ISSN 0002-306X. Proc. of the RA NAS and NPUA Ser. of tech. sc. 2022. V. LXXV, N3

UDC 004.33 **MICROELECTRONICS**

DOI: 10.53297/0002306X-2022.v75.3-415

A.V. BABAYAN

VALIDATION AND TEST TIME REDUCTION APPROACH FOR BUILT-IN-SELF-TEST AND REPAIR OF MEMORIES VIA A SHARED INTERFACE

A memory built-in self-test and repair (MBISTR) is a key component in improving a system-on chip (SoC) yield under daily raising new challenges in IC industry where memories very often make up the bulk of a SoC. Once a given SoC has a huge number of memories, it is convenient to test them via a common test network. Meantime, some designs do not allow to insert this network into an already existing functional design via adding new nodes in the existing paths but do allow to add it to the design via already existing functional connection interfaces instead. For example, a shared interface for connecting a central processing unit (CPU) and cache memories in SoCs is already used for testing these memories via special MBISTR engines connected to this shared interface. These MBISTR engines have specific features which may impact essentially the validation and test time.

A way for validation and test time reduction is proposed in this paper for the multi-port memories connected to MBISTR engine via a shared interface.

***Keywords*:** memory BIST, test and repair, multi-port memory testing, shared bus architecture, test and validation time reduction, system-on-chip, parallel testing.

Introduction. The experience of several decades shows that the Built-in self-test (BIST) solutions keep up the IC reliability and yield [1]. Once up to 80% of a system-on-chip (SoC) may consist of memories [1] which in that case have a high impact on the SoC yield, one of the possible solutions could be the use of the memory BIST (MBIST) infrastructure [2,3]. This infrastructure has multiple advantages and some of them are listed below:

- A unified interface with MBIST engine (processor) which wraps memory peculiarities via a special component of the infrastructure named “wrapper” (Fig. 1) and allows to use the same processor instructions for essentially differing memories.

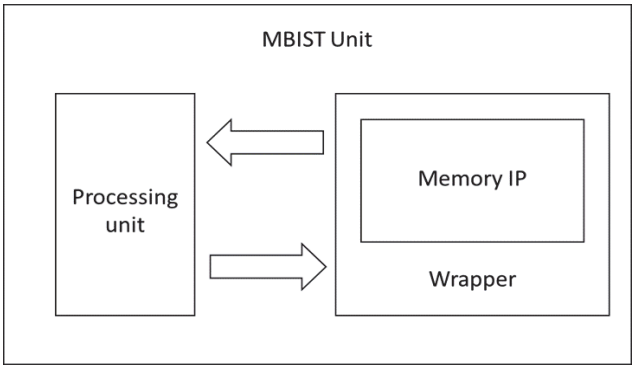

Fig. 1. MBIST processor

- A possibility of merging different processing components of the infrastructure which in the SoC are located sufficiently far from each other in one common test network via a special component named "server". Moreover, it becomes possible to build a hierarchy of test levels within the network and combine parallel and sequential test of sub-networks inside the network.
- A possibility of merging the test of memories, logic BIST and the test for analog and mixed signal IPs in one network.
- A simultaneous use of different test algorithms within the same test network.

Meantime, the insertion of this infrastructure into a functional design of the given SoC might not be desirable for some customers due to an impact on the reliability and yield of the SoC. In such cases it is preferable to perform built-in test via an interface shared between the functional design and MBISTR [4, 5].

The usage of shared interface has some limits and the deep validation of it must be applied. Testing memories through shared interface takes a relatively long time compared with SMS where the user can test all the memories in parallel. In its turn, the test time has a direct impact on the validation time.

In this paper the memory testing and validation methods are discussed for shared interface architecture. The first paragraph describes the structure of SoC with shared interface. The second and third paragraphs are related to algorithm execution on this architecture. Hence, the validation steps are pointed out. In the last chapter an optimization method is suggested which reduces the validation and test time for multi-port memories.

