Embedded Memory Test & Repair: Infrastructure IP for SOC Yield

Yervant Zorian

Virage Logic
46501 Landing Parkway, Fremont, CA 94538, USA
zorian@viragelogic.com

Abstract:
Today’s System-on-Chip typically embeds memory IP cores with very large aggregate bit count per SoC. This trend requires using dedicated resources to increase memory yield, while containing test & repair cost and minimizing time-to-volume. This paper summarizes the evolution of such yield optimization resources, compares their trade-offs, and concentrates on on-chip Infrastructure IP. To maximize the repair efficiency, this Infrastructure IP need to leverage the memory design knowledge and the process failure data. The ideal solution is to integrate the memory IP and its Infrastructure IP into a single composite IP that yields itself effectively.

1. Introduction:
The growth in demand for System-on-Chips (SoCs) has spurred a flood of better, faster, smaller chips. The creation of such SoCs necessitates finer fabrication processes and new material. One of the greatest obstacles to the widespread adoption of SoCs based on such processes is achieving adequate manufacturing yields. Today’s semiconductor fabrication processes are reaching defect susceptibility levels that result in lowering SOC yield and reliability, consequently lengthening production ramp-up period, and therefore time-to-volume (TTV).

In order to optimize yield and reach acceptable TTV levels, the semiconductor industry started adopting advanced yield optimization solutions. These are solutions that are implemented at the design stage and utilized at different phases of the chip realization flow [17]. These advanced yield optimization solutions necessitate embedding a special type of IP blocks in a chip. Such IP blocks are called Infrastructure-IP (I-IP) [18]. Unlike the Functional-IP (F-IP) cores used in SoCs, such as embedded processors, embedded memories, or embedded FPGAs, the I-IP cores do not add to the main functionality of the chip. Rather, they are intended to ensure the manufacturability of the SoC and to achieve lifetime reliability [19]. The industry in the recent past has introduced a wide range of I-IP blocks. Examples of such I-IP blocks are process monitoring IP [3], test & repair IP, diagnosis IP [9], timing measurement IP [13], and fault tolerance IP [4]. Embedding such blocks into the SoC design helps optimize yield and reliability during manufacturing and in the field. The reason I-IP cores are needed is due to the fact that the traditional semiconductor manufacturing, which is based on external equipments and processes, alone is insufficient to handle today’s yield optimization challenges.

The first usages of I-IP cores were either at the wafer level or at the top-level of an SoC [19]. However, the need to produce finer and denser chips moved certain I-IP cores towards decentralization at the SoC level, resulting in multiple distributed I-IP cores, where each I-IP is coupled with a corresponding F-IP core to monitor and control the peripheries of the F-IP, such as in the case of hierarchical embedded test methodology [9]. Today’s SoC complexity has reached a level causing certain types of I-IP to move even further, in order to be fully integrated with the corresponding F-IP block. This resulted in a new type of a single composite IP comprised of the integrated F-IP and I-IP. An example of this is today’s composite embedded memory cores that contain the test and repair I-IP integrated with the memory F-IP.

The reason for integrating memory cores with their corresponding test and repair IP is that, embedded memories in today’s SoCs typically need repair at manufacturing level, diagnosis for process improvement and field repair capabilities. Traditionally yield optimization was based on general-purpose test, diagnosis and repair solutions. However, these solutions have limited repair efficiency and are also very expensive, for today’s SOC processes. This created the need for a test and repair I-IP customized to the embedded memory and provided as an integral function of the embedded memory solution. Such a dedicated I-IP provides higher repair efficiency because it leverages the memory design knowledge and failure history.

This paper introduces the composite IP concept, i.e. integrated F-IP with its dedicated I-IP, and then presents an effective SoC yield optimization solution for embedded memories that are capable of providing manufacturing repair, diagnosis for process improvement, and field repair capabilities.
The paper is organized in the following way. Section 2 discusses the trends and challenges of today’s embedded memories. Section 3 introduces three yield optimization loops used at three different phases of the IC realization flow. Section 4 shows the evolution of embedded memory test and repair methodologies and compares their trade-offs. Section 5 introduces a new I-IP for embedded memories and describes its architecture and operation. Section 6 discusses the effectiveness of this I-IP on manufacturing repair, process improvement and field repair. Finally, Section 7 summarizes the impact of this composite IP on the SoC design challenges.

