



# FORUM

Vol. 13 No.4 Dec. 2006  
Published by FSA \$60 (U.S.)

*BEST PRACTICES FOR COST-EFFECTIVE TEST AND YIELD OPTIMIZATION OF EMBEDDED MEMORIES*

*DESIGN METHODOLOGY IS THE KEY TO SUCCESS AT 90NM*

*COLLABORATION BETWEEN FABLESS AND FOUNDRY LEADS TO SUCCESSFUL SOP AND SIP SOLUTIONS*

*BENCHMARKING SEMICONDUCTOR MANUFACTURING*

*UNDERSTANDING THE VALUE OF INDUSTRY STANDARDS FOR PREDICTING PROCESSOR PERFORMANCE*



*Benchmarking for Success*

# BEST PRACTICES FOR COST-EFFECTIVE TEST AND YIELD OPTIMIZATION OF EMBEDDED MEMORIES

STEPHEN PATERAS, SENIOR DIRECTOR, CORPORATE PRODUCT MARKETING, LOGICVISION, INC.

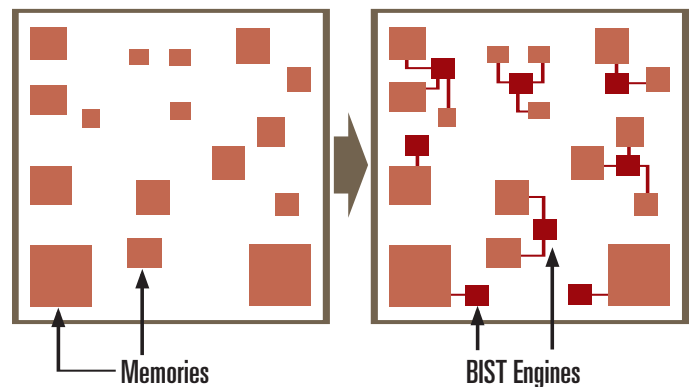
The industry is experiencing increasing quality and yield challenges as it moves to deep nanometer scale process nodes. As embedded memories represent a significant and growing percentage of the die, memory test and yield optimization solutions are of particular importance. A comprehensive embedded memory solution must incorporate not only high-quality memories, but also highly effective embedded test and repair capabilities.

One of the oldest and most commonly used forms of design for test (DFT) is memory built-in self-test (BIST). In its simplest form, memory BIST consists of an on-chip engine placed next to each embedded memory that writes algorithmically generated patterns to the memory and then reads these patterns back to discover and possibly log any defects. The memory BIST engine is typically designed to generate patterns based on a pre-determined memory test algorithm. Over the years, commercial memory BIST solutions have evolved from this simple approach to very sophisticated automation and intellectual property (IP) solutions needed to deal with today's complex design and test challenges. Several important capabilities should be evaluated when choosing a memory BIST solution to ensure a cost-effective memory test and yield optimization strategy.

## DESIGN AUTOMATION

Many of today's designs have literally hundreds of memories of different types and sizes spread throughout the chip. Adding test and repair capabilities to these memories can be a daunting task. To ensure both a timely and efficient integration of test and repair capabilities, it is important that a comprehensive automation flow be adopted. With such a large number of memories on a chip, it becomes necessary to share BIST engines across multiple memories. The optimum allocation of BIST engines to memories will depend on several factors, including the physical location of the memories, the clock domains within which the memories reside, the power ratings of the memories, the size and type of each memory, and the maximum desired test time. A key automation capability is the ability to automatically take these factors into consideration and create an optimal allocation of BIST engines to memories (Figure 1).

Figure 1. Optimum Allocation of BIST Engines



This allocation should also properly configure each BIST engine to support the memories allocated to it. Having this kind of automation can reduce memory BIST integration time on large designs from days to hours.