1. Shared interface architecture. Testing memories can be applied using the functional connections existing in SoC. In this architecture SoC consists of shared interface and the memories. As shown in Fig. 2, there are functional connections between that interface and the memories. The main question is how to test these memories using the existing connections.

To test these memories, information about memory locations is required [6]. Using that information, it is possible to apply test algorithms under considering the limits of the functional connections. In general, the mentioned information should contain the shared interface and memory related descriptions in some specified format.

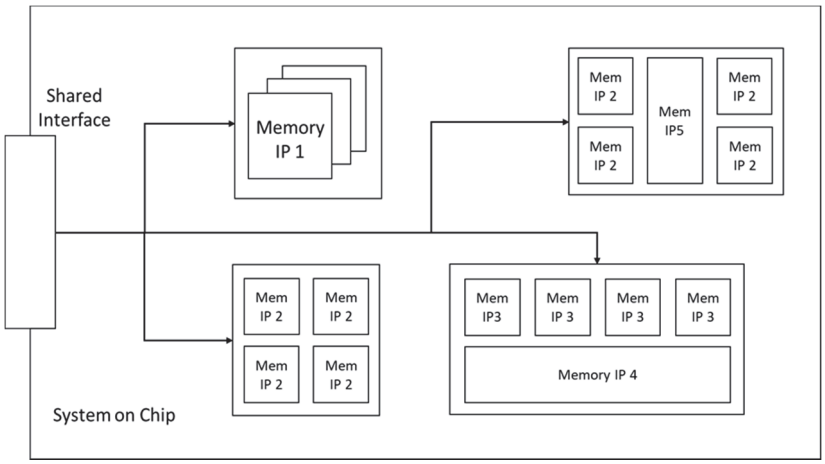

Fig.2. SoC with memories

2. Test algorithm limitations. The architecture specific limitations may exist in the SoCs with shared interface. Some memory ports can be unreachable to that interface or have indirect connections which might change the algorithm execution sequence. Thus, the definition of test algorithms for the case should take into account the peculiarities of the considered architecture.

Application of test operations inside a given algorithm for the considered case also has some specifics.

3. Test operations. The operations used in test algorithm are applied through the shared interface. To implement single read and write operations the corresponding connections should be present in SoC (Table 1, Fig. 3).

Usually the chip-select control for multiple memory instances is implemented via the address bus bits. The "i" in the third line of Table 1 is the corresponding address bit on the shared interface.

Table 1

The bus to memory connections

Bus	Memory Port	Operation
WE_1	WE	Write
RE_1	RE	Read
ADDR_1[i]	CS	Chip-select

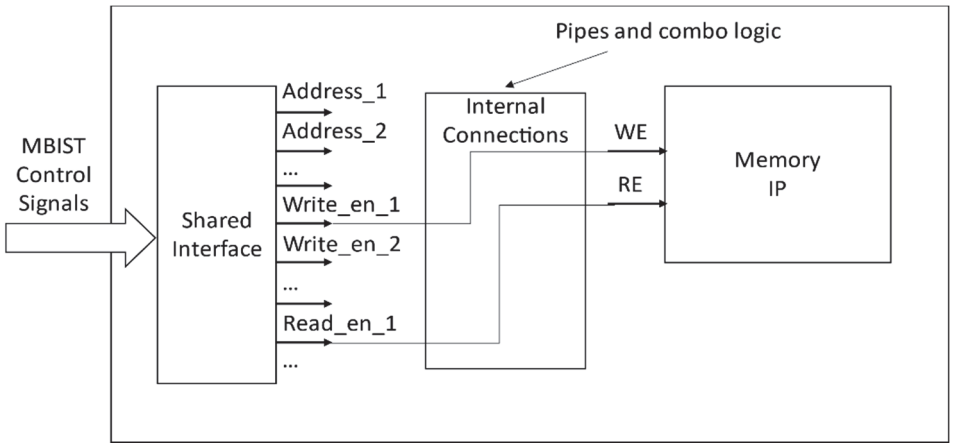

Fig. 3. The bus to the memory connection for a single port memory

If the memory has multiple read and write ports, all read and write ports should be connected to some of the interface outputs. If the memory has two read and write ports then all these ports should be accessed by the shared interface as in Fig. 4, otherwise the memory test is forced to work with the memory as with a single-port memory.

