Efficient Test Data Compression and Decompression for System-on-a-Chip using Internal Scan Chains and Golomb Coding[1]

Anshuman Chandra and Krishnendu Chakrabarty
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708, USA
{achandra, krish}@ee.duke.edu

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 chains 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 use of the internal scan chain for decompression obviates the need for a CSR. 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.

1 Introduction

System-on-a-chip (SOC) designs consisting of intellectual property (IP) cores present a number of difficult test challenges [1]. 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. However, the I/O channel capacity, speed and accuracy, and data memory of automatic test equipment (ATE) are severely limited. New techniques are therefore needed for decreasing test data volume in order to overcome memory bottlenecks and to reduce testing time.

A promising approach for reducing test data volume for SOCs is based on data compression techniques [2, 3]. 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 ATE memory. An on-chip decoder is used for pattern decompression to generate T_D from T_E during pattern application.

Test data can be more efficiently compressed by exploiting the fact that the number of bits changing between successive patterns in a test sequence is generally very small. This observation was used in [3], where a "difference vector" sequence T_{diff} determined from T_D was compressed using run-length coding. A test architecture employing difference vectors and based on cyclical scan registers (CSRs) is sketched in Figure 1. A drawback of the compression method described in [3] is that it relies on variable-to-fixed-length codes, which are less efficient than more general variable-to-variable-length codes [5, 6]. Furthermore, it is inefficient for cores with internal scan chains that are used to capture test responses; in these circuits, separate CSRs increase hardware overhead.

A more efficient compression and decompression method was used in [7, 8], where T_{diff} was compressed using variable-to-variable-length Golomb codes. However, this approach also requires separate CSRs and is therefore inefficient for cores that use the same internal scan chains for applying test patterns and capturing test responses.

In this paper, we present an improved test data compression/decompression method that makes effective use of Golomb codes and the internal scan chains of the core under test. No separate CSR is required for pattern decompression. 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 [4], but it is also significantly smaller than the compressed test sets obtained using Golomb coding in [7, 8].

We extend the decompression architecture of [7] to an interleaving scheme that allows multiple cores to be tested in parallel with a single ATE I/O channel. Finally, we present analytical results to 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.

2 Compression method and test architecture

We first review Golomb coding and its application to test data compression [5, 7]. The first step in the compression procedure is to derive T_{diff} from T_D, where T_D = $\{t_1, t_2, t_3, ..., t_n\}$, is the (ordered) precomputed test set. T_{diff} is defined as follows: $T_{diff} = \{d_1, d_2, \dots, d_n\} = \{t_1, t_1 \oplus t_2, t_2 \oplus t_3, ..., t_{n-1} \oplus t_n\}$, where a bit-wise exclusive-or operation is carried out between patterns t_i and

[1]This research was supported in part by the National Science Foundation under grant number CCR-9875324.

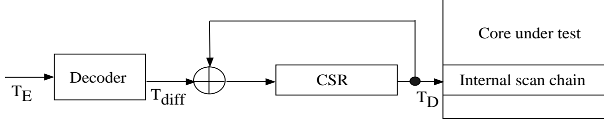

Figure 1. Decompression architecture based on a cyclical scan register (CSR).

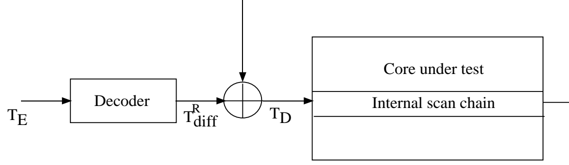

Figure 2. Test architecture based on Golomb coding and the use of internal scan chains.

t_{i+1}. This assumes that the CSR starts in the all-0 state. (Other starting states can be considered similarly.)

The next step in the encoding procedure is to select the Golomb code parameter m, referred to as the group size. If the input data stream is random with 0-probability p, then m should be chosen such that $p = 0.5$ [6]. However, since T_{diff} does not satisfy the randomness assumption, the best value of m for test data compression can only be determined through actual experimentation.

Once m is determined, the runs of 0s in the test data stream 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 0s in T_{diff}. 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 [6]. To each group A_k, we assign a group prefix of $(k-1)$ 1s followed by a 0. We denote this by $1^{(k-1)}0$. If m is chosen to be a power of 2 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.

The proposed method differs from [7] 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^R_{diff}. 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^R_{diff} is now given by: $T^R_{diff} = \{d_1, d_2, \ldots, d_n\} = \{t_1, r_1 \oplus t_2, r_2 \oplus t_3, ..., r_{n-1} \oplus t_n\}$, 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^R_{diff} is shown in Figure 2.

As observed in [7], test data compression is more effective if T_D consists of test cubes containing don't-care bits. In order to determine T^R_{diff} 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 0s and use the logic simulation engine of FSIM [9] to generate the fault-free response r_1.