## TEST ALGORITHM PROGRAMMABILITY

Continued decreases in process feature sizes and associated increases in memory densities are resulting in a growing number of memory defect types. Many of these new defect mechanisms are difficult to predict and are therefore difficult to test. Unfortunately, these defects are increasingly being discovered during the production testing of a device or, worse, during the analysis of field returns. This can result in significant quality and cost issues if the predetermined test algorithm used by the BIST engine does not detect a newly discovered defect type. In many cases, functional tests have to be added to the manufacturing test flow at significant cost. Because of this growing problem, some commercial memory BIST solutions now provide programmable BIST engines. With these engines it is possible to download (on the tester or in system) program code that implements an arbitrary memory test algorithm. This allows new or enhanced algorithms to be applied, as needed, to specific memories as new defect mechanisms need to be addressed. To maintain a simplified manufacturing test flow, these programmable BIST

engines will typically support predetermined default algorithms as well. This removes the need to program the BIST engine if the default algorithm is sufficient. Only when new defect mechanisms are discovered does it become necessary to program each BIST engine before having it execute the memory test. The programmable BIST engines are larger than the more traditional hard-coded ones and therefore should only be used when the need is justified. Usage tends to be with either very large memories or very dense ones such as the increasingly popular 1T embedded DRAMs.

## SELF-REPAIR

Device yields are increasingly dependent on embedded memory yields, as the memories not only represent the densest components of the chip, but also represent a significant and growing percentage of the die. Increasingly, therefore, embedded memories are being fitted with spare elements – spare rows, columns or both. These spare elements can then be used to replace corresponding faulty elements and salvage the memory.

The manufacturing flow for supporting a memory repair strategy can be expensive. During wafer sort, memories not only have to be tested, but all failure information must be extracted and analyzed to determine if the memory is repairable and how to repair it (which elements to replace with available spares). Once the necessary fuse-based repair information is calculated, the wafer containing the bad die must be moved to a laser repair station to blow the necessary fuses for each bad die. The wafer must then be retested to ensure all the repairs were successful. Alternatively, the good and repaired die can be sliced and packaged and sent to final test for verification. In the latter case, there will be additional (and expensive) final test fallout, as some unsuccessfully repaired die will have been packaged.

Advanced commercial memory BIST solutions now support a fully embedded memory repair flow. This approach, referred to as built-in self-repair (BISR), can significantly reduce, if not completely eliminate, the complexities and costs described above. The most advanced BISR solutions will test and permanently repair all defective memories in a chip using no external resources. The on-chip architecture for one of these solutions is shown in Figure 2. Central to this architecture is the concept of a programmable fuse pool. Electrical or programmable fuses are becoming increasingly

popular as they are smaller than laser-based fuses, and they can be programmed without the need of any external (laser) equipment. Pooling fuses together is also becoming popular to reduce overhead. Because most memories will typically need little to no repair on any given die, sharing a pool of fuses for all memories allows for much better fuse utilization. Memories needing little to no repair will require little to no fuse information to be stored, freeing the fuse storage for other memories. To simplify the fuse data allotment, standard data compression techniques are used to implicitly allocate the necessary amount of fuse storage per memory.

A fuse box controller performs on-chip management of a centralized programmable fuse pool. This controller, along with one or more BIST engines, performs all necessary activities for testing and repairing memories. In this architecture, the BIST engines are enhanced not only to test the memories, but also to analyze how to repair faulty ones. This capability is typically referred to as built-in repair analysis (BIRA). A typical flow with these embedded capabilities is as follows:

1. BIST engines are executed (typically all in parallel) to test and analyze the repair requirements for their assigned memories. The repair analysis generates fuse data that must be applied to the repair port of each memory. This fuse data is stored in local BIRA registers.
2. The data in the local BIRA registers is transferred to serial BISR registers that are connected to a chip-wide serial BISR register chain, which in turn is connected to the fuse box controller.
3. The fuse box controller scans and compresses the incoming fuse data from the serial BISR register and programs this compressed data into the fuse pool.
4. To verify the programmed contents of the fuse pool and the repair analysis, the fuse box controller is then used to scan out and uncompress the fuse data from the fuse pool back into the serial BISR registers.
5. With local BISR registers now driving the correct repair input to all memories, the BIST engines are executed again to make sure all memories are properly repaired.

