

GRAAL: A TOOL TO DESIGN HIGHLY DEPENDABLE SRAM ARCHITECTURE FOR AEROSPACE APPLICATIONS

A.S.Brogna, P.Bellandi¹⁾, F.Bigongiari¹⁾, S.Chiusano²⁾, P.Prinetto²⁾, R.Saletti

University of Pisa – Dip. Ingegneria dell’Informazione,
Via Diotallevi, 2 – 56122 Pisa, Italy

¹⁾ Aurelia Microelettronica S.p.A. – a C.A.E.N. S.p.A. Company
Via Giuntini, 13 – 56023 Navacchio (PI), Italy

²⁾ Politecnico di Torino – Dip. di Automatica e Informatica,
C.so Duca degli Abruzzi, 24 – 10129 Torino, Italy

Abstract

The scientific space missions require high-speed and low-power on-board data storage capabilities that are obtained with cheap and non qualified Components Off The Shelf (COTS) memories, in which a change in the stored data can unfortunately occur because of the radiation effects. This paper presents an implementation tool (GRAAL) to design a high-reliability memory architecture that consists in a wrapper placed around the target memory. The memory collar together with the target memory behave like a *Dependable Memory Core* that can be embedded in complex systems.

1. Introduction

Memories are key components in highly dependable applications as space, defense, automotive and biomedical systems, since they play a crucial role in terms of *availability* and *serviceability*. They appear in a big variety of sizes, technologies (SRAMs, DRAMs, ...) and packaging (IP cores, chips, dedicated boards). However, regardless of their actual implementation, memories are the most critical devices with respect to both *permanent* and *transient* faults due to the environmental stresses and interferences.

GRAAL (a high-reliability RAM cell Generator for Life and safety-critical Applications) is a tool for the automatic generation of highly dependable memories, by which the designer can define a *dependable SRAM architecture*, which achieves the target dependability requirements and satisfies design constraints. *Built-In-Self-Test* (BIST) logic for *OFF-line* and/or *ON-line* memory testing is included. In the case of *ON-line* testing, both *Concurrent* and *Not-Concurrent* test strategies are supported.

Moreover, the dependable SRAM architecture can also include the *Built-In Self Repair* (BISR) logic for functional memory repairing. The memory repairing functionality is out of the purpose of the GRAAL project, however a dependable SRAM architecture could include the BISR logic, if available.

Commercial tools are presently available for the automatic insertion of testing architectures in complex systems [1], [2], [3]. Memory testing is one of the aspects addressed by the tools, which typically propose a limited set of testing solutions. They mainly allow the insertion of BIST logic for OFF-line testing, but only a few March-Tests are supported. Moreover, none of them usually provides the insertion of logic for ON-line memory testing. Therefore, even if their effectiveness is demonstrated in the majority of the applications, these tools do not guarantee the proper reliability levels in safety critical systems including memories. GRAAL is complementary to the available commercial tools, since it tackles aspects usually not covered in memory testing, such as the automatic insertion of OFF-line and ON-line BIST architectures. The set of architectures can easily be extended to meet the dependability requirements or to include the most innovative and recent testing algorithms and strategies.

The innovative aspect in GRAAL is its flexibility. It has been designed, developed and implemented targeting the following issues:

- *independence* on target technology, system embedding the memory and EDA design environment;
- *upgradability*, *maintainability* and *advanced design reusability* because of open architecture;
- *dependability-cost trade-off* in the design process.

To achieve the above issues, we defined a suitable dependable memory architecture: the target memory is wrapped by a *dependable memory collar*, embedding the logic to guarantee the required dependability levels, which acts as interface between the memory and the external environment. The collar is an open architecture, with a highly modular structure, with no limitations on its possible implementation. The proposed approach achieves the *upgradability* and *maintainability* properties, as well as the *independence* on both the *embedding system* and the *target memory*.

During the design flow, the designer selects one of the architectures stored into the library. Using GRAAL, the designer can identify the possible architecture that meets the design requirements, trading-off dependability and implementation costs.

2. The Dependable Memory Architecture

The dependable SRAM architecture consists in a wrapper placed around the target RAM that is designed to achieve the target design and dependability requirements. Such a wrapper, named *Dependable Memory Collar* (DMC), interfaces on one hand the memory itself and on the other hand the system environment in which the memory is located. The DMC together with the target SRAM represent a *dependable SRAM core* that includes the *Built-In-Self-Test* (BIST) logic for memory testing in both cases of *OFF-line* as well as *ON-line* testing strategies. In the case of ON-line testing, the implementation of both Concurrent and Not-Concurrent test strategies is supported. Inside the DMC, the OFF-line and ON-line testing strategies can be implemented either individually or combined. The DMC supports the execution of both internal and external tests.

Moreover, the wrapper also includes the *Built-In Self Repair* (BISR) logic for functional memory repairing, useful in applications requiring very high reliability levels. The Repairing Layer will be mentioned in the rest of the work, but no architectures will be presented for it.

