Evaluating the Yield of Repairable SRAMs for ATE

Marco Ottavi, *Member, IEEE*, Luca Schiano, *Student Member, IEEE*, Xiaopeng Wang, Yong-Bin Kim, *Senior Member, IEEE*, Fred J. Meyer, *Member, IEEE*, and Fabrizio Lombardi, *Senior Member, IEEE*

***Abstract*—An accurate yield evaluation is essential in selecting redundancy allocation and testing strategies for memories. Yield evaluation can resolve the many issues revolving around cost-effective built-in self-test (BIST) and automatic test equipment (ATE)-based solutions for a higher test transparency. In this paper, two yield-calculation methodologies for SRAM arrays are proposed. General yield expressions for VLSI chips are initially presented. The regular and repetitive structure of an SRAM array is exploited, and substantial yield improvements can be achieved by the introduction of redundancy. Two repair yield-evaluation methods for one-dimensional redundant memory arrays are introduced and compared for ATE application. The first method is based on the sum of the probabilities of all repairable fault patterns; the second method is based on Markov modeling. Using industrial data, it is shown that these methods are applicable to ATE usage under different conditions of defect rate in the possible defects. Different features of the proposed methods are discussed.**

***Index Terms*—Automatic test equipment (ATE), manufacturing, Markov modeling, redundancy, SRAM, yield.**

I. Introduction

MANUFACTURING of chips requires accurate monitoring of the different processes for preserving the quality of the shipped product. An accurate estimate of the yield (i.e., the percentage of working chips in a production batch) is an essential feature in manufacturing and selecting appropriate repair strategies (if required). This is applicable to memories; starting from the expected yield, different strategies can be employed for spare allocation and related testing processes. For redundancy allocation, an expected high yield can be used to justify a small number of spare rows/columns. To maintain the defect level of the shipped product at an acceptable value, additional redundancy may be required at design level. Furthermore, a yield estimate impacts the operation of testers such as an automatic test equipment (ATE). Memory testers are designed for enhancing a parallel operation to exploit modularity for a high throughput at low test time. Currently, an ATE can test up to 128 DUT simultaneously to achieve an operational speed of up to 500 MHz.

Manuscript received June 15, 2004; revised March 18 2006. This work was supported by the IEEE.

M. Ottavi, L. Schiano, Y.-B. Kim, and F. Lombardi are with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115 USA (e-mail: lombardi@ece.neu.edu).

X. Wang is with the Microelectronics Division, IBM, Essex Junction, VT 05402 USA (e-mail: xiaopeng@us.ibm.com).

F. J. Meyer is with the Electrical and Computer Engineering Department, Wichita State University, Wichita, KS 67260 USA.

Digital Object Identifier 10.1109/TIM.2006.880315

With the dramatic increase in memory size, efficient solutions are required to maintain an acceptable production yield [1]. Moreover, testing of embedded memories (SRAM and DRAM) incurs in additional costs due to the large test time and ATE performance (such as for "at speed" testing). To lower test costs and to allow tester designers to meet memory requirements, discrete Fourier transform (DFT)/built-in self-test (BIST) and built-in self-repair (BISR) solutions have been proposed [2]–[5]. Yield evaluation can be used in selecting an appropriate testing strategy; when the expected yield is high, BIST or BISR solutions may be preferred; otherwise, more expensive ATE solutions are available. Generally, the former technique leads to a low test transparency, i.e., the fraction of defects not detected by testing is higher. However, the reduction in test/repair costs can justify the use of BISR, provided a higher yield is expected. An expected low yield may require the use of purely ATE-based test and repair techniques. Although they are more expensive, in most cases, they can guarantee a lower defect level of the shipped product; high defect levels are expected in the earliest stages of manufacturing, so a yield estimate should be pursued prior to utilizing an ATE. Yield estimate for integrated circuits (IC) has been extensively studied; the initial work of [6] has recognized that the yield of an IC does not follow the simple Poisson statistics and has introduced the use of compound Poisson statistics. The use of compound Poisson statistics has been then further studied and improved in [7]–[9]. Moreover, the so-called clustering effect has been extensively analyzed with respect to the sizes of the cluster [10]–[12], [14]. In [15], a unified model has been provided. The yield estimate of a repairable memory stems from the study of IC yield and has been addressed in [9], [13], [14], [17], and [22]. Memory repair has been proved to be an NP-hard problem [19]. Many algorithms have been proposed for optimizing spare row and column allocations in a RAM [20], [21], [23], [24]. Most of these repair algorithms are based on greedy heuristics, because they are advantageous in terms of computation time.

