ISSN 0002-306X. Proc. of the RA NAS and NPUA Ser. of tech. sc. 2024. V. LXXVII, N1

UDC 004.33 **MICROELECTRONICS**

DOI: 10.53297/0002306X-2024.v77.1-69

A.V. BABAYAN, S.K. SHOUKOURIAN

LEARNING METHODOLOGY FOR VALIDATION OF MEMORY BIST SOLUTIONS VIA A SHARED INTERFACE

Memory built-in self-test (MBIST) continues to have its unique place in IC industry. It provides test and repair capabilities which significantly increase IC manufacturing yield.

In general, the integration of MBIST solutions in a system on chip (SoC) is done via automated flows. Possible issues occurring in the flow should not be skipped as it will degrade the performance of SoC or even may disrupt its functioning. The probability of issues is increased if the SoC has specific structure adding limitations for MBIST solution such as testing the memories by shared interface. Dedicated validation environments (VE) help to overcome these issues.

Validation challenges of the MBIST solution via a shared interface for a specific case of multi memory bus BIST engines (MMBBE) are discussed, and a solution is proposed. To avoid the exhaustion increase for the integration scenarios random choices combined with some exhaustions are done depending on a given feature's priority. Meantime, it is mentioned that for big configurations the usage of random values of parameters might bring to missing some corner cases. Due to that it is recommended to use further some learning methods for reducing the VE iterations and exhaustion within a given iteration. In all the cases, a targeted analysis should be done and decisions should be additionally taken to find out a reasonable number of these iterations.

In this paper, a methodology is proposed to soften the mentioned above exhaustion. A new algorithm of learning is proposed, and the results of the algorithm application are adduced which justify its efficiency in reducing the exhaustion.

***Keywords*:** MBIST, test and repair validation, shared interface test, test integration, SoC test, randomization.

Introduction. The continuous increase in the requirements set to IC performance creates new challenges for its maintenance for various criteria, in particular, reliability. Test and repair solutions [1,2] which increase IC reliability, typically do not meet IC vendors' specifications. Particularly, the latter do not accept any modification in the system which brings additional limitations when inserting a testing infrastructure. In such systems, the testing units are connected to the shared interface which allows to have access to them, and the test can be applied through those connections [3]. Considering these, the multi-memory bus BIST engine (MMBBE) solutions are proposed [4,5] for shared interfaces.

1. MMBBE structure. To test multiple memory groups, MMBBE uses an already implemented shared interface (Fig. 1). It consists of common signals which are necessary to test memories. The type and number of memories and interfaces may vary depending on SoC specification. Having the description of memories and interfaces, the proper MMBBE is created.

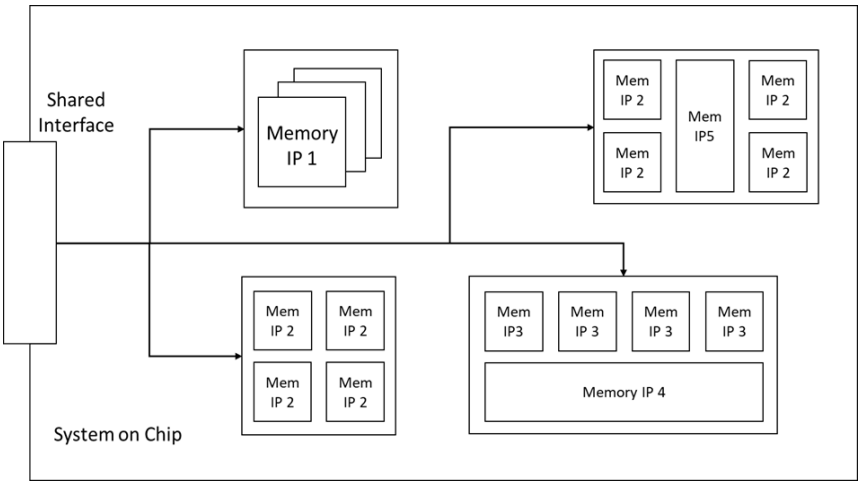

Fig. 1. SoC with memories

Each memory group may have different addressing, read/write and sharing mechanisms. A memory in a group may have its own banks, several ports, and other features (Fig. 2). As per [6] MMBBE is connected to the group via a bus which supports pipeline stages, memory latencies, shared connections within a group, etc.