2. Embedded Memory Trends & Challenges:
Today’s SoCs are moving from logic-dominant chips to memory-dominant ones. Fig. (1) shows a plot extrapolated from the International Technology Roadmap for Semiconductors (ITRS) [1]. In year 2002, the usage of embedded memories is shown to be more than half of the die area for a typical SoC. They are predicted to take up even more space on the die in the future reaching up to 94% by year 2014.

Area Share

<table>
<thead>
<tr>
<th>Year</th>
<th>99</th>
<th>02</th>
<th>05*</th>
<th>08*</th>
<th>11*</th>
<th>14*</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Area Memory</td>
<td>40%</td>
<td>32%</td>
<td>52%</td>
<td>71%</td>
<td>53%</td>
<td>4%</td>
</tr>
<tr>
<td>% Area Reused Logic</td>
<td>64%</td>
<td>16%</td>
<td>16%</td>
<td>13%</td>
<td>9%</td>
<td>4%</td>
</tr>
<tr>
<td>% Area New Logic</td>
<td>16%</td>
<td>32%</td>
<td>4%</td>
<td>4%</td>
<td>3%</td>
<td>4%</td>
</tr>
</tbody>
</table>

Fig. (1) Embedded Memory Usage

2.1 Yield Challenge
Embedding large number of memory bits per chip creates a more powerful SoC that adapts better to today’s memory hungry applications. But it brings with it the problem of large die size and poor yields.

2.1.1 Need for Manufacturing Repair: because embedded memories are designed with aggressive design rules, they tend to be more prone to manufacturing defects and field reliability problems than any other cores on the chip. This is why embedded memories are considered as defect magnets. The overall yield of an SoC relies heavily on the memory yield. Hence, securing high memory yield is critical to achieving lower cost silicon. Even though, intrinsic or virgin memory yield may be unsatisfactory, but memory and therefore, overall die yield, can be improved. The lower curve on Fig. (2) shows the memory yield as a function of aggregate memory bit-count in SoC. For instance, in this example, the yield for 24Mbit of embedded memory is close to 20%. The example assumes a chip of size 12cmX12cm with memory defect density of 0.8 and logic defect density of 0.4. Traditionally, embedded memories were made testable, but not repairable. Similar to stand-alone memories, yield improvement can be obtained by offering redundancy in embedded memories, i.e. spare elements [7]. Determining the adequate type and amount of redundant elements for a given memory requires both memory design knowledge and failure history information for the process node under consideration. While this is a challenge by itself, providing the right redundant elements does not solve the whole problem. The know-how of how to detect and locate the defects in a memory, and how to allocate the redundant elements require manufacturing know-how in terms of defect distributions [11]. In order to achieve the higher curve in Fig. (2), i.e. optimized yield solution, one needs to utilize test & repair algorithms that contain this know-how, without which the yield can only be improved to an intermediate level, as in Fig. (2).

Fig. (2) Memory Yield Optimization Curve

2.1.2 Need for Process Improvement: In addition to manufacturing repair via redundancy, embedded memories often contribute to a solution for another major challenge, namely the process yield improvement. New systematic defects are often manifested as yield-limiting faults resulted from shrinking geometries and introduction of new material into the fabrication process. To discover the root causes of the yield-limiting factors, adequate diagnosis and failure analysis is needed and accordingly process improvement steps are performed.

2.1.3 Need for Field Repair: Furthermore, the very deep submicron technologies made devices more susceptible to a range of post manufacturing reliability failures. This challenge may be resolved by allowing periodic field level repair and power-up soft repair for the embedded memories.
memories. This type of repair may utilize the remaining redundant elements, if any.

