ISSN 0002-306X. Proc. of the RA NAS and NPUA Ser. of tech. sc. 2019. V. LXXII, N2.

UDC 004.451.3 MICROELECTRONICS

D.L. HAYRAPETYAN

VERIFICATION OF THE MEMORY TEST AND DIAGNOSIS FLOW IMPLEMENTATION IN SOFTWARE POST-SILICON ANALYSIS AUTOMATION TOOLS

With rapidly increasing density and capacity of nanoscale memory devices embedded in modern system-on-chips (SoC), new design problems are being introduced, as well as the requirements are strengthened towards test and diagnosis for achieving higher quality and increased yield. This leads to modification of existing and/or development of new memory test, fault detection, localization and classification flows that are being implemented in various software post-silicon analysis automation tools. In this paper, an approach for verification of those tools is proposed.

Keywords: Verification, Memory test, Memory diagnostics, Memory faults, Test pattern.

Introduction. The rapid increase of density and capacity in memory IP cores embedded in modern system-on-chip (SoC) creates new challenges of preserving test and repair cost while also minimizing time-to-market. The on-chip infrastructure IP is proposed in [1] to maximize the test and repair efficiency utilizing the memory design knowledge and providing the analysis on failure data. Considering the increasing complexity of SoC design, it becomes crucial for silicon embedded memory test and repair solutions to keep up with the technology advances in order to consistently provide superior chip quality and yield optimization [2]. Some aspects of implementation of the corresponding solutions for post-silicon analysis automation that extend the mentioned infrastructure to cover challenges of today's designs which are much bigger, faster, hierarchical and sensitive to area, timing and power are considered in the current work.

With the technology shrinking, new types of memory defects and corresponding memory fault models [3] for the memory test have been observed during post-silicon analysis [1]. That posed new challenges in the test and diagnosis of embedded memories of systems-on-chip (SoC) using all-in-one solutions [2, 4]. We follow the approach of task distribution between the hardware (HW) memory built-in self-test (MBIST) engine and software (SW) automation tools adduced in [2], where the management and control of test and diagnosis flow is implemented via SW, while the actual at-speed basic test and diagnosis procedures are performed by the MBIST engine.

The interaction between SW and HW sides of this mature solution is managed via creation of test patterns at the SW side, their application to MBIST engine via standard interfaces [5-7] and analysis of obtained results/chains from the MBIST engine at the SW side. The proposed mature test and diagnosis flow [8] comprises three main phases: fault detection, classification and localization, where each phase requires specific test patterns to be created and analyzed, so that results can be propagated to the next phase preparation step. Specific march test algorithms and march-like test algorithms should be developed and used for each phase for generating the corresponding test patterns, which in their turn, will be passed to the MBIST engine for further at-speed execution.

Since it is crucial to ensure the correctness of diagnosis flow implementation before it is applied to a real SoC, a fault-prone environment for pattern verification is required for modeling the test pattern execution on the MBIST engine using some accurate models of memory faults to be tested by the MBIST engine. It is also essential to estimate the execution run time for being time-efficient and allowing a variety of different scenarios to be quickly passed through the MBIST engine.

The aim of this paper is to build the mentioned environment for test patterns and obtained chain verification.

With the introduction of FinFET technology, new types of memory defects have been observed [9-11]. Test and diagnosis flows designed for faults present in previous designs were not applicable as they were not able to provide the necessary coverage and required modification of detection, classification and localization phases. At the same time, solutions elaborated for current designs will face the same issue in future because of continuous changes in memory designs due to technology shrinkage. A natural demand for prediction of new fault types and modifications of solutions required for the test and diagnosis arose. The issue was addressed with introduction of multidimensional prediction mechanism for memory fault classification [12], that systemizes all known memory faults in periodic manner and gives a view on impending new faults that may appear in memories with new technology nodes. Furthermore, the mechanism offers a generic flow for efficient new march test algorithm generation for the new faults based on a test algorithm template. The use of the mentioned mechanism is added to the generation process of memory fault models for the SW side.

A System Verilog environment [13] is considered as a basis for implementation of the environment for pattern verification. The existing tools [14] are enriched to cover the new requirements above. The application of the enriched tools for generation of several memory faults inherent to modern manufacturing technologies [15] is also considered.