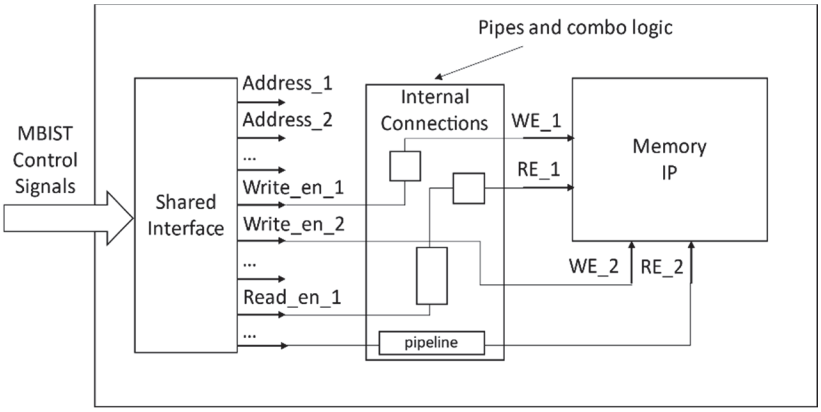

Fig. 4. A multi-port memory connection

4. Validation. Once the MBISTR engine is created, it should be validated: whether the memories in the SoC are tested properly. The validation of such systems includes the generation of the SoC, MBISTR engine, and specific steps of simulation for test algorithms and test environment. From this point of view the SoC generation should create memories with corresponding connections to the shared interface. Next, the MBISTR engine generation step should create a corresponding controller or processor which applies test algorithms to the SoC memories, i.e., should execute these algorithms.

The validation flow has also other steps such as logic synthesis and post synthesis simulation steps which do not impact the considerations in this paper and are not discussed here.

5. A proposed solution. For increasing the scope of validation of the MBIST processor, the SoC generation scenarios must sensitize a large number of different cases, particularly, covering an important branch of testing multi-port memories.

In general, the multi-port memories require more than one driver on the shared interface. For example, if there are two dual-port memory instances which have 64-bit data port (Fig. 5), the shared interface must have two 64-bit data buses.

The simplest way of testing multi-port memories is to apply test algorithms on each memory port for each memory instance located in the SoC. As it was noticed above some of the MBISTR engine buses are factually unused during the validation of test functionality which leads to additional test time and, thus, test time wise is not efficient.

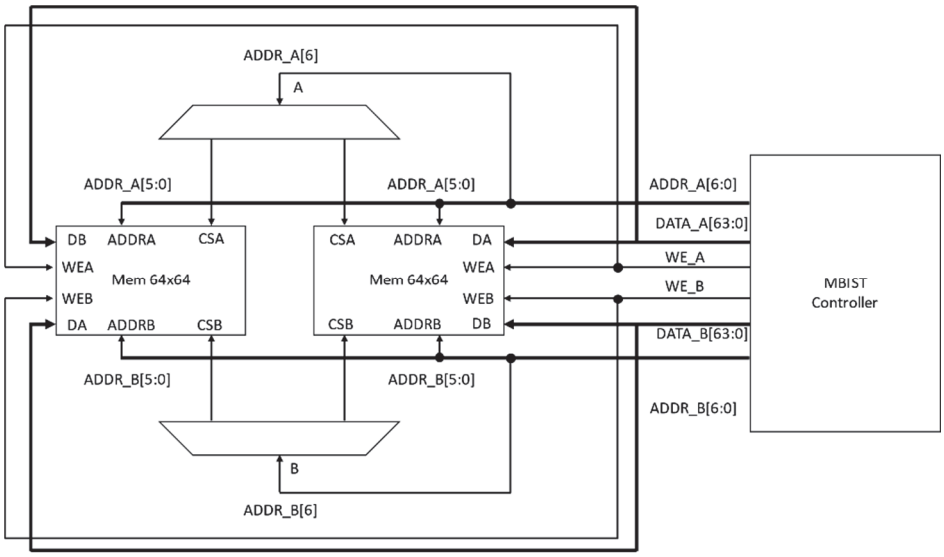

Fig. 5. Two dual-port memories with MBIST controller

On the other hand, there are algorithms that use different ports of one memory instance simultaneously during the memory stress test when the MBISTR engine buses are used for. Nevertheless, it is possible to optimize validation time for the cases, where only one of the memory ports is used.