2.2 Manufacturing Cost Challenge

The most common approach to perform memory repair is using external test & repair method. Fig. (3) illustrates the four steps that comprise this method:
a. Wafer-level memory test is applied (either by a memory tester or by BIST) to the embedded memory with redundancy and collects back the responses and builds a failed bit map to be stored in a large capture memory on the Memory Tester. If defects are found, the failed bit map is then utilized by general-purpose redundancy allocation software to either determine that the given defective memory is unrepairable or to determine the best way to allocate redundant resources to replace any defective locations and as a result generate the reconfiguration data.
b. The reconfiguration data strings are then transferred to the Laser Repair equipment, which programs them by blowing the fuses in the laser fuse boxes corresponding to the defective memories.
c. When the memories are repaired a retest is needed to ensure that the repair was successfully implemented. This is done by the second Memory Tester, in Fig. (3).
d. Finally, the Logic Tester performs the test for the random logic portions of the SoC. This is the last step at the wafer level prior to packaging.

This external test and repair method relies on extensive use of equipment. This constitutes as much as 40 percent of the overall manufacturing cost of a semiconductor chip. Therefore, keeping these expenses down is key to lowering the cost of manufacturing. This is especially important for high-volume consumer electronics or networking products and any cost-sensitive applications. New cost effective methods will be discussed in Section 4.

2.3 Time-to-Volume (TTV) Challenge

The continuous increase in SoC complexity and the simultaneous increase in time-to-market pressure force SoC providers to look into volume production as the most critical challenge. TTV is comprised of two periods [19]: SoC design time and production ramp up time.

Reducing SoC design time has been a topic of discussion for long time. Reusing pre-designed F-IP cores and ensuring ease of integration (interoperability) is a viable way to address the growing SoC design gap [16]. Obtaining embedded memories from IP providers is a common practice today. However, this is not sufficient to ensure a minimal SoC design time. One has also to obtain all the necessary views and models of a given memory to simplify SoC integration and minimize design time.

Traditionally, the yield optimization is done during the ramp up period, i.e. following the design stage. During ramp up, the yield problems are detected, analyzed, and corrected. As a result, the yield slowly ramps up to a mature level. After which, usually, the volume production starts.

Due to time-to-market pressure, the ramp up period of an SoC may start before achieving the traditional defect densities, and hence prior to reaching the corresponding yield maturity levels. Because the yield is not sufficiently mature and the SoC is typically complex, the traditional ramp up can take considerably longer. However, the ramp up period can be reduced, and hence the TTV, if the yield optimization effort starts before the ramp up period. This can be realized for embedded memories, if a memory IP provider performs the yield optimization effort at the IP design and characterization phases, prior to SoC ramp up:
- First: by fabricating F-IP test chips, characterizing them and applying knowledge from the fabrication process to improve the yield of the F-IP block. This results in silicon proven F-IP before the SoC production ramp up starts.
- Second: by designing into the FIP and the SoC all necessary memory repair functions in advance. Using this memory repair augments the volume of fault-free SoCs, and hence simplifies the ramp up effort.
- Third: by designing into the F-IP all necessary diagnosis and failure analysis functions based on which to perform process improvement during ramp up period.

In summary to address these three challenges, today’s embedded memories require solutions capable of addressing the yield and reliability needs: repair at manufacturing level; diagnosis for process improvement; and field repair capabilities. At the same time, these solutions need to minimize the manufacturing cost and reduce TTV. Section 3 presents three yield optimization loops corresponding to each of the above needs and Section 5 describes an IP solution to address all three challenges discussed in this Section.

Fig. (3) External Test & Repair Method
3. Yield Optimization Loops:

The above challenges affect different phases of the SoC realization flow. In order to address them a number of yield feedback loops are needed. A yield feedback loop, i.e. a yield optimization solution, encompasses three consequent functions: Detection, Analysis and Correction (DAC). Naturally, all three functions or resources are necessary to obtain yield improvement. To increase their effectiveness, some of these DAC functions are embedded into the design and hence are Infrastructure-IP, whereas others are located off-chip and realized by external equipment and resources [19]. Fig. (4) shows the SoC realization flow and specifies three yield feedback loops to address the yield and reliability needs for embedded memories. The three loops are as follows -

3.1 Manufacturing Repair Loop:

This yield optimization loop is represented by feedback loop 1 in Fig. (4). It takes place during back-end manufacturing phase, more precisely at wafer-level test. The embedded memory repair flow described in Section 2.2 is one possible realization of this feedback loop. In that realization, the three DAC components are external to the SoC, hence the manufacturing cost is too high. Section 4 and 5 show that maximizing repair efficiency, reducing manufacturing cost and minimizing TTV necessitate a fully embedded loop, i.e. that the three DAC components need to reside in the SoC under repair, as an IP for embedded test & repair.

