

Logic BIST Diagnostics Using Simple Synchronised MISR Unload

Chris Hill and Thomas Rinderknecht

Design For Test Group of Mentor Graphics Corporation

8005 SW Boeckman Road, Wilsonville, Oregon 97070

Abstract - We present a commercial logic BIST diagnostic approach; using an enhanced BIST controller that supports simple synchronised observation of the MISR on a per pattern basis, reinitialising the BIST controller. Selected patterns are unloaded and diagnostic inference identifies candidate faults and their locations. The approach is flexible and can be tailored to fit constraints of the test environment. At-speed BIST is supported, as is diagnostics through a low speed interface, such as IEEE 1149.1.

I Introduction

This paper presents a pragmatic approach to the extraction of diagnostic information from a failing logic BIST session. Our approach uses an innovative diagnostic mode within the BIST controller to expedite the search for multiple input signature register (MISR) values that observe faults from the design under test (DUT). This removes the need to perform complex searches to identify failing BIST patterns.

Logic BIST is mostly used for system test. However, as the hardware is there it can be used as part of the manufacturing test flow too[12]. Being able to locate the source of failures is important during manufacturing test, it can also render useful feedback on reliability issues when system test fails.

Logic BIST[1][2] provides a compact pass/fail signature[3] value. Much has been written about extracting diagnostic information from a BIST run[3][4][5][6][7][8][9][10][11]. The weakness of these solutions is their hardware overhead, behaviour in the context of multiple faults in the DUT or the need for multiple BIST runs to extract diagnostic information.

A new approach is proposed by Wohl[13], using STIL[14][15] and interval unloads. Our diagnostic approach can be used in a similar manner. However, we are able to pre-screen directly to failing patterns without significant overhead. Also, our BIST controller diagnostic mode provides synchronisation enabling diagnostics to be performed when BIST is being clocked from a free running clock, such as an on chip clock generator (at speed).

II BIST Controller Architecture for Diagnostics

Our BIST controller follows the STUMPS[16] architecture and has a state machine that controls the BIST session. The

BIST controller is designed for use from an 1149.1[17] compliant TAP interface. We provide a two register logical interface. A *selection* register controls the internal register accessed through the *data* register port. So, to interact with the value of MISR one must load the *selection* register with MISR_SELECT value and then access the data register. Both the *selection* register and the *data* register are fully shadowed and provide synchronisation between the slow clock domain of the logical interface and that of the BIST controller. The exact implementation of this interface is beyond the scope of this paper. Our BIST controller supports scan chain concatenation, presenting the many short DUT scan chains as a smaller number of longer ones. It can also be configured to present a single concatenated scan chain for TAP access. The BIST controller is designed to run from an at-speed clock, it provides internal clock control and is capable of generating complex at-speed capture clock waveforms and a slow speed shift clock.

There are two novel features that provide direct support for the diagnostic process. Our BIST controller can be initialised to run a window over any part of the overall BIST session. When in diagnostic mode the BIST session supports synchronisation points, sensitive to activity on the controller's physical interface.

A Restartable BIST Controller

Rather than using masking to protect the MISR from being polluted by the unknown scan chain contents sampled during the first pattern of a window we assert a hold control on the MISR.

When performing a BIST run over a window we preload the MISR with the predicted initial value for the start of the window and initialise the pseudo random pattern generator (PRPG), the pattern counter and stop value for the window. At the end of a fault free window the MISR will have the expected fault free value. The pattern counter start and stop values are used, rather than just a count of the number of patterns as the absolute pattern number controls the selection of capture clock behaviour and test point activation. The MISR, PRPG and DUT scan chain values can be computed once, at BIST insertion time.

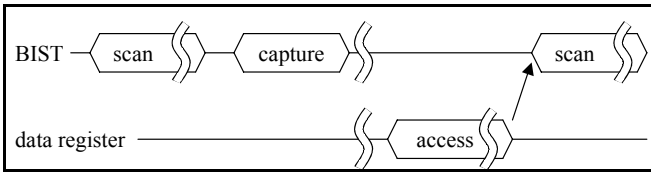


Fig. 1. BIST and data register synchronisation

B Diagnostic Synchronisation Points

When the BIST controller is in diagnostic mode it waits, at the end of each BIST pattern, for the logical interface to perform a data register load/unload shift sequence, as shown in Fig. 1. Once the sequence has been performed the BIST controller continues to the next pattern. It is this waiting that provides synchronisation between the execution of the BIST patterns and activity occurring externally that is gathering the information for the diagnostic process.