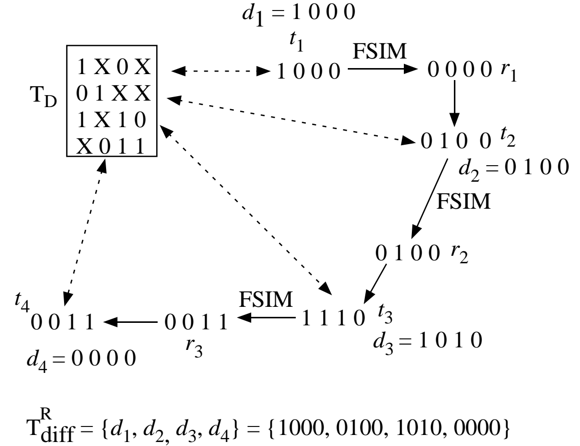

Figure 3. An example to illustrate the procedure for deriving T^R_{diff}.

The problem of determining the best test ordering is equivalent to the NP-Complete Traveling Salesman problem. Therefore, we use a greedy procedure. Suppose a partial ordering $t_1 t_2 \ldots t_i$ 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 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. We continue this process until all test patterns in T_D are placed in the ordered list. Figure 3 illustrates the procedure for obtaining T^R_{diff} from T_D.

For most scan chains in IP cores, the number of inputs driven by the scan cells is not equal to the number of outputs which feed the scan chain. The compression procedure can be easily augmented to handle these cases. However, the details are not presented here due to lack of space.

An on-chip decoder decompresses the encoded test set T_E and produces T^R_{diff}. The exclusive-or gate and the internal scan chain are used to generate the test patterns from the difference vectors. 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 4 flip-flops and 34 combinational gates [7].

3 Analytical results

In this section, we analyze the testing time for a single scan chain when Golomb coding is employed with the test architecture shown in Figure 2. From the state diagram of the Golomb decoder [7], we note that each '1' in the prefix part takes m cycles for decoding, each separator '0' takes

one cycle, and the 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 1s in T^R_{diff}. T_E contains r tail parts, r separator 0s, and the number of prefix 1s 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:

$$\mathcal{T}_{max} = (n_c - r(1 + \log_2 m))m + r + mr$$
$$\mathcal{T}_{min} = (n_c - r(1 + \log_2 m))m + r + \gamma r$$

Hence, the difference between $\mathcal{T}_{max}$ and $\mathcal{T}_{min}$ is given by $\delta\mathcal{T} = \mathcal{T}_{max} - \mathcal{T}_{min} = r(m - \log_2 m - 1)$. We will make use of this result in Section 4.

A major advantage of Golomb coding is that on-chip decoding can be carried out at scan clock frequency f_{scan} while T_E can be fed to the core under test with external clock frequency $f_{ext} < f_{scan}$. 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 [10], and $f_{scan} = mf_{ext}$, where the Golomb code parameter m is usually a power of 2. We now present an analysis of testing time using $f_{scan} = mf_{ext}$.

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 $\mathcal{T}_{ATPG}$ equals pn external clock cycles. Next, suppose the difference vector T^R_{diff} obtained from the uncompacted test set contains r 1s and its Golomb-coded test set T_E contain 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

$$\tau = \mathcal{T}_{max}/f_{scan} = (mn_c - r(m \log_2 m - 1))/f_{scan}$$

and the testing time τ' (seconds) for external testing with ATPG-compacted patterns is given by $\tau = pn/f_{ext} = pnm/f_{scan}$. If testing is to be accomplished in $\tau^\star$ seconds using Golomb coding, f_{scan} must equal $\mathcal{T}_{max}/\tau^\star$, i.e. $f_{scan} = (mn_c - r(m \log_2 m - 1))/\tau^\star$. This is achieved using a slow external tester operating at frequency $f_{ext} = f_{scan}/m$. On the other hand, if only external test is used with the p ATPG-compacted patterns, the required external tester clock frequency f'_{ext} equals $pn/\tau^\star$. Let us take the ratio of f'_{ext} between f_{ext}:

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

Experimental results presented in Section 5 show that f'_{ext} is much greater than f_{ext}, thus Golomb coding allows us to decrease the volume of test data and use a slower tester without increasing testing time.

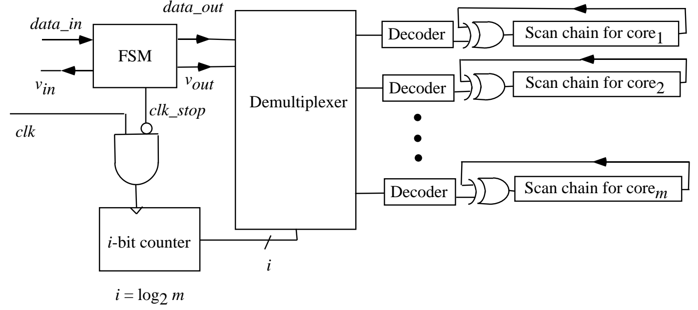

Figure 4. SOC channel selector for application to multiple cores and multiple scan chains.

