An Architecture for Combined Test Data Compression and Abort-on-Fail Test

Erik Larsson and Jon Persson
Embedded Systems Laboratory
Department of Computer and Information Science
Linköpings Universitet, Sweden

Abstract[1]

The low throughput at IC (Integrated Circuit) testing is mainly due to the increasing test data volume, which leads to high ATE (Automatic Test Equipment) memory requirements and long test application times. In contrast to previous approaches that address either test data compression or abort-on-fail testing, we propose an architecture for combined test data compression and abort-on-fail testing. The architecture improves throughput through multi-site testing as the ATE memory requirement is constant and independent of the degree of multi-site testing. For flexibility in modifying the test data at any time, we make use of a test program for decompression; only test independent evaluation logic is added to the IC. Major advantages compared to MISR (Multiple-Input Signature Register) based schemes are that our scheme (1) allows abort-on-fail testing at clock-cycle granularity, (2) does not impact diagnostic capabilities, and (3) needs no special care for the handling of unknowns (X).

1 Introduction

The technology development makes it possible to manufacture Integrated Circuits (ICs) where transistor count, transistors per area unit, wiring layers, die size, clock rate and power consumption increase while feature size and voltage decrease. The complex manufacturing leads to low throughput mainly due to the high test data volume needed to ensure fault-free ICs.

The high test data volume leads to long test application times and high ATE (Automatic Test Equipment) memory requirements. Multi-site testing, several devices are tested in parallel, increases throughput but also ATE memory requirement.

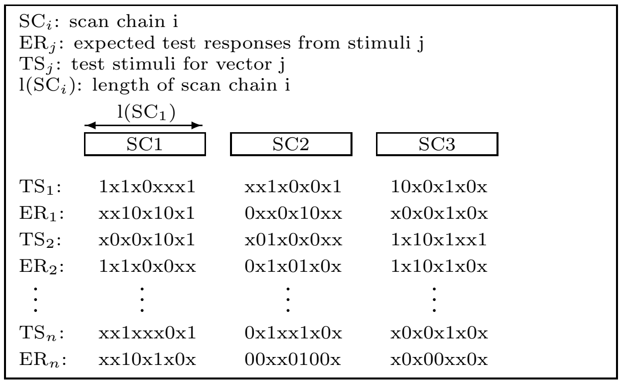
Test scheduling based on abort-on-fail testing, the testing is terminated as soon as a fault is detected, lower the test application times. Larsson *et al.* defined an abort-on-fail scheduling technique where defect probabilities for the testable units are taken into account when ordering the tests such that the expected test time is minimized [1]. Ingelsson *et al.* showed that by using abort-on-fail at finer granularity, the savings in test time becomes significant [2].

Test data compression is useful to address the high test data volume. Several approaches have been proposed. For example, Chandra and Chakrabarty use FDR (Frequency-Directed Run-Length) code [3] and Gonciari and Al-Hashimi [4] use Huffman-coding. Balakrishnan and Touba [5] use matrix operations, and Jas and Touba [6] use an embedded processor for decompression.

These compression schemes make use of MISRs (Multiple-Input Signature Registers) for test response compression (compaction). There are disadvantages with MISRs. First, the usage of time saving abort-on-fail testing is limited as the test result is known only at the end of the testing when the signature is produced. Second, diagnostic capabilities are reduced as it is difficult to determine where the fault is in the DUT (device under test) based on a faulty signature. And finally, when the unspecified bits in the test stimuli are defined, the expected test responses are defined after simulation. Unfortunately, simulation cannot always determine all bits in the test responses to 0 or 1. In these cases the responses are simply *unknown (X)*. A fault-free design may produce a 0 or 1 at such an X-position where both are correct; however, it corrupts the signature in the MISR. Several masking approaches have been proposed; however they become test dependent and cannot guarantee masking all Xs.