The goal of this paper is to introduce and analyze two different methods for estimating the yield of SRAMs while considering clustering effects and the benefits of a repair mechanism. Different from most of the previous works, both proposed methods calculate the yield by considering the SRAM repair solutions. This improves the level of accuracy compared with the traditional yield methodologies, which either consider repair operations based on probabilistic assumptions or ignore the effect of overlapping defects in the redundancy allocation. To reduce the complexity of the allocation algorithm, only one-dimensional redundancy (spare rows) has been considered in this paper. The first proposed approach [25] is a fast industrial-based method for embedded SRAMs; it computes the yield by considering design features of so-called compiler-based embedded SRAMs. This is an approximate approach (hereafter also referred to as "method A") and is the basis of the compiler-based array yield analysis (CAYA) tool described in [25]. The second approach (hereafter also referred to as "method B") provides a highly accurate yield estimate through a Markov-based analysis. Previous works on Markov-based analysis can be found in [16]–[18]. As this approach provides a high level of accuracy, the yield found by method B has been used as a reference to evaluate the industrial-based approach represented by method A. The effectiveness and accuracy of the two proposed methods are then compared with respect to the industrial design data for the defects.

This paper is organized as follows. Section II briefly presents yield-modeling issues. In Section III, the two proposed methods are described in detail. In Section IV, both of these methods are compared. Finally, conclusions are drawn in Section V.

II. Global Yield Model

The evaluation of the effects of manufacturing defects on an IC is related to the defect density (given by D), the extent by which defects are clustered (parameterized by α), and the critical area (denoted as A) exposed to the defects [26], [27]. The expected average number of faults on a chip is given by $\lambda_0 = AD$. At manufacturing, defects can be categorized into two main classes [29].

1) Pinhole Defects: These are defects in which the size is not important, such as defects that cause dielectric pinholes of junction leakage. These defects are rather easy to model.
2) Photolithographic defects: These are defects that are comparable in size with photolithographic patterns. In this case, defect sensitivity depends on the defect size.

The feature of a critical area is related to the presence of a defect as a possibly fatal event (i.e., a fault); hence, the size of the circuit pattern in a chip is used to establish whether a defect can cause a fault. In [26] and [27], they have been analyzed through a function fault probability kernel $K(x)$, which is zero when a defect does not cause a fault and one otherwise. The critical area of a circuit is, then, the product of the actual area and the integral of the product of the defect size distribution and the kernel function [26], [27]. The integral is replaced in many cases by a so-called proportionality function. Thus, the critical area is assumed to be simply proportional to the actual area by ignoring any detailed spatial interaction [14]. For manufacturing, the critical area of an SRAM is commonly calculated from the layout using an extraction tool, such as those based on shape expansion, or a Monte Carlo simulation [25], [28].

Structural defects (whose number is denoted by n) and their expected density can be obtained from the manufacturing-line data. Functional faults describe the effect of defects on memory cells, word/bit lines, and peripheral circuitry. Therefore, the number of defects in an SRAM can be calculated by relating defects to the functional faults. Consider the six functional faults as reported in Table I and acquired through industrial practice [25]. The number of faults present in an SRAM can be calculated as follows. Let Λ be a (6×1) matrix representing the average number of functional faults; let A be a $(6 \times n)$ critical area matrix, with an entry $A_{i,j}$ denoting the critical area of a functional fault type j in a critical area i; and let D be a $(n \times 1)$ defect-density matrix for n structural defects. Using the defect-density matrix (obtained from the manufacturing line), the number of faults (of different types) is given as

$$\Lambda^T = (\lambda_{\text{ck}}, \lambda_{\text{sc}}, \lambda_{\text{hp}}, \lambda_{\text{vp}}, \lambda_{\text{row}}, \lambda_{\text{col}}) \tag{1}$$

where ck, sc, hp, vp, row, and col are the functional fault types as described in Table I, and

$$\Lambda = AD. \tag{2}$$

TABLE I
Functional Faults for SRAM Arrays

Functional fault	**Description**
chip kill (ck)	unrepairable fault
single cell (sc)	only one faulty cell
horizontal pair (hp)	two adjacent faulty cells horizontally
vertical pair (vp)	two adjacent faulty cells vertically
single word line (row)	one faulty word line
single data column (col)	one faulty data column (bit lines, DIOs, AIO)

TABLE II
Fault Types Percentages

Chip Kill Fault	$\lambda_{ck} = 0.05 \cdot \lambda_0$
Single Cell Fault	$\lambda_{sc} = 0.45 \cdot \lambda_0$
Horizontal Pair Fault	$\lambda_{hp} = 0.1 \cdot \lambda_0$
Vertical Pair Fault	$\lambda_{vp} = 0.1 \cdot \lambda_0$
Single Row Fault	$\lambda_{row} = 0.15 \cdot \lambda_0$
Single Column Fault	$\lambda_{col} = 0.15 \cdot \lambda_0$

