A Physical Design Tool for Built-in Self-Repairable Static RAMs

Kanad Chakraborty
Server Dev. Division
IBM E. Fishkill
Hopewell Junction, NY 12533

Anurag Gupta
MPG Design Tech. Group
Intel Corp.
Santa Clara, CA 95052

Mayukh Bhattacharya,
Shriram Kulkarni, Pinaki Mazumder
Department of EECS
The University of Michigan
Ann Arbor, MI 48109

Abstract

*A novel physical design tool, **BISRAMGEN**, that generates layout geometries of parametrized built-in self-repairable SRAM modules, producing significant improvement in testability, reliability, production yield and manufacturing cost of ASICs and microprocessors with embedded RAMs, is presented.*

Key words and phrases: *Built-in self-repair (BISR), yield, reliability*

1 Introduction

This paper describes a novel physical design tool, **BISRAMGEN**, which synthesizes layout geometries of built-in self-*repairable* static RAMs. Built-in self-repair (BISR) refers to a range of algorithms and circuit techniques used to replace faulty elements in a VLSI circuit with redundant fault-free ones; it is usually deployed in conjunction with a built-in self-testing (BIST) technique that identifies faults. BISR produces a dramatic improvement in the production yield, field-related reliability and fault-tolerance of VLSI devices. With deep submicron technologies characterized by shrinking effective channel lengths of transistors below 0.18μ, the raw production yields are very low, hence RAM BISR is of great value both for commodity and embedded (i.e. on-chip) RAMs used in high-density ASICs and microprocessors.

RAM compilers were introduced in 1986 by Texas Instruments [1]. The first reported compiler, RAMGEN, generated layouts, simulation models, symbols and datasheets (for setup and hold times, read access times and write times, and supply currents and voltages) for RAMs, ROMs, and PLAs. Since this effort, RAM compilers have become progressively more powerful and sophisticated. CMOS SRAM compilers have been designed by Motorola and Mentor Graphics (the Memorist compiler [2]), Cascade Design Automation (the CDA RAM compiler [3]), and at the University of Michigan (the Aurora GaAs MESFET RAM compiler (ARC) [4]). Some of these compilers, such as the CDA and the ARC, try to achieve process independence; that is, the ability to generate layouts for a wide range of input process technologies and design rules. In 1992, Kebichi, et al. [5] reported a compiler for built-in self-testable RAMs. In this paper, we describe a new compiler, **BISRAMGEN**, for built-in self-repairable static RAMs. **BISRAMGEN** is design-rule independent and can generate efficient layouts and simulation models for built-in self-testable and self-repairable RAM arrays. It achieves low area overheads for BIST and BISR (at most 7% for realistic RAM array sizes). Our calculations show that it achieves significant improvement in production yield and reliability, and lowers the manufacturing cost of dense ASICs and microprocessors with embedded RAMs.

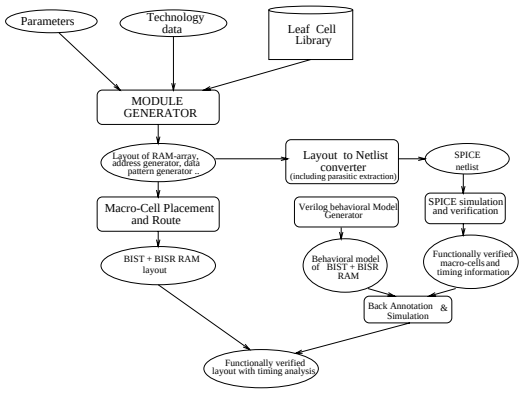


Figure 1: Overview of BISRAMGEN

2 Overview of BISRAMGEN

Figure 1 gives an overview of the layout synthesis flow in **BISRAMGEN**. From a set of user-specified geometry parameters and CMOS technology data, **BISRAMGEN** builds a library of leaf cells that are subsequently used for generating layout modules (or macrocells) in a bottom-up (hierarchical) fashion.