In order to address these problems, we propose a MISR-free test compression scheme that supports abort-on-fail testing with termination at clock-cycle granularity and has good diagnostic capabilities. We propose a novel approach to test response compression and make use of a processor, placed on-chip or at the ATE loadboard. The processor receives the compressed test data volume from the ATE, decompresses and applies it to the DUT. The test evaluation is performed on-chip by added test independent logic.

[1]The research is partially supported by the Swedish STRINGENT project.



SC_i: scan chain i
ER_j: expected test responses from stimuli j
TS_j: test stimuli for vector j
$l(SC_i)$: length of scan chain i

$l(SC_1)$

	SC1	SC2	SC3
TS_1:	1x1x0xxx1	xx1x0x0x1	10x0x1x0x
ER_1:	xx10x10x1	0xx0x10xx	x0x0x1x0x
TS_2:	x0x0x10x1	x01x0x0xx	1x10x1xx1
ER_2:	1x1x0x0xx	0x1x01x0x	1x10x1x0x
⋮	⋮	⋮	⋮
TS_n:	xx1xxx0x1	0x1xx1x0x	x0x0x1x0x
ER_n:	xx10x1x0x	00xx0100x	x0x00xx0x

Figure 1: Scan chains and their test data.

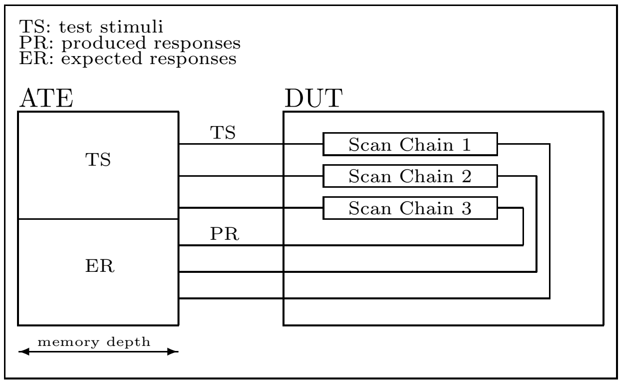


Figure 2: A DUT at test application.

A major advantage with the proposed architecture is that the ATE memory requirement is constant regardless of the degree of multi-site testing. Further, it is straight forward to modify (increase/decrease) the tests as the added evaluation logic is test independent and decompression is performed by a test program running on a processor. The possibility to modify tests is important as the development of tests ranges through stages as rapid silicon prototyping, first silicon, volume production, and in-field test. Unavoidable re-spins, introduction of new technologies as well as moving the production to a new site makes it important to have the possibility to modify the tests.

In order to verify the architecture, we have implemented the facsimile compression algorithm and made experiments on ISCAS designs and an industrial design. The test data volume for the industrial design with 87% unspecified bits was compressed 69%. The test decompression program is only 88 bytes in size; hence when test is to start, low loading time is needed.

The paper is organized as follows. The prior architecures are outlined in Section 2 and the proposed architecture is described in Section 3. The results are in Section 4 and conclusions are in Section 5.

2 Prior Test Architectures

For illustration we use an IC with three scan chains where the number of scan-chains and their length along with test data are given, see Figure 1. The test data volume, test stimuli (TS) and expected test responses (ER), contains specified bits (0 and 1) and unspecified bits (x).

2.1 Architecture without Compression

Figure 2 shows a typical setup when compression is not used. Prior to test application, the test data volume, TS and ER, is arranged in the ATE to correspond to the scan chain configuration in the DUT. Figure 3 shows the arrangment of TS_1 (the first test stimulus) in the ATE for the DUT in Figure 2.

At test application, TS are fed to the DUT and the PR (produced test responses) are sent to the ATE to be compared with ER. The comparison between PR and ER can be done at clock cycle granularity as PR are produced each clock cycle but at the capture cycles. Hence, the setup is suitable for abort-on-fail testing as it is possible to terminate the testing at clock-cycle granularity. Further, the setup is suitable for diagnosis. In the case of a fault, it is possible to determine where in the DUT the fault is.