Let's suppose that an algorithm's execution time is T_A for one port. It is evident that the test time of this algorithm for two dual-port memories is $4T_A$ when each port is tested separately. This time can be reduced if different memory ports of different instances are tested at the same time. To perform this, an additional

switching logic should be added to the MBIST processor. This switching mechanism decides whether the access sharing between multiple memory instances is possible. For the described case, the simulation time is twice reduced by setting different chip select (CSA,CSB) values on the memories. As a result, the test sequence will be executed in this order.

- Run test algorithm with CSA0 := 1, CSA1 := 0; CSB0 := 0, CSB1 := 1.
- Run test algorithm with CSA0 := 0, CSA1 := 1; CSB0 := 1, CSB1 := 0.

Finally, if there are N=pk memory instances (k is a natural number) with p > 1 ports, the test time will be reduced by p times.

In fact, the switching unit in the MBIST processor controls the chip select sharing mechanism. It distributes memory chip select signals if the algorithm does not require multiple port access.

Without loss of generality, further considerations will be carried out for three-port memories. The conventional and proposed approaches for testing these memories through A,B and C ports are presented below (Fig. 6, Table 2).

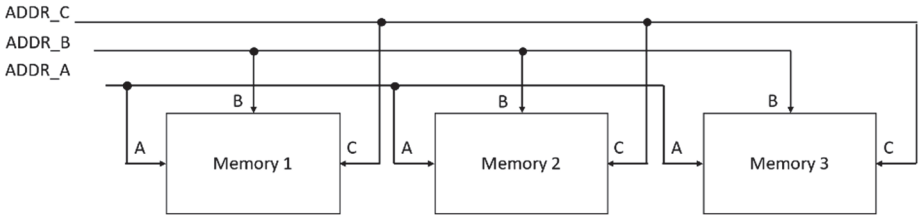

Fig. 6. Connection of three port memories

Table 2

Test Sequence Execution

Step	Access Port Conventional Method	Access Port Proposed Method
1	A1 (A port, first memory)	A1,B2,C3 (3 different memories)
2	B1 (B port, first memory)	A2,B3,C1 (3 different memories)
3	C1 (C port, first memory)	A3,B1,C2 (3 different memories)
4	A2 (C port, second memory)	-
5	B2(C port, second memory)	-
6	C2(C port, second memory)	-
7	A3(C port, third memory)	-
8	B3(C port, third memory)	-
9	C3(C port, third memory)	-

Via this approach, the number of test sequence executions is reduced from 9 to 3. If there are 6 three-port memories, the number of iterations is reduced from 18 to 6. To activate these A, B and C ports, the corresponding chip-select values should be applied in such a way (Table 3).

Table 3

Testing three-port memories

STEP	Chip Select	Memory 1	Memory 2	Memory 3
1	CSA	Active	Inactive	Inactive
	CSB	Inactive	Active	Inactive
	CSC	Inactive	Inactive	Active
2	CSA	Inactive	Active	Inactive
	CSB	Inactive	Inactive	Active
	CSC	Active	Inactive	Inactive
3	CSA	Inactive	Inactive	Active
	CSB	Active	Inactive	Inactive
	CSC	Inactive	Active	Inactive

Actually, the shortening of the validation time reduces on-chip test time too. The switching unit allows the memories to be tested in parallel. Thus, the additional hardware which makes the validation faster does not affect actual memory testing but improves it.