3.2 Process Improvement Loop:

Feedback loop 2 in Fig. (4) corresponds to the process improvement loop. This is meant to detect the root cause of failures (D), perform analysis (A) and apply correcting steps (C) to the fabrication process. As shown in Fig. (4), this loop links the failure analysis phase back to the fabrication phase. In the case of embedded memories, the D component of DAC need to be implemented as Infrastructure IP, whereas the A and C utilize off-chip resources. Section 4 and 5 discuss the benefits of implementing D as an embedded diagnosis IP and offering it as an integral part of the memory F-IP.

3.3 Field Repair Loop:

Feedback loop 3 in Fig. (4) operates in the field. It is meant to execute field self-repair to resolve certain post manufacturing reliability failures. This loop typically becomes operational upon power up. The DAC components in this loop need to be fully embedded in the SoC as an HP, since it needs to be shipped with the product to the field. The same embedded test & repair IP used in loop 1 can typically be reused for loop 3.

4. Memory Test & Repair Evolution:

While the external test and repair method is commonly used for repairing stand-alone memories and was lately extended to SoCs with embedded memories, which typically have extra, unused or redundant locations at the time of manufacturing. But, this method requires expensive equipment, which represents a large percentage of the total manufacturing cost (around 40%), as indicated in Section 2.2. Also, it provides limited repair efficiency for today’s embedded memories, due to the fact that it relies on general-purpose redundancy allocation algorithms; it typically has limited bandwidth to access diagnostic data; and does not allow field repair.

The expense of external test & repair equipment makes it an infeasible solution. As a result, built-in self-test (BIST) engines are being adopted; however, BIST technology has been limited to detecting memory failures. For system-on-a-chip (SoC) devices, this can mean that a single failure in a single memory leads to discarding the unit under test. Thus in addition to BIST, repair is needed. Using BIST and external repair still necessitates expensive set of equipment and resources, similar to the ones shown in...
Fig.(3), which include large capture memory for failed bit map storage, external redundancy allocation process, etc.

Two advanced methods are described in this Section, namely I-IP for Single-Time Repair and I-IP for Multi-Time Repair. Both do not face the yield, cost and TTV limitations as the above method does. Hence, they are able to address today’s embedded memories needs.

4.1 Infrastructure-IP for Single-Time Repair
This method does not require a Memory Tester with large capacity memory for failed bit map storage nor external software for redundancy allocation. Here, the failed bits are diagnosed and the redundancy allocation is performed real-time, by an on-chip dedicated processor for Self-Test and Repair, called STAR Processor, see Fig. (5). The STAR Processor interacts with the embedded memory autonomously to find out if the memory is fault free, determine if it is un-repairable, or generate its repair signature.

a. During loop 1, a simple Logic Tester is sufficient to handshake with the processor in order to obtain one of the three results including the repair signature.

b. The repair signature is then transferred to the laser repair equipment, which blows the laser fuses in the laser fuse boxes corresponding to the defective memories. Laser fuses can be programmed only a single time, hence the name of this method.

c. After the laser fuses are blown, a memory retest is done by the STAR Processor. During the same step, the random logic is also tested using a Logic Tester.

In this method, the manufacturing cost is reduced, because the need for external test & repair equipment and resources are less, compared to Fig. (3). On the other hand, the Infrastructure-IP in the SoC is comprised of a STAR Processor, a Laser Fuse Box and an advanced memory, called the STAR memory. This memory includes a number of self-test & repair features. The specific IP functions are discussed in Section 5.

4.2. Infrastructure-IP for Multi-Time Repair
In order to further reduce the manufacturing cost and improve the repair efficiency, the I-IP for Multi-Time Repair method is introduced. In addition to all the advantages of Single-Time Repair, this method does not require Laser Repair equipment, see Fig. (7).