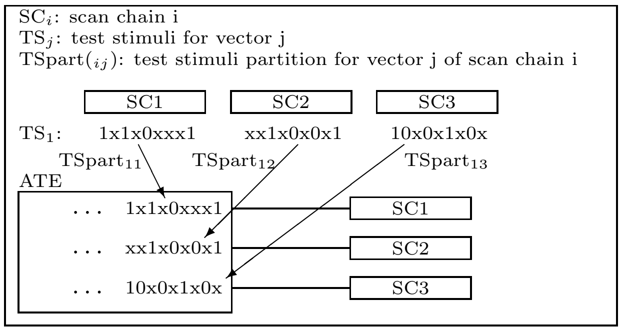
Figure 2 shows single-site testing; a single DUT is tested at a time. In order to increase the throughput, several DUTs can be tested concurrently. However, the ATE memory requirement will then increase. If TS and ER are duplicated for each n degree of multi-site test, the memory requirement will be $n \times (TS + ER)$. If TS is broadcasted to n devices, the ATE memory requirement is $TS + n \times ER$. The memory requirement increases linearly with the degree of multi-site test (n).

2.2 Architecture with Compression

The common approach to test data compression is to fill the high number of unspecified bits (x) in the TS such that high compression ratio is reached. Once the unspecified bits in the TS are defined, the unspecified bits in ER are determined through simulation and then the on-chip response compressor, the MISR, can be designed for the defined ER.

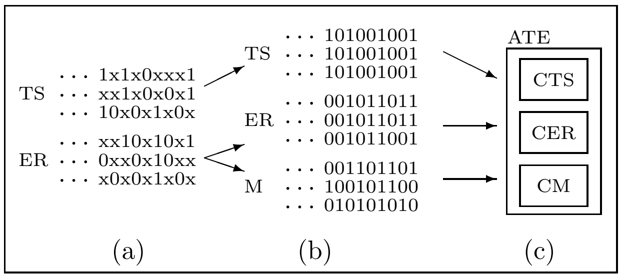
At test application, the added decompression hardware receives CTS (compressed test stimuli) from the ATE and produces DTS (decompressed test stimuli), which are fed to the DUT. PR are compressed into a MISR. At the end of the testing, the MISR signature is shifted out and compared to the expected signature stored in the ATE.

Disadvantages are that test result is only known at the end of the testing, when the signature is produced; hence the usage of time saving abort-on-fail testing is limited. Further, the diagnostic capabilities are lim-



SC$_i$: scan chain i
TS$_j$: test stimuli for vector j
TSpart$(_{ij})$: test stimuli partition for vector j of scan chain i
SC1
SC2
SC3
TS$_1$: 1x1x0xxx1 xx1x0x0x1 10x0x1x0x
TSpart$_{11}$
TSpart$_{12}$
TSpart$_{13}$
ATE
... 1x1x0xxx1
... xx1x0x0x1
... 10x0x1x0x
SC1
SC2
SC3

Figure 3: ATE arrangement for test stimuli TS$_1$.



TS ... 1x1x0xxx1 ... xx1x0x0x1 ... 10x0x1x0x
ER ... xx10x10x1 ... 0xx0x10xx ... x0x0x1x0x
TS ... 101001001 ... 101001001 ... 101001001
ER ... 001011011 ... 001011011 ... 001011001
M ... 001101101 ... 100101100 ... 010101010
ATE
CTS
CER
CM
(a)
(b)
(c)

Figure 4: Test data handling: (a) arranging test data for ATE, (b) unspecified bit filling, mask definition, and (c) compressed test data in ATE.

ited as it is; in case of a faulty signature, difficult to determine the fault location and which TS detected the fault. And finally, if an unknown (X) is produced; it might corrupt the MISR signature. Additional logic can be added; however, only a limited number of Xs can be handled.

