOC). The proposed approach makes effective use of Golomb coding and the internal scan chain(s) of the core under test and provides significantly better results than a recent compression method that uses Golomb coding and a separate cyclical scan register (CSR). The major advantages of Golomb coding of test data include very high compression, analytically predictable compression results, and a low-cost and scalable on-chip decoder. The use of the internal scan chain for decompression obviates the need for a CSR, thereby reducing hardware overhead considerably. In addition, the novel interleaving decompression architecture allows multiple cores in an SOC to be tested concurrently using a single ATE I/O channel. We demonstrate the effectiveness of the proposed approach by applying it to the ISCAS 89 benchmark circuits.**

Index Terms—**Automatic test equipment (ATE), decompression architecture, embedded core testing, precomputed test sets, response vectors, system-on-a-chip testing, test set encoding, testing time, variable-to-variable-length codes.**

I. Introduction

Embedded cores are becoming commonplace in large system-on-a-chip (SOC) designs [1]. Along with the benefits of higher integration and shorter time to market, intellectual property (IP) cores pose several difficult test challenges. The volume of test data for an SOC is growing rapidly as IP cores become more complex and an increasing number of these cores are integrated in a chip. In order to effectively test these systems, each core must be adequately exercised with a set of precomputed test patterns provided by the core vendor. However, the input/output (I/O) channel capacity, speed and accuracy, and data memory of automatic test equipment (ATE) are severely limited.

The testing time for an SOC depends on the test data volume, the time required to transfer the data to the cores, and the rate at which it is transferred (measured by the cores test data bandwidth and ATE channel capacity). Lower testing time increases production capacity as well as reduces test cost and time to market for an SOC. New techniques are therefore needed for decreasing test data volume in order to overcome memory bottlenecks and to reduce testing time.

An attractive approach for reducing test data volume for SOCs is based on the use of data compression techniques [2]–[4]. In this approach, 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 the ATE memory. An on-chip decoder is used for pattern decompression to generate T_D from T_E during pattern application. Test data compression using statistical coding of test sequences for synchronous sequential (nonscan) circuits was presented in [2] and for full-scan circuits in [3]. While the compression method in [2] is restricted to sequential circuits with a large number of flip–flops and relatively few primary inputs, the work presented in [3] does not conclusively demonstrate that statistical coding provides greater compression than standard ATPG compaction methods for full-scan circuits [5], [6].

Test data can be more efficiently compressed by taking advantage of the fact that the number of bits changing between successive test patterns in a test sequence is generally very small. This observation was used in [4], where a "difference vector" sequence $T_{\rm diff}$ determined from T_D was compressed using run-length coding. A drawback of the compression method described in [4] is that it relies on variable-to-fixed-length codes, which are less efficient than more general variable-to-variable-length codes [7], [8]. Furthermore, it is inefficient for cores with internal scan chains that are used to capture test responses; in these circuits, separate CSRs must be added to the SOC, thereby increasing hardware overhead. A more efficient compression and decompression method was used in [9], where $T_{\rm diff}$ was compressed using variable-to-variable-length Golomb codes. However, this approach requires separate CSRs and is therefore also inefficient for cores that use the same internal scan chains for applying test patterns and capturing test responses.

In this companion paper to [9], we present an improved test data compression and decompression method for IP cores in an SOC. The proposed approach makes effective use of Golomb codes and the internal scan chain(s) of the core under test. No separate CSR is required for pattern decompression. The difference sequence $T_{\rm diff}^R$ is derived from the given precomputed test set T_D using the fault-free responses R of the core under test to T_D. Golomb coding is then applied to $T_{\rm diff}^R$. The resulting encoded test set T_E is much smaller than the original precomputed test set T_D. We apply our compression approach to test sets for the ISCAS 89 benchmark circuits and show that T_E is not only considerably smaller than the smallest test sets obtained using ATPG compaction [5], but it is also significantly smaller than the compressed test sets obtained using Golomb coding in [9].

The proposed compression approach for reducing test data volume is especially suitable for system-on-a-chip containing IP cores since it does not require gate-level models for the embedded cores. Precomputed test sets can be directly encoded without any fault simulation or subsequent test generation. This is in contrast to other recent techniques, such as LFSR-based reseeding for BIST [10] and scan broadcast [11], which require structural models for fault simulation and test generation. The mixed-mode BIST technique in [10] relies on fault simulation for identifying hard faults and test generation to determine test cubes for these faults. The scan broadcast technique in [11] also requires test generation.

We extend the decompression architecture of [9] to an interleaving scheme that allows multiple cores to be tested in parallel with a single ATE I/O channel. We also present analytical results for test data compression and testing time. Finally, we show that test data compression not only reduces the volume of test data but it also allows a slower tester to be used without any penalty on testing time.

Manuscript received December 21, 2000; revised July 10, 2001. This work was supported in part by the National Science Foundation under Grant CCR-9875324. This paper was presented in part in *Proc. Design, Automation and Test in Europe (DATE) Conference*, pp. 145–149, Munich, Germany, March 2001. This paper was recommended by Associate Editor R. Aitken.

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

Publisher Item Identifier S 0278-0070(02)04700-0.

0278-0070/02$17.00 © 2002 IEEE

Group	Run-length	Group prefix	Tail	Codeword
A_1	0	0	00	000
	1		01	001
	2		10	010
	3		11	011
A_2	4	10	00	1000
	5		01	1001
	6		10	1010
	7		11	1011
A_3	8	110	00	11000
	9		01	11001
	10		10	11010
	11		11	11011
...	...	...	...	...

Fig. 1. An example of Golomb coding for $m = 4$.

II. Compression Method and Test Architecture

We first review Golomb coding and its application to test data compression [9]. The first step in the encoding procedure is to select the Golomb code parameter m. The choice of m has received a lot of attention in the information theory literature; for certain distributions of the input data stream (T_{diff}^R in our case), the group size m can be optimally determined. For example, if the input data stream is random with 0 probability p, then m should be chosen such that $p^m \approx 0.5$ [8]. However, since the difference vectors for precomputed test sets do not satisfy the randomness assumption, the best value of m for test data compression must be determined experimentally. Nevertheless, it has been shown in [9] that the best value of m can be approximated analytically.