The structure of the paper is as follows: section 1 provides brief information on the approach for building verification environment for test patterns used for post-silicon analysis. Fault model generation and injection flows are outlined in section 2. The verification step for test patterns is defined in section 3 along with the verification step of the output chain analysis of the test and diagnosis flow in section 4. The experimental results are provided in section 5. The use of the approach for the predicted faults is proposed in section 6. Section 7 concludes the paper.

1. An approach for building the verification environment for post-silicon analysis of test patterns. Throughout this paper, we will consider the test patterns used for the test and diagnosis flow described in [8]. Each step of the test and diagnosis flow, when implemented in test automation tools, requires creation of a corresponding test pattern, application of the pattern on the BIST system and the analysis of the obtained chains. The analyzed information is utilized at creating the test patterns in the next step.

However, test pattern creation and chain analysis are fault-prone for implementation. Therefore, they require an adequate environment for being tested and verified before interactions with manufactured SoCs. Since the test pattern generation in each phase of the flow depends on the results of test pattern execution in the previous phase, verification of this flow implementation is essential. The verification environment is constructed on the basis of MBIST register transfer level representation (Fig. 1). The process of building the environment can be divided into 3 parts which will be discussed further, throughout this paper.

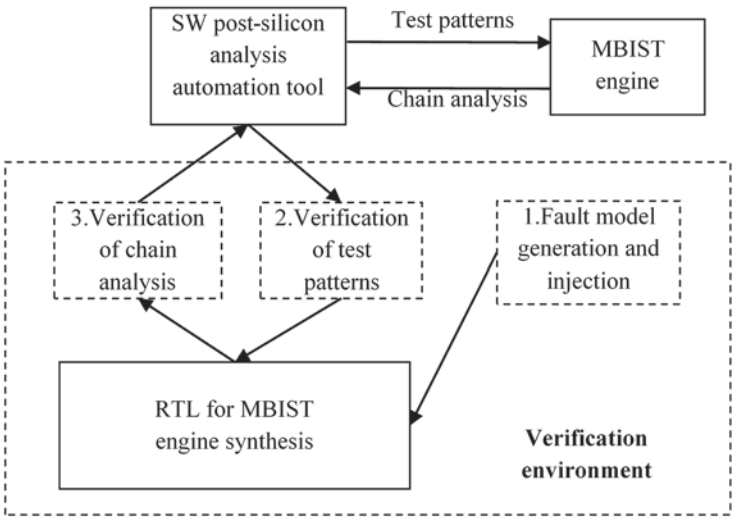

Fig. 1. Verification flow for the software post-silicon analysis automation tool

2. Fault model generation and injection. The fault DFA generation flow [14] was used in this step which consists of the following major steps:

1. Parsing input files and storing the obtained data as separate constructs: fault primitives (FPs) [3], fault injection memory cell addresses, memory configuration file.

2. Generating a fault description table (FDT) for FP (<S/F/R> for single-cell and <S_a: S_v/F/R for coupling faults>). FDT is a table representation of the DFA model, where each row of the table contains information on DFA state and transitions going out from it.

3. Generating the System Verilog code for FDT and including it into the memory test bench, while considering the information provided in point 1.

The fault DFA is intended to model the memory internal faults. It is shown how the fault model is extended by increasing the number of the affected cells and operations in FP [16]. In addition, fault model extension for linked faults that comprise multiple FPs [17] was derived in [18].

2.1. Fault injection. During the inclusion of fault model in RTL, the test bench fault memory cell address must be provided. Since memories generally use words for storage of the information rather than single bits, the memory cell address can be provided via memory word address and the position of the faulty bit in it. Fault models with several memory cells require multiple addresses and bit positions to be specified. Furthermore, memories may consist of multiple banks, thus the bank index must be provided. Finally, as soon as multiple instances of a memory device may be present in the MBIST network, the hierarchical path to a memory must also be provided (Fig. 2).

```
server{
[processor1 [1]]
  MEMORY [1]
    //   Address in range: [1023 : 0], Number of Bits = 133

    single cell faults:
    ADDR = 3, BIT = 2, FTYPE = <R0/0/1>
    ADDR = 2, BIT = 1, FTYPE = <0/1/->

    coupling faults:
    ADDRA = 900, BITA = 5, ADDRV = 915, BITV = 2, FTYPE = <1W1;0/1/->

[processor2 [2]]
  MEMORY2 [1]
    //   Address in range: [4991 : 0], Number of Bits = 61

  MEMORY2 [2]
    //   Address in range: [4991 : 0], Number of Bits = 61
}
```