The user can select a design rule-set from a range of manufacturing processes with three routing layers. Some examples are: Cascade Design Automation processes CDA.5μ3m1p^{TM} and CDA.7μ3m1p^{TM}, and the MOSIS

process mos.6u3m1pHP^{TM}. **BISRAMGEN** allows the user to input the values of the circuit parameters for a wide-word RAM employing column-multiplexed addressing. In column-multiplexed addressing, a single column stores multiple bits, the number of bits per column being denoted by *bpc*. The number of bits per memory word is denoted by *bpw*. A $\log bpc$-to-*bpc* column decoder chooses exactly one out of *bpc* bitline pairs from each of *bpw* I/O sub-arrays, producing a *bpw*-bit word. The parameters explicitly specified by the user include: *bpc*, *bpw*, number of words, and number of spare rows. Critical components in the RAM circuitry, such as the precharge transistors and the word line drivers, are made larger than minimum size. Furthermore, for a given buffer size, the *N* and *P* transistors are automatically sized to balance the rise and fall times. **BISRAMGEN** implements fast-access and high-bandwidth circuit techniques, such as current-mode sensing and column-multiplexed addressing.

3 BIST technique

BISRAMGEN uses a low-area overhead, microprogrammed BIST circuit for applying the IFA-9 test on a static RAM array. The test circuitry consists of a ***Test Data Background Generator***, a ***Test Address Generator***, and a combined ***Test and Repair Controller PLA***, which implements the finite state machine used for generating control signals during self-test and self-repair. This circuit can send a signal to the microprocessor to generate a reset state during which data retention testing is done.

The test involves two passes. In the first pass, the memory array is tested and faulty addresses are stored in a translation lookaside buffer (TLB). In the second pass, the array is re-tested along with the mapped redundant addresses. Any fault detected in the second pass produces a 'Repair Unsuccessful' status signal, which implies either too many faults in the memory array (such as due to a faulty column) or faulty spares.

The march notation for the IFA-9 test is as follows [6]: $\{\Uparrow(w0), \Uparrow(r0,w1), \Uparrow(r1,w0), \Downarrow(r0,w1) \Downarrow(r1,w0),$ Delay,$\Uparrow(r0,w1),$ Delay, $\Uparrow(r1)\}$. This test consists of two delays of 100 ms each for data retention testing. For a wide-word RAM, this test has to be performed once for multiple background patterns in order to test pairwise couplings between cells in the same row. To run this test, a binary up-down counter for address generation and a Johnson counter for background data generation are used.

4 BISR technique

The faulty row addresses detected by BIST are stored in a hardware TLB. This circuit uses an innovative design (international patent pending) that associates a sequence of faulty addresses with a unique, predetermined sequence of redundant addresses. The faulty addresses are stored in the TLB during the first pass of the BIST, and are compared with incoming addresses during the second pass. If a match is found, then an address diversion occurs to a redundant location. The match operation is performed by a CAM (content addressable memory) built into the TLB.

The TLB produces a modest delay penalty (of about 1.2 ns with 4 spare rows and a 0.7μ technology) for matching and mapping the incoming addresses during normal operation. This small delay will, however, *not* result in stretching of the RAM access time. This is due to the fact that the TLB operation can be completely overlapped with other components in the read or write cycle. In case of an asynchronous RAM, it can be overlapped with the precharge phase following an address transition detection (ATD) signal of a read or write cycle (note: often RAM bit-lines are precharged in anticipation of read in order to reduce the access time, even though the actual operation may have been a write). In case of a synchronous RAM with a level-sensitive clock, the address bits are sent simultaneously to the address register and the TLB, and the operations of the TLB (address comparison and spare address outputting) are done in parallel with those of the address register.

The controller consisting of D flip-flops and a pseudo-NMOS NOR-NOR PLA, is found to occupy a very tiny fraction of the memory array area (less than 0.1%) for a 16 kilobyte RAM.

