A tool for teaching memory testing based on BIST

A tool for teaching memory testing based on BIST

Authors: Fischerova M., Pikula T., Simlastik M., Bosio A., Di Carlo S., Di Natale G.,

Published in the Proceedings of the IEEE International Biennal Baltic Electronics Conference (BEC), 2-4 Oct. 2006, Tallin, EE.

DOI: 10.1109/BEC.2006.311094

A tool for teaching memory testing based on BIST

Mária Fischerová[1], Tomáš Pikula[1], Martin Šimlaštík[1],
Alberto Bosio[2], Stefano di Carlo[2], Giorgio di Natale[2]

[1]*Institute of Informatics, Dúbravská cesta 9, 845 07 Bratislava, Slovakia*
[2]*Politecnico di Torino, Corso duca degli Abruzzi 24, I-10129 Torino, Italy*

ABSTRACT: The paper presents a tool that explains and demonstrates the essentials of RAM testing and memory built-in self-test. It also generates the BIST structure for the given memory matrix together with a march test which is provided by the march test generator according to the defined list of faults. The developed system was implemented as a Java applet what means its good compatibility regarding different hardware and operating system platforms, its safety and accessibility while it is placed on Internet. The presented tool has been utilised as the educational instrument in laboratory works.

1 Introduction

Memories, one of the most important components in digital systems like systems on chip (SoC), are extremely vulnerable to physical defects due to the high density of their cell arrays. Memory testing and design-for-test (DfT) became one of the crucial tasks in the design of complex and heterogeneous SoCs.

The semiconductor industry needs test engineers with high skills in memory testing, and the academic world needs to fulfil this requirement. Politecnico di Torino and the Institute of Informatics have a wide experience in the field of RAM testing (i.e., automatic march test generation, fault simulators, memory BIST generators etc.). This work takes advantage of the joint experience of our research groups in developing an interactive educational tool for the students that should introduce standard and well-known methods of memory testing based on built-in self-test (BIST).

Two individual tools, MemBIST and the March Test Generator, designed and implemented at the two above-mentioned institutions were merged into one tool in order to facilitate its usage also by the professionals.

2 MemBIST

The operation and testing of memories are different from logic. Memory testing needs special testing algorithms to generate the required memory test patterns (a sequence of write logical 0/1 and read logical 0/1 operations) since many fault models have to be covered. Many testing algorithms were developed in the past (e.g., Zero-one, Walking, Galloping, or Checkerboard patterns) but at present, mostly many types of march like algorithms are realistic to be used in testing of bit or word oriented memories [1], [2].

The memory embedded into SoC is usually difficult to test because of its poor controllability and observability. The proper test solution is the usage of the BIST method – the testing algorithm and the test response analysis are implemented on the chip [1], [2], [3].

The MemBIST is a software tool that demonstrates principles of RAM testing and memory BIST (MBIST) architectures and generates the memory BIST structure for a given memory [4]. The developed and implemented applet consists of two separate modules – Learning and Generation.

The *Learning module* demonstrates RAM faults and March C- testing algorithm, explains the BIST method for RAMs, and presents its specific structure and functionality by an interactive animation. It is divided into two functionally independent parts.

In the part called Learning, fundamentals of MBIST are taught while following successively components of the memory array and the BIST architecture. It starts with explanations of the memory model and the most common fault models in memories (Figure 1). The next step is setting the address, data and control multiplexer's parameters. The control unit is composed of the March C- algorithm (Figure 2) and an address generator. The March C- algorithm can be implemented as a finite state machine (FSM). As the address generator, the linear feedback shift register (LFSR) was chosen, due to its easy hardware implementation (Figure 3). The user can define the characteristic polynomial and the initial value of LFSR and observe the generated patterns. The last explained component of MBIST is a comparator of fault-free value with the output of the memory.

