

# Debugging High Performance Embedded Memories

Rob Aitken  
Artisan Components  
Sunnyvale, CA, USA  
aitken@artisan.com

## Abstract

Embedded memories fail because of defects, process variation and design marginality. Finding root cause requires a solid design methodology, the correct design-for-debug features, and a structured debug process. This paper/presentation will show some proven techniques for doing this.

## 1. Introduction to memory debug

In their simplest abstraction, memories consist of rows and columns of bit cells, plus access logic. Figure 1 shows a stylized example of a 4-bit memory. Cells rows (C0-C3) are accessed horizontally by word lines, and vertically by pairs of bit lines (B0-B3 and their complements). For a given memory access, the bit lines are precharged, and then the appropriate word line is activated. For a read operation, the cell discharges either the true or complement bit line, and the differential (as opposed to a solid logic signal) is then sensed by the sense amplifiers and latched. For a write operation, the appropriate bit line (true or complement) is discharged, and the 0 value is written into the bit cell. In practice, most memories include a column multiplexing capability also, so that a physical row contains multiple words (e.g. 4, 8, 16, 32, etc.).

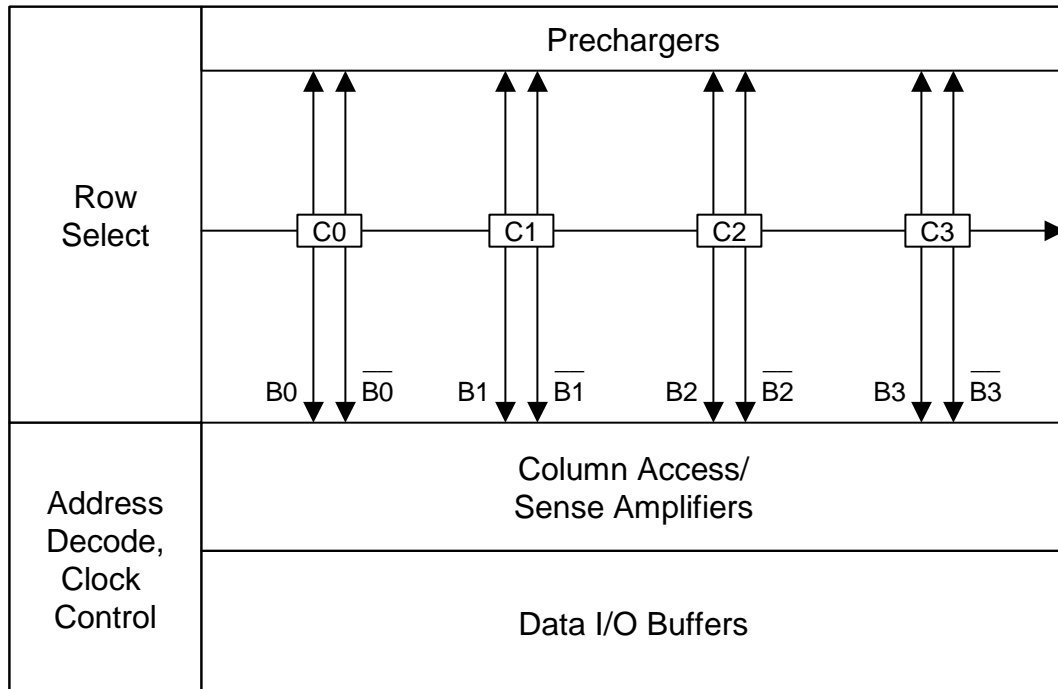


Figure 1 Abstracted memory architecture

The total access time for a memory is split among these operations. The relative importance of each depends on the array size, performance targets, and multiplexing factors. Most memories are self-timed, and so two internal timing paths are important: the

self-timing path and the data timing path. The self-timing path is designed to track process variations in the data path, independent of the nature of the clock signal (duty cycle, slew, etc.). See [1][2] for more information on memory architecture and design.

In a custom memory design, each of these factors can be optimized for a given application, providing the best tradeoff between performance area and power for that application. In practice, though, special purpose memory generators create the vast majority of memories. These generators use an architecture that is designed to cover a large design space and achieve high performance at reasonable area. A standard generator might produce memories from 512 bits to 512 Kbits (e.g. 256x2 to 8192x64). As a result, the number of bit cells connected to a bit line can vary by a factor of 16, and the number connected to a word line can vary by a factor of 32, with the driven capacitances varying correspondingly. The word and column drivers are usually the same for all cases (they must be sized for the largest instance), but the amount of time they will be on depends on the individual instance layout (smaller memories can operate faster).