Our self-repair scheme performs only row repair, but can be easily extended to perform column repair. In this case, it would be necessary to add extra columns to the RAM, and to enhance the control logic to perform column testing in addition to row-based testing. However, for wide-word, column-multiplexed RAMs, column redundancy and repair are *not* recommended. Such RAMs are the most practical ones in embedded applications because these fast RAMs provide adequate data bandwidth and are thus commonly used in on-chip caches. For such RAMs, even a single redundant (multi-bit) column would cause unacceptable silicon area overhead. Even if we exclude the circuitry needed for column repair, the area overhead due to a single redundant multi-bit column would be as high as $100/bpc$ %, (where *bpc* is usually between 4 and 64), since all the bits of a multi-bit column have to be simultaneously accessed. On the other hand, having single-bit (instead of multi-bit) redundant columns in a wide-word, column-multiplexed RAM array would destroy the symmetry and regularity of the custom-designed array; moreover, the circuitry used for column testing would cause an access delay penalty. The technique of row-repair performed by **BISRAMGEN** can *detect* faulty columns by producing a 'Repair Unsuccessful' signal at the end of the second pass of self-testing and self-repair, described before.

5 Fault coverage

The IFA-9 test [6] performed on the RAM array covers a wide range of hard functional faults caused by layout defects. Layout defects such as gate oxide shorts and bridges can give rise to stuck-at and stuck-open faults, transition faults and state coupling faults. In addition, the BIST circuitry performs data-retention testing. Column failures (e.g. due to a sense amplifier or a column multiplexer stuck-on) are catastrophic and can be detected by the 'Repair Unsuccessful' status, but not diagnosed or repaired by the BISR scheme.

6 Yield improvement produced by BISRAMGEN

BISRAMGEN generates CMOS layouts of built-in self-repairable RAM arrays. The circuits synthesized can repair a maximum of S faulty rows, S being the number of available spare rows. A built-in self-repairable RAM produced by **BISRAMGEN** is said to be 'good' if the manufacturer can guarantee that the spares are all good and the number of faulty words is at most equal to the number of spare words. Since **BISRAMGEN** is unable to perform more than one round of spare substitution (i.e., it is unable to substitute faulty spare words by fault-free spare words), we use the above notion of 'goodness' of BISR'ed RAMs.

Suppose we use the Poisson model of a single cell yield, i.e.,

$$Y_{sc} = e^{-\lambda_{sc}},$$

where λ_{sc} represents the average number of faults per cell. Let us also assume the well-known yield formula due to Stapper [7, 8] to calculate the original yield of the memory array without built-in self-repair:

$$Y_0 = (1 + A\delta/\alpha)^{-\alpha},$$

where δ is the defect density, A is the area of the RAM array, and α is some clustering factor of the defects. For a given manufacturing process, let P_r be the probability function for a defect pattern to be repairable with respect to the fault-free spare rows available. Then the yield Y_{BISR} is as follows:

$$Y_{BISR} = Y_0' + (1 - Y_0')P_r$$

where

$$Y_0' = [1 + (A + B)\delta/\alpha]^{-\alpha},$$

where B denotes the area of the additional circuitry (BIST/BISR), assuming that the clustering factor α remains unchanged. This is a reasonable assumption since α is a function of the manufacturing process which remains constant.

The probability of not having a failing bit in a $bpw \cdot bpc$-bit row is given by $Y_{sc}^{bpw \cdot bpc}$, and the probability that at least one bit fails is given by $1 - Y_{sc}^{bpw \cdot bpc}$. A defect pattern can be repaired successfully if and only if the number of faulty rows is at most equal to the number of spare rows, and the spares required are themselves fault-free. For a given number d of defects and a single memory cell, λ_{sc} exponentially decreases with the product of N, bpc and bpw – it is proportional to $exp(-d/(N \cdot bpc \cdot bpw))$.