After generating the repair signature similar to the Single-Time Repair method, the STAR Processor here programs the Fuse Box autonomously, i.e. instead of transferring the repair signature to the external equipment, it internally programs it into the Fuse Box. This is only possible, because, instead of a Laser Fuse Box, this Multi-Time Repair method uses a Non-Volatile Fuse Box. The non-volatile memories are electrically alterable using a special charge pump. The charge pump is embedded in the SoC and is considered among the I-IP resources. Thus, in this method, the manufacturing cost is further reduced, because the need for external repair equipment is totally eliminated. The Logic Tester alone is sufficient to execute the embedded memory test & repair process followed by testing the random logic blocks. Here, The embedded memory test & repair flow for loop 1 and 3 includes: test, diagnosis, redundancy allocation, repair signature generation, non-volatile fuse box programming and retest.

**Fig. (5) I-IP for Single-Time Repair**
In this method, the manufacturing cost is reduced, because the need for external test & repair equipment and resources are less, compared to Fig. (3). On the other hand, the Infrastructure-IP in the SoC is comprised of a STAR Processor, a Laser Fuse Box and an advanced memory, called the STAR memory. This memory includes a number of self-test & repair features. The specific IP functions are discussed in Section 5.

**Fig. (6) I-IP for Multi-Time Repair**
In addition to the reduction in manufacturing cost, this method allows higher repair efficiency, due to the fact that the Non-Volatile Fuse Box can be reconfigured multiple times. The multi-time repair extends the repairability to different tests, such as corner conditions (or screens) for
different temperatures, voltages, frequencies, etc. It is not limited only to wafer-level test types. But rather, it can be extended to post packaging tests, including field tests.

5. Infrastructure-IP for Embedded Memories

The two Infrastructure-IP based methods presented in Section 4 address today’s embedded memory challenges. While their dependence on external test & repair resources is drastically reduced, see Fig. (5) and Fig. (6), these two methods require an advance embedded IP solution to complement the reduced external resources.

Such an I-IP solution, as described in this Section, contains the necessary capabilities to perform manufacturing repair (yield optimization loop 1), help process improvement to fix the fabrication deficiencies (loop 2), and execute field repair to resolve certain post manufacturing reliability failures (loop 3). Therefore, the I-IP is comprised of embedded test & repair functions to perform the DAC components for loops 1 and 3. And it has embedded diagnosis functions to perform the D component of loop 2; whereas A and C of loop 2 are supported by external resources.

The I-IP that performs these functions is a comprehensive IP. It is comprised of a number of hardware components, see Fig. (7), including: a STAR Processor, a Fuse Box, Intelligent Wrappers (IW), and also it includes several self-test and repair features built into STAR memories, examples of which are shown in Section 6.

The STAR Processor performs all the appropriate test & repair coordination of a STAR memory. This Processor is programmed by a set of instructions to control the operation of the internal modules, as shown in Fig. (8). The BIST module performs the tests using memory specific test algorithms. The test algorithms may be pre-configured in the BIST module or made re-configurable via microcode residing outside the processor, such as in a non-volatile memory, named the NV Test Box. The Built-In Self-Diagnosis (BISD) module determines the location of the memory defects (if any) and provides error logging by scanning out failure data if required for failure analysis during loop 2.

The BIRA module (Built-In Redundancy Allocation) identifies available redundant rows and columns and determines the optimum redundancy configuration when a failure(s) occur. It pulls the information from its process failure history, which is a database of failure data from the specific foundry that contains pertinent defect information. Coupled with address mapping and available redundancy, the STAR processor can find the defect and repair it, as shown in Fig. (10). The Reconfiguration Data module translates redundancy allocation into a memory-specific repair signature that gets programmed into the Fuse Box.

As discussed in Section 4, the Fuse Box may be made of laser fuses to allow Single-Time Repair or may be built of non-volatile memory to perform Multi-Time Repair. In the second case, the on-chip programming support, such as the charge pump and its control, are considered a part of the I-IP function.

A single STAR processor can test and repair several memory instances, either running in parallel, one at a time or any particular schedule. In the design example shown in Fig. (7), two STAR processors are used. One is assigned to four memory instances, and another is assigned to a single memory instance. Grouping the memory instance with a shared processor on an SoC depends on several factors, such as area, power, speed, system clock, busses and the chip’s floor plan. A single Fuse Box serves all memory instances on this SoC.