## **1.1 Yield**

Yield has two main components: random (caused by process contamination such as particles) and systematic (where circuit deformation is a product of its design). Neither is inherently more important than the other. A systematic yield problem can cause observed yield for one or more parts to be significantly lower than predicted yield, while a random problem can cause excursions on a fab's yield on all products at a given time. Systematic yield issues require design changes, while random yield issues require improvements to the process.

Repairability can improve memory yield, and addresses the immediate question of how to get a given chip working, but it is not helpful for resolving the fundamental problems, unless the failure information used for repair is also made available for yield improvement.

## **1.2 Previous work**

Bit map failure patterns have been used for many years to debug memories (e.g. [3][4]). Successful embedded memory debug requires access to the memory, complete failure information on a per-cell basis, and ideally the ability to develop and apply new tests locally. Many of the necessary approaches are well-known to practitioners (e.g. March tests), while others are more obscure. Successful debug requires knowledge of the mapping of logical addresses to physical, as well as knowledge of physical attributes such as bit line twisting. This paper concentrates on classifying observed failure information, and assumes that access, test vectors, and response collection have been solved elsewhere.

## **2. How memories fail**

Three basic, but interrelated issues can cause memory failure: defects, process variation, and design marginality.

### **2.1 Defects**

Defects can disable a memory completely, or partially. A low resistance short on a clock circuit, for example, will usually cause catastrophic failure. On the other hand, a

short in a bit cell can be repaired, or if its resistance is high enough, not affect memory operation at all. Memories are dense circuits, and defects are to be expected. They can become a concern when yields are lower than predicted, when they appear to be related to a systematic problem, or where they result in test escapes.

## **2.2 Process Variation**

Memories are designed to tolerate a wide spread in process variation, at minimum that characterized by the process SPICE models, but often much more than that. For example, bitcell leakage variation of 5-10X over specified values is often tolerated, in order to account for unexpected process drift, or to allow aggressive process skewing for high performance products. When variation becomes excessive, though, failure will occur (if for example, leakage gets too high, read current may be unable to create enough differential on a bit line pair for correct sensing).

## **2.3 Design Marginality**

Design marginality is in some ways a function of defects and process variation, along with variation in operating environment. Product designs are specified to meet operating standards across a range of defined variations. These variations are derived from the expected customer operating variations and manufacturing variations. Common variations include process, voltage, temperature, and transistor mismatch. Variations can be global (between arrays) or local (within an array). In order to ensure that a circuit functions properly across the operating range, it is necessary to stress variations beyond the operating range. The difference between the design space defined by the stress variations and the operating variations is its margin. Inadequate margin can result in failure due to unintended conditions (e.g. IR drop, poor cooling, etc.), process changes (intended or otherwise), or other stresses. In each case, a marginal design will fail where one with more margin would not.

## **3. Debug basics: bitmaps, shmoo, data**

Two key elements to successful debug are bitmaps, and shmoo plots [5]. Bitmaps bound the circuitry affected by a problem, while shmoo plots characterize the relationship between the problem and operating environment. In general, as much data as possible should be collected prior to developing theories of root cause of a problem. Memories are complex circuits and often fail for complex reasons. Data sources include silicon measurement, EDA tools, foundries, and IP vendors. Particularly in a fabless model, numerous organizations may be involved in identifying root cause of a problem.

### **3.1 General Issues**

The nature of a failure gives some clues about what might cause it. The table below discusses some of these. For each potential location, hints about when to suspect it are given, along with exclusions. The exclusions are specific to the suspected values.

Location	When to suspect it	When to exclude it
Bit cell	Single cell failure Odd-shaped contiguous group of cell failures Isolated cell failures	Non-random failure patterns
Bit cell contact	Two/four adjacent cells failing	
Bit line	One or two adjacent columns failing (whole or partial)	Entire I/O failing
Sense amp, I/O	Block of columns fail	
Word line, drivers	Whole or partial row failure	
Addressing	One or more rows/columns failing (whole or partial), especially when separated by regular spacing (e.g. 8, 16)	
Clocking, control	Entire memory, or substantial portions of it, fail	

#### 4. Defect signatures

The classic defect signature is a single failing bit. As feature sizes shrink, some defects are becoming larger than bit cells. Single point defects can also look like failing rows, columns, or blocks of memory. Signatures of types of defects (e.g. resistive vias fail at lower temperatures more often than high ones). The need to develop defect-specific tests, especially for bit cell defects.