The Exercise part illustrates the testing performance of March C- algorithm during fault detection and localisation of an injected fault. First, the user defines memory parameters – the number of rows, the number of columns and the memory cell bit-width. After defining the memory, the user injects selected faults into the memory matrix and then defines the polynomial and the seed of LFSR. The last step is the simulation of the configured memory with injected faults (Figure 4). It can run automatically, or the user can manually control the

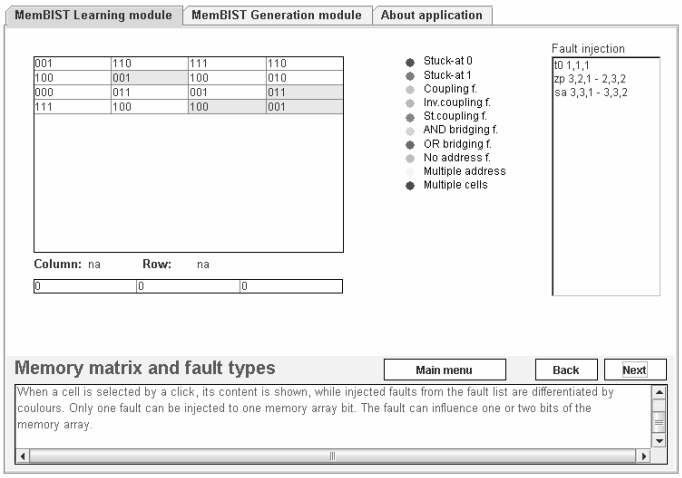

Fig. 1. MemBIST Learning module – fault injection.

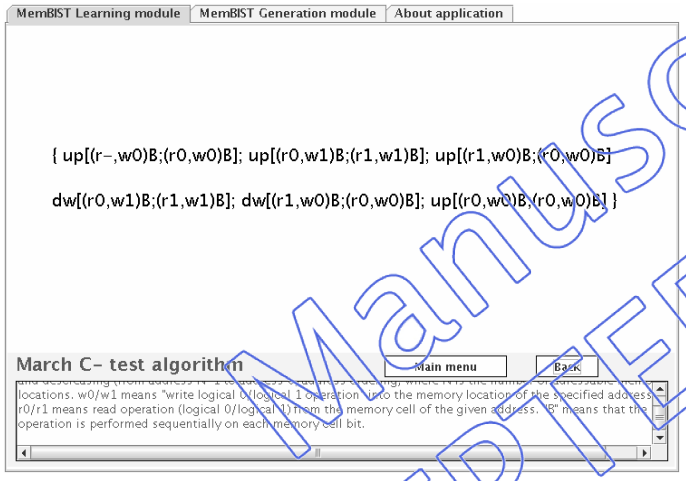

Fig. 2. MemBIST Learning module – March C- algorithm.

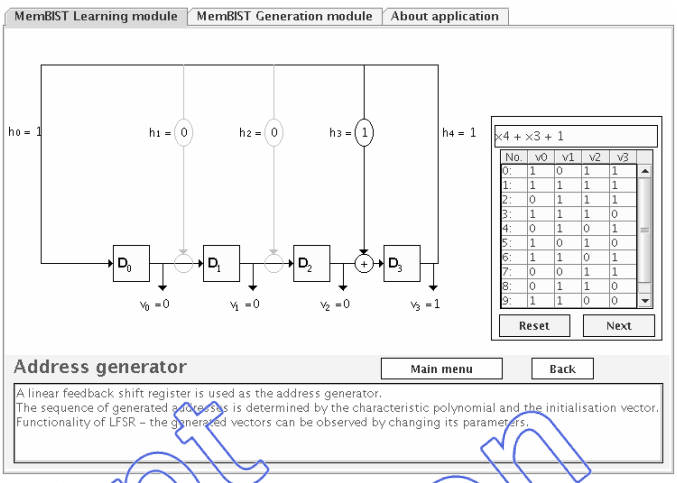

Fig. 3. MemBIST Learning module – address generator.

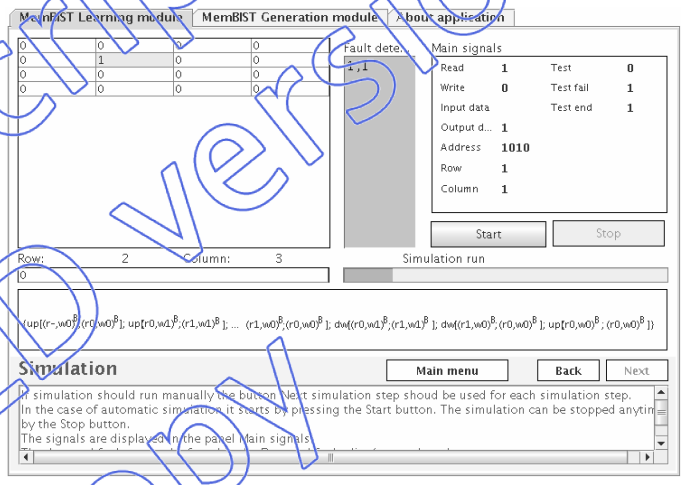

Fig. 4. Learning module – fault simulation.

simulation steps. The simulation can be configured to stop after the first detected fault, or to run until the end to observe all detected and localised faults.