These synchronisation points allow diagnostics to be performed from the external interface to the BIST controller when the BIST process is being clocked at-speed from a free running clock source, such as a phase locked loop (PLL) clock generator. All that is required externally is to wait sufficient time for the next synchronisation point to have been reached, no other external handshaking is required. This makes it possible to consider diagnostics of an embedded DUT, where only an 1149.1 TAP is available.

Using this diagnostic mode we are able to load/unload the MISR and observe it, reloading it with the fault free value. Hence the MISR value observed for each BIST pattern is made independent and no adjustment is required after a failing MISR value has been observed.

Another way to use the diagnostic mode is to unload the contents of the DUT scan chains, which will contain the values that would be observed in the MISR after the next BIST pattern had unloaded them there. So, it is possible for a given range of BIST patterns to fully unload and observe the DUT scan chains. Given sufficient time and storage capabilities a fully exhaustive unload run of the actual BIST patterns could be performed. Note, in this scenario no load data is specified to fill the scan chain contents, as they are fully refilled when the next BIST pattern is triggered. Also, the MISR value is ignored

III Diagnostic Process

Fig. 2 shows the steps in the process. These steps are described below:

1. Run the BIST controller to perform the BIST session

At BIST creation time test vectors to run the full BIST session and observe the MISR value are created. These vectors are used. A failure observe the expected MISR value indicates that the DUT contains observable faults.

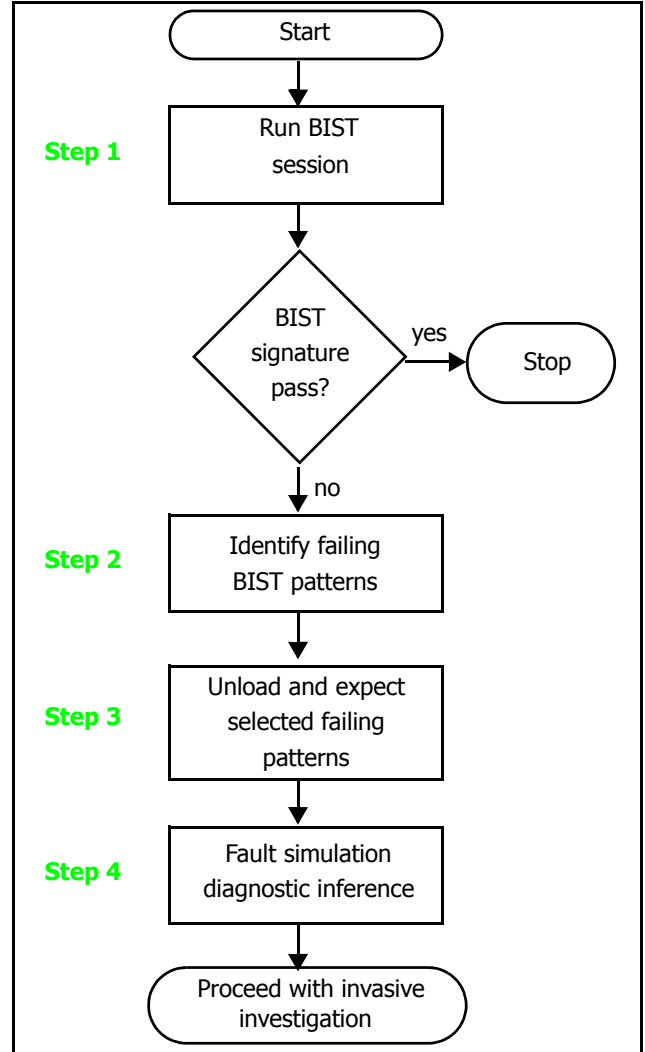


Fig. 2. Four Step Logic BIST Diagnostic Process

These full BIST session vectors are compact in nature and perform the operations as shown in the pseudo code below:

```

Reset BIST controller
Initialise BIST controller
Start BIST run
Wait for BIST run to have completed
Observe BIST signature and verify it is
expected value

```

2. Identify the failing BIST patterns

Use the BIST controller in diagnostic mode to identify the BIST patterns that cause the MISR to observe a faulty value. A second set of test vectors, generated at BIST creation time, run the BIST in diagnostic mode. These vectors initialise the BIST controller and then perform the actions described in the pseudo code below:

```

Reset BIST controller
Initialise BIST controller
For each BIST pattern
    Wait for 1 BIST pattern to complete
    Unload MISR and reload expected value
    Verify BIST signature is expected

```

These vectors are compact, the bulk of them only contain expected MISR values. If this set of vectors is too large for the test environment it can safely be split into a number of subsets. Just like conventional ATPG vector based ATE application these failing pattern discovery vectors can be prematurely terminated once a sufficient[21] quantity of failing patterns have been discovered.