Once the group size m is determined, the runs of zeros in precomputed test set are mapped to groups of size m (each group corresponding to a run length). The number of such groups is determined by the length of the longest run of zeros in the precomputed test set. The set of run lengths $\{0, 1, 2, \ldots, m-1\}$ forms group A_1; the set $\{m, m+1, m+2, \ldots, 2m-1\}$, group A_2; etc. In general, the set of run lengths $\{(k-1)m, (k-1)m+1, (k-1)m+2, \ldots, km-1\}$ comprises group A_k [8]. To each group A_k, we assign a group prefix of $(k-1)$ 1s followed by a zero. We denote this by $1^{(k-1)}0$. If m is chosen to be a power of two, i.e., $m = 2^N$, each group contains 2^N members and a $\log_2 m$-bit sequence (tail) uniquely identifies each member within the group. Thus, the final code word for a run length L that belongs to group A_k is composed of two parts, a group prefix and a tail. The prefix is $1^{(k-1)}0$ and the tail is a sequence of $\log_2 m$ bits. It can be easily shown that $(k-1) = (L \bmod m)$ i.e., $k = (L \bmod m) + 1$. The encoding process is illustrated in Fig. 1 for $m = 4$.

The next step in the compression procedure is to derive the difference vector set T_{diff} from T_D, where $T_D = \{t_1, t_2, t_3, \ldots, t_n\}$ is the (ordered) precomputed test set. The ordering is determined using a heuristic procedure described later. T_{diff} is defined as follows:

$$\begin{aligned} T_{\text{diff}} &= \{d_1, d_2, \ldots, d_n\} \\ &= \{t_1, t_1 \oplus t_2, t_2 \oplus t_3, \ldots, t_{n-1} \oplus t_n\} \end{aligned}$$

where a bit-wise exclusive-or operation is carried out between patterns t_i and t_j. This assumes that the CSR starts in the all-zero state. (Other starting states can be considered similarly.) The details of the Golomb coding procedure are presented in the companion paper [9], hence omitted here.

The proposed method differs from [9] in that no separate CSR is used; instead the internal scan chain is used for pattern decompression and the fault-free responses of the core under test are used to generate a difference vector set T_{diff}^R. Given an (ordered) precomputed test set T_D, the set of corresponding fault-free responses $R = \{r_1, r_2, \ldots, r_n\}$ is used to generate the test patterns. The difference vector set T_{diff}^R is now given by

$$\begin{aligned} T_{\text{diff}}^R &= \{d_1, d_2, \ldots, d_n\} \\ &= \{t_1, r_1 \oplus t_2, r_2 \oplus t_3, \ldots, r_{n-1} \oplus t_n\} \end{aligned}$$

where r_i is the fault-free response of the core under test to pattern t_i. A test architecture based on the use of T_{diff}^R is shown in Fig. 2.

As observed in [9], test data compression is more effective if T_D consists of test cubes containing don't-care bits. In order to determine T_{diff}^R in such cases, we need to assign appropriate binary values to the don't-care bits and perform logic simulation to obtain the corresponding fault-free responses. (In general, the simulation model for the core provided by the core vendor can be used to obtain the fault-free responses.) First, we set all don't-care bits in t_1, the first pattern in T_D, to zeros and use the logic simulation engine of FSIM [12] to generate the fault-free response r_1. The ordering algorithm described below is then used to generate the successive test patterns.

The problem of determining the best ordering is equivalent to the NP-Complete Traveling Salesman problem. Therefore, a greedy algorithm is used to generate an ordering and the corresponding T_{diff}^R. Suppose a partial ordering $t_1 t_2 \cdots t_i$ has already been determined for the patterns in T_D. To determine t_{i+1}, we first determine r_i using FSIM and then calculate the Hamming distance $HD(r_i, t_j)$ between r_i and all patterns t_j that have not been placed in the ordered list. We define $HD(r_i, t_j)$ as the number of bit positions for which r_i and t_j have different (specified) binary values. We select the pattern t_j for which $HD(r_i, t_j)$ is minimum and add it to the ordered list, denoting it by t_{i+1}. All don't-care bits in t_{i+1} are set to the corresponding specified bit in r_j. In this way, a fully specified test pattern is obtained and the smallest number of ones is added to the difference vector sequence. We continue this process until all test patterns in T_D are placed in the ordered list. Fig. 3 illustrates the procedure for obtaining T_{diff}^R from T_D.

For most cores, the number of inputs $|I_{\text{core}}|$ driven by the scan cells is not equal to the number of outputs $|O_{\text{core}}|$ that feed the scan chain. (I_{core} and O_{core} refer to the sets of inputs driven by the scan chain and outputs feeding the scan chain, respectively.) Consider the following two cases.

Case 1—$|I_{\text{core}}| > |O_{\text{core}}|$: Assume without loss of generality that the outputs in O_{core} drive scan elements that are located at the beginning of the scan chain. Let $t_i = \langle \tilde{t}_{i,1}, \tilde{t}_{i,2}, \ldots, \tilde{t}_{i,n} \rangle$ and $r_i = \langle \tilde{r}_{i,1}, \tilde{r}_{i,2}, \ldots, \tilde{r}_{i,k} \rangle$ denote the ith test pattern and the ith fault-free response, respectively. The encoding procedure is modified as follows to generate the difference vector $d_{i+1} = \langle \tilde{d}_{i+1,1}, \tilde{d}_{i+1,2}, \ldots \tilde{d}_{i+1,n} \rangle$, where

$$\begin{aligned} \tilde{d}_{i+1,1} &= \tilde{t}_{i+1,1} \oplus \tilde{r}_{i,1} \\ \tilde{d}_{i+1,2} &= \tilde{t}_{i+1,2} \oplus \tilde{r}_{i,2} \\ \cdots\ &\ \cdots \\ \tilde{d}_{i+1,k} &= \tilde{t}_{i+1,k} \oplus \tilde{r}_{i,k} \\ \tilde{d}_{i+1,k+1} &= \tilde{t}_{i+1,k+1} \oplus \tilde{t}_{i,k+1} \\ \cdots\ &\ \cdots \\ \tilde{d}_{i+1,n} &= \tilde{t}_{i+1,n} \oplus \tilde{t}_{i,n}. \end{aligned}$$