Note that this approach is not applicable for the test algorithms which require access to few memory ports simultaneously, e.g., for testing the memory under stress conditions when an initial approach to the test should be used.

Let us consider an example when there are 10 test algorithms and 2 of them have simultaneous operations on multiple ports (Fig. 7). For the rest - 8 test algorithms, only one port is used during the test run. We consider a memory with 1024 words; each operation takes one cycle on each word. In the initial case where no support is present, each algorithm spends T_{Ai} time. Using the proposed approach, the 8 algorithms will be run two times faster than the initial algorithms. The overall test time is reduced by 37.72%.

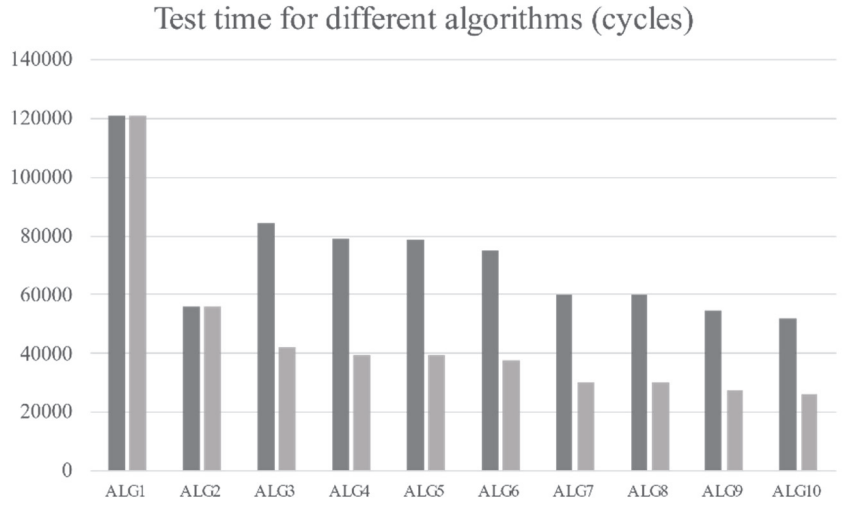

Fig. 7. Test time reduction for dual-port memories

This solution is not applicable for the flows where there are no test algorithms accessing only one of the memory ports.

Conclusion. For multi-port memories tested via a shared interface a corresponding MBISTR engine validation method is suggested for test algorithms that do not deal simultaneously with multiple ports. This method requires some minor changes in the MBISTR engine and reduces the validation time ***p*** times as well as the on-chip test time where ***p*** is the number of memory access ports.

REFERENCES

1. 2015 International Technology Roadmap for Semiconductors (ITRS).
2. **Zorian Y., Shoukourian S.** Embedded-memory test and repair: infrastructure IP for SoC yield // IEEE Design & Test of Computers. – May-June, 2003. – Vol. 20, no. 3. – P. 58-66.
3. **Shoukourian S., Vardanian V., Zorian Y.** SoC Yield Optimization via an Embedded-Memory Test and Repair Infrastructure // IEEE Design & Test of Computers. - May-June 2004.-Vol. 21, no. 3. - P. 200-207.
4. https://resources.sw.siemens.com/en-US/video-automating-physical-mapping-on-arm-ip-with-tessent
5. https://news.synopsys.com/2014-10-21-Synopsys-STAR-Memory-System-Multi-Memory-Bus-Processor-Enables-10-Percent-Die-Size-Reduction-for-Marvell-SoC
6. **McLaurin T., Knoth R.** The Challenges of Implementing an MBIST Interface: A Practical Application // IEEE International Test Conference (ITC) . – 2019. – P. 1-6.

National Polytechnical University of Armenia. The material is received on 06.09.2022.