The *Generation module* allows generating the memory BIST architecture for a given tested memory. There are two possible inputs into the module. The size and structure parameters of the memory matrix and its cells (the number of row and column, the cell bit-width) can be either typed directly into the module dialog window or inserted in the form of a VHDL entity. The another parameters, which have to be set in the following dialog windows, are the characteristic polynomial and the seed for LFSR, and the type of faults the march test has to cover.

The generator can build the BIST circuit for single port memories of arbitrary size (preferably the size of 2^N). As the BIST architecture is based on shifting, the tested memory can even be word-wide [1].

The MBIST architecture contains the type-1 (external-XOR) LFSR, which is responsible for the address generation. This type of LFSR was chosen due to the possibility of generating the maximum-length address sequence including the all-zero pattern and its reverse ordered sequence. The maximum-length sequence depends on the selection of the characteristic polynomial that must be primitive.

An important issue of using the BIST architecture is an efficient test, which has to guarantee the high fault coverage with minimal area overhead and performance penalty. The user selects the fault types from the list of classic fault models typical for memories or defines own fault models using the fault primitive's formalism (Figure 5) [5]. The march test covering selected faults is then generated by the March Test Generator [6].

All set parameters are considered in the generation of the BIST circuit, especially its finite state machine (FSM). The results of the March Test Generator directly influence the VHDL description of FSM within the memory BIST.

The output of the presented tool is a hierarchical HDL description of the generated BIST blocks - components (address generator, test generator, control logic, output response compactor and analyser) for the tested RAM on the behavioural level (Figure 6).

The resulting VHDL code can be simulated in commercial VHDL simulators (e.g. ModelSim) and is fully synthesizable in commercial or freely available

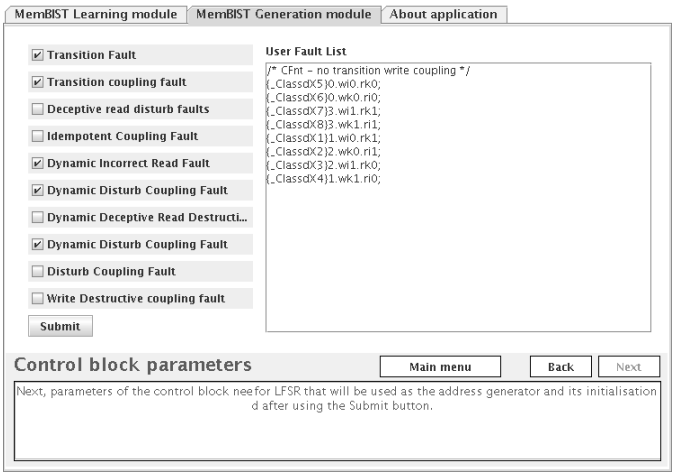

Fig. 5. MemBIST Generation module – fault list selection.

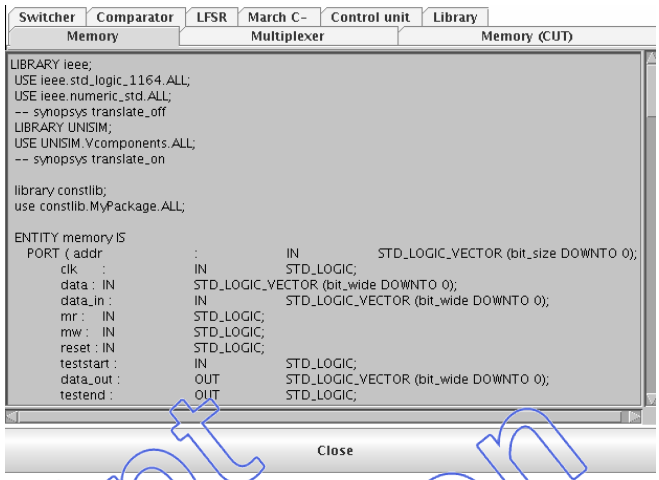

Fig. 6. Generated MemBIST VHDL code

design tools. It can be used as an example implementation of the memory BIST in the educational process.