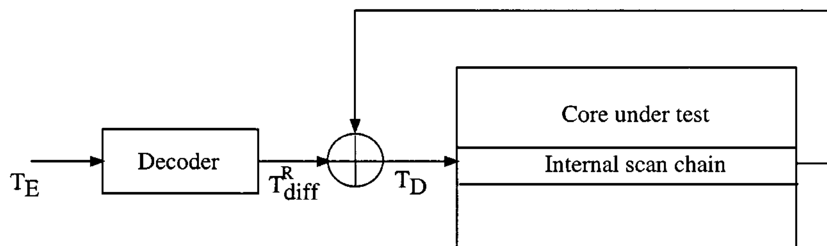

Fig. 2. Test architecture based on Golomb coding and the use of internal scan chains.

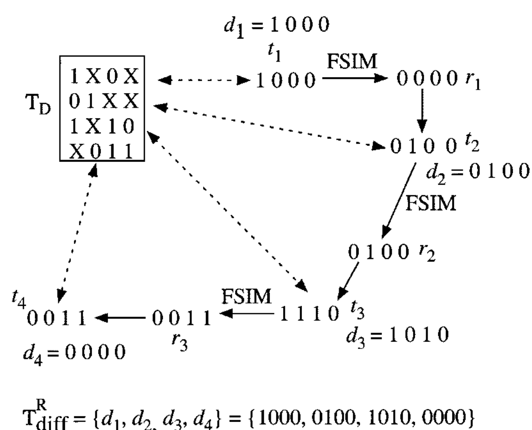

Fig. 3. An example to illustrate the procedure for deriving $T^R_{\rm diff}$.

Case 2—$|I_{\rm core}| < |O_{\rm core}|$: In this case, additional zeros must be inserted into the difference vector sequence as follows:

$$\begin{aligned}\tilde{d}_{i+1,1} &= \tilde{t}_{i+1,1} \oplus \tilde{r}_{i,1}\\ \tilde{d}_{i+1,2} &= \tilde{t}_{i+1,2} \oplus \tilde{r}_{i,2}\\ \cdots &\quad \cdots\\ \tilde{d}_{i+1,k} &= \tilde{t}_{i+1,k} \oplus \tilde{r}_{i,k}\\ \tilde{d}_{i+1,k+1} &= 0\\ \cdots &\quad \cdots\\ \tilde{d}_{i+1,n} &= 0.\end{aligned}$$

An on-chip decoder decompresses the encoded test set T_E and produces $T^R_{\rm diff}$. The exclusive-or gate and the internal scan chain are used to generate the test patterns from the difference vectors. As discussed in the companion paper [9], the decoder can be efficiently implemented by a $\log_2 m$-bit counter and a finite-state machine (FSM). The synthesized decode FSM circuit contains only four flip–flops and 34 combinational gates [9]. For any circuit whose test set is compressed using $m = 4$, the given logic is the only additional hardware required other than the two-bit counter.

III. Analysis of Test Application Time and Test Data Compression

In this section, we analyze the testing time for a single scan chain when Golomb coding is employed with the test architecture shown in Fig. 2. From the state diagram of the Golomb decoder [9], we note the following.

- Each "1" in the prefix part takes m cycles for decoding.
- Each separator "0" takes one cycle.
- The tail part takes a maximum of m cycles and a minimum of $\gamma = \log_2 m + 1$ cycles.

Let n_c be the total number of bits in T_E and r be the number of ones in $T^R_{\rm diff}$. T_E contains r tail parts, r separator zeros, and the number of prefix ones in T_E equals $n_c - r(1+\log_2 m)$. Therefore, the maximum and minimum testing times ($\mathcal{T}_{\max}$ and $\mathcal{T}_{\min}$, respectively), measured by the number of cycles, are given by

$$\begin{aligned}\mathcal{T}_{\max} &= (n_c - r(1+\log_2 m))m + r + mr\\ &= mn_c - r(m\log_2 m - 1)\\ \mathcal{T}_{\min} &= (n_c - r(1+\log_2 m))m + r + \gamma r\\ &= mn_c - r(m(1+\log_2 m) - (1+\gamma)).\end{aligned}$$

Therefore, the difference between $\mathcal{T}_{\max}$ and $\mathcal{T}_{\min}$ is given by

$$\begin{aligned}\delta\mathcal{T} &= \mathcal{T}_{\max} - \mathcal{T}_{\min}\\ &= r(m - \log_2 m - 1).\end{aligned}$$

We will make use of this result in Section IV.

A major advantage of Golomb coding is that on-chip decoding can be carried out at scan clock frequency $f_{\rm scan}$ while T_E can be fed to the core under test with external clock frequency $f_{\rm ext} < f_{\rm scan}$. This allows us to use slower testers without increasing the test application time. The external and scan clocks must be synchronized, e.g., using the scheme described in [13], and $f_{\rm scan} = mf_{\rm ext}$, where the Golomb code parameter m is usually a power of 2. This allows the bits of $T^R_{\rm diff}$ to be generated by the decoder at the frequency of $f_{\rm scan}$. We now present an analysis of testing time using $f_{\rm sys} = mf_{\rm ext}$ and compare the testing time for our method with that of external testing in which ATPG-compacted patterns are applied using an external tester.

Let the ATPG-compacted test set contain p patterns and let the length of the scan be n bits. Therefore, the size of the ATPG-compacted test set is pn bits and the testing time $\mathcal{T}_{\rm ATPG}$ equals pn external clock cycles. Next, suppose the difference vector $T^R_{\rm diff}$ obtained from the uncompacted test set contains r ones and its Golomb-coded test set T_E contains n_c bits. The maximum number of scan clock cycles required for applying the test patterns using the Golomb coding scheme is $\mathcal{T}_{\max} = mn_c - r(m\log_2 m - 1)$.