3. Unload failing pattern contents

This step requires a third set of vectors, these can be swiftly generated once the failing BIST pattern set has been identified. This step uses the failing patterns, or a subset of them, and a selection of expected passing patterns. The vectors of this step use the BIST controller to initialise the scan chains within the core under test and perform the BIST capture cycle. Then, the vectors unload the scan chain contents and comparing them to the expected scan chain contents for the pattern. This is shown in the pseudo code below:

```
Reset BIST controller
For each failing BIST pattern
    Initialise BIST controller pattern
    Wait for BIST pattern to complete
    Unload core scan chain contents
    Verify core scan chain against
    expected value
```

For a large number of failing BIST patterns this vector set may become sizable. Since diagnostic inference can produce reliable deductions with a small number of failing patterns, it is expected that only a subset of failing BIST patterns will be unloaded for further analysis.

In order to ensure that the BIST controller is operating correctly the vector set contains a number of significant known passing patterns, within the scan chain integrity test¹ and each of the MTPI phases. If these patterns are not unloaded correctly then it is most likely that a defect within the BIST hardware is preventing correct operation of the BIST.

Access to the DUT scan chains is through either the concatenated single scan chain or the DUT scan chains presented at the level of the BISTed DUT. When the BIST controller is in diagnostic mode and waiting for synchronisation the scan chains are held in scan mode. Clocking the BIST controller's `diag_clk` input causes the scan chain to shift once.

4. Diagnostic Inference

Take the diagnostic failure map and logic BIST pattern set into fault simulation and run diagnostic inference. This final part of the process is identical to the step that would be performed during conventional diagnostic analysis using functional or structural vectors, rather than the logic BIST patterns.

1. During these patterns the scan enable input is held high during the capture cycle, so that scan shifting problems can easily be identified.

A Static vs Dynamic Vector Generation

The process described above requires the generation of test vectors to perform the DUT scan chain unload step, for the BIST patterns that cause the MISR to observe a failing value. This may not be convenient, if for example the BIST pattern, PRPG and MISR database is not available at this time. In such circumstances a more conventional window based search of the BIST pattern space and unload of the DUT scan chain contents for the failing windows can be employed. The vector sets for this operation can be created at BIST creation time. In this case step two and three of the process above would be replaced with:

2. Identify the failing BIST pattern regions

Divide the BIST pattern space into a number of regions². Run the BIST controller to locate the BIST pattern space regions that cause MISR signature to diverge.

This second collection of vector sets is static in nature and can be generated during the BIST controller generation process. Each of these vector sets are very compact, containing only initialisation, partial run and a MISR signature unload. Their behaviour is as shown in the pseudo code below:

```
Reset BIST controller
Initialise BIST controller for window
Run BIST controller for window
Unload MISR value
Verify expected MISR value for window end
```

The size of the region is determined by the capabilities of the ATE and the limitations placed on the unload described in the next step. If a BIST pattern region is found to fail, producing a mismatch with expected signature when the vectors are run, then it can be added to the collection of windows for further investigation.

3. Unload failing pattern regions

Use the BIST controller to initialize the scan chains in the core under test with each BIST pattern in the failing region and unload and compare the scan chains with expected values.

The vectors of this step use the BIST controller to initialise the scan chains within the core under test and perform the BIST capture cycle. Then, the vectors unload the scan chain contents and comparing them to the expected scan chain contents for the pattern. This is shown in the pseudo code below:

```
Reset BIST controller
For each failing BIST pattern
    Initialise BIST controller for pattern
    Wait for BIST pattern to complete
    Unload DUT scan chain contents
    Verify DUT scan chain
```

The vectors required for this step are static in nature and can be generated during BIST controller generation proc-

2. These BIST pattern regions should be sized to fit the constraints of the ATE.

ess. The previous step identifies exactly which vector sets, from the sets for all regions of the BIST pattern space, need to be exercised on the ATE. Normally sufficient diagnostic information will be contained within the first few failing BIST pattern regions.