The Intelligent Wrapper (IW) associated with each memory is used in conjunction with the STAR Processor.
to perform test and repair of the memory as well as allow normal memory functioning in the system. The IW contains functions, such as address counters, registers, data comparators and multiplexers. The architectural partitioning between the functions contained in IW vs the STAR Processor depends on the I-IP bandwidth requirements. The low bandwidth interactions are held via the Test Access Mechanism (TAM) between the Processor and the IW; whereas the TAM between the IW and the corresponding memory are mostly of high bandwidth nature. The IW is placed close to the memory core to allow at-speed testing.

6. I-IP Impact on Yield and Reliability
In the previous section, the functionality of the embedded memory I-IP is described. The effectiveness of such an I-IP on optimizing yield and reliability depends on several factors. These factors consist of: type and amount of redundant elements; fault detection and location algorithm; redundancy allocation algorithm; repair strategy; and reconfiguration mechanism. The selection of each one of these factors has an impact on the repair efficiency of the embedded memory and/or the ability to perform effective failure analysis. To increase this impact and hence maximize the yield and reliability, the embedded memory design and manufacturing knowledge base is utilized to select the above factors. This section addresses each factor.

6.1 Redundant Elements
The repair efficiency depends on the type and amount of redundancy in a given memory, and also the amount of repairable memories versus the un-repairable ones in a given SoC.

There are several types of redundancies. The most popular ones are: wordline redundancy, bitline redundancy, and word redundancy [10]. Wordline redundancy provides the possibility to replace one or more rows with spare ones. Bitline redundancy is similar to wordline, but spare columns are added to the embedded memory, instead of rows. Usually, it is difficult to design an efficient architecture to implement bitline redundancy. Therefore, a simplified implementation has been developed, using I/O redundancy. Another type of redundancy is word redundancy, which is based on adding a few redundant flip-flops based words, each of which corresponds to a logical address of the RAM.

Simpler embedded memories utilize a single type of redundant elements, such as rows-only or columns-only [7]; whereas others utilize a combination of multiple types [6][12], see Fig. (9). In case of large embedded memories made up of hierarchical blocks, certain redundant elements may be dedicated to the block level and others to the memory level. For instance, the embedded memory discussed in [12] uses spare columns at the block level and spare rows at the memory level.

For a given memory design and a process technology, redundancy requirements analysis is needed to determine, which type(s) of redundant elements to include, and to which level of memory design hierarchy to assign. Such an analysis leverages the memory design knowledge and utilizes the process defect history. This needs a close relationship between the redundant memory F-IP provider and the semiconductor foundry and requires a learning curve for a process technology. An advanced method to perform redundancy requirements analysis is based on critical area analysis [11]. This method utilizes the defect densities for every process layer.

Typically, an SoC with memories less than 1 Mbit (aggregate bit count) does not require memories with redundancy. Redundancy requirements start between 1Mbit and 2Mbit depending on the process and the type of memory cells. Therefore, an SoC may contain a mix of embedded memories, some without redundancy (up to a certain threshold level) and some with redundancy for the remaining ones beyond that threshold level.

The overall yield of an SoC is determined by its non-memory yield and memory yield segments. To calculate the yield of each segment in an SoC, three parameters are needed: effective defect density for each segment ($D_{o,i}$), segment area ($A_i$), and complexity factor of the process ($N_i$), where $i=M$ refers to the memory portion and $i=L$ refers to the non memory portion of the SoC. $D_o$ and $N_o$ are typically provided by the foundry. They directly influence the yield of the given process technology.

To predict yield for memories, one takes as input $D_{o,M}$, $A_{M}$, and $N_i$ and predicts the virgin (un-repaired) yield, as well as the post-repaired yield.
takes into account the ability of the I-IP to repair various types of defects encountered in an embedded memory. Each type of defect, such as single bit, row, or column, requires a corresponding type of redundant element [7]. The post-repair yield target helps determine the amount of redundant elements of each type per unit of size, example 2 columns for each 1 Mbit, and one row for each 4 Mbit.