Fig. 2. A fault injection file example

Another important issue to be considered is that memory address used for fault injection is generally a logical address. Thus, two bits logically preceding one another (within the same word or two words logically neighboring each other), are not necessary neighboring each other inside the memory device. This behavior is determined by the memory address and data scrambling that is usually present in most modern memories [19]. Assuming the data on memory address and data scrambling information is provided, the second address for coupling the faults is calculated by providing the logical address of either aggressor or victim cell, calculating the logical addresses of the neighboring cells (Fig. 3), using the scrambling information and picking the address of the second cell from the list of calculated logical addresses.

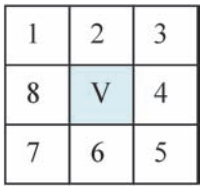

Fig. 3. Victim cell V, with potential aggressor cells 1,2,3,4,5,6,7,8

Fault injection or the process of including the DFA fault model in the RTL test bench requires knowledge of memory device pins for addressing, obtaining and shifting in the data, application of read and write operations. This information can be provided with the memory configuration file [14].

3. Verification of test patterns. Since the test patterns command the MBIST to run march test algorithms/march-like adaptive test algorithms on the memory devices, a way to determine if the test pattern was successfully executed is to observe the behavior of the fault model injected into the MBIST.

The fault model is traversable through the transition flags in FDT representation [14]. Each time a transition from a state occurs, the transition flag of current FDT row for corresponding operation is set to 1. After the simulation is over, the resulting FDT is compared with a reference FDT which shows what is expected after the execution of the pattern (Fig. 4). This template is created on the basis of generated fault model FDT and is traversed separately via the march test algorithm that is used in the test pattern.

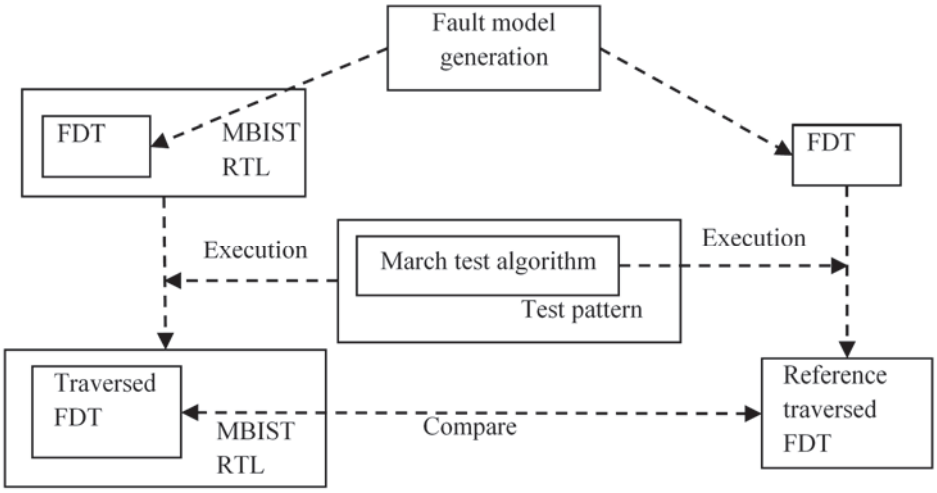

Fig. 4. Verification of test patterns

4. Verification of chain analysis for test and diagnosis flow. Since the memory test and diagnosis is time-sensitive, parallel testing of multiple memory instances is crucial. BIST instructions for multiple memory instances that need to be executed simultaneously are usually combined in groups during the test pattern creation.

Generally, the test and diagnosis flow implementation in test automation tools consists of four phases (test, detection, classification and localization) [8] described below via test pattern templates. Templates are used to generate a pattern instance for the corresponding phase. Each phase requires specific analysis of the MBIST output chains.