As previously, in order to ensure that the BIST controller is operating correctly one would like to observe at least some passing patterns unloaded from the failing BIST pattern regions. If these patterns are not encountered one can not be certain that the failures are not caused by a defect within the BIST controller.

B Using 1149.1 TAP

The BIST controller does not use the 1149.1 RUNBIST instruction. A BIST session is controlled directly from the data registers within the BIST controller. To initiate a BIST session one needs only to load the `RUN_BIST` value into the BIST controller's *selection* register. When this value is in the *selection* register the MISR is accessible through the *data* register. The BIST session will run through to completion, or until the next diagnostic synchronisation point, unless the value in the *selection* register is changed. Once enabled, operation of the BIST is independent of the TAP.

When running in diagnostic mode all that is needed to synchronise to the BIST controller is a suitable delay, so that by the time the TAP reaches the Capture-DR state, with the BIST *data* register access instruction loaded, the BIST controller has reached the next synchronisation point. Passing through the Update-DR state triggers the next BIST pattern. If accessing the DUT scan chain contents, through the single scan chain, one must force a visit to Updated-DR TAP state for BIST controller's *data* register. A fake access to the MISR, will accomplish this.

IV Advantages of this Approach

A Using BIST vectors

During the diagnostic process we use the BIST hardware to create and apply each pattern to the DUT. Observation is then done either using the MISR or the directly through the DUT scan chains. In this way any fault observable to the BIST can be diagnosed and the full capability of the BIST hardware to exercise the faults is available, including at-speed clock control and complex capture windows. No complex test environment is required for the unit under test. No diagnostic resolution is lost when a failing part is located on an embedded subsystem with limited access, through 1149.1 TAP or some other test interface.

B Flexibility

Using the restartable nature of the BIST controller one can perform a traditional search of the BIST pattern space, using independent windows, for the failing BIST patterns. Using the diagnostic synchronisation mode of the BIST controller

allows easy observation of the MISR value or DUT scan chain contents for a BIST window of BIST patterns.

Both of these capabilities put together allow one to tailor one's diagnostic approach to the exact constraints of the test interface and test environment being used at diagnostic time. A full BIST run can be used with diagnostic synchronisation and MISR unload to determine the BIST patterns that observe failures in the MISR. These failing patterns can then be rerun and the actual DUT scan chain contents observed.

If the constraints of the test access method are exceeded by this direct approach then the BIST pattern space can be divided into windows and the approach used within these windows. Kapur[18] suggests using non uniform window sizes, where the window is sized to include the a similar number of new fault detects. Our overall BIST architecture and the use of multi-phased test point insertion (MTPI)[19] would suggest that initial window sizes coincident with the test point phase size should be considered.

C Leveraging ATPG Diagnostic Inference

Diagnostic inference, from regular scan based test vectors, is a mature art. Our diagnostic approach leverages on this capability. Once a set of diagnostic unload vectors have been exercised using the BIST hardware and the observed failures reported this failure information is mapped into a format that can be taken directly into fault simulator based diagnostic inference, with respect to the original BIST patterns. This process then produces the candidate fault sites for the observed failures.

As our approach uses the actual DUT core scan chain values captured by the failing BIST patterns there is no diagnostic resolution lost. The same resolution is available as would be if the BIST patterns and capture procedures had been applied directly to the DUT. Our approach is effective irrespective of the number of faults within the DUT. The entire diagnostic approach is independent of the fault model being used. It works just as well for stuck-at-faults as for transition-faults. Being that we are aiming to make the BIST patterns as diagnosable as conventional ATPG patterns the difference between fault models is a matter for fault simulation alone.

D Vector Volume

BIST runs tend to require a large volume of pattern data, given the pseudo random nature of that data. Compare this to the volume of test data required for a similar fault coverage when using deterministic ATPG vectors. As an example we took a design of 32,000 gates (6,000 scan elements) with 342 inputs and 208 outputs. ATPG was able to reach 98.2% fault coverage in just 383 scan patterns. This represents a total vector volume of $383 \times (342 + 208 + 6000 \times 2) = 4806650$ symbols.