Ա.Վ. ԲԱԲԱՅԱՆ

ԸՆԴՀԱՆՈՒՐ ԿԱՊՈՒՂԻՈՎ ՄԻԱՑՎԱԾ ՀԻՇՈՂ ՍԱՐՔԵՐԻ ՆԵՐԿԱՌՈՒՑՎԱԾ ԹԵՍՏԱՎՈՐՄԱՆ ՀԱՄԱԿԱՐԳԻ ՎԱՎԵՐԱՑՄԱՆ ԵՎ ԹԵՍՏԱՎՈՐՄԱՆ ԺԱՄԱՆԱԿԻ ԿՐՃԱՏՄԱՆ ՄՈՏԵՑՈՒՄ

Հիշող սարքերի ներկառուցված թեստավորման և վերանորոգման համակարգը (MBISTR) կարևոր բաղադրիչ է բյուրեղի ելքի տոկոսը բարձրացնելու համար՝ ինտեգրալային սխեմաների արդյունաբերության նորանոր մարտահրավերների պայմաններում, որտեղ հիշող սարքերը գերակշիռ մաս են կազմում են: Համակարգում առկա բազմաթիվ հիշող սարքերը հարմար է ստուգել ընդհանուր թեստավորման ցանցի միջոցով: Միևնույն ժամանակ, որոշակի նախագծերում չի թույլատրվում ավելացնել այս թեստավորող ցանցը նոր միացումներով, փոխարենը՝ առաջարկելով օգտագործել առկա ֆունկցիոնալ միացումներով ինտերֆեյսները: Օրինակ, կենտրոնական մշակման հանգույցը (CPU) և քեշ հիշողությունները իրար միացնող ընդհանուր կապուղին և կապուղն միացված հատուկ MBISTR

համակարգերն արդեն իսկ օգտագործվում են այդ հիշողությունները թեստավորելու համար: Այս MBISTR համակարգերն ունեն որոշակի առանձնահատկություններ, որոնք կարող են զգալի կերպով ազդել վավերացման և թեստավորման ժամանակի վրա:

Առաջարկվում է ընդհանուր կապուղուն միացված բազմապորտ հիշողությունները թեստավորող համակարգի վավերացման և թեստավորման ժամանակի կրճատման մոտեցում:

Առանցքային բառեր. հիշողությունների թեստավորում, թեստավորում և վերանորոգում, բազմապորտ հիշողությունների թեստավորում, ընդհանուր կապուղով ճարտարապետություն, թեստավորման և վավերացման ժամանակի կրճատում, համակարգ բյուրեղի վրա, զուգահեռ թեստավորում:

А.В. БАБАЯН

ПОДХОД К СОКРАЩЕНИЮ ВРЕМЕНИ ПРОВЕРКИ И ТЕСТИРОВАНИЯ ДЛЯ ВСТРОЕННОЙ СИСТЕМЫ САМОТЕСТИРОВАНИЯ И ВОССТАНОВЛЕНИЯ ПАМЯТИ ЧЕРЕЗ ОБЩИЙ ИНТЕРФЕЙС

Встроенная система самотестирования и восстановления памяти (MBISTR) является ключевым компонентом в повышении производительности системы на кристалле (SoC) в условиях появления новых задач в промышленности интегральных схем, где память очень часто составляет основную часть кристалла. Когда у данного кристалла огромное количество памяти, удобно тестировать их через общую тестовую сеть. Между тем некоторые проекты не позволяют вставить эту тестовую и восстановительную сеть в уже существующий функциональный проект путем добавления новых узлов в существующие, но вместо этому позволяют добавить ее в проект через уже существующие функциональные соединения с интерфейсом. Например, общий интерфейс для подключения центрального процессора (CPU) и кэш-памяти в кристалле уже используется для тестирования этой памяти через специальные механизмы MBISTR, подключенные к общему интерфейсу. Эти механизмы MBISTR имеют определенные функции, которые могут существенно повлиять на время проверки и тестирования.

Предлагается способ сокращения времени проверки и тестирования многопортовой памяти, подключенной к механизму MBISTR через общий интерфейс.

***Ключевые слова*:** встроенное самотестирование памяти, тестирование и восстановление, тестирование многопортовой памяти, архитектура с общим интерфейсом, сокращение времени проверки и тестирования, система на кристалле, параллельное тестирование.