Test. Pattern template:

1. SELECT_MEMORY_GROUP
2. LOAD_TEST_ALGORITHM
3. RUN_BIST
4. READ_FAILED_MEMORY_INFO

Description: This pattern loads the test algorithm into the BIST, while instructing to run it on the specified memory groups. Information on failed memories is obtained from BIST, which can help to narrow the set of memories for further diagnostics, by excluding them from memory groups for the next test patterns.

Verification of the chain analysis: Verification of the reported information on failed memories is based on the conformity with memories used for fault injection in the fault injection file.

Detection. Pattern template:

1. SELECT_MEMORY_GROUP
2. LOAD_TEST_ALGORITHM

3. SET_SONE I
4. RUN_BIST
5. READ_DIAGNOSTIC_INFORMATION

Description: This pattern also loads the test algorithm into the BIST while specifying the value of stop on N-th error register. This will command the BIST to stop the test algorithm execution after the N-th error has been encountered (totally N read operation of march test algorithm have returned the value that was not expected). Diagnostic information on the last failed memory cell and applied test operation is obtained from BIST as a binary chain with the last instruction of the test pattern, and further evaluated by software tools for convenient representation. The general information provided is:

1. Memory failed word address.
2. Memory failed bit.
3. March test failed operation.

Verification of the chain analysis: The verification of the reported diagnosis information is based on the conformity with banks, addresses and bits used for fault injection in the fault injection file.

Classification. Description: The fault classification step uses the test pattern template which was the described point **a**, while applying fault classification march test algorithms specifically designed for this diagnosis phase, **n** number of pattern executions need to be made while modifying value **I** to make sure that **n** read operations have been applied on the faulty cell, where **n** is the number of read operations present in test algorithm [8]. The test syndrome is generated as a result, which is **n**-bit signature, where the order of bits corresponds to the sequence of the read operations in the test algorithm. The faults are classified based on the obtained signature.

Verification of the chain analysis: The verification of this phase is done by checking the conformity of the classified fault type with FP used in the fault injection file.

Localization.

The test pattern template for this phase is as follows:

1. SELECT_MEMORY_GROUP
2. APPLY_ADAPTIVE_TEST_ALGORITHM
3. READ_DIAGNOSTIC_INFORMATION

Description: This pattern is generally used with coupling faults for locating the aggressor cell. An adaptive march-like algorithm is applied [8] to the neighborhood of a memory victim cell (Fig. 2), usually determined after execution of pattern **b**. The idea of the algorithm is:

1. Set the potential aggressor cell values opposite to the fault activation initial value of aggressor for the coupling fault.

2. Apply the sequence of the fault activating operations to the potential aggressor or victim cell depending on the type of the coupling fault.

3. Read the value of the victim cell.

4. If the fault is not triggered, repeat steps 1-3 for the next potential aggressor.

5. The algorithm stops when the faulty value is finally observed on the victim cell.

Verification of the chain analysis: The verification is based on the conformity with banks, addresses and bits of aggressor cells used in fault injection files.

5. Experimental results. Diagnosis flow implementation along with the implementation of fault model generation flow were verified with VLP1(26N), VLP2(26N), VLP3(22N), March FFDD (42N) algorithms [5]. The MBIST flow was simulated separately for each fault and faults were detected. Using the fault classification step of diagnosis flow FinFET-specific faults were successfully determined and classified.

An example of FinFET-specific <R0R0R0R0R0/1/1> fault traversed by March FFDD algorithm is provided in Fig. 5.