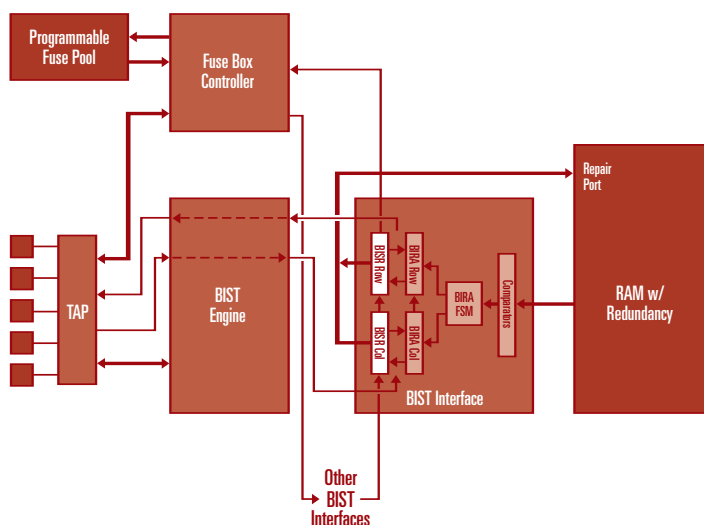
The five steps above are all performed in a single wafer sort test insertion. The need for any fail data extraction and analysis, any additional test insertions or any additional external equipment is completely eliminated.

The fuse box controller is also used at every power-on-reset (POR) event to sequence the fuse data in the fuse pool to all memories needing repair. This is done using Steps 4 and 5 above. This architecture also supports incremental repair. That is, the fuse data stored in the fuse pool can be used as a baseline on every POR event. The BIST engines can then be used to detect any new failures, and the combined fuse data can be stored in the local BISR registers to repair the memories. This approach allows for improving long-term device reliability.

## DIAGNOSTICS

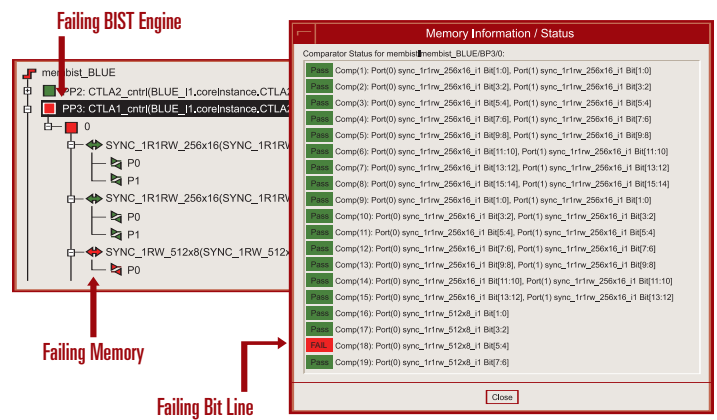
The ability to precisely and efficiently diagnose memory failures is a key component of understanding and correcting yield issues. This is becoming particularly important as nominal yields continue to drop

**Figure 2. Self-Repair Architecture**



with each successive process node. Therefore, a highly automated memory diagnostic approach is a key ingredient of an embedded memory strategy. The most advanced diagnostic solutions consist of both an IP and software component. The IP component refers to additional capabilities within the BIST engine for the logging of failure data. The most common of these is a “stop-on-error” capability. This enables the BIST engine to stop on the detection of each error so all pertinent failure information, such as the failing row and column addresses, can be scanned out of the engine and logged on the tester. Making use of this embedded diagnostic capability can be a significant challenge without appropriate automation. Correctly generating the necessary tester patterns to control all of the stop-on-error activity within a specific BIST engine connected to a specific memory can be both time consuming and error prone. For this reason, a diagnostic solution that also incorporates control software running directly on the tester is highly beneficial. The graphical interface of one such commercial solution is shown in Figure 3. In this system, the BIST engines and memories they test are shown graphically. Test and diagnostic results also are displayed graphically using a color-coding approach, as well as placed into a datalog file for later analysis. The most important aspect of this system, however, is the completely automated and seamless diagnostic environment it provides. Diagnostic operations are specified through the graphical interface, and feedback is provided in real time. No test patterns need to be generated, and no tester fail data output has to be processed or analyzed. The software automatically handles all interactions with the BIST engines and processing of failure data.

**Figure 3. Automated Diagnosis Interface**



A number of aspects must be considered when choosing and implementing an embedded memory test and yield management solution. When dealing with large numbers of memories and the more advanced process nodes, adopting the right solution can result in significant cost and time-to-market savings. ■

#### **About the Author**

*Stephen Pateras, senior director of corporate product marketing at LogicVision, Inc., has more than 15 years of extensive background in DFT, BIST and fault tolerance, and holds many journal and conference publications in these areas. Dr. Pateras received his Ph.D. in electrical engineering from McGill University. He can be reached at pateras@logicvision.com.*