The hardware design languages like VHDL and Verilog are the industry standards used for hardware modelling from the abstract to any particular level. The BIST structures are included in the professional CAD tools, but apply the basic testability architectures to circuit under test on the structural level. In the presented tools, the BIST techniques are applied at the register transfer level using VHDL models [3], [7], [8].

2 March test generator

The March Test Generator module (Figure 7) is able to generate march tests starting from a user defined list of faults. The march tests are particularly simple memory test algorithms that use the regular structure of SRAMs to reduce the test complexity [9]. Several march tests targeting different set of memory faults have been proposed [10]. Most of them have been generated by hand, but with the occurrence of new and more complex fault models, the task of hand writing test algorithms is becoming harder and it may lead to non-optimal results.

The march test generation process starts from the

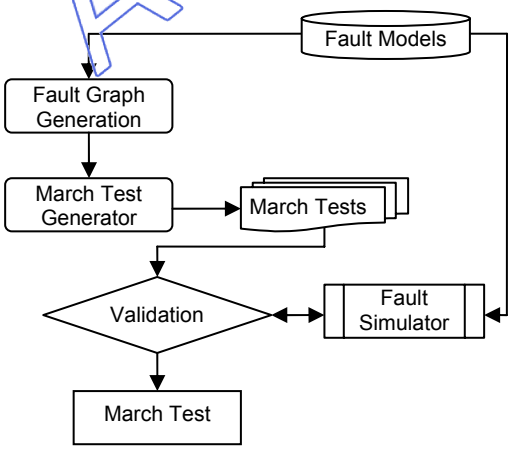

Fig. 7. March Test Generator.

definition of the list of faults to be tested. Besides classic models, user defined faults expressed in terms of fault primitives [4] are supported. The March Test Generator is able to deal with:

1. static and dynamic faults
2. linked and unlinked faults
3. single and multiple port memories

Given the list of faults to be tested, the March Test Generator is able to generate a non-redundant march test covering the selected faults. Each generated march test is then fault simulated to check its coverage and to eventually optimize the final test.

Using the March Test Generator students can become familiar with one of the most used memory test algorithm in the industry.

3 Conclusion

The new MemBIST tool has been utilized in the educational process at the Faculty of Informatics and Information Technologies of the Slovak University of Technology. It has been regularly used for practical exercises in the testing area as a new educational concept at the lab works in the basic course Diagnostics and Reliability of Digital Systems for undergraduate students, in the advanced course Testing of Digital Systems for graduate students and in diploma works. The Web-based applet simulates the learning subject in a well illustrative graphical form that is self-explanatory, takes the advantage of learning by doing and involves interaction possibilities. Using such tools during the laboratory works makes the course more attractive to the students. Students' opinions, remarks and suggestions have been gathered and analysed in order to improve the MemBIST modules.

In a similar way the same tool has been used at Politecnico di Torino during lab sessions of the course Digital Systems Dependability for master students of Electronics and Computer Science Engineering. By comparing the interest of students with regard to the previous editions of the same course not using the

MemBIST applet, more interest gained from the possibility of applying theoretical notion was explained during lectures on real test cases.

MemBIST is the part of a tool set for understanding testability methodologies as the BIST (deterministic TPG construction based on LFSR and CA, signature compaction techniques, test response analysers) and DfT techniques (compliant with the recommended IEEE standards) that has been developed and implemented at the Institute of Informatics (Figure 8) [11], [12].

Since the tool set is freely accessible on Internet [13], teachers and students from other technical universities are also encouraged to exploit the modules in the teaching and learning process.

The work on MemBIST will continue with implementing the neighbourhood pattern sensitive fault model and the particular algorithm that is based on Eulerian or Hamiltonian sequences for detecting neighbourhood pattern sensitive faults. Further applet improvement resides in the optimisation of the generated VHDL code in terms of speed and area. The MemBIST learning module should also contain some exercises for localisation of hidden faults (stuck-at, bridging and coupling) in the memory matrix with respect to the behaviour of the memory cells.