We next analyze the amount of compression that is achieved using Golomb coding of a precomputed test set T_D. Theorem 1 provides an easy-to-compute bound on the size of the encoded test set T_E. (The proof is omitted due to lack of space.) 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 5.

Theorem 1 *Let the number of don't cares in T_D be n_ϕ. If T^R_{diff} is encoded using Golomb code with parameter m, an upper bound on the length G of the encoded sequence is given by $G \leq n/m + (n - n_\phi) \log_2 m + (n - n_\phi)(1 - 1/m)$.*

4 Interleaving decompression architecture

We next present an interleaving decompression architecture which allows the concurrent testing of multiple cores using a single ATE I/O channel, thereby increasing the ATE I/O channel capacity.

As shown in [7], the FSM in the decoder runs the counter for m decode cycles whenever a 1 is received and starts decoding the tail as soon as a 0 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 (or feed m scan chains) together, such that the decoder corresponding to each core is fed with encoded prefix data after every m cycles. Whenever the tail is to be decoded (identified by a 0 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 a FSM is used for interleaving; see Figure 4.

First, the encoded test data for m cores are combined to generate a composite bit stream T_C that is stored in the ATE. T_C is obtained by interleaving the prefix parts of the compressed test sets of each core, but the tails are included unchanged. 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.

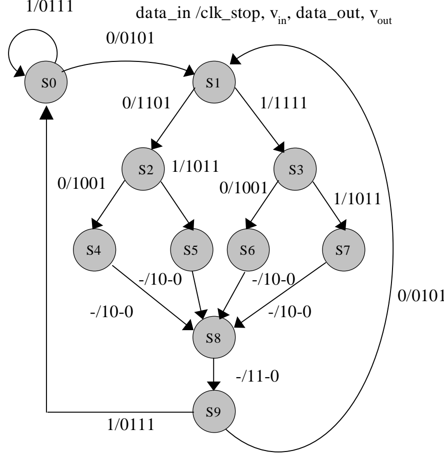

Figure 5. State diagram for the SOC channel selector FSM ($m = 4$).

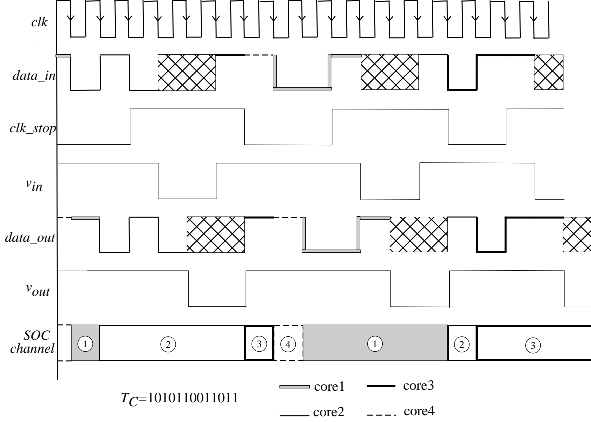

Figure 6. Timing diagram for the SOC channel selector FSM ($m = 4$).

The FSM 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_{in} and v_{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. If the FSM detects that a portion of the tail has arrived, the 0 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 $m = 4$ is shown in Figure 5. The FSM is fed with T_C corresponding to four different cores. It remains in state $S0$ as long as it receives the 1s corresponding to the prefixes. As soon as a 0 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. As shown in the timing diagram of Figure 6, whenever a 0 is received, the SOC channel selection remains unchanged for the next $(1 + m)$ cycles. We developed a Verilog model for the FSM for $m = 4$ and simulated it using $T_C = 1010110011011$. The gate-level circuit obtained using Synopsys Design Compiler consists of only 4 flip-flops and 17 gates.

As discussed in Section 3, 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, i.e. $m \leq 8$, $\delta t \leq 4$. In this case, the decode FSM can be easily modified by introducing additional states to the Golomb decoder FSM of [7] such that the tail decoding always takes m cycles and $\delta t = 0$. For example, only three additional states are required to make tail and prefix decoding cycles equal for $m = 4$.