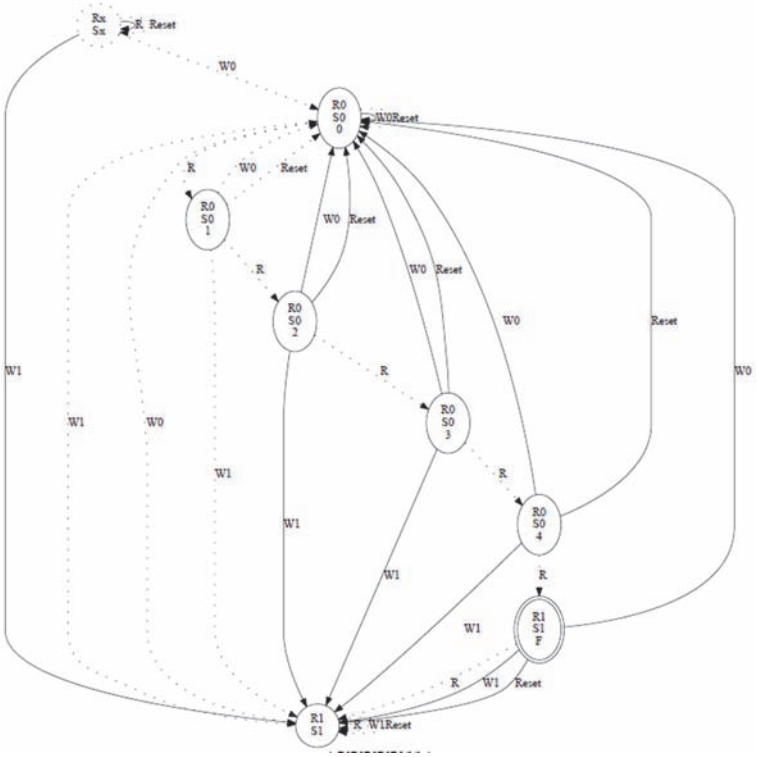

Fig. 5. <R0R0R0R0R0/1/1> fault traversed by March FFDD

6. Use of the approach for the predicted faults. As it was mentioned above, a multidimensional prediction mechanism for memory fault classification introduced in [12] allows to predict new types of faults in memory. The suggested verification approach can avail of the mechanism if several additional changes will be made in the fault model generation and injection part to reflect the knowledge reflected in the prediction mechanism.

Conclusion. A built verification environment for test and diagnosis flow implementation within software post-silicon analysis automation tools is described in this paper. The test results on the fault DFA model behavior on recently considered fault types, using the fault classification flow are provided.

The verification environment is implemented in DesignWare STAR Memory System Yield Accelerator [2] tool that is currently used both in the development of MBIST and during the test and diagnosis of SoCs with an embedded MBIST engine. Some further work connected with the extension of the approach for the predicted new types of faults is outlined too.

REFERENCES

1. **Zorian Y., Shoukourian S.** Embedded memory test and repair: infrastructure IP for SOC yield // IEEE Design and Test of Computers.- 2003.- Issue 6.- P. 58-66.
2. **Zorian Y., Shoukourian S.** Test Solutions for Nanoscale Systems-on-Chip: Algorithms, Methods and Test Infrastructure // Selected papers of Ninth International Conference on Computer Science and Information Technologies.- IEEE.- 2013.- P. 1-3.
3. **Van de Goor A.J.** Testing Semiconductor Memories: Theory and Practice.- Wiley & Sons Inc, 1998.- 512 p.
4. Quality assurance in memory built-in self-test tools / **A. Au, A. Pogiel, J. Rajski, et al** // 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems.- 2014.- P. 39-44.
5. 1149.1-2013 - IEEE Standard for Test Access Port and Boundary-Scan Architecture - IEEE Standard.
6. 1500-2005 - IEEE Standard Testability Method for Embedded Core-based Integrated Circuits - IEEE Standard.
7. 1687-2014 - IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device - IEEE Standard.
8. **Harutyunyan G., Martirosyan S., Shoukourian S., Zorian Y.** Memory Physical Aware Multi-Level Fault Diagnosis Flow // IEEE Transactions on Emerging Topics in Computing.- 2018.- P. 1-12.
9. **Liu Y., Xu Q.** On Modeling Faults in FinFET Logic Circuits // IEEE International Test Conference.- 2012.- P. 1-9.