Now, the maximum testing time τ (seconds) when Golomb coding is used is given by

$$\begin{aligned}\tau &= \frac{\mathcal{T}_{\max}}{f_{\rm scan}}\\ &= \frac{mn_c - r(m\log_2 m - 1)}{f_{\rm scan}}\end{aligned}$$

and the testing time τ' (seconds) for external testing with ATPG-compacted patterns is given by

$$\begin{aligned}\tau &= \frac{pn}{f_{\rm ext}}\\ &= \frac{pnm}{f_{\rm scan}}.\end{aligned}$$

If testing is to be accomplished in $\tau^\star$ seconds using Golomb coding, the scan clock frequency $f_{\rm scan}$ must equal $\mathcal{T}_{\max}/\tau^\star$, i.e.,

$$f_{\rm scan} = \frac{mn_c - r(m\log_2 m - 1)}{\tau^\star}.$$

This is achieved using a slow external tester operating at frequency $f_{\rm ext} = f_{\rm scan}/m$. On the other hand, if only an external test is used with the p ATPG-compacted patterns, the required external tester clock frequency $f'_{\rm ext}$ equals $pn/\tau^\star$. Let us take the ratio of $f'_{\rm ext}$ between $f_{\rm ext}$

$$\frac{f'_{\rm ext}}{f_{\rm ext}} = \frac{pn/\tau^\star}{f_{\rm scan}/m} = \frac{pn}{n_c - r\log_2 m + r/m}.$$

Experimental results presented in Section V show that $f'_{\rm ext}$ is much greater than $f_{\rm ext}$, therefore, demonstrating that the use of Golomb coding allows us to decrease the volume of test data and use a slower tester without increasing testing time.

We next analyze the amount of compression that is achieved using Golomb coding of a precomputed test set T_D. The following three lemmas will lead to the main result in Theorem 1. As in [9], we assume without loss in generality here that the difference sequence always ends in one.

Lemma 1: Let T_D be the given precomputed test set, and let $T^R_{\rm diff}$ be the bit stream derived from T_D and the set of fault-free responses. Let the number of don't cares in T_D be n_ϕ. The number of zeros in $T^R_{\rm diff}$ is at least n_ϕ.

Proof: The lemma follows from the fact that every don't care in T_D can be mapped to a zero in $T^R_{\rm diff}$, while ones and zeros in T_D must be selectively mapped to ones or zeros in $T^R_{\rm diff}$, depending on the fault-free response. □

Lemma 2 [9]: If an n-bit data stream S containing r ones is encoded using Golomb code with parameter m, an upper bound on the length G_S of the encoded sequence is given by

$$G_S \le \frac{n}{m} + r\log_2 m + r\left(1 - \frac{1}{m}\right).$$

Lemma 3: Let S be any binary sequence and let $S^\star$ be a binary sequence derived from S by replacing one or more ones in it by zeros. Let S_E ($S^\star_E$) be the Golomb-coded sequence corresponding to S ($S^\star$). Then $len(S_E) > len(S^\star_E)$, where $len(S_E)$ and $len(S^\star_E)$ are the number of bits in S_E and $S^\star_E$, respectively.

Proof: Suppose we complement a 1 in S that separates two runs of zeros of length l_1 and l_2 ($l_1, l_2 \ge 0$), respectively, to obtain $S^\star$. We now have a run of ($l_1 + l_2 + 1$) zeros in $S^\star$. The number of bits N required to encode the two runs of zeros of length l_1 and l_2 is given by

$$N = \left\lfloor \frac{l_1}{m} \right\rfloor + 1 + \log_2 m + \left\lfloor \frac{l_2}{m} \right\rfloor + 1 + \log_2 m = \left\lfloor \frac{l_1}{m} \right\rfloor + \left\lfloor \frac{l_2}{m} \right\rfloor + \log_2 m + 2.$$

Similarly, the number of bits in $N^\star$ required to encode the single run of ($l_1 + l_2 + 1$) zeros in $S^\star$ is given by

$$N^\star = \left\lfloor \frac{(l_1 + l_2 + 1)}{m} \right\rfloor + \log_2 m + 1.$$

This implies that

$$\begin{aligned} len(S_E) - len(S^\star_E) &= N - N^\star \\ &= \left\lfloor \frac{l_1}{m} \right\rfloor + \left\lfloor \frac{l_2}{m} \right\rfloor - \left\lfloor \frac{(l_1 + l_2 + 1)}{m} \right\rfloor + \log_2 m + 1 \\ &\ge \left\lfloor \frac{l_1}{m} \right\rfloor + \left\lfloor \frac{l_2}{m} \right\rfloor - \frac{(l_1 + l_2 + 1)}{m} + \log_2 m + 1 \\ &\ge \left(\frac{l_1}{m} - 1\right) + \left(\frac{l_2}{m} - 1\right) - \frac{(l_1 + l_2 + 1)}{m} + \log_2 m + 1 \\ &= \log_2 m - 1 - \frac{1}{m}. \end{aligned}$$

This implies that $len(S_E) - len(S^\star_E) > 0$ if $m > 2$. For the special case of $m = 2$, we note that

$$\begin{aligned} len(S_E) - len(S^\star_E) &= \left\lfloor \frac{l_1}{m} \right\rfloor + \left\lfloor \frac{l_2}{m} \right\rfloor - \left\lfloor \frac{(l_1 + l_2 + 1)}{2} \right\rfloor + \log_2 2 + 1 \\ &\ge \frac{(l_1 - 1)}{2} + \frac{(l_2 - 1)}{2} - \frac{(l_1 + l_2 + 1)}{2} + \log_2 2 + 1 \\ &= 0.5. \end{aligned}$$

Therefore, complementing a single one to a zero always decreases the length of the Golomb-coded sequence. This argument can be easily extended using transitivity to show that $len(S_E) > len(S^\star_E)$ whenever one or more ones in S are changed to zeros to obtain $S^\star$. □

We now present an upper bound on the amount of expression that is obtained via Golomb coding of $T^R_{\rm diff}$. The proof of the theorem follows from Lemmas 1–3.