Let λ_0 be the sum of the average number of faults of each type in the SRAM. From the analysis performed on an industrial embedded SRAM [25], the values of Table II have been computed for each type of fault. Given λ_0, the probability of having k faults on a chip can be approximated by the discrete Poisson distribution of a random variable $X = k$ [9], [13]:

$$P\{X = k\} = \frac{e^{-\lambda_0}\lambda_0^k}{k!}. \tag{3}$$

It can be proved that the mean and variance of (3) are both equal to λ_0. The yield is defined as the probability of having no fault on a chip. If the chip has no redundancy, the yield is given by (3), which is computed for $k = 0$, i.e.,

$$Y = P\{X = 0\} = e^{-\lambda_0}. \tag{4}$$

However, as observed from the experimental data, the mean value and variance of the fault distribution do not match λ_0, as predicted by a Poisson distribution [9]. This is caused by the effect of a nonconstant distribution of the defect density on the wafer, i.e., the value of λ_0 varies for different chips. This issue is known as the clustering effect. Therefore, a mixed Poisson distribution is applied using a gamma distribution as mixing function. The result is modeled by a Polya–Eggenberger distribution as

$$P\{X=k\} = \frac{\Gamma(k+\alpha)\left(\frac{\lambda_0}{\alpha}\right)^k}{k!\Gamma(\alpha)\left(1+\left(\frac{\lambda_0}{\alpha}\right)\right)^{\alpha+k}}. \tag{5}$$

The mean and the variance of this distribution function are given by $E(X=k)=\lambda_0$ and $\mathrm{Var}(X)=\lambda_0\cdot(1+\lambda_0/\alpha)$, respectively. Therefore, the yield for a nonredundant chip is derived from (5) for $k=0$ as

$$Y = P\{X=0\} = \left(1+\frac{\lambda_0}{\alpha}\right)^{-\alpha}. \tag{6}$$

This is generally known as the negative-binomial yield model. In (6), α represents the clustering effect of the mean fault density λ_0 on different chips; a value of α, which has been widely adopted in industry, is ≈ 2. As

$$\lim_{\alpha\to\infty}\left(1+\frac{\lambda_0}{\alpha}\right)^{-\alpha} = e^{-\lambda_0}$$

this reduces to (4). When $\alpha\to\infty$, the values of λ_0 for different chips are totally uncorrelated, and therefore, the probability distribution function of λ [i.e., $f(\lambda)$] can be considered as a Dirac pulse on λ_0. Thus, $f(\lambda)=\delta(\lambda-\lambda_0)$.

III. Yield Models

In this section, the description of the two proposed yield models is reported. The considered models are based on two different approaches: The first approach [25] (method A) reduces the computational complexity by introducing some approximations, while the second approach (method B) provides accurate results.

A. Method A

A fault pattern is defined as a vector $\mathrm{FP}=(i_1,i_2,\ldots,i_F)$, where i_k is the number of type-k faults. The concept of fault pattern (or FP) has been introduced in [9] and [22]. Given the average number of failures occurring on a chip, λ_0 can be split into a set of different functional faults, i.e., the value λ_0 can be seen as the sum of λ_i, as the distinct average occurrences of F possible fault types (with $i=1,2,\ldots,F$), i.e.,

$$\lambda_0 = \sum_{i=1,F}\lambda_i. \tag{7}$$

By considering a set of fault types for the SRAM (as described previously in Section II), (7) can be written as

$$\lambda_0 = \lambda_{\mathrm{sc}}+\lambda_{\mathrm{vp}}+\lambda_{\mathrm{hp}}+\lambda_{\mathrm{row}}+\lambda_{\mathrm{col}}+\lambda_{\mathrm{ck}}. \tag{8}$$

Method A is consist of two steps.

1) The first step evaluates the yield as the sum of the probabilities of all repairable fault patterns on the chip (by considering a Poisson distribution of each fault).
2) The second step considers the clustering effect by performing the inversion of the results obtained in the first step and using λ_{r} (as the average number of faults left unrepaired) into a negative-binomial distribution function.

In the first step, the yield is calculated as the sum of the probabilities of all repairable fault patterns (based on repairable faults or RFs) of a chip, each occurring with a Poisson probability distribution. Using the FPs, the yield of a repairable chip is, therefore, the probability of having all RFs. So, if the FPs are assumed to be disjoint, then

$$Y = \sum_{\mathrm{RF}} Pr\{\mathrm{FP}_i\}. \tag{9}$$

Therefore, the proposed analysis consists of estimating the yield that is attainable by using an exhaustive repair algorithm. Assuming a Poisson distribution for k faults of type i, then

$$P_i(k) = \frac{e^{-\lambda_i}\lambda_i^{-k}}{k!}. \tag{10}$$

From (9), the yield after repairing the SRAM using the provided redundancy (without considering the clustering effect) is given by

$$Y_{\mathrm{r}} = P_{\mathrm{ck}}(0)\sum_{C_F}P_{\mathrm{sc}}(i)P_{\mathrm{vp}}(j)P_{\mathrm{hp}}(k)P_{\mathrm{row}}(l)P_{\mathrm{col}}(m). \tag{11}$$