The proposed approach relies on a set of elementary building blocks, each one targeting different functionalities of the collar. The blocks have been designed in such a way to be developed, updated and debugged as standalone blocks, but, at the same time, as easily *composable* to create a proper collar. Ad-hoc defined *rules* drive the building process, and collars with various characteristics can be generated composing the blocks in different ways. The elementary building blocks in the DMC are named *Layers*.

As sketched in Fig. 1, the DMC consists in a four layers hierarchy, in which the layers are placed in an *onion skin-like* mode. The *SRAM* represents the target memory and the *System Environment* is the application context in which the dependable SRAM core is embedded. The DMC is first decomposed into three layers: a middle layer named *Test Layer* (TL) and two external layers, named *System Interface Layer* (SIL) and *Memory Interface Layer* (MIL).

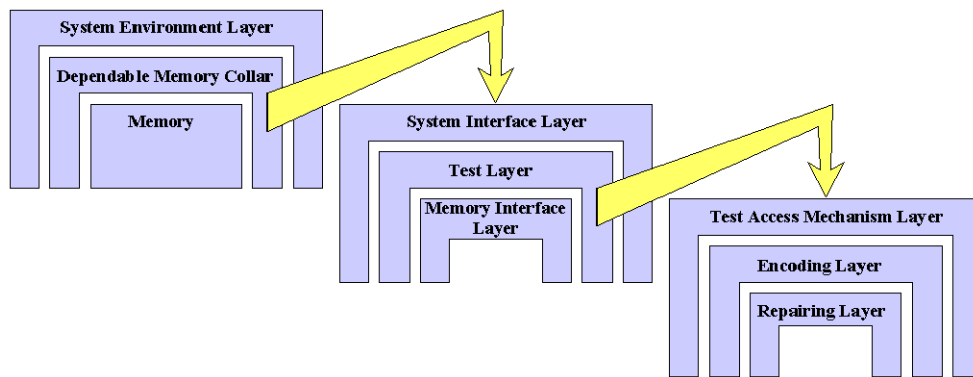


Fig. 1: The GRAAL Architectural Diagram

The *Test Layer* is the kernel of the architecture since it contains the BIST testing logic; it allows the execution of OFF-line as well as ON-line (both Concurrent and Not-Concurrent) memory testing. Moreover, it contains the BISR logic for functional memory repairing. The remaining two wrappers, the System Interface Layer and the Memory Interface Layer, have been introduced to create a standard interface between the Test Layer and both the external system environment and the target SRAM.

The *System Interface Layer* (SIL) interfaces the Test Layer with the external system environment in which the dependable SRAM core will be located. The SIL is in charge of sending/receiving control/data signals between the Test Layer and the system environment, using the specific I/O transmission protocol

required by that particular system environment. During this process, the SIL also inserts the appropriate timing information, according to the communication protocol of the system environment. The SIL is not fixed but it has to be designed for each specific system environment.

The *Memory Interface Layer* (MIL) interfaces the Test Layer and the target SRAM. The MIL is in charge of sending/receiving control/data signals between the Test Layer and the SRAM, using the I/O transmission protocol required by that specific SRAM. During this process the MIL also inserts the appropriate timing information, according to the communication protocol of the target SRAM. The MIL also is not fixed but it has to be designed for each specific target SRAM. Let us assume that for each specific memory, a Memory Interface Layer is provided.

2.1 The Test Layer structure

The *Test Layer* (TL) is the kernel of the dependable SRAM architecture. It includes the Built-In-Self-Test (BIST) logic for memory testing. The OFF-line and ON-line testing methodologies can be implemented individually or they can be combined. In the case of ON-line testing, the implementation of both Concurrent and Not-Concurrent test strategies is supported. The Test Layer supports the execution of both internal and external tests.

Finally, the Test Layer can also include the Built-In Self Repair (BISR) logic for functional memory repairing. The repairing strategy is typically combined with at least one of the testing strategies, in order to be able to find out the occurrence of faulty cells. Since the design of BISR logic was out of the scope of the GRAAL project, no architecture of it is presented.

To further increase the modularity of the dependable memory wrapper, the Test Layer has been organized as a hierarchy of three different layers, as shown in Fig. 1. Each layer embeds the logic to implement a different testing functionality and guarantees different dependability properties.

- The Test Access Memory (TAM) Layer: includes the BIST logic for *OFF-line* and *Not-Concurrent ON-line* memory testing. As an example, different architectures of the TAM layer can be obtained implementing different kind of March Tests.
- The Encoding Layer (EL) includes the BIST logic for *Concurrent ON-line* memory testing. The Encoding layer is mainly based on encoding the data stored into the memory. Using the code, the Encoding Layer verifies the correctness of the memory data whenever a memory cell is read.
- The Repairing Layer: includes the logic for *functional memory repairing*.

The TAM Layer interfaces with the SIL, and the Repairing Layer with the MIL.

Different implementations of the Encoding Layer can be obtained resorting to different algorithms for data encoding: the Encoding Layer not only detects faulty memory cells, but it can also correct errors.