For those bit cells other than the classical single port 6T SRAM bit cells, the RAM density conversion factor (DF) can be calculated based on the relative bit cell size with respect to 6T SRAM bit cell size. The reference DFs are as follows: The DF for two port Register Files and Dual Port SRAM bit cells (8T) is close to 2x of 6T bit-cell size. Hence, the corresponding amount of redundancy is predicted. This prediction is very important because having too much redundancy means wasted silicon area and having too little will lead to poor yield.

6.2 Fault Detection and Localization
Conventional memory test algorithms are designed to detect functional faults that are likely to occur, in order to determine that a chip is defective or not. However for repairable memories, fault detection is not enough. Fault localization is needed to determine which cells need to be replaced [15]. The more the fault localization coverage is, the higher the repair efficiency becomes, and hence the yield obtained. Three enhancements, to increase the fault localization coverage, are presented below.

The first enhancement adds dedicated I-IP modes to the memory F-IP, in order to improve fault localization. The STAR memory, shown in Fig. (9), embeds in its layout dedicated HP modes, called Test Modes. These modes include: read margin control, stress test, ground and substrate isolation, adjustable setup and hold time, range of supply voltage operation, bypass self-time clock, etc. This requires directly influencing the memory F-IP design.

The second enhancement does not add to a memory design, but leverages its design information, specifically its scrambling knowledge, to determine the topological background data. The background data is meant to locate coupling faults between cells and between bitlines and for detecting weaknesses in memory periphery. The memory design information required to determine the exact scrambling includes: bitline twisting, folding, contact and well sharing, column multiplexing, decoder optimization, and placement of redundant elements [14]. This type of enhancement is not possible, unless one has intimate knowledge of the memory design and its compilation.

The third enhancement is meant to optimize the fault detection and localization algorithm for a given memory [15]. This may be realized either by leveraging the memory design information, at the layout level, to perform inductive fault analysis on it and consequently generate a corresponding fault detection and localization algorithm; and/or by leveraging the failure history of a given process technology and cell design using test chips or real SoCs. This information helps generate a dedicated test algorithm. Since, the pre-defined test algorithms are not always sufficient to detect all defects in memories because subtle process variations cannot be predicted and accounted for ahead of time. When such faults are uncovered after design tape out, there is a need to be able to program the STAR Processor with new test algorithms that can detect them. As in Fig. (8), such test algorithms can be stored in a Non-Volatile Test Box and scanned to the STAR Processor prior to BIST execution. With NV Test Box, one may program a new algorithm, if the failure history changes after tape out or even in the field.

These three enhancements complement each other, and hence can all be leveraged for the same memory F-IP. All three enhancements can only be implemented, if intimate knowledge of the memory design and manufacturing data is available.

6.3 Redundancy Allocation:
Having appropriate amount of redundant elements in a memory and having the enhanced fault localization solution are not enough to guarantee repair. The analysis phase, which determines the best way to allocate redundant elements in order to replace any defective locations, is very critical. This phase is known as redundancy allocation. If only a single type of redundant elements is used, such as column-only or rows-only, the redundancy allocation is simple [7]. However, if a mix of redundant element types and multiple hierarchy levels are used, optimal redundancy allocation becomes complex.

Unlike the off-line or post processing algorithms used with the external test & repair method, the redundancy allocation algorithms that are embedded in the I-IP need to run on-the-fly, i.e. real-time, while the detection and localization is being executed. Since, the I-IP has no storage space for the failed-bit map. The redundancy allocation algorithm needs to understand the topology of the STAR memory and the process failure history, and as well to know how to utilize the available redundancy in the decision making process, see Fig. (10). These algorithms are typically of two types: primitive algorithms and intelligent ones. The first comes with a predetermined sequence of redundant element allocations based on failure history, such as in [6]; whereas the second performs analysis at every step before allocating a redundant element, such as in [2]. If a solution is not found with the first allocation, there are typically secondary attempts. The secondary attempts may be executed either in parallel with the first to reduce the execution time, or serially to save HP area. In order to select the most optimal redundancy allocation algorithm, a methodology for evaluating the efficiency of algorithms is needed, such as the methodology described in [12].
6.4 Repair Strategies