In (11), Y_{r} is the yield after repairing $(i+j+k+l+m)$ faults of different types. Repair effectively translates into a process by which most (in some cases all) faults can be corrected. Let the number of faults left unrepaired be given by λ_{r} and Y_{p} denote the so-called perfect yield, i.e., the probability that there is no fault left unrepaired. As the average number of faults to be repaired is λ_0, then

$$Y_{\mathrm{p}} = P(0) = (1+\lambda_0/\alpha)^{-\alpha}. \tag{12}$$

Starting from (12), the second step of method A introduces the clustering effect that was not considered in the computation of Y_{r} [by (11)]. λ_{r} can be considered as the average number of faults left unrepaired after the first step; this is obtained by inverting the Poisson expression computed for $k=0$, i.e., $Y_{\mathrm{r}}=\mathrm{e}^{-\lambda_{\mathrm{r}}}$. By substituting the computed λ_{r} into (12) and by considering the clustering effect, the final yield is given by

$$Y_f = (1+\lambda_{\mathrm{r}}/\alpha)^{-\alpha}. \tag{13}$$

B. Method B

Similar to method A, method B is made of two consecutive steps to estimate the memory yield. The SRAM is again modeled by utilizing a given row redundancy and a probabilistic characterization of the faults. As for method A, the clustering

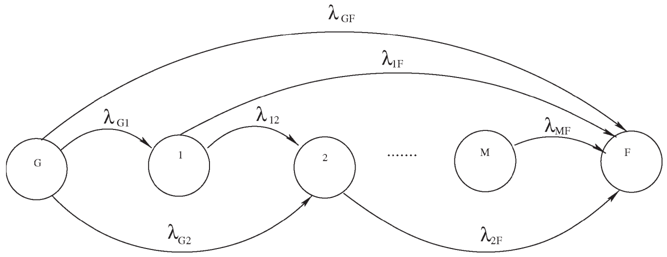


Fig. 1. Markov chain for spare rows case.

effect is also taken into account. Method B consists of two steps.

1) In the first step, the yield after repair is computed by solving a Markovian model.
2) In the second step, the previously obtained results are integrated with the gamma distribution function to obtain the yield (including the clustering effect).

Consider in more detail these two steps. The repair process can be described as a series of states representing the different configurations of the memory, depending on the number of faults that have been repaired. In [16], an estimate of the yield is found by analyzing a finite-state Markov chain (representing all possible chip repair states). The case of a memory array with only spare rows can be considered as an M out of N reliability problem, i.e., M elements out of N must be operative. This type of problem is typically modeled using Markov chains; thus, the repair process can also be modeled as a continuous time Markov chain (CTMC).

The model shown in Fig. 1 has $M+2$ states, where M is the number of available spare rows, and two additional states represent the initial "good chip" state (i.e., G) and the fail state (i.e., F). Therefore, each state corresponds to the level of redundancy currently available in the memory. The occurrence of a fault causes a state transition to the corresponding memory configuration after (possibly) performing a repair operation. Thus, the proposed repair algorithm processes faults one at a time. It is straightforward to prove that, if only spare rows are provided (as in the considered case), a sequential algorithm is also exhaustive.

The transition rates for the Markov chain model are shown in Table III. The same set of possible faults as in Table I has been considered. However, the values of the fault rates (marked with an asterisk in Table III) must be divided by either the number of columns or rows (or their product). As all the faults are assumed to be independent, their weighted probabilities are summed when characterizing a transition rate. In Table III, s_{r} denotes the number of spare rows, n_{c} denotes the number of columns, and n_{r} denotes the number of rows. For the terms related to vertical pair faults in Table III, consider

$$\lambda'_{\mathrm{vp}}(i,i+1) = n_{\mathrm{c}}n_{\mathrm{r}}\lambda^*_{\mathrm{vp}}\left[1-\left(\frac{i}{n_{\mathrm{r}}}\frac{i-1}{n_{\mathrm{r}}-1}+\frac{n_{\mathrm{r}}-i}{n_{\mathrm{r}}}\frac{n_{\mathrm{r}}-i-1}{n_{\mathrm{r}}-1}\right)\right]$$

$$\lambda'_{\mathrm{vp}}(i,i+2) = n_{\mathrm{c}}n_{\mathrm{r}}\lambda^*_{\mathrm{vp}}\left(\frac{n_{\mathrm{r}}-i}{n_{\mathrm{r}}}\cdot\frac{n_{\mathrm{r}}-i-1}{n_{\mathrm{r}}-1}\right). \quad (14)$$

To understand the transition rates reported in Table III, the following general rule can be used: For an edge between two