10. **Lin C.-W., Chao M. C.-T., Hsu C.-C.** Investigation of Gate Oxide Short in FinFETs and The Test Methods for FinFET SRAMs // VLSI Test Symposium.- 2013.- P. 1-6.
11. **Chi M.-H.** Challenges in Manufacturing FinFET at 20nm Node and Beyond // Globalfoundries.- 2012.-P. 1-2.
12. **Harutyunyan G., Shoukourian S., Zorian Y.** Fault Awareness for Memory BIST Architecture Shaped by Multidimensional Prediction Mechanism // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.- 2019.- Vol. 38, No. 3.- P. 562-575.
13. Synopsys Inc. // https://www.synopsys.com/verification/simulation/vcs.html, VCS Functional Verification Solution.
14. **Hayrapetyan D., Manukyan A., Tshagharayn G.** Implementation of memory static, coupling and dynamic fault models at the register transfer level // IEEE East-West Design & Test Symposium.- 2018.- P. 744-748.
15. **Harutyunyan G., Tshagharayn G., Vardanyan V., Zorian Y.** Fault Modeling and Test Algorithm Creation Strategy for FinFET-Based Memories // IEEE 32nd VLSI Test Symposium.- 2014.- P. 1-6.
16. **Hayrapetyan D., Manukyan A.** Modeling dynamic single-cell and coupling faults via automata models // Computer Science and Information Technologies (CSIT).- 2017.- P. 65-68.
17. **Hamdioui S., Al-Ars Z., Van de Goor A.J., Rodgers M.** Linked Faults in Random Access Memories: Concept, Fault Models, Test Algorithms, and Industrial Results // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.- 2004.- Vol. 23, No. 5.- P. 737 – 757.
18. **Hayrapetyan D.** Modeling linked faults via automata models // IEEE East-West Design & Test Symposium.- 2017.- P. 237-241.
19. US 7,768,840 B1. Memory modeling using an intermediate level structural description / **K. Aleksanyan, K. Amirkhanyan, S. Shoukourian, et al.- 2010.**

Yerevan State University. The material is received on 01.04.2019.

Դ.Լ. ՀԱՅՐԱՊԵՏՅԱՆ

ՀԵՏԻԼԻԿՈՆԱՅԻՆ ՎԵՐԼՈՒԾՈՒԹՅԱՆ ԱՎՏՈՄԱՏԱՑՄԱՆ ԾՐԱԳՐԱՅԻՆ ԳՈՐԾԻՔՆԵՐՈՒՄ ՀԻՇՈՂՈՒԹՅԱՆ ԹԵՍՏԱՎՈՐՄԱՆ ԵՎ ԱՐԱՏՈՐՈՇՄԱՆ ԳՈՐԾԸՆԹԱՑՆԵՐԻ ԻՐԱԿԱՆԱՑՄԱՆ ՍՏՈՒԳՈՒՄԸ

Արդի բյուրեղի վրա համակարգերում (SoC) ներկառուցված նանոչափական հիշող սարքերի արագորեն աճող խտությամբ և հզորությամբ պայմանավորված՝ ի հայտ են գալիս նախագծման նոր խնդիրներ, ինչպես նաև, բարձր որակը և արտադրողականությունը ապահովվելու նպատակով, խստացվում են պահանջները թեստավորման և արատորոշման նկատմամբ: Սա հանգեցնում է հետսիլիկոնային վերլուծության ավտոմանտացման ծրագրային տարբեր գործիքներում իրականցված հիշողության թեստավորման, սխալների հայտնաբերման, տեղայնացման և դասակարգման առկա գործընթացների փոփոխման և/կամ նորերի ստեղծման անհրաժեշտությանը: Այս հոդվածում առաջարկվում է այդ գործիքների ստուգման մոտեցում:

Առանցքային բառեր. ստուգում, հիշողության թեստավորում, հիշողության արատորոշում, հիշողության սխալներ, թեստավորման կաղապար:

Д.Л. АЙРАПЕТЯН

ВЕРИФИКАЦИЯ РЕАЛИЗАЦИИ ПРОЦЕССОВ ТЕСТИРОВАНИЯ И ДИАГНОСТИРОВАНИЯ ПАМЯТИ В ПРОГРАММНЫХ ИНСТРУМЕНТАХ АВТОМАТИЧЕСКОГО ПОСТСИЛИКОНОВОГО АНАЛИЗА

Резкое увеличение плотности и мощности нанометровых устройств памяти, встроенных в системы на кристалле (СнК), приводит к новым проблемам проектирования, а также ужесточению требований к тестированию и диагностированию. Это вызывает необходимость модификации имеющихся или реализации новых процессов тестирования; нахождения, локализации и классификации ошибок в программных инструментах автоматического постсиликонового анализа. В статье предлагается подход к верификации этих инструментов.

Ключевые слова: верификация, тестирование памяти, диагностирование памяти, ошибки памяти, тестовый шаблон.