Another factor that impacts repair efficiency is the selection of a repair strategy. A strategy describes the conditions under which the redundancy allocation is determined and repair is performed. Four such strategies are show in Fig. (11) and summarized below:

**Hard Repair:**
Hard repair requires the use of a permanent storage to remember the repair information after power is turned off. Hard repair does not need retest every time at power up. The STAR Processor only has to scan the repair signature into the reconfiguration register within a STAR memory. As shown in Section 4, there are two methods for hard repair. The first is the Single-Time Repair. This uses a laser fuse box. Here, the repair can only be done once in the factory before packaging. The second is the Multi-Time Repair. This uses a re-programmable NV fuse box in which the repair signature may be changed electrically. It is possible to repair during manufacturing and in the field.

**Soft Repair:**
Soft repair allows a repair signature to be changed any number of times. In fact, this repair happens every time upon power up. No fuse box is needed to implement it because the repair information is not remembered upon power down. Soft repair cannot uncover all types of faults, especially those related to high temperature and high voltage stresses. Therefore, it is always recommended to augment this with an extensive burn-in testing or use it in conjunction with hard repair. Soft repair may find different faults at different times and therefore can be used to improve the yield in the field.

**Combination Repair:**
This delivers the best of both hard and soft repair. Combination repair uses hard repair in the factory to start off the repair and cover faults that cannot be detected by soft repair in the field. Soft repair follows and may happen at the factory or field and uses the existing repair signature stored upon the hard repair as a starting point. Soft repair uses it to build a new signature based on new defects that may be uncovered during this phase. The new signature constitutes the combination repair information and is shifted into the reconfiguration register in the STAR SRAM. This methodology is better than hard or soft repair alone, but the information resulting from testing and repairing each time soft repair is done is not retained and used the next time soft repair is initiated.

**Cumulative Repair:**
To build on the information, each time a memory requires storing the repair data in a re-programmable NV Fuse Box and using the prior data at the start of the next test & repair cycle. The repair signature is changed every time in the NV Fuse Box, if test & repair is completed and more faults are covered over time to produce a very high yield.

The purpose of programming multiple times rather than once is to be able to catch defects under different environmental conditions pertaining to voltage and temperature. The Single-Time Repair method cannot produce this kind of comprehensive repair because it can only be programmed once. The highest repair efficiency is attained when a cumulative repair methodology is employed.

6.5 Reconfiguration Mechanism:
The quality of the repair depends also on whether the reconfiguration mechanism allows true repair or uses the bypass technique. Fig. (12) compares the two reconfiguration mechanisms with a single redundant I/O for repair. The bypass technique relies on multiplexing I/Os externally and hence avoiding the defective locations while leaving room for latent defects to surface in the
memory. This leaves open room for field failures even though external reconfiguration has been made in the factory. If a short occurs between bit line and power or ground, merely multiplexing the bit line does not solve the problem. Over time the leakage current from the bypassed bit line increases to the point where the memory cannot function reliably and its performance degrades. Neighboring bit lines are also affected and the memory fails over time.

Contrast this with the STAR memory using true repair, shown on the right side in Fig. (12). Here the faulty bit line is not only replaced; it is also fully disconnected from the power source. The resulting repair is permanent and leads to no memory performance degradation over time. The external bypass technique also uses redundancy multiplexors and comparators, implemented in RTL, outside the SRAMs. This leads to a performance and area penalty compared to the STAR memory, where the multiplexors are integrated into the memory layout.

7. Conclusion

Producing SoC’s requires embedded memory IPs supported by I-IP to perform embedded test & repair and diagnosis. This Infrastructure IP, as described in this paper, can only reach adequate yield and reliability levels, if it leverages the memory design information and the process failure data. Thus the solution needed is a composite IP that includes the memory F-IP and its test & repair I-IP. Integrating this composite IP into the SoC design requires adequate interoperability with the rest of the functional and test infrastructure on-chip. The STAR Processor solves the test interoperability problem by adopting the standard core test interface IEEE P1500 [5][16], similar to the adoption of IEEE P1500 by other types of embedded cores [8].

References


Fig. (12) Bypass technique vs true repair