Theorem 1: Let T_D be the given precomputed test set, and let $T^R_{\rm diff}$ be the n-bit data stream derived from T_D and the set of fault-free responses. Let the number of don't cares in T_D be n_ϕ. If $T^R_{\rm diff}$ is encoded using Golomb code with parameter m, an upper bound on the length G of the encoded sequence is given by

$$G \le \frac{n}{m} + (n - n_\phi)\log_2 m + (n - n_\phi)\left(1 - \frac{1}{m}\right).$$

Theorem 1 provides an easy-to-compute bound on the size of the encoded test set T_E. This bound depends only on the precomputed test set T_D and is independent of the fault-free response. It can therefore be obtained without any logic simulation. We list these bounds for several ISCAS 89 circuits in Section V.

IV. Interleaving Decompression Architecture

We now present a novel interleaving decompression architecture, which enables testing of multiple cores or the loading of multiple balanced scan chains in parallel. The same decoder can be used to drive equal-length scan chains in one or more cores in parallel. An important constraint here is that the same value of m must encode test sequences for all the scan chains. The proposed decompression architecture not only reduces the testing time and the size of the test data to be stored in the ATE memory, but also allows testing of multiple cores using a single ATE I/O channel, thereby increasing the ATE I/O channel capacity.

As discussed in Section II, when Golomb coding is applied to a block of data containing a run of 0s followed by a single 1, the code word contains two parts—a prefix and tail. For a given code parameter m (group size), the length of the tail ($\log_2 m$) is independent of the run length. Note further that every one in the prefix corresponds to m zeros in the decoded difference vector. Thus the prefix consists of a string of ones followed by a zero, and the zero can be used to identify the beginning of the tail.

As shown in [9], the FSM in the decoder runs the counter for m decode cycles whenever a one is received and starts decoding the tail as soon as a zero is received. The tail decoding takes at most m cycles. During prefix decoding, the FSM has to wait for m cycles before the next bit of the prefix can be decoded. Therefore, we can use interleaving to test m cores together, such that the decoder corresponding to each core is fed with encoded prefix data after every m cycles. (This can also be used to feed multiple scan chains in parallel as long as the capture cycles of the scan chains are synchronized.) Whenever the tail is to be decoded (identified by a zero in the encoded bit stream), the respective decoder is fed with the entire tail of $\log_2 m$ bits in a single burst of $\log_2 m$ cycles. The SOC channel selector consisting of a demultiplexer, a $\log_2 m$ counter, and an FSM is used for interleaving; see Fig. 4. This interleaving scheme works as follows.

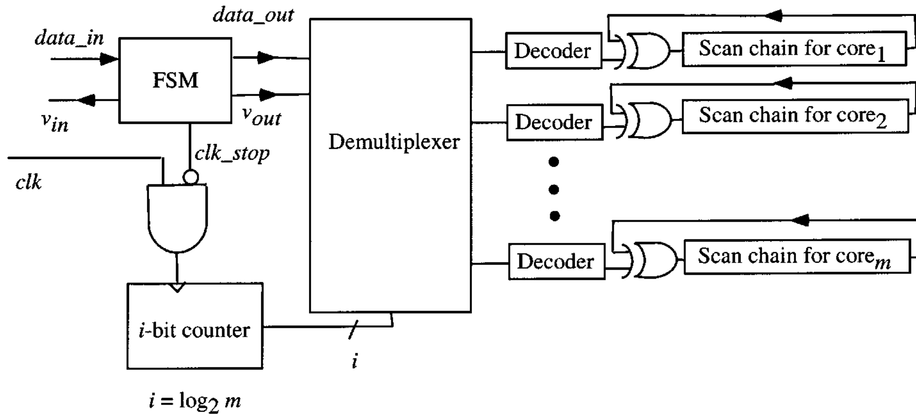

Fig. 4. SOC channel selector for application to multiple cores and multiple scan chains.

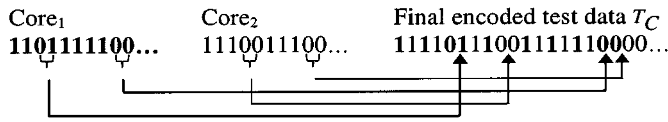

Fig. 5. Composite encoded test data for two cores with group size $m = 2$.

First, the encoded test data for m cores are combined to generate a composite bit stream T_C that is stored in the ATE. Next, T_C is fed to the FSM, which is used to detect the beginning of each tail and to feed the demultiplexer. An i-bit counter ($i = \log_2 m$) is used to select the outputs to the decoders of the various cores. T_C is obtained by interleaving the prefix parts of the compressed test sets of each core, but the tails are included unchanged in T_C . An example is shown in the Fig. 5 where compressed data for two cores (generated using group size $m = 2$) have been interleaved to obtain the final encoded test set to be applied through the decompression scheme for multiple cores.

We now describe the SOC channel selector in more detail. The FSM, the i bit counter, and the demultiplexer together constitute the SOC channel selector. The FSM is used to detect the beginning of the tail and generates the clk_stop signal to stop the i-bit counter. The $data_in$ is the input to the FSM, $data_out$ is the output, and signals $v_{\rm in}$ and $v_{\rm out}$ are used to indicate that the input and output data is valid. The i-bit counter is connected to the select lines of the demultiplexer and the demultiplexer outputs are connected to the decoders of the different scan chains. Every scan chain has a dedicated decoder. This decoder receives either a one or the tail of the compressed data corresponding to the various cores connected to the scan chain. If the FSM detects that a portion of the tail has arrived, the zero that is used to identify the tail is passed to the decoder and the clk_stop goes high for the next m cycles. The output of the demultiplexer does not change for this period and the entire tail of length $\log_2 m$-bits is passed on continuously to the appropriate core.

The state diagram of the FSM for $m = 4$ and the corresponding timing diagram are shown in Figs. 6 and 7, respectively. The FSM is fed with T_C corresponding to four different cores. It remains in state $S0$ as long as it receives the ones corresponding to the prefixes. As soon as a zero is received, it outputs the entire tail unchanged and makes clk_stop high. This stops the i-bit counter and prevents any change at demultiplexer output. It is shown in the timing diagram (Fig. 7) that whenever a zero is received, the SOC channel selection remains unchanged for the next $(1 + m)$ cycles.