To achieve 97.0% fault coverage with BIST required the use of 4,096 BIST patterns. In this example we used a 64 bit

MISR, hence the controller has a 64 bit data register. To run the entire BIST session requires the scan loading of 5 BIST data register values and the final observation of the MISR. This represents $5 \times (64 + 4) + 64 = 409$ symbols. The MISR unload patterns for the entire BIST run would require the same amount of initialisation data then the observation of 4095 MISR values, this represents $5 \times (64 + 4) + 4095 \times 64 = 262420$ symbols. Finally, unloading the DUT scan chains for a mixture of 64 failing and passing patterns would require $64 \times (5 \times (64 + 4) + 6000) = 405760$ symbols. This represents a reduction of the diagnostic data volume by approximately 90%. Compare this to the requirement of ordinary interval unloads, using a window size of 256 patterns, which would require $\frac{4095}{256} \times (5 \times (4 + 64) + 64) \approx 6462$ symbols to evaluate all of the windows and then $5 \times (4 + 64) + 256 \times 6000 = 1536340$ symbols to fully unload each failing window, where each unloaded window will render at least one failing pattern. To obtain the first failing pattern would require about twice the data volume as our approach.

E Aliasing

Aliasing of MISR values, a sequence of faulty DUT scan cell values causing the MISR to observe a fault free value is a well understood and studied problem in BIST[20]. Given that the MISR is proportioned to have an acceptably low probability of aliasing over the entire BIST pattern space the probability of aliasing occurring within the compression of a single BIST pattern is further reduced.

Typically a single fault will be observed in multiple failing BIST patterns and hence multiple failing MISR values. The probability of aliasing occurring for each MISR value that would otherwise observe the failure rapidly tends towards zero, because of the conjunction of the respective single pattern aliasing probabilities.

F Failure Mapping

Presently there is no standard format for ATE failure information to be passed off to other tools. Each test environment produces a unique representation of any captured failure information. This problem is currently solved, on an ad hoc basis, by those performing diagnosis using regular functional or structural test vectors. A mapping must be made from the ATE failure information back into the domain of the source test patterns, suitable for consumption by fault simulation for diagnostic inference.

With BIST diagnostics this mapping must be made back not to the applied test vectors but to the underlying BIST patterns. Our BIST diagnostic approach currently uses meta-comments and annotations within the source test vectors to maintain the information needed to perform this mapping. When a stand-

ard approach for driving this information forward into the test environment is finalised this will be used instead.

Failure mapping takes the reported time of a failure, which can be either a tester cycle count or absolute time value, and converts this into information more meaningful to the diagnostic step being performed. During the diagnostic MISR unload steps is just a matter of determining which BIST pattern is triggering the failing MISR value to be observed. For the DUT scan chain content unload step the failure mapping must relate the failure back to the DUT core scan chain and element position within it. This mapping may need to take account of any scan chain concatenations present in the BISTed DUT, even being able to map back from the single scan chain used through the 1149.1 TAP.

V Disadvantages of this Approach

As we stated above, to make effective use of the diagnostic unloading of MISR values and then to be able to unload only the targeted failing patterns and the selection of passing patterns, requires access to the BIST database of pattern, MISR and PRPG values. If this is not possible at the point at which the diagnostic data is being gathered then this approach will not work. One can fall back to the alternative window based unload approach also outlined above.

Running diagnostics through the 1149.1 tap interface may take some time, considering the volume of data to move. However, extraction of diagnostic information outside of the manufacturing test flow is not a time critical task.

Performing BIST diagnostics using the MISR unload mechanism outlined above is disruptive, each failing part will require it's own failing pattern unload vector set. Hence, it can only be performed as an off line activity. Using the non disruptive interval unload approach may require a large amount of response/failure storage, because of the data volume involved.

The MISR unload mechanism will not work when transient defects¹ cause failures, unless these defects are repeatable using an isolated pattern. Diagnostic information for such defects can be captured using a windowed BIST pattern run terminating on the pattern that observes the defect effect. The DUT scan chains can then be unloaded to locate the scan element that made the observation. Multiple BIST runs may be necessary to locate the failing pattern.