TABLE III
DEFECT INTENSITIES

$\lambda_{G,F}$	$n_c\lambda^*_{col}+\lambda^*_{ck}$
$\lambda_{G,1}$	$n_cn_r\lambda^*_{sc}+n_r\lambda^*_{row}+n_cn_r\lambda^*_{hp}$
$\lambda_{G,2}$	$n_cn_r\lambda^*_{vp}$
$\lambda_{1,F}$	$n_c\lambda^*_{col}+\lambda^*_{ck}$
$\lambda_{1,2}$	$n_c(n_r-1)\lambda^*_{sc}+(n_r-1)\lambda^*_{row}+$ $n_c(n_r-1)\lambda^*_{hp}+\lambda'_{vp}(1,2)$
$\lambda_{1,3}$	$\lambda'_{vp}(1,3)$
$\lambda_{2,F}$	$n_c\lambda^*_{col}+\lambda^*_{ck}$
$\lambda_{2,3}$	$n_c(n_r-2)\lambda^*_{sc}+(n_r-2)\lambda^*_{row}+$ $n_c(n_r-2)\lambda^*_{hp}+\lambda'_{vp}(2,3)$
$\lambda_{2,4}$	$\lambda'_{vp}(2,4)$
$\lambda_{i,F}$	$n_c\lambda^*_{col}+\lambda^*_{ck}$
$\lambda_{i,i+1}$	$n_c(n_r-i)\lambda^*_{sc}+(n_r-i)\lambda^*_{row}+$ $n_c(n_r-i)\lambda^*_{hp}+\lambda'_{vp}(i,i+1)$
$\lambda_{i,i+2}$	$\lambda'_{vp}(i,i+2)$
$\lambda_{M,F}$	$n_c(n_r-s_r)\lambda^*_{sc}+(n_r-s_r)\lambda^*_{row}+$ $n_c\lambda^*_{col}+\lambda^*_{ck}+(n_r-s_r)\lambda^*_{row}+$ $n_c(n_r-s_r)\lambda^*_{hp}+\lambda'_{vp}(s_r,s_r+1)$

states, its transition rate is the weighted sum of λ^* of all possible faults that can cause that transition, and the weight is the number of elements that are exposed to that fault. As an example, consider the transition rate $\lambda_{G,1}$ from a state with no allocated spare row (G) to a state with only one allocated spare row. In this case, the transition rate is given by the sum of three terms.

1) $n_{\mathrm{c}}n_{\mathrm{r}}\lambda^*_{\mathrm{sc}}$: This is the probability of having a single cell fault in any of the $n_{\mathrm{c}}\times n_{\mathrm{r}}$ cells of the array.
2) $n_{\mathrm{r}}\lambda^*_{\mathrm{row}}$: This is the probability of having a row fault in any of the n_{r} rows of the array.
3) $n_{\mathrm{c}}n_{\mathrm{r}}\lambda^*_{\mathrm{hp}}$: This is the probability of having a horizontal cell fault in any of the $n_{\mathrm{c}}\times n_{\mathrm{r}}$ cells of the array; this is not dependent on the location of the faulty pair of cells within a row, because repair utilizes an entire row.

The case in which vertical pair faults contribute to a transition rate is more complex. Equation (14) defines the probability of increasing the number of allocated rows by one $(\lambda'_{\mathrm{vp}}(i,i+1))$ or two $(\lambda'_{\mathrm{vp}}(i,i+2))$ spares, respectively. For $\lambda'_{\mathrm{vp}}(i,i+2)$, this rate is given by the product of the vertical pair fault rate of each possible cell $(n_{\mathrm{c}}n_{\mathrm{r}}\lambda^*_{\mathrm{vp}})$ and the probability that the pair of affected rows was not already repaired $[(n_{\mathrm{r}}-i/n_{\mathrm{r}})\cdot(n_{\mathrm{r}}-i-1/n_{\mathrm{r}}-1)]$. Else, the considered vertical pair may increase the number of allocated rows by at most one. For $\lambda'_{\mathrm{vp}}(i,i+1)$, this rate is given by the product of the vertical pair fault rate of each possible cell $(n_{\mathrm{c}}n_{\mathrm{r}}\lambda^*_{\mathrm{vp}})$ and the probability that the repair of this fault increases the allocation of spare rows by one. This last rate is given by the probability of not utilizing either a pair of adjacent rows already affected by the fault or a pair in which neither row is already affected $(\{1-[(i/n_{\mathrm{r}})\,(i-1/n_{\mathrm{r}}-1)+(n_{\mathrm{r}}-i/n_{\mathrm{r}})\,(n_{\mathrm{r}}-i-1/n_{\mathrm{r}}-1)]\})$.

To solve the proposed Markov model, all transitions are assumed to be slow, i.e., the transitions occur with an exponential probability density function, and the rates (λ_{ij}) are constant during the simulation time. The CTMC is then solved by computing the solution of the following set of differential equations:

$$P'(\lambda) = P(\lambda)\mathbf{A} \tag{15}$$

where $P(\lambda)$ is a vector whose elements are the probabilities of being in a state $(G, 1, 2, \ldots, M, F)$, and $\boldsymbol{A}$ is the so-called generating matrix whose elements are the transitions' rates

$$\mathbf{A} = \begin{pmatrix} \lambda_{11} & \lambda_{12} & \cdots \\ \lambda_{21} & \lambda_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

The numerical solution of (15) is obtained by considering its solution as

$$P(\lambda) = P(0)e^{(A\lambda)}.$$

For $\Delta\lambda \ll 1$, the above equation can be written as

$$P(\Delta\lambda) = P(0)(I - A\Delta\lambda)$$

and therefore, in general, for any given λ

$$P(\lambda + \Delta\lambda) = P(\lambda)(I - A\Delta\lambda). \tag{16}$$

Equation (16) can be solved numerically (albeit this approach requires a matrix multiplication at each step), and the yield after repair is calculated as the probability of not being in the fail state (F state).