3 Proposed Test Architecture

The proposed architecture allows, in contrast to prior architectures, integrated test data compression and abort-on-fail testing. It handles any number of Xs, does not limit diagnosis, and has a constant ATE memory requirement that is independent on the degree of multi-site testing.

3.1 Test Preparations

The preparation prior to application is outlined in Figure 4 using the example in Figure 1. First the test data, TS and ER, are arranged for the ATE to fit the scan-chains (Figure 4(a)). The unspecified bits in TS are filled to result in high compression (Figure 4(b)). The way we handle TS is similar to previous compression schemes.

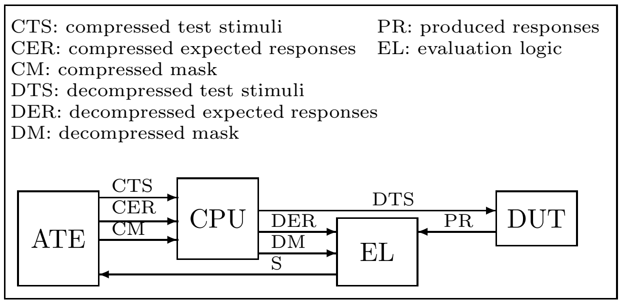
Different from previous approaches is our handling of ER. Instead of compressing PR on-chip, we compress ER and store it off-chip in the ATE. In order to find a scheme that allows arbitrary filling of the bits in ER, we introduce a mask (M), see Figure 4 (b).

We explain M. For each bit in ER there is a corresponding bit in M (Figure 4 (b)). Initially all bits in M are set to 0. For any specified bit (0 or 1) in ER, the corresponding bit in M is set to 1.

Finally, TS, ER and M are compressed into CTS (compressed test stimuli), CER (compressed expected test responses), and CM (compressed mask). CTS, CER and CM are all stored in the ATE (Figure 4(c)).

3.2 Test Application

The scheme at test application is outlined in Figure 5. Given is an ATE where CTS, CER, and CM are stored. At test application, CTS, CER, and CM



CTS: compressed test stimuli
CER: compressed expected responses
CM: compressed mask
DTS: decompressed test stimuli
DER: decompressed expected responses
DM: decompressed mask
PR: produced responses
EL: evaluation logic
ATE
CTS
CER
CM
CPU
DTS
DER
DM
S
EL
PR
DUT

Figure 5: Illustration of the scheme at test application where the DUT is as in Figure 2.

are sent and decompressed by a test program running on a processor.

The processor is an on-chip processor or a processor placed on the ATE loadboard. The processor is used for decompression and must be tested prior to testing the DUT. In the case of an on-chip processor it is tested first, and in the case of a processor on the loadboard it is tested once for the batch of DUTs (all dies) that are to be tested.

The test program takes CTS, CER, and CM as inputs and produces DTS (decompressed test stimuli), DER (decompressed expected responses) and DM (decompressed mask). The DTS are sent and applied to the DUT and the PR (produced responses) are received by added EL (evaluation logic). The EL hardware receives as input the PR, DER and DM and produces a signature (S) that indicates if a fault is detected or not (0 if fault-free, 1 if fault). In the fault-free case, the test process proceeds; however, in the case of a detected fault, the testing can be aborted, and if desirable, the response bits can be shifted out for diagnostics.

The test evaluation logic (EL) in Figure 5 is detailed in Figure 6. For each produced test response bit, PR_i, the corresponding decompressed mask bit DM_i, and