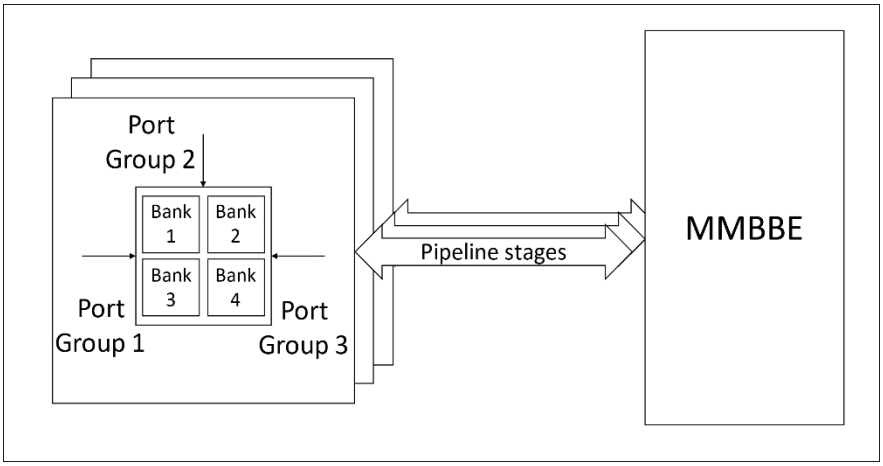

Fig. 2. MMBBE to memory group connections

The integration of the MMBBEs with the described features may have some mismatches which can cause unacceptable consequences. One covers the system's description from test perspective including testing units, i.e., memories. Insufficient or inappropriate description can lead to wrong implementation of the MMBBE.

To assure that the test solution through the shared interface is properly implemented, a validation methodology was suggested in [4]. The methodology tried to solve two problems:

1. find the parameters which describe the system's important features and perform exhaustive variations on those values with an expectation to provide a reasonable coverage;

2. avoid huge variations by selecting random values for the rest of the parameters.

The methodology has defined the primary parameters which indicate the existence of some features in the system. For these features, a validation environment (VE) has been created which allows to create the described above scenarios with configurable value ranges.

It is mentioned that for big configurations to avoid exhaustion increase, random values of parameters are used, which might bring to missing some corner cases. In particular, the MMBBE VE enables iterations both for primary and non-primary parameters. The variations on parameters whose values are chosen randomly, can be performed multiple times. To estimate these variations, it is proposed to use learning for outlining the scope of varied parameters for localization of failures and for bypass from repeating iterations. That will reduce the VE iterations, and an exhaustion within a given iteration according to some criteria.

In the next section, the mentioned approach and a learning algorithm for VE usage are described in detail.

2. The proposed solution. Basically, MMBBE VE iterates through the primary binary parameter (PBP) values and, for each such specific scenario of values named further **PBP vector,** randomly selects acceptable values of the non-primary parameters.

To represent the VE functionality in more detail, the following variables are defined:

n – the number of PBPs;

i – iteration index for VE actions;

For each iteration i the following variables are defined:

N_i – the number of random variations for a PBP vector;

F_i – the number of the observed failures;

k_i – the number of essential PBPs with fixed values;

P_i – the percentage of the observed failures.

After defining othe above variables, the following actions are sequentially executed using the VE:

1. Assigning the initial value: i=0.
2. Assessing the value and assigning the value to N_i.
3. Generating $N_i \cdot 2^{n-k}$ testcases for simulations using MMBBE VE ($k=k_i$, $k_0=0$).
4. Performing simulations and determining the F_i set.
5. Calculating $P_i = F_i/(N_i \cdot 2^{n-k}) \cdot 100\%$ ($k=k_i$).
6. Performing essential ordering of parameters and their values via analysis of PBP vectors that bring to failures during simulations:

- the parameter which occurs with the same value most often in the considered PBP vectors is named the most essential;
- in the case when the occurrence is similar for more than one parameter or for two different values of the same parameter, the determination of their ordering is carried out using other criteria (e.g., via secondary parameter analysis, less time-consuming during runs, etc.).

7. Increment i.
8. Assessing the k_i value as follows; the fixed scenario composed of k_i bits of a PBP vector should yield more than half of the failures from F_i and $P_i > P_{i-1}$. If there are several similar combinations, the greatest is chosen. If all scenario failures have close values, the exhaustion of all values in iterations is inevitable.
9. Inverting the PBPs which are chosen at iteration i-1, and fixing for the rest of the iterations.
10. Repeating steps 2-9 until the $k_i=n$.

3. Experimental results. The application of the proposed method is applied to the SMS MMB processor compiler of Synopsys using MMBBE VE. The 8 PBPs are described below.

1. *Multi interface design*. Specifies whether the testcase contains a single or multiple MBIST interfaces.
2. *Multi memory design.* Specifies whether the testcase contains an interface which is connected to the multiple memory instances or not.
3. *Repairable design*. Specifies the ability to repair at least one memory instance.
4. *Design with latencies.* Specifies whether there is a memory instance with operation latency or not.
5. *Design with memory masks*. If specified, there is at least one memory instance which has a data masking port.
6. *Design with partially connections.* If specified, there is at least one memory instance whose address/data bits are not fully accessible through MBIST interface.