In the second step of the proposed method, the numerical integration with the gamma distribution function is required. The actual yield (inclusive of the clustering effect) is thus obtained, i.e.,

$$\begin{aligned} Y_t(\lambda_0, \alpha) &= \int_0^\infty Y(\lambda) \cdot g(\lambda, \lambda_0, \alpha) \mathrm{d}\lambda \simeq \\ &= \sum_{n=0}^{n=\text{large}} Y(n\Delta\lambda) \cdot g(n\Delta\lambda, \lambda_0, \alpha)\Delta\lambda. \end{aligned} \tag{17}$$

$g(\lambda, \lambda_0, \alpha)$ is the gamma probability distribution function, and its general definition is given in [30] as

$$g(x) = \gamma x^{b-1} e^{-cx} U(x) \tag{18}$$

where $\gamma = [c^b/\Gamma(b)]$, $U(x)$ is the step function, b and c are positive numbers, and $\Gamma()$ is the gamma function. The gamma function is defined as

$$\Gamma(b+1) = \int_0^\infty y^b e^{-y} dy$$

with $b > -1$. $\Gamma()$ is also referred to as the generalized factorial, because $\Gamma(b+1) = b\Gamma(b)$ [if b is an integer, $\Gamma(b+1) = b!$, $\Gamma(1) = 1$].

As reported in [14], (18) is used to weight the clustering effect in the yield analysis by assuming, for the parameters x, c, b, the following values: $x = \lambda$, $c = (\alpha/\lambda_0)$; finally, $b = \alpha$ [9], [14]. Therefore

$$g(\lambda, \lambda_0, \alpha) = \frac{\alpha^\alpha}{\lambda_0^\alpha \cdot \Gamma(\alpha)} \lambda^{\alpha-1} e^{-\frac{\alpha}{\lambda_0}\lambda}.$$

The numerical formulation of the actual yield in (17) is then

$$Y_t(\lambda_0, \alpha) = \sum_{n=0}^{n=\text{large}} Y(n\Delta\lambda) \cdot \frac{\alpha^\alpha}{\lambda_0^\alpha \cdot \Gamma(\alpha)} \lambda^{\alpha-1} \mathrm{e}^{-\frac{\alpha}{\lambda_0} n\Delta\lambda} \Delta\lambda. \tag{19}$$

The numerical solution of the first and second steps has been performed by Matlab simulation [31]. The computational complexity of these steps in the proposed method is dependent on the selected simulation step (given by $\Delta\lambda$).

IV. Comparison

In this section, a comparison of the yield estimates obtained by the two proposed methods is provided. In summary, the proposed methods are described in the following:

1) Method A: Fault pattern (step 1) and inversion approximated solution (step 2);
2) Method B: Markov model (step 1) and numerical gamma-function integration (step 2).

Both methods consist of two steps. The first step calculates the yield by assuming a Poisson distributed failure probability, and the second step introduces suitable modifications to the values obtained in the first step to take into account the clustering effect. The two methodologies differ in many aspects. Method B relies on the dynamic evolution of a repair algorithm, while method A is based on a static probabilistic analysis. For the considered case, the presence of only spare rows implies that the sequential algorithm used in the Markov chain is exhaustive, and therefore, the results are comparable to those obtained by method A. However, when both spare columns and rows are present, method B is more flexible to allow different repair algorithms in the yield evaluation. For example, a greedy algorithm could be evaluated with method B as an efficient alternative in terms of performance.

Also, the two methods differ in execution by considering the memory array size and aspect ratio when computing the final yield. Both methods use fault data obtained industrially by critical area analysis (whose values are reported in Table II). These data depend on the size and aspect ratio of the considered memory array and are applicable for all the configurations of the reported memory array (1024 × 512, 1024 × 1024, and 2048 × 1024). Other than the fault data, method A does not take into further account issues such as size and aspect ratio. Method B is based on parametric fault rates (found by dividing the values in Table III by the number of rows, columns, or their product as applicable) and can be used to accurately consider actual sizes and aspect ratios; thus, it can be extended to different array configurations.