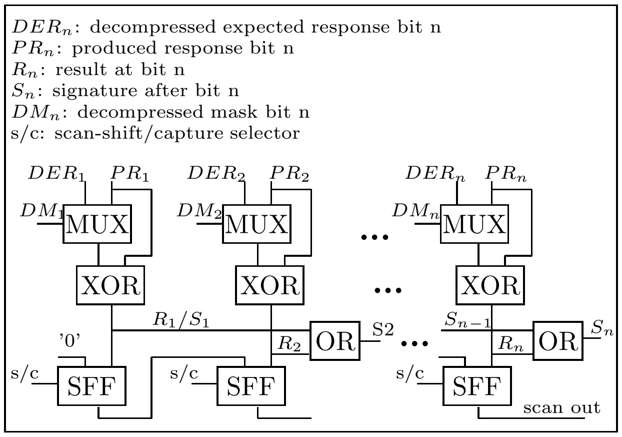

DER_n: decompressed expected response bit n
PR_n: produced response bit n
R_n: result at bit n
S_n: signature after bit n
DM_n: decompressed mask bit n
s/c: scan-shift/capture selector
DER_1 PR_1 DER_2 PR_2 DER_n PR_n
DM_1 DM_2 DM_n
MUX
XOR
OR
SFF
R_1/S_1
'0'
R_2
S2
S_{n-1}
R_n
S_n
s/c
scan out

Figure 6: The test evaluation logic.

decompressed expected response bit DER_i defines a result bit R_i. R_i is stored in an added scan flip-flop and S_i is used to produce the signature.

Table 1 contains the truth table for the evaluation of a response bit where the comments are:

A: DM_i= 0 and PR is masked

B: PR=ER - no fault

C: PR $\neq$ ER - a fault is detected

At each clock cycle the n evaluation bits are combined by or-operation into a single bit signature that indicates pass or fail. In our example with three scan chains there are three bits produced each clock cycle. Assume at a given clock cycle the PR corresponding to the initial ER "x1x" are "010". The initial ER is filled such that when decompressed it is "110". Even though ER "110" and PR "010" are different, the mask, which for ER "x1x" is "010" will ensure that only the middle bit is used for test evaluation, and as the middles bits in expected responses and produced responses are equal there is no fault. The mask will ensure that only specified bits in ER are compared with PR.

In the case of a fault, it is possible to terminate the testing (abort-on-fail) at clock-cycle granularity or terminate the testing for diagnosis.

In the case of diagnosis, the position of a fault is stored in the added scan flip-flops, and by shifting out the response bits stored in EL, it is possible to identify at which scan-chain the fault appeared.

Note, as the mask is used to identify the initially specified bits in ER the problem with X-handling; the simulation cannot define if an output should be 0 or 1, is solved. Only specified bits in the ER are used for test evaluation as the evaluation logic masks away unspecified bits using the mask data.

The overhead is for each bit an added MUX, an XOR, an OR (but for the first bit) and a scan flip-flop (SFF). If n bits are produced as test responses, n MUXes, n XORs, $n-1$ ORs and n SFFs are needed. The hardware cost is linear to n (number of produced response bits per clock cycle). Note, that the EL is test independent; hence the order of tests and the tests themselves can be modified at any time.

Finally, as test evaluation is performed on-chip, the test data stored in the ATE, CTR, CER, and CM, can be broadcasted to n parallel DUTs; hence the memory requirement is constant in the case of multi-site testing.

DER_i	PR_i	DM_i	S_i	**Comment**
-	-	0	0	Comment A
0	0	1	0	Comment B
1	1	1	0	Comment B
0	1	1	1	Comment C
1	0	1	1	Comment C

Table 1: Truth table for DER$_i$ (decompressed expected response), PR$_i$ (produced response), and DM$_i$ (decompressed mask) at TAM wire i.

4 Experimental Results

The objectives with the experiments are (1) show that our scheme produces similar compression ratio as previous approaches on stimuli compression, (2) demonstrate that good compression can be achieved when stimuli, expected responses including the mask are compressed, and (3) illustrate ATE memory savings with the proposed scheme at multi-site test.

For the experiments, we have made use of the ISCAS circuits and one industrial design. The efficiency of data compression is computed as:

$$\text{Compression (\%)} = \frac{\text{Original Bits - Compressed Bits}}{\text{Original Bits}} \times 100.$$

The proposed scheme assumes a test program (software) for decompression. We have in this paper made use of facsimile compression algorithm (facsimile coding standard is the ITU-T Group 3 standard) [8]. The basic idea is that a line on a printed paper is similar to the line just above and therefore only the difference is sent. A dot on the paper is coded to be either white or black, also known as Bi-Level images.