As discussed in Section III, the difference in $\mathcal{T}_{\max}$ and $\mathcal{T}_{\min}$ is given by $\delta\mathcal{T} = r(m - \log_2 m - 1)$. Therefore, the difference between maximum and minimum testing times for a single tail is $\delta t = (m - \log_2 m - 1)$. If we restrict m to be small, $m \leq 8$, $\delta t \leq 4$. In this case, the decode FSM can be easily modified by introducing additional states to the Golomb decoder FSM of [9] such that the tail decoding always takes m cycles and $\delta t = 0$. To make tail and prefix decoding equal for $m = 4$, three additional states are required as shown in Fig. 8. The additional states do not adversely affect the testing time and the hardware overhead significantly. There are m cores in parallel and each separator zero and tail takes $(1 + m)$ cycles to decode. Therefore, for m cores, the decoding time $t_{\rm tail}$ for the separator and the tail is given by

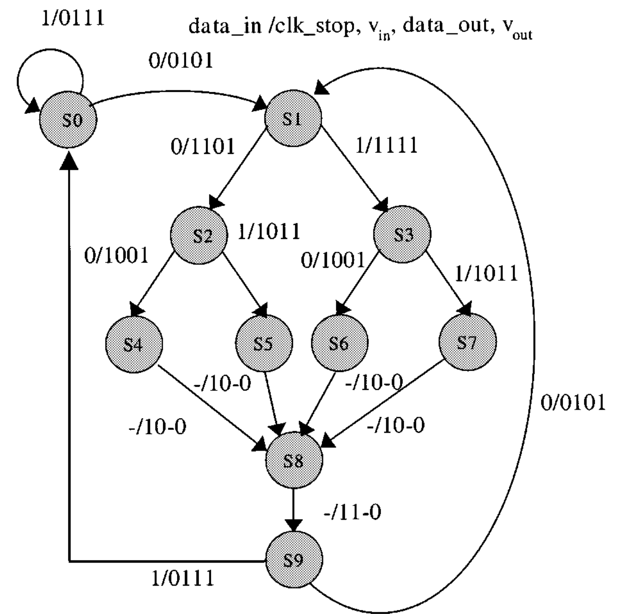

Fig. 6. State diagram for the SOC channel selector FSM ($m = 4$).

$$t_{\rm tail} = \sum_{j=1}^{m} (r_j + m r_j)$$

$$= (1 + m) \sum_{j=1}^{m} r_j$$

$$= (1 + m)R$$

where $R = \sum_{j=1}^{m} r_j$. Since all the prefixes of the cores are decoded in parallel, the number of cycles $t_{\rm prefix}$ required for decoding all the prefixes in T_C is equal to the number of ones in the prefix of the core with the largest encoded test data. Therefore

$$t_{\rm prefix} = \max\{(n_{C,\,i} - r_i(1 + \log_2\, m))m\}$$

$$= (n_{C,\,\max} - r_{\max}(1 + \log_2\, m))m$$

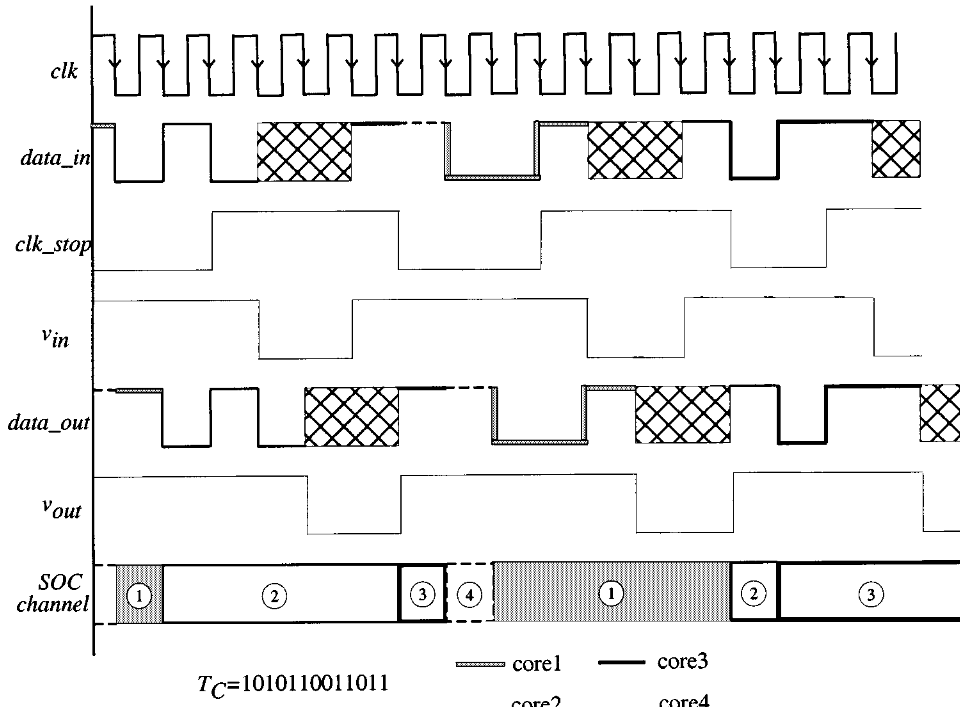

Fig. 7. Timing diagram for the SOC channel selector FSM ($m = 4$).