7. *Design with multi-port memory*. Specifies whether there is a memory with multiple read/write ports or not.

8. *Design with multi-bank memory*. If specified, there is at least one multi-bank memory instance.

For the first iteration, N_0 was chosen 50 and $50*2^8$ testcases were generated. Running these testcases it became clear that F_0=241 (P_0=1.88%). Since the generated cases are numbered, the priority list of failures can be composed observing these numbers. Considering MMBBE structure, 6 PBPs were fixed to the values which were most common in failing testcases. The features corresponding to the first two fixed values were 8^{th} and 5^{th} PBPs as enabled. The rest of the fixed PBPs had more even distribution, and the adjustment of those was necessary to be carried out. To highlight the impact of these four fixed values, the previous two PBPs were inverted. The second iteration was done with N_1=3200 to keep the same number of testcases. As expected, the number of failures increased due to localization becoming F_1=541 (P_1=4.22%). For the second iteration, 4^{th}, and 6^{th} PBPs with enabled values were common which could be inverted for the third iteration. The details for all iterations are described in Table.

Table

Validation Environment Iterations

Iteration (i)	**Fixed bits (k_i)**	**Testcases ($N_i*2^{(n-ki)}$)**	**Failed cases (F_i/P_i)**
1	0	50*256	241/1.88%
2	6	3200*4	541/4.22%
3	6	1600*4	201/3.14%
4	6	800*4	427/13.34%

For the last two iterations, the number of fixed bits (k_3 and k_4) are the same, because the combinations give approximately similar failure percentages. After 4^{th} iteration, corner cases remained which will be manually analyzed.

In fact, each next iteration takes less time and requires less effort to analyze failures. The proposed method can be used for different number of parameters, and for each iteration, the reducing scheme may vary depending on system structure and its representation form.

Conclusion. A validation methodology is proposed for memory test and repair solutions working with shared interfaces. The steps of the proposed approach estimate and reduce the number of execution iterations of a validation environment as follows:

1. dependencies between some elements of the PBP are determined and exhaustion is performed based on the consideration of these dependencies;

2. after an estimation of an impact for a given dependence, the specific values of the PBP corresponding elements which highlighted the consideration of the dependence are inverted, and a search for new dependencies is initiated;

3. due to the application of the approach, the number of considered random test-cases is reduced by 30% on average;

4. the performed experiments have confirmed the efficiency of the proposed approach.

REFERENCES

1. **Zorian Y., Shoukourian S.** Test solutions for nanoscale Systems-on-Chip: Algorithms, methods and test infrastructure // IEEE Ninth International Conference on Computer Science and Information Technologies Revised Selected Papers. – September, 2013. – P. 1-3.
2. **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. – March, 2019. – Vol. 38, no. 3. – P. 562-575.
3. **McLaurin T., Knoth R.** The Challenges of Implementing an MBIST Interface: A Practical Application // IEEE International Test Conference (ITC) . – 2019. – P. 1-6.
4. https://news.synopsys.com/2014-10-21-Synopsys-STAR-Memory-System-Multi-Memory-Bus-Processor-Enables-10-Percent-Die-Size-Reduction-for-Marvell-SoC
5. https://resources.sw.siemens.com/en-US/video-automating-physical-mapping-on-arm-ip-with-tessent
6. **Babayan A.** Validation and Test Challenges for Multi-Memory Bus BIST Engines // IEEE East-West Design & Test Symposium (EWDTS). – September, 2023. – P. 1-6.

National Polytechnical University of Armenia. The material is received on 10.03.2024.

Ա.Վ. ԲԱԲԱՅԱՆ, Մ.Կ. ՇՈՒՔՈՒՐՅԱՆ

ՈՒՍՈՒՑԱՆՎՈՂ ՄԵԹՈԴԱԲԱՆՈՒԹՅՈՒՆ ՀԻՇՈՂ ՍԱՐՔԵՐԻ ԸՆԴՀԱՆՈՒՐ ԿԱՊՈՒՂՈՎ ՆԵՐԿԱՌՈՒՑՎԱԾ ԻՆՔՆԱԹԵՍՏԱՎՈՐՄԱՆ ԼՈՒԾՈՒՄՆԵՐԻ ՎԱՎԵՐԱՑՄԱՆ ՀԱՄԱՐ

Հիշող սարքերի ներկառուցված թեստավորումը (MBIST) շարունակում է զբաղեցնել իր ուրույն տեղը ինտեգրալ սխեմաների (ԻՍ) արդյունաբերությունում: Այն ընձեռում է թեստավորման և վերականգնման հնարավորություններ, որոնք զգալիորեն բարձրացնում են ԻՍ արդյունաբերության պիտանի ելքի տոկոսը:

Ընդհանուր առմամբ, բյուրեղում (SoC) MBIST լուծումների ինտեգրումը կատարվում է ավտոմատացված հոսքուղիների միջոցով: Հոսքուղում առաջացող հնարավոր խնդիրները չպետք է բաց թողնվեն, քանի որ դրանք կվատթարացնեն կամ, նույնիսկ, կխափանեն բյուրեղի աշխատանքը: Խնդիրների հավանականությունը մեծանում է, եթե բյուրեղն ունի հա-

տուկ կառուցվածք, որը սահմանափակումներ է ավելացնում MBIST լուծման համար, օրինակ՝ հիշող սարքերի թեստավորումն ընդհանուր կապուղու միջոցով: Համապատասխան վավերացման միջավայրերն (VE) օգնում են՝ հաղթահարելու այս խնդիրները:

Քննարկվել են MBIST լուծման վավերացման մարտահրավերներն ընդհանուր կապուղով թեստավորման համակարգերի դեպքում, և առաջարկվել է լուծում: Ինտեգրման սցենարների հատարկումների աճից խուսափելու համար հատկանիշների հատարկումները զուգորդվում են պատահական ընտրությունների հետ՝ կախված տվյալ հատկանիշի առաջնահերթությունից: Միևնույն ժամանակ նշվում է, որ մեծ կոնֆիգուրացիաների դեպքում պարամետրերի արժեքների պատահական ընտրության պարագայում կարող է բաց թողնվել որոշ անկյունային դեպքերի դիտարկումը: Այդ իսկ պատճառով առաջարկվել է օգտագործել որոշակի ուսուցանվող մեթոդներ՝ վավերացման միջավայրի սցենարների հատարկումը նվազեցնելու համար: Բոլոր դեպքերում, անհրաժեշտ են նպատակաուղղված վերլուծություն և որոշումներ՝ հատարկումների ողջամիտ թիվը որոշելու համար:

Առաջարկվում է մեթոդաբանություն, որը մեղմում է վերը նկարագրված հատարկումը: Առաջարկվում է ուսուցանվող նոր ալգորիթմ, իսկ ալգորիթմի կիրառման արդյունքները ներկայացվում են, որոնք հավաստում են դրա արդյունավետությունը հատարկման նվազեցման գործում:

Առանցքային բառեր. MBIST, թեստավորման և վերականգնման վավերացում, ընդհանուր կապուղով թեստավորում, թեստավորման ինտեգրում, բյուրեղի վրա թեստավորում, պատահականացում:

А.В. БАБАЯН, С.К. ШУКУРЯН

ОБУЧАЕМАЯ МЕТОДОЛОГИЯ ВАЛИДАЦИИ РЕШЕНИЙ ПО ВСТРОЕННОМУ САМОТЕСТИРОВАНИЮ УСТРОЙСТВ ПАМЯТИ ЧЕРЕЗ РАЗДЕЛЯЕМЫЙ ИНТЕРФЕЙС

Встроенные системы самотестирования памяти (MBIST) продолжают занимать уникальное место в индустрии интегральных схем (ИС). Они предоставляют возможности тестирования и восстановления ИС, что значительно увеличивает выход годных ИС во время их производства.

Как правило, интеграция MBIST в систему на кристалле (SoC) осуществляется посредством организации автоматизированного потока для процесса интеграции. Не следует игнорировать возможными проблемами при прохождении через поток, поскольку они могут ухудшить производительность SoC или даже нарушить функционирование SoC. Вероятность появления таких проблем увеличивается, если структура SoC накладывает дополнительные ограничения на MBIST, например, такие, как тестирование памяти через разделяемый интерфейс. Нужны специально построенные среды валидации (VE) для преодоления отмеченных проблем.

Обсуждены проблемы валидации MBIST через разделяемый интерфейс в конкретном случае использования процессора MBIST (MMBBE) для нескольких устройств памяти, связанных через разделяемую шину, и предложено соответствующее решение – среда валидации. Во избежание увеличения перебора для сценариев интеграции выполнялись случайные выборки в зависимости от приоритета данной функции в сочетании с определённым перебором. При этом упоминалось, что для больших конфигураций использование случайных значений параметров может привести к пропуску определённых граничных случаев. В связи с этим было предложено в дальнейшем использовать обучающую среду валидации для уменьшения итераций VE и перебора в пределах данной итерации. Во всех случаях необходимо провести целенаправленный анализ и принять решения для нахождения разумного количества таких итераций.

В данной статье предлагается методология смягчения упомянутого выше перебора. Предложен новый обучающийся алгоритм и приведены результаты его применения, обосновывающие его эффективность для снижения перебора.

***Ключевые слова*:** MBIST, валидация тестирования и восстановления, тестирование с общим интерфейсом, интеграция тестирования, тестирование системы на кристалле, рандомизация.