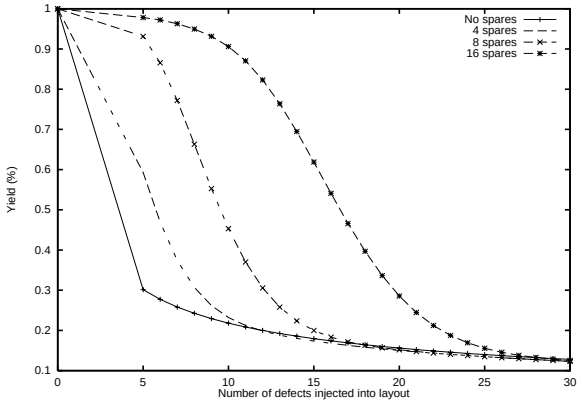


Figure 2: Yield vs. number of defects for a narrow RAM array with 1024 rows, bpc=4 and bpw=4

Figure 2 shows the improvement of yield due to BISR for a RAM with 1024 rows, and with $bpc = bpw = 4$. The BISR circuitry produced by **BISRAMGEN** is suitable for wide-word RAMs that are typically used as embedded macrocells within high-density microprocessors and application-specific integrated circuits (ASICs), for example, those used in telecommunication and local-area network applications. Almost all modern microprocessors have on-chip data and instruction caches or combined caches. In addition, some microprocessors use different levels of caches (called, L1 and L2 caches), such as Alpha™ 21164. Some microprocessors, such as Motorola 68060, use a 'branch cache' for storing branch prediction addresses. Our calculations [9] show that the yield improvement of on-chip RAMs in high-density microprocessors and ASICs would cause a significant reduction in the cost-per-die and would thereby produce an impressive decrease in the total manufacturing cost (sometimes by as much as 10%).

7 Reliability improvement produced by BISRAMGEN

BISR is useful not only during manufacture but also during the operational life-time of a chip, provided enough spares are available. Hard (or permanent) failures that occur during field use can often be repaired using BISR. Transient failures (such as those due to alpha particle hits, cosmic rays, static discharges, etc.) and intermittent failures (such as those caused by resistance and capacitance variations and crosstalk), affect the system for too short a

duration to be detected and repaired by BISR techniques. We begin by recapitulating a few definitions.

Definition 1 *The **reliability** (also known as survivability) $R(t)$ of a system as a function of time t, is the probability of correct functioning of the system* until *time t.*

Expressed mathematically:

$$\begin{aligned} R(t) &= 1 - P(X \leq t), \text{with} \\ R(0) &= 1, \text{and} \\ R(\infty) &= 0 \end{aligned}$$

(**Note:** In the above, X is a random variable denoting the time at which a failure occurs, and $P(X \leq t)$ denotes the probability that $X \leq t$).

Definition 2 *The **failure probability density function**, $f(t)$, is defined as the rate of decrease of reliability with time t; i.e. the negative derivative of $R(t)$.*

Hence, $f(t) = -\frac{dR(t)}{dt}$.

Definition 3 *The **mean time to fail** $MTTF$, is defined as the expected value of the time at which failure occurs.*

In other words, $MTTF = \int_0^\infty t\, f(t) dt$.

Integrating the above by parts, we conclude that:

$MTTF = \int_0^\infty R(t) dt$. (For example, if $R(t) = e^{-\lambda t}$, then $MTTF = 1/\lambda$.)

Suppose, a memory module generated by **BISRAMGEN** consists of N regular rows, S spare rows, bpc bits per column, and bpw bits per word. Assume that the failure rate per bit per unit time is λ. Then a single RAM cell is fault-free with a probability of $e^{-\lambda t}$. Therefore, the probability that a bpw-bit RAM word is faulty at time t, denoted by $P^f_{word}(t)$, is as follows:

$P^f_{word}(t) = 1 - e^{-\lambda \cdot t \cdot bpw}$

The RAM module will survive until time t if and only if at most $S \cdot bpc$ of the regular words are faulty until time t, and the $S \cdot bpc$ spare words are themselves fault-free until this time. Let $X(t)$ be a random variable denoting the number of faulty words at time t. Then the reliability function $R(t)$ of the built-in self-repairable RAM module is as follows:

$$\begin{aligned} R(t) &= P(X(t) \leq S \cdot bpc) \cdot (1 - P^f_{word}(t))^{S \cdot bpc} \\ &= \left[\sum_{i=0}^{S \cdot bpc} \binom{N \cdot bpc}{i} \cdot (P^f_{word}(t))^i (1 - P^f_{word}(t))^{N \cdot bpc - i} \right] \\ &\quad \times (1 - P^f_{word}(t))^{S \cdot bpc} \end{aligned}$$

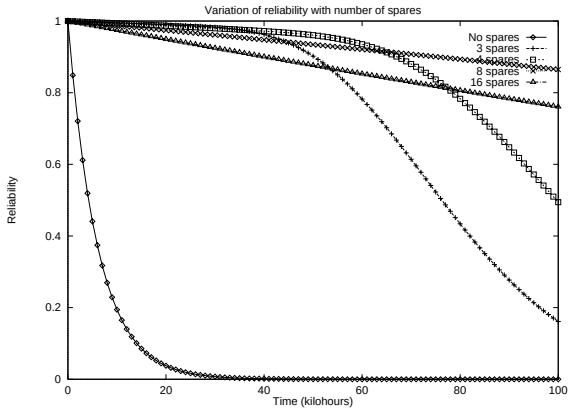


Figure 3: Reliability for a RAM with BISR with a defect rate of 10^{-5} per kilo-hour per memory cell; the RAM has 1024 regular rows, with bpc=4 and bpw=4

Table 1: BISR overhead with 4 spare rows, process chosen for illustration: CDA0.7u3m1p

Size of RAM	Geometry	Area Overhead with BISR
64 Kb	$1024 \times 8 \times 8$	7.01%
128 Kb	$1024 \times 8 \times 16$	3.64%
512 Kb	$512 \times 32 \times 32$	3.22%
1 Mb	$512 \times 64 \times 32$	3.07%
2 Mb	$1024 \times 64 \times 32$	1.90%
4 Mb	$2048 \times 64 \times 64$	0.60%

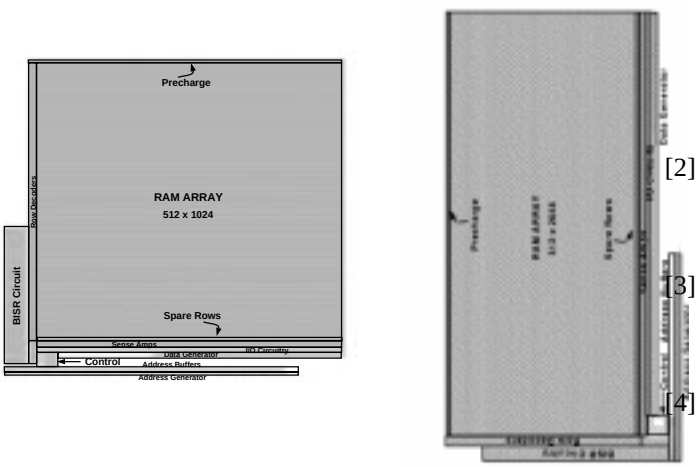
The MTTF is computed by integrating the reliability function $R(t)$, from $t = 0$ to infinity. The result is as follows: $MTTF = \sum_{i=0}^{S \cdot bpc} \sum_{k=0}^{i} \binom{N \cdot bpc}{i} \cdot \binom{i}{k} \cdot \frac{1}{\lambda \cdot bpw \cdot [(N+S) \cdot bpc - i + k]}$

Evaluation of the reliability equation reveals that the reliability typically increases with the number S of spares only after a period of several years after manufacture. Initially the reliability is found to decrease with the number of spares.

Figure 3 illustrates the reliability as a function of time (i.e. the age of the device), for a RAM with 1024 regular rows, $bpc = bpw = 4$. The defect rate per cell has been fixed at 10^{-5} per kilo-hour. This plot indicates that it is better to use 8 spare rows instead of 4 only if the chip survives for more than 8 years (70 kilohours) after manufacture.