VI Conclusion

The novel hardware within the BIST controller allows the BIST to be run using any number of independent windows of arbitrary size. It also provides a diagnostic mode that allows

1. Such as power supply noise, triggered by activity in previous patterns.

access to the MISR and DUT scan chain contents on a per BIST pattern basis, with exceedingly simple synchronisation semantics, without the need to reinitialise the BIST controller for each pattern. These capabilities when used with our supporting software tools provide a very flexible diagnostic environment that can be used to extract complete diagnostic information in a wide range of flows, from a very low data volume binary search and unload of failing patterns through to a high volume unload of all patterns (with or without the use of windowing).

We also propose a new technique that allows failing patterns to be identified and then targeted directly for DUT scan chain observation, using the diagnostic mode of the BIST controller.

The diagnostic approach is valid in situations where there is good access to the failing part as well as those where the failing part is embedded and access is only available through a constrained test interface, such as 1149.1 TAP. Because of the flexibility of the approach the diagnostic process can be tailored to match the constraints of the test access method and testing environment.

The diagnostic approach provides the same level of diagnostic resolution as would be available had the BIST patterns been applied directly to the DUT. It can be used off line on failing parts during manufacturing test or as a reliability diagnosis tool for parts that fail after manufacture.

VII References

1. C Stroud, *Automated BIST for Sequential Logic Synthesis*, IEEE Design and Test of Computers 1988
2. B Koenemann, T Mucha and G Zwiehoff, *Built in Logic Block Observation Techniques*, Proceedings of the International Test Conference 1979
3. W H McAnney and J Savir, *There is Information in Faulty Signatures*, Proceedings of the International Test Conference 1987
4. Y Wu and S Adham, *BIST Fault Diagnosis in Scan-Based VLSI Environments*, Proceedings of the International Test Conference 1996
5. Y Wu and S Adham, *Scan Based BIST Fault Diagnosis*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, No 2 February 1999
6. J Rajski and J Tyszer, *On the Diagnostic Properties of Linear Feedback Shift Registers*, IEEE Transactions on Computer-Aided Design Oct 1991
7. J Rajski and J Tyszer, *Diagnosis of Scan Cells in BIST Environment*, IEEE Transactions on Computers Vol 48, No 7 July 1999
8. J Ghosh-Dastidar, D Das and N Touba, *Fault Diagnosis in Scan Based BIST Using Both Time and Space Information*, Proceedings of the International Test Conference 1999
9. J Ghosh-Dastidar and N Touba, *A Rapid and Scalable Diagnosis Scheme for BIST Environments with a Large Number of Scan Chains*, Proceedings of the VLSI Test Symposium 2000
10. J Savir, *Salvaging Test Windows in BIST Diagnostics*, IEEE Transactions on Computers Volume: 47 Issue: 4 April 1998
11. J Savir, *Salvaging Test Windows in BIST Diagnostics*, Proceedings of the VLSI Test Symposium 1997
12. Xijiang Lin, R Press, J Rajski, P Reuter, T Rinderknecht, B Swanson and N Tamarapalli, *High-frequency, at-speed scan testing*, Design & Test of Computers, IEEE, Volume: 20, Issue: 5, Sept.-Oct. 2003
13. P Wohl, J Waicukauski, S Patel and G Maston, *Effective Diagnostics Through Interval Unloads in a BIST Environment*, Proceedings of the Design Automation Conference 2002
14. *IEEE Standard Test Interface Language (STIL) for digital test vector data*, IEEE Std 1450-1999
15. P Wohl and N Biggs, *P1450.1: STIL for the simulation environment*, Proceedings of the VLSI Test Symposium 2000
16. P.H. Bardell, W.H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudo Random Techniques*, Wiley Interscience, New York, 1987
17. *IEEE standard test access port and boundary - scan architecture*, IEEE Std 1149.1-1990
18. R Kapur, T Williams and M Mercer, *Directed-Binary Search in Logic BIST Diagnostics*, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition 2002
19. N Tamarapalli and J Rajski, *Constructive multi-phase test point insertion for scan-based BIST*, Proceedings of the International Test Conference 1996
20. S K Gupta, *Recent advances in BIST*, VLSI Test Symposium, 1992. '10th Anniversary. Design, Test and Application: ASICs and Systems-on-a-Chip', Digest of Papers., 1992 IEEE, Vol., Iss., 7-9 Apr 1992
21. Zhiyuan Wang, Kun-Han Tsai, M Marek-Sadowski, J Rajski, *An efficient and effective methodology on the multiple fault diagnosis*, Proceedings of the International Test Conference 2003