We made an analysis on the test data to find the most suitable coding words as the codewords in the facsimile standard are based on paper copies characteristics. Further, as the order in which the test vectors are applied does not impact the test result, we order the test data volume to achieve better compression. The size of the decompression program is 88 assembly instructions only. First, we compare test stim-

Circuit	FDR [3]		Matrix [5]		Linear [7]		Prop. Scheme	
	Comp. bits	% Comp.	Comp. bits	% Comp.	Comp. bits	% Comp.	Comp. bits	% Comp.
s13207	30880	81.30	33470	79.99	9920	94.44	25204	84.68
s15850	26000	66.22	23552	67.88	11168	87.65	19832	66.54
s38417	93466	43.26	69556	56.00	30432	82.58	66428	54.11
s38584	77812	60.91	66838	65.15	30208	84.25	72103	56.80
s5378	12346	48.02	10390	59.20	5696	81.39	11005	48.57
s9234	22152	43.59	16888	53.49	9280	74.27	14209	48.17

Table 2: Comparing test stimuli compression.

Circuit	Size (bits)	Stimuli only		Size (bits)	Responses only	
	(stimuli only)	Comp. bits	% Comp.	(responses only)	Comp. bits	% Comp.
s838	5092	2218	56.44	2584	789	69.47
s9234	27417	14209	48.17	27750	12815	53.82
s38584	166896	72103	56.80	166896	72095	56.80
s13207	164500	25204	84.68	185650	28936	84.41
s15850	59267	19832	66.54	66348	23676	64.32
s5378	21400	11005	48.57	22800	10860	52.37
s35932	21156	3502	83.45	24576	3553	85.55
s38417	144768	66428	54.11	151554	71640	52.73

Table 3: Separate test stimuli compression and expected test responses compression.

Circuit	Original test data volume (stimuli and expected responses) (bits)	Compressed test data volume (stimuli, expected responses, and mask) (bits)	Compression ratio (%)
s838	7676	3979	48.16
s9234	55167	40754	26.13
s38584	333792	238922	28.42
s13207	350150	117236	65.52
s15850	125615	72680	42.14
s5378	44200	36556	17.29
s35932	45732	10434	77.18
Industrial design	14056068	4398738	68.71

Table 4: Combined compression of test stimuli and test responses.

uli compression to demonstrate that facsimile compression produces compression ratio similar as previously proposed compression techniques. We compare against the methods by Chandra and Chakrabarty [3], Balakrishnan and Touba [5] and Balakrishnan and Touba [7]. The results collected in Table 2 show that facsimile produces results in similar range as previous approaches.

Second, we compared the compression ratio of test stimuli compression and expected test response compression. The objective is to demonstrate that similar compression ratio can be achieved for compression of expected test responses as for compression of test stimuli. We made experiments where stimuli as well as responses are compressed separately and the result can be found in Table 3. Column two gives the original size of the test set, column three and four gives the compressed size and the percentage. Column five through seven shows the same for the responses. The results confirm that compression of expected test responses reaches similar compression ratio as test stimuli compression.

Third, we demonstrate the proposed scheme where the test stimuli, expected test responses and the mask are compressed. The results are collected in Table 4. The second column shows the original test data volume (test stimuli and test responses). For each design the compressed data size and the compression ratio (percentage) in relation to the initial test data volume are shown. Note that original test data volume includes only the original test stimuli and expected test responses, and not the mask data, while the compressed test data includes test stimuli, test responses, and mask data.

Fourth, we show the ATE memory requirement at n-degree multi-site test. We compare (1) a standard approach that duplicates TS and ER based on the degree (n) of multi-site test, (2) a broadcast approach that duplicates ER only, and (3) the proposed approach. From Figure 7 the memory saving becomes clear using the proposed approach has it has constant ATE memory requirement independent of the degree of multi-site test.