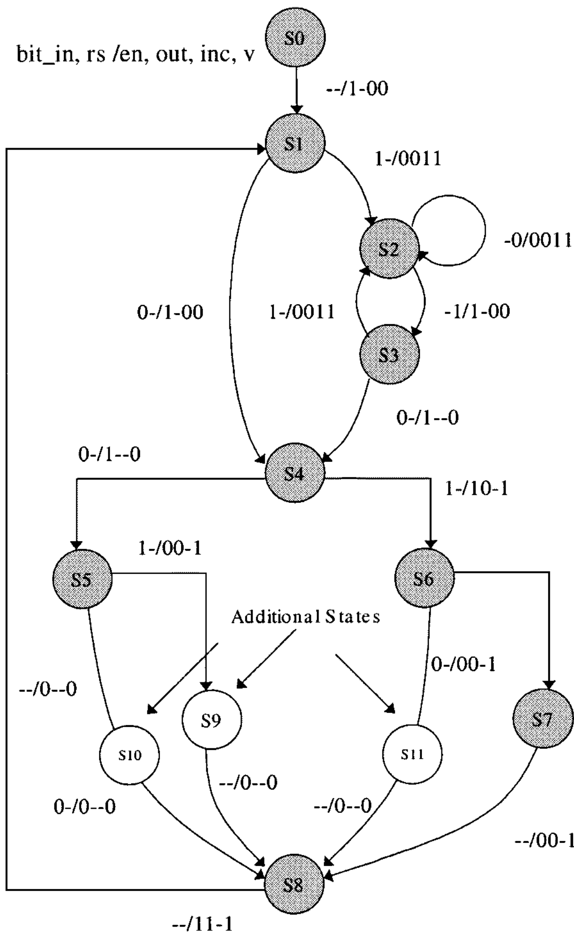

Fig. 8. Modified state diagram of the decode FSM to make tail and prefix decode cycles equal.

where $n_{C,\,i}$ and r_i are the number of encoded bits in T_E and number of ones in $T_{\rm diff}$ for the ith core, respectively, and $n_{C,\,\max}$ and $r_{\max}$ are the number of encoded bits in T_E and number of ones in $T_{\rm diff}$ for the core with the largest encoded test data. Therefore, total testing time $\mathcal{T}_I$

TABLE I
EXPERIMENTAL RESULTS ON TEST DATA COMPRESSION USING GOLOMB CODES (COMPARISON WITH [9])

Circuit	Golomb coding using test pattern to generate next pattern [9]				Golomb coding using response pattern to generate next pattern				Size of Mintest test set (bits)	Upper bound (bits)
	Best value of m	Size of test set T_{diff} (bits)	Best compression (percent)	Size of T_E (bits)	Best value of m	Size test set T^R_{diff} (bits)	Best compression (percent)	Size of of T_E (bits)		
s9234	4	39273	43.34	22250	4	39750	43.40	22495	25935	30273
s13207	16	165200	74.78	41658	32	186440	81.16	35122	163100	54589
s15850	4	76986	47.11	40717	8	86184	64.51	30581	57434	51150
s35932*	512	4007299	98.51	59573	2048	4655104	99.42	26885	19393	32355
s38417	4	164736	44.12	92054	4	172458	47.18	91088	113152	134950
s38584	4	199104	47.71	104111	8	235280	61.79	89884	161040	120350

*The test set used here is obtained from the Atalanta ATPG program [14]. (The Mintest test set with dynamic compaction is almost fully compacted.)

for m cores when tested in parallel using the interleaving architecture is given by

$$\mathcal{T}_I = t_{\text{prefix}} + t_{\text{tail}} = (n_{C,\max} - r_{\max}(1 + \log_2 m))m + (1+m)R. \quad (1)$$

Let us now find the testing time $\mathcal{T}_{NI}$ (NI denotes noninterleaved) required if all the cores were tested one by one independently using a single ATE I/O channel

$$\begin{aligned}\mathcal{T}_{NI} &= \sum_{j=1}^{m}\{(n_{C,j} - r_j(1+\log_2 m))m\} + (1+m)R \\ &= m|T_C| - m\sum_{j=1}^{m} r_j \log_2 m - m\sum_{j=1}^{m} r_j + (1+m)R \\ &= m|T_C| - mR\log_2 m + R \\ &= m|T_C| - R(m\log_2 m - 1) \quad (2)\end{aligned}$$

where $|T_C|$ denotes the number of bits in T_C. The difference between the interleaved and the noninterleaved testing times is given by

$$\begin{aligned}\mathcal{T}_{NI} - \mathcal{T}_I &= m|T_C| - Rm\log_2 m + R - n_{C,\max} \\ &\quad + mr_{\max}(1+\log_2 m) - R - mR \\ &= m(|T_C| - n_{C,\max}) - m(1+\log_2 m)(R - r_{\max}) \\ &= m((|T_C| - n_{C,\max}) - (1+\log_2 m)(R - r_{\max})) \\ &\approx m(|T_C| - n_{C,\max}) \gg 0\end{aligned}$$

since $n_{C,\max} \gg r_{\max}$ and $T_C \gg R$.

Consider a hypothetical example of four cores with encoded test data size equal to $n_{C,1} = 40$, $n_{C,2} = 60$, $n_{C,3} = 80$, $n_{C,4} = 100$ and number of ones equal to $r_1 = 4$, $r_2 = 6$, $r_3 = 8$, $r_4 = 10$. Therefore, $n_{C,\max} = 100$, $r_{\max} = 10$, $m = 4$, $R = 28$ and $|T_C| = 280$. Therefore $\mathcal{T}_{NI} - \mathcal{T}_I = 4((280-100) - (1+2)(28-10)) = 504$. It is evident from the above analysis that interleaving architecture reduces testing time and increases the ATE channel bandwidth. We developed a Verilog model for the FSM for $m = 4$ and simulated it for several T_C sequences. The gate-level schematic (derived using Synopsys Design Compiler) of the channel selector FSM consists of only four flip–flops and 17 gates. The additional hardware overhead is therefore very small.

V. EXPERIMENTAL RESULTS

In this section, we present experimental results on Golomb coding of the precomputed test sets for the six largest ISCAS 89 benchmark circuits. We used test cubes (with dynamic compaction) obtained using the Mintest ATPG program [5]. The difference between the size of the test sequences here and in [9] can be explained as follows. Since the number of inputs driven by the scan chains is less in every case than the number of outputs that feed the scan chains, additional (dummy) zeros are inserted in the difference vector sequence T^R_{diff}. This procedure was explained in Section II.