The two proposed methodologies are hereafter compared with respect to complexity and accuracy. As for the computational complexity, let F be the number of possible fault types [as in (7)] and s_r be the number of available spare rows. The complexity of the first step of method A is given by $O(s_\mathrm{r}^F)$. For a given number of faults and available spare rows, the exhaustive algorithm from which method A is based searches whether, for each possible fault, there is a repair configuration permitted by the available spare rows. By also considering the clustering effect, the computational complexity of method A does not increase because the algorithm only needs to compute two more operations. Method B instead is based on matrix multiplications to solve (15) for the Markovian model. The number of matrix multiplications to solve (16) is related to the required accuracy (which is higher for lower values of the execution step $\Delta\lambda$) and the maximum value of the average fault number λ_0. Given λ_0 and an execution step $\Delta\lambda$ and $n_\mathrm{max} = (\lambda_0/\Delta\lambda)$, then the computational complexity is $O(n_\mathrm{max} \cdot s_\mathrm{r}^3)$. Similarly, the computation complexity of the second step is related to the solution of (17). Thus, it requires additional $O(n_\mathrm{max}^2)$ operations on the previously computed yield after repair.

As for the accuracy of the proposed two methods, two features must be considered. For the first step, as previously mentioned, method B can be reliably used for evaluating arrays of different size and aspect ratio. Moreover, it provides an evaluation over a more accurate set of possible fault patterns than method A. Method A does not consider possible overlaps of the same types of faults; for example, neither two single cell faults on the same row nor two vertical pairs overlapping on a row are taken into account. As for the second step, method B introduces a higher level of accuracy with the gamma-function integration because this is more suitable in evaluating the clustering effect than the inversion-based approach. As described previously, this results in a higher computational complexity.

In the remainder of this section, the results of the Matlab-based simulations [31] to compare the two methods are presented. In particular, the two methods have been compared in both steps 1 and 2 by calculating the yield for differently sized RAM arrays. The chosen sizes are 0.5 Mbit (1024 × 512), 1 Mbit (1024 × 1024), and 2 Mbit (2048 × 1024). For each of these memory arrays, redundancies of zero, one, two, four, and eight spare rows have been considered.

From the reported values, it can be observed that the first steps of the considered methods give close results for small values of λ_0 (Tables IV–VI). The values of λ_0 reported in the tables have been chosen according to the values of the defect density in manufacturing lines of a mature industrial process. The plot reported in Fig. 2 shows that yield estimates overlap for a wide range of λ_0. In particular, in this plot, simulation results are shown for a 1024 × 1024 memory array by varying the average fault rate λ_0 from 0 to 15 with a step $\Delta\lambda = 10^{-3}$.

The results of step 2 in both methods are compared in Tables VII–IX. These tables show that the yield estimates, which now take into account the clustering effect, are still very close for a small λ_0. However, the effect of the approximation introduced in the second step of method A is evident at higher values. This can be clearly observed in Fig. 3 for a 1-M SRAM array (1024 × 1024). For higher values of λ_0, the yield calculated by method A tends to be underestimated. For a new manufacturing line, the yield is likely to be very low because the process is not yet finely tuned. Thus, it is

TABLE IV
ARRAY SIZE 0.5 M (1024 ROWS 512 COLUMNS) AND $\lambda_0 = 0.1$ FOR STEP 1

	Method A	Method B	Difference (B-A)
no spare	9.0479215e-01	9.0438208e-01	-4.1007e-04
one row	9.6819314e-01	9.6833027e-01	1.3713e-04
two rows	9.7944832e-01	9.7949935e-01	5.103e-05
four rows	9.8019289e-01	9.8017626e-01	-1.663e-05
eight rows	9.8019671e-01	9.8017904e-01	-1.767e-05

TABLE V
ARRAY SIZE 1 M (1024 ROWS 1024 COLUMNS) AND $\lambda_0 = 0.2$ FOR STEP 1

	Method A	Method B	Difference (B-A)
no spare	8.1873075e-01	8.1864883e-01	-8.192e-04
one row	9.3335306e-01	9.3338220e-01	2.914e-05
two rows	9.5775123e-01	9.5776794e-01	1.671e-05
four rows	9.6075544e-01	9.6075240e-01	-3.04e-06
eight rows	9.6078944e-01	9.6078559e-01	-3.85e-06

TABLE VI
ARRAY SIZE 2 M (2048 ROWS 1024 COLUMNS) AND $\lambda_0 = 0.4$ FOR STEP 1

	Method A	Method B	Difference (B-A)
no spare	6.7032005e-01	6.701859e-01	-1.3415e-04
one row	8.5800966e-01	8.5803862e-01	2.896e-05
two rows	9.1109901e-01	9.1112907e-01	3.006e-05
four rows	9.2281806e-01	9.2281423e-01	-3.83e-06
eight rows	9.2311629e-01	9.2310890e-01	-7.39e-06