The additional states do not adversely affect the testing time and the hardware overhead significantly. For m cores, the decoding time t_{tail} for the separator and the tail is given by $t_{tail} = \sum_{j=1}^{m}(r_j + mr_j) = (1 + m)R$, where $R = \sum_{j=1}^{m} r_j$. Since all the prefixes of the cores are decoded in parallel, the number of cycles t_{prefix} required for decoding all the prefixes in T_C is equal to the number of 1s in the prefix of the core with the maximum encoded test data. Therefore, $t_{prefix} = \max_i\{(n_{C,i} - r_i(1 + \log_2 m))m\} = (n_{C,max} - r_{max}(1 + \log_2 m))m$, where $n_{C,i}$ and r_i are the number of encoded bits in T_E and number of 1s in T_{diff} for the ith core respectively, and $n_{C,max}$ and r_{max} are the number of bits in T_E and number of 1s in T_{diff} for the core with the largest encoded test data. Therefore, the total testing time $\mathcal{T}_I$ for m cores using the interleaving architecture is given by

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

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

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

where $|T_C|$ denotes the number of bits in T_C. The difference between the interleaved and the non-interleaved testing times is given by $\mathcal{T}_{NI} - \mathcal{T}_I \approx m(|T_C| - n_{C,max}) \gg 0$, since $n_{C,max} \gg r_{max}$ and $T_C \gg R$.

It is evident from the above analysis that the interleaving architecture reduces testing time and increases the ATE channel bandwidth.

5 Experimental results

In this section, we present experimental results on Golomb coding for the six largest ISCAS 89 benchmark

Circuit	Golomb coding using test patterns only [7]				Golomb coding using test patterns and internal scan chains					Size of Mintest test set (bits)
	Best value of m	Size of test set (bits)	Best compression (percent)	Size of T_E (bits)	Best value of m	Size test set (bits)	Best compression (percent)	Size of of T_E (bits)	Upper bound (bits)	
s9234	4	39273	43.34	22250	4	39750	43.40	22495	30273	25935
s13207	16	165200	74.78	41658	32	186440	81.16	35122	54589	163100
s15850	4	76986	47.11	40717	8	86184	64.51	30581	51150	57434
s38417	4	164736	44.12	92054	4	172458	47.18	91088	134950	113152
s35932*	512	4007299	98.51	59573	2048	4655104	99.42	26885	32355	19393
s38584	4	199104	47.71	104111	8	235280	61.79	89884	120350	161040

*The test set used here is obtained from Atalanta [11]. (The Mintest test set with dynamic compaction is almost fully compacted.)

Table 1. Experimental results on test data compression using Golomb codes.

circuits. We used test cubes (with dynamic compaction) obtained using Mintest [4]. The results shown in Table 1 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. 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 2 demonstrates that Golomb coding allows us to use a slower tester without incurring any testing time penalty. As discussed in Section 3, Golomb coding provides three important benefits: (i) it significantly reduces the volume of test data, (ii) the test patterns can be applied at the scan clock frequency f_{scan} using an external tester that runs at frequency $f_{ext} = f_{scan}/m$, and (iii) 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 2.

6 Conclusions

We have shown that the use of Golomb codes and the internal scan chains of the embedded cores offers significant test data compression for SOCs, leading to reduction in 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. We have shown that 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. We are currently extending the test architecture to ensure that certain patterns are not applied to the core under test due to constraints such as bus contention.

Acknowledgment

The authors acknowledge Prof. H.-J. Wunderlich of University of Stuttgart, Germany for first suggesting the use of the internal scan chain for pattern application.

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

Table 2. Comparison between f_{ext} required for Golomb-coded test data and f'_{ext} required for external testing using ATPG-compacted patterns (for the same testing time).

References

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

[2] V. Iyengar, K. Chakrabarty and B. T. Murray, "Deterministic built-in pattern generation for sequential circuits", *JETTA*, vol. 15, pp. 97–115, August/October, 1999.

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

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

[5] S. W. Golomb, "Run-length encoding", *IEEE Trans. Inf. Theory*, vol. IT-12, pp. 399-401, 1966.

[6] H. Kobayashi and L. R. Bahl, "Image data compression by predictive coding, Part I: Prediction Algorithm", *IBM Journal of R&D*, vol. 18, pp. 164, 1974.

[7] A. Chandra and K. Chakrabarty, "Test data compression for system-on-a-chip using Golomb codes", *Proc. IEEE VLSI Test Symp.*, pp. 113-120, 2000.

[8] A. Chandra and K. Chakrabarty, "System-on-a-chip test data compression and decompression architectures based on Golomb codes", *IEEE Trans. CAD*, vol. 20, March 2001 (accepted for publication).

[9] H. K. Lee and D. S. Ha, "An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation", *Proc. Int. Test Conf.*, pp. 946-955, 1991.

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

[11] H. K. Lee and D. S. Ha. "On the Generation of Test Patterns for Combinational Circuits" *Tech. Report No. 12_93, Dept. Electrical Eng.*, Virginia Poly. Inst. and State Univ.