TABLE II
COMPARISON BETWEEN THE EXTERNAL CLOCK FREQUENCY f_{ext} REQUIRED FOR GOLOMB-CODED TEST DATA AND THE EXTERNAL CLOCK FREQUENCY f'_{ext} REQUIRED FOR EXTERNAL TESTING USING ATPG-COMPACTED PATTERNS (FOR THE SAME TESTING TIME)

Circuit	m	r	n_c	pn	f'_{ext}/f_{ext}
s9234	4	5113	22495	25935	1.914
s13207	32	5111	35122	163100	16.768
s15850	8	5542	30581	57434	3.921
s38417	4	18924	91088	113152	1.951
s38584	8	16814	89884	161040	3.875

The results shown in Table I demonstrate that significant amount of compression is achieved if Golomb coding is applied to difference vectors obtained from the test set and the fault-free responses. In five out of six cases, we achieve better results than ATPG compaction using Mintest. In addition, the proposed method outperforms [9] in five out of the six cases. The upper bound values (derived from Theorem 1) represent the worst case compression that can be achieved using Golomb codes. The upper bound is an important parameter which can be used to determine the suitability of the proposed method.

Table II demonstrates that Golomb coding allows us to use a slower tester without incurring any testing time penalty. As discussed in Section III, Golomb coding provides three important benefits: 1) it significantly reduces the volume of test data; 2) the test patterns can be applied to the core under test at the scan clock frequency f_{scan} using an external tester that runs at frequency $f_{\text{ext}} = f_{\text{scan}}/m$; and 3) in comparison with external testing using ATPG-compacted patterns, the same testing time is achieved using a much slower tester. The third issue is highlighted in Table II.

We next compare our results with a recent parallel scan design technique aimed at reducing test data volume and testing time [11]. A direct comparison is difficult since the two methods employ different strategies. Nevertheless, a comparison with the published results in [11] shows that the proposed method outperforms [11] for five out of the six largest ISCAS 89 benchmark circuits; see Table III. Moreover, the scan broadcast approach in [11] requires a structural model of a core for test generation and for determining aliased faults, a restriction that does not affect the proposed compression technique.

TABLE III
COMPARISON WITH THE SCAN BROADCAST SCHEME IN [11]

Circuit	Size of compressed test data (bits) [11]	Size of compressed test data for proposed method (bits)
s9234	34882	22495
s13207	82546	35122
s15850	76030	30581
s35932	9136	26885
s38417	127932	91088
s38584	129580	89884

Testing time comparison between the two methods is especially difficult. The testing time in [11] is presented in terms of clock cycles, whereas the proposed method employs two different clock rates: a faster on-chip decode clock and a slower off-chip tester clock for feeding T_E. Hence, even though four to six times more clock cycles are required here compared to [11], we claim that the testing time is less since $f_{\rm scan}$ is much larger than $f_{\rm ext}$ and the use of 16 scan chains as in [11] can offer significantly more parallelism. (We assumed single scan chains for all the benchmarks in our experiments.)

The compression method presented here is directed at IP cores in SOCs, which are not BIST-ed and whose structural models are not available. Only the precomputed test sets are available to the system integrator. Greater compression can be achieved with LFSR-based reseeding in a BIST environment [10]. This, however, requires that the cores be BIST-ed and structural models be made available for fault simulation to identify easy faults and for test generation to determine test cubes for hard faults.

VI. CONCLUSION

We have presented a new test data compression and decompression method for testing embedded cores in an SOC. We have shown that the proposed scheme makes efficient use of Golomb codes and the internal scan chain(s) of the core under test to achieve high test data compression for SOCs and to save ATE memory and testing time.

We have also presented a novel interleaving decompression architecture that allows testing of multiple cores in parallel using a single ATE I/O channel. This reduces the testing time of an SOC further and increases the ATE I/O channel capacity. The additional logic for the SOC channel selector is small and easy to implement. In addition, it is independent of the multiple cores under test and their corresponding precomputed test sets. We also show that apart from reduction in the volume of test data, test data compression also allows a slower tester to be used without any reduction in testing time.

Experimental results for the ISCAS benchmarks show that the proposed scheme is very efficient for compressing test data. The results also show that ATPG compaction may not always be necessary for saving ATE memory and reducing testing time.

ACKNOWLEDGMENT

The authors acknowledge Prof. H.-J. Wunderlich of the University of Stuttgart, Germany, for discussions on the use of the internal scan chain for pattern application. The authors would also like to thank S. Swaminathan for help in carrying out the experiments.

REFERENCES

[1] Y. Zorian, E. J. Marinissen, and S. Dey, “Testing embedded-core based system chips,” in *Proc. Int. Test Conf.*, 1998, pp. 130–143.

[2] V. Iyengar, K. Chakrabarty, and B. T. Murray, “Deterministic built-in pattern generation for sequential circuits,” *J. Electron. Testing: Theory and Applications (JETTA)*, vol. 15, pp. 97–115, Aug./Oct. 1999.

[3] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, “Scan vector compression/decompression using statistical coding,” in *Proc. IEEE VLSI Test Symp.*, 1999, pp. 114–120.

[4] 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.

[5] I. Hamzaoglu and J. H. Patel, “Test set compaction algorithms for combinational circuits,” in *Proc. Int. Test Conf.*, 1998, pp. 283–289.

[6] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, “On compacting test sets by addition and removal of vectors,” in *Proc. VLSI Test Symp.*, 1994, pp. 202–207.

[7] S. W. Golomb, “Run-length encoding,” *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 399–401, 1966.

[8] H. Kobayashi and L. R. Bahl, “Image data compression by predictive coding, Part I: Prediction algorithm,” *IBM J. Res. Devel.*, vol. 18, p. 164, 1974.

[9] 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.

[10] S. Hellebrand, H.-G. Liang, and H.-J. Wunderlich, “A mixed-mode BIST scheme based on reseeding of folding counters,” in *Proc. Int. Test Conf.*, 2000, pp. 778–784.

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

[12] H. K. Lee and D. S. Ha, “An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation,” in *Proc. Int. Test Conf.*, Oct. 1991, pp. 946–955.

[13] 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. IEEE VLSI Test Symp.*, 1998, pp. 234–238.

[14] H. K. Lee and D. S. Ha, “On the generation of test patterns for combinational circuits,” Dept. Electrical Eng., Virginia Polytechnic Inst. State Univ., Tech. Rep. 12_93.