4 Acknowledgement

The presented work has been supported by Slovak-Italian Science and Technology Co-operation project "Testable and Reconfigurable Digital Cores" and by Slovak national project VEGA 2/5123/25 "Methods and Algorithms for Optimum Testing of Digital System on Chip".

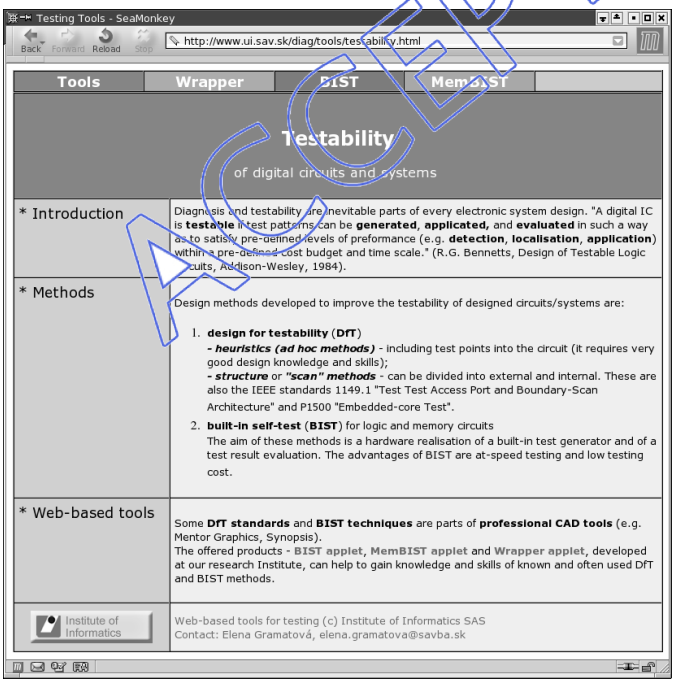

Fig. 8. Set of testability tools.

References

[1] A.J. van de Goor, "Testing Semiconductor Memories, Theory and Practice", ComTex Publishing, Gouda, The Netherlands, 1998.

[2] R.D. Adams, "High Performance Memory Testing. Design Principles, Fault Modeling and Self-Test," Kluwer Academic Publishers, Bostom / Dordrecht / London, 2003.

[3] M.L. Bushnell, V.D. Agrawal, "Essential of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits", Kluwer Academic Publisher, Bostom / Dordrecht / London, 2000.

[4] M. Fischerová, M. Šimlaštík, "MemBIST applet for learning principles of memory testing and generating memory BIST", EuroMicro Conference on Digital System Design, August 30 - September 3, 2005, Porto, Portugal, pp. 276–279.

[5] A. J. van de Goor, Z. Al-Ars, "Functional Memory Faults: A Formal Notation and a Taxonomy", 18th IEEE VLSI Test Symposium, 2000, pp. 281-289.

[6] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "Automatic March tests generation for static and dynamic faults in SRAMs", 10th IEEE European Test Symposium, May 22-25, 2005, Tallinn, Estonia, pp. 122–127.

[7] L.S. Hurst, "VLSI Testing Digital and Mixed Analogue/Digital Techniques", The Institution of Electrical Engineers, London, United Kingdom, 1998.

[8] A.L Crouch, "Design for Test for Digital IC's and Embedded Core Systems", Prentice Hall PTR, New Jersey, USA, 1999.

[9] A. J. van de Goor, "Using March Tests to Test SRAMs", IEEE Design & Test of Computers, pp. 8–14, 1993.

[10] A.J. van de Goor, B. Smit, "Automatic verification of March Tests", IEEE International Workshop on Memory Technology, Design and Testing, 1993, pp.131–136.

[11] M. Baláž, T. Pikula, R. Lauko, M. Fischerová, E. Gramatová, "eLearning and eTraining Tools for Testability Techniques of Digital Circuits and Systems", 5th International Conference Virtual University, Bratislava, Slovak Republic, December 16-17, 2004, pp. 95-100.

[12] M. Baláž, E. Gramatová, T. Pikula, M. Fischerová, "eTool for Teaching and Application of Digital System Testability Techniques", IEEE Region 8 EUROCON 2005 Conference, Belgrade, Serbia and Montenegro, 2005, pp. 831-834.

[13] www.ui.sav.sk/diag/tools/testability.html