Each layer represents a sort of black box, which can be filled by different architectures, able to guarantee the functionality proper of the layer, but differing in terms of design and dependability properties. The design properties are technology-dependent characteristics, as area and power consumption. The dependability properties include the performed testing strategies, the detected faults, and the capability of correcting the memory data when affected by a fault. As an example, different architectures for the TAM layer are obtained depending on the executed March-Test [4]; in the case of the EL, the implementations differ according to the implemented encoding algorithm [5]. Finally, the MIL and the SIL are specific for each target SRAM and system environment.

To establish the architecture of the Test Layer, the designer defines the structures of its three layers. Each layer can be “bypassed” if the designer does not include the corresponding functionality. The TL contains all the dependability logic and therefore represents the kernel of the DMC. The MIL and the SIL synchronize data exchanged between the TL and the upper and lower layers, and converts the TL inputs and outputs into the proper protocol required by the memory and the system environment, respectively.

The main advantages of the proposed approach relies on the fact that the Test Layer can exchange control/data signals using a fixed I/O communication protocol, which is independent of the specific I/O transmission protocols required by any system environment and target SRAM.

The introduction of two interfacing wrappers makes the Test Layer independent of the specific system environment in which the memory is located as well as of the specific technology selected to implement the memory. As a consequence, the Test Layer does not have to be redesigned when changing either the characteristics of the system environment or the memory implementation technology.

Using the proposed approach, the designer can define a Test Layer to be applied to a SRAM functionally characterized by a given size (i.e. number of cells and memory words), in order to achieve certain design and dependability parameters; then, the Test Layer can be embedded in different

application contexts by redefining each time the System Interface Layer, only. Moreover, the Test Layer can be applied to the SRAM, independently of the selected implementation technology. The designer has only to redefine an appropriate Memory Interface Layer, if the selected memory technology is characterized by a different I/O communication protocol.

3. Performance evaluation

The GRAAL development and its debugging process is completed: some fault injection experiments validate the proposed DMC structure as well as the tool, and show the expected design performances.

To execute the experiments, we developed a set of architectures to be stored in the Knowledge Library. We developed architectures executing a set of March-Tests and a Transparent Test [6] for the TAML, and architectures executing the parity code and the Hamming codes [5] for the EL. Combining the TAML and EL architectures, we are able to generate up to 32 different structures of the dependable collar.

Concerning the area introduced by the DMC, it mainly consists of the testing logic located in the TL. We verified that the TL area is almost equivalent to the area introduced by architectures manually designed resorting to the same testing algorithms and structures. Therefore, the automation of the design process does not increase the area overhead. Additional area is given by the SIL and MIL. However, area is negligible since these two layers mainly perform signal synchronization and protocol conversion. Moreover, the designer when embedding the testing architecture together with the memory in a system typically inserts a wrapper with functionality equivalent to the SIL. Finally, commercial tools addressing the insertion of memory BIST logic often insert a wrapper to interface the inserted logic and either the memory or the environment. The DMC introduces a negligible extra delay when the system is in normal mode and thus it does not degrade significantly the performance.

Finally, as far as the design time is concerned, the execution of the design flow strongly depends on the designer experience. The tool addresses both expert and not expert designers. The former can exploit the interactivity of the tool to progressively identify the target architecture by specifying the target dependability properties and modifying the design at each step of the flow. The latter can avoid the interactivity, speeding up the design flow. However, the time spent for the automatic generation of the memory collar is negligible with respect to the time needed to manually design an equivalent architecture.

4. Conclusions

This paper presented a tool named GRAAL for the automatic generation of highly dependable memories. A proper structure has been defined for both the tool and the dependable memory, in order to achieve a very high degree of flexibility. The easy upgrade and maintainability of the tool makes it open to always support the most advanced testing strategies and architectures; the design flow allows the designer to identify the proper architectures, trading off dependability and cost. As compared to the commercial tools for automatic insertion of testing structures, GRAAL allows inserting logic for ON-line memory testing. Therefore GRAAL is a suitable tool to achieve the proper reliability levels in safety critical systems using memories.

The main advantages of the proposed approach is based on the fact that the Test Layer is internally structured into three layers, which exchange signals using a fixed I/O communication protocol: therefore, GRAAL has an architecture flexible and open because it is obtained by combining the architectures of the three layers. The architecture of each layer can be defined almost independently of the other layers. Moreover, the implementation of each layer can be *modified* without affecting the implementation of the remaining layers. GRAAL is not customized in implementing a few and specific testing strategies, but different testing approaches can be realized. In fact, the Test Layer is ready to implement innovative and new testing and/or repairing algorithms and solutions.

5. References

- [1] LogicVision ©, *memBIST-IC User Manual*, January 1999
- [2] Mentor Graphics Corporation ©, *MBISTArchitect Manual*, 1999
- [3] www.synthest.com
- [4] A.J. Van de Goor, *Testing semiconductor memories: theory and practice*, Wiley, 1991
- [5] F. J. Macwilliams, N. J. A. Sloane, *The Theory of Error-Correcting Codes II*, North-Holland Mathematical Library, Vol.16
- [6] M. Nicolaidis, *Theory of transparent BIST for RAMs*, IEEE Transactions on Computers, vol.45, n.10, Oct. 1996, Page(s): 1141-1156