8 Area overhead

BISRAMGEN produces BIST/BISR circuitry with low silicon area overhead. Table 1 gives some examples of the area overhead including BIST and BISR. In this table, the parameters used are: W (= the number of words), bpc, and bpw the geometries being specified as $W \times bpc \times bpw$. Note that the spare rows constitute less than 0.8% of the total RAM area.



(*a*) (*b*)

Figure 4: SRAM arrays with: (*a*) *bpc* = 32, *bpw* = 32, 16384 words, 4 spares; (*b*) *bpc* = 64, *bpw* = 32, 32768 words, 4 spares (**Note:** plots are not to scale)

The BISR area overheads are found to be comparable with results published by other researchers [10]. Most researchers base their results on either a custom layout for a specific process technology or a very rough transistor count metric. Using **BISRAMGEN**, we have experimentally generated BISR RAM layouts for a wide range of CMOS processes; therefore, we feel that our results are very reliable. Note that the area overhead values shown in Table 1 translates to only about 1% of the *total chip area*, as obtained using die floorplan analysis for a range of microprocessors and ASICs [9]. Typical layout plots produced by **BISRAMGEN** are shown in Figure 4.

9 Conclusion

The RAM layouts produced by **BISRAMGEN** use efficient circuit techniques for fast access and high data bandwidth. Such techniques include column multiplexed addressing and current-mode sense amplification for reading data out at high speed. The area overhead with BIST and BISR is very modest (only about 1% of the total chip area). Besides, the BIST algorithm microprogrammed into the control PLA achieves a high fault coverage. The production yield and reliability with built-in self-repair are significantly greater than without built-in self-repair. Currently, the task of integrating BIST and BISR circuitry with RAM layouts is usually done manually by CAD designers, and is thereby a major bottleneck in the design cycle. A tool like **BISRAMGEN** is therefore, a great help in speeding up the time-to-market.

References

[1] Swartz, W.P., et al., "CMOS RAM, ROM and PLA Generators for ASIC Applications," presented at the *Proc. of the 1986 Custom Integrated Circuits Conf.*, 1986.

[2] Tou, J., "Submicrometer CMOS Embedded SRAM Compiler," *IEEE J. Solid-State Circuits*, pp. 417-424, 1992.

[3] Shinohara, H., et al., "A Flexible Multiport RAM compiler for Data Path," *IEEE J. Solid-State Circuits*, vol. 26, no. 3, March 1991, pp. 343-349.

[4] Chandna, A., "GaAs MESFET Static RAM Design for Embedded Applications," Ph.D. Dissertation, Department of Electrical Engineering, The University of Michigan, Ann Arbor, 1995.

[5] Kebichi, O. and Nicolaidis, M., "A Tool for Automatic Generation of BISTed and Transparent BISTed RAMs," *Proc. Intl. Test Conf.*, pp. 570-575, 1992.

[6] Shen, J.P., et al., "Inductive Fault Analysis of CMOS Integrated Circuits," *IEEE Design & Test of Computers*, 1985, pp. 13-26.

[7] Stapper, C.H., McLaren, A.N. and Dreckmann, M., "Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and Partially Good Product," *IBM J. Research & Development*, v. 24, May 1980, pp. 398-409.

[8] Stapper, C.H., "On Yield, Fault Distributions, and Clustering of Particles," *IBM J. Research & Development*, v. 30, no. 3, May 1986, pp. 326-338.

[9] Chakraborty, K., "BISRAMGEN: A Silicon Compiler for Built-in Self-Repairable Random-Access Memories," Ph.D. Dissertation, Department of Computer Science and Engineering, The University of Michigan, Ann Arbor, May 1997.

[10] Chen, T. and Sunada, G., "Design of a Self-Testing and Self-Repairing Structure for Highly Hierarchical Ultra-Large Capacity Memory Chips," *IEEE Trans. on VLSI Systems*, vol. 1, no. 2, June 1993, pp. 88-97.