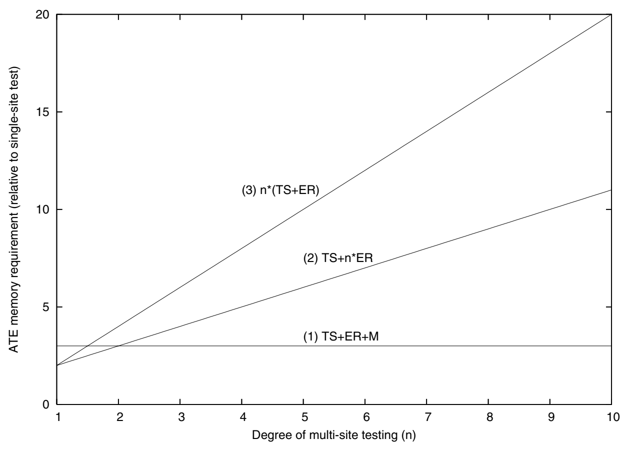


Figure 7: Relative ATE memory requirement at multi-site (n) for the approaches (1) proposed, (2) stimuli broadcast, and (3) standard.

5 Conclusions

Low throughput when testing Integrated Circuits (ICs) is a problem mainly due to the increasing test data volume. High test data volume leads to long test times and high ATE memory requirements. Multi-site testing can increase throughput; however, ATE memory requirement increases. The long test application times can efficiently be addressed by making use of abort-on-fail testing; the testing is terminated as soon as a fault is detected. The high test data volume can be addressed by test data compression; however, compression techniques prohibit the usage of time saving abort-on-fail testing.

The paper proposes an architecture that allows test data compression and abort-on-fail testing. We make use of a novel way to handle test responses. The compressed test data volume is stored on the ATE and fed to a test program that decompress test data and feds the device under test. Added test independent evaluation logic determines at clock cycle granularity the result. Advantages are that the technique does not limit diagnosis, handles any number of unknowns (X-tolerance), and needs constant ATE memory; independent of the degree of multi-site testing.

The experiments on ISCAS designs and one industrial results show that the proposed scheme performs well compared to previous compression schemes in terms of compression ratio.

References

[1] E. Larsson, J. Pouget, and Z. Peng, "Defect-Aware SOC Test Scheduling," in *VLSI Test Symposium*, 2004, pp. 359–364.

[2] U. Ingelsson, S. Goel, E. Larsson, and E. J. Marinissen, "Test Scheduling for Modular SOCs in an Abort-on-Fail Environment," in *European Test Symposium*, 2005, pp. 8–13.

[3] A. Chandra and K. Chakrabarty, "Test Data Compression and Test Resource Partitioning for System-on-a-Chip Using Frequency-Directed Run-Length (FDR) Codes," *Transactions on Computers*, vol. 52, no. 8, pp. 1076–1088, Aug 2003.

[4] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-Chip Test Data Compression/Decompression," in *Design, Automation and Test in Europe*, March 2002, pp. 604–611.

[5] K. J. Balakrishnan and N. A. Touba, "Matrix-Based Test Vector Decompression Using an Embedded Processor," in *Defect and Fault Tolerance in VLSI Systems*, Nov 2002, pp. 159–165.

[6] A. Jas and N. A. Touba, "Deterministic Test Vector Compression/Decompression for Systems-on-a-Chip Using an Embedded Processor," *Journal on Electronic Testing: Theory and Applications (JETTA)*, vol. 18, no. 4/5, pp. 503–513, Aug 2002.

[7] K. J. Balakrishnan and N. A. Touba, "Deterministic Test Vector Decompression in Software Using Linear Operations," in *VLSI Test Symposium*, April-May 2003, pp. 225–231.

[8] S. Kahlid, *Introduction to Data Compression*, 2nd ed. Morgan Kaufmann Publishers, 2000.