Defect	When to suspect it	When to exclude it
General	No pattern to observed failures when compared with process variation	Regular, consistent failure pattern
Metal short	Any failure mode Fail behavior worse with increasing temperature Column failures more likely as distance from sense amp increases. Linear static current relationship to supply voltage (if measurable)	Bidirectional fails in bit cells or bit lines during same test.
Break	All cells above a point in a column fail both directions All cells to left or right of a point in a row fail both directions	Failure point moves with changing conditions.
Resistive contact/via	Failure gets worse with decreasing temperature Failure occurs after multiple repeats of same operation (e.g. multiple reads on same cell)	

Scratch	Mostly linear swath of cell failures.	
Parametric	Failing cells are adjacent, but do not follow architectural pattern (e.g. 15 fails in one row, 16 in the next, 14 in the third)	Regular, especially rectangular failing pattern

## 5. Process variation signatures

Common variations: transistor length/ $V_t$ , metal width/thickness. Byproducts of these variations are reduced performance, increased leakage, “weak” bits. How to find them (e.g. to disprove a bit cell leakage problem, identify failures when all cells in a column hold the same value; to stress leakage, read from a cell that holds the opposite value of all others in a column).

Variation	When to suspect it	When to exclude it
General	Wafer parametric data at or near process limits	No pattern to failing devices (some slow, some fast, some in the middle)
Excess leakage	Higher than expected $I_{off}$ Failures at high temperature Failures at fast process corner	Successful read of 1 when all other bits in a column hold 0, plus the inverse

## 6. Design margin signatures

Memory failures can result from design margin issues within the memory and within the logic surrounding the memory. How to differentiate the two. Logic margin issues: setup/hold time violations, IR drop, crosstalk. Relationship between process variation, defects, and design margin.

Issue	When to suspect it	When to exclude it
General	Failures only at extremes of temperature, voltage, or process variation	Not a regular signature, no pattern to circuits that fail.
Logic problem	Fail depends on values on memory I/O, but not on memory contents Fail occurs during BIST or normal operation, but not both	If failure can be isolated within array
Setup problem	Static timing analysis suggests problem Slowing clock rate fixes problem	
Hold problem	Static timing analysis suggests problem Problem more prevalent in fast silicon	

IR drop	Narrow power buses Few power connections Problem worse when nearby circuits switching	Power network is robustly and regularly connected to memory
Crosstalk	Failure dependent on unrelated circuitry switching Poor memory shielding	

## 7. Design for debug features

As a consequence of increasing chip performance requirements and the inherent complexity of integrating design IP from multiple sources, achieving timing closure is a dominant problem in SoC design. In addition, measuring timing accurately needs to consider signal integrity effects such as crosstalk, IR drop, and glitches. Careful analysis using high quality tools is needed to avoid these during design. Finally, there is an issue with built-in self-repair with respect to marginal timing issues: If a memory is tested at start-up, its temperature will be lower than later during operation (e.g. 25C versus 100C). Timing defects could be present, but not active, at the lower initial temperature.

These issues can be addressed within a memory for both debug and normal operation by adjusting memory timing. Some generators provide an adjustable timing mechanism to account for this. This works by delaying the timing of the self-timing path in the sense amp circuitry, which allows the bit cells to have additional time to discharge either the bit line or its complement, resulting in a higher differential voltage and a more robust read, even for a weakened cell.

Outside the cell, debug features include the ability to generate special debug patterns, or indeed any customized patterns, either from an external APG in a tester, or from BIST circuitry.

In all cases, bitmapping capability is vital for successful memory debug. If this is not present in BIST, some form of direct access needs to be provided.

## 8. Conclusions

Debugging memories is a complex task, but careful consideration during design and careful analysis later can greatly simplify the problem.

## 9. References

- [1] A. Sharma, *Advanced Semiconductor Memories: Architectures, Designs, and Applications*, IEEE Press (Wiley), 2002.
- [2] R.D. Adams, *High Performance Memory Testing: Design Principles, Fault Modeling and Self-Test*, Kluwer, 2002.
- [3] B.B. Sindahl, "Interactive Graphical Analysis of Bit Fail Map Data Using Interactive Pattern Recognition", *Proc. Int. Test Conf.*, pp. 687-695, 1987.
- [4] T. Zanon et al, "Analysis of IC Manufacturing Process Deformations: An Automated Approach Using SRAM Bit Fail Maps", *Proc. Int. Symp. Test and Failure Analysis*, pp. 232-241, Nov. 2003.
- [5] K. Baker, J. van Beers, "Shmoo Plots - the Black Art of IC Test", pp. 90-97, *IEEE Design and Test*, No. 3, 1997.

R. Aitken, Debugging High Performance Embedded Memories, SDD Workshop, 2004