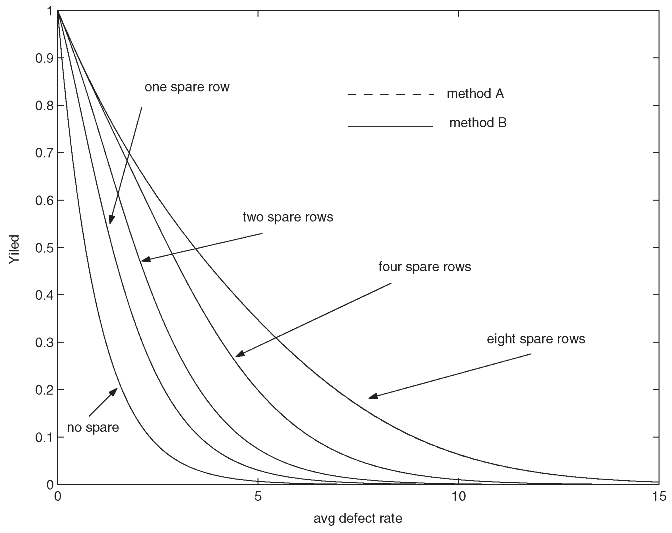


Fig. 2. Step 1: A yield comparison varying λ_0 for a 1024 × 1024 memory array.

TABLE VII
ARRAY SIZE 0.5 M (1024 ROWS 512 COLUMNS) AND $\lambda_0 = 0.1$ FOR STEP 2

	Method A	Method B	Difference (B-A)
no spare	9.0702948e-01	9.0695293e-01	-7.655e-05
one row	9.6842654e-01	9.6748070e-01	-9.4584e-04
two rows	9.7954620e-01	9.8024805e-01	7.0185e-04
four rows	9.8029211e-01	9.8024805e-01	-4.406e-05
eight rows	9.8029605e-01	9.8026077e-01	-3.528e-05

TABLE VIII
ARRAY SIZE 1 M (1024 ROWS 1024 COLUMNS) AND $\lambda_0 = 0.2$ FOR STEP 2

	Method A	Method B	Difference (B-A)
no spare	8.2644628e-01	8.2636277e-01	-8.351e-05
one row	9.3443883e-01	9.3164755e-01	-2.79128e-03
two rows	9.5819119e-01	9.5670682e-01	-1.48437e-03
four rows	9.6113543e-01	9.6104487e-01	-9.056e-05
eight rows	9.6116878e-01	9.6115662e-01	-1.216e-05

TABLE IX
ARRAY SIZE 2 M (2048 ROWS 1024 COLUMNS) AND $\lambda_0 = 0.4$ FOR STEP 2

	Method A	Method B	Difference (B-A)
no spare	6.9444444e-01	6.9432656e-01	-1.1788e-04
one row	8.6281062e-01	8.5649907e-01	-6.31155e-03
two rows	9.1301623e-01	9.0801210e-01	-5.00413e-03
four rows	9.2426897e-01	9.2360873e-01	-6.6024e-04
eight rows	9.2455615e-01	9.2454467e-01	-1.1480e-05

worth considering higher values of λ_0 at the early stages of the manufacturing process; as an example, Table X shows the values of the yield computed for an average of four defects per array.

V. CONCLUSION

In this paper, two methods for calculating the yield of repairable RAM arrays have been proposed and compared. The analyzed methods have been chosen to be representatives of two different approaches. Method A provides the basis of the CAYA tool described in [25]. It is simple, and its computation is faster, but it is based on an approximation. Method B is more complex; its computational complexity is higher, but it is more flexible and accurate.

The yield computation of these two methods is accomplished in two steps. In the first step, the yield under a given defect rate is computed without considering the clustering effect. In the second step, the effect of clustering is introduced by suitable procedures. The intermediate and final results of the two methods have been compared. Results are shown to be very close for low values of the average number of defects as reflecting manufacturing lines of a mature process.

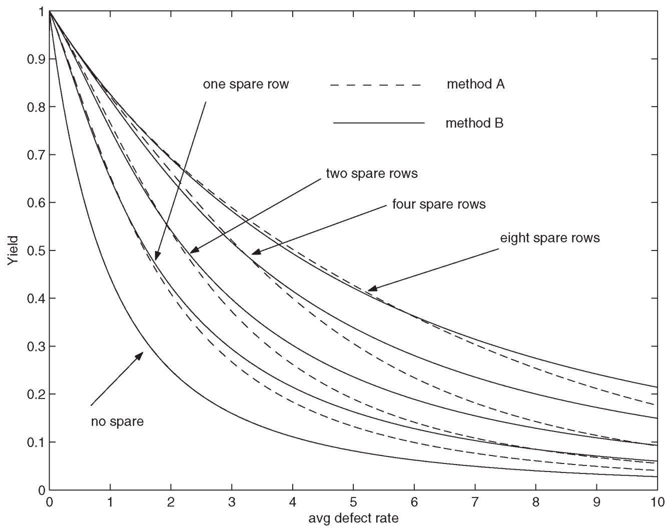


Fig. 3. Step 2: A yield comparison varying λ_0 for a 1024 × 1024 memory array.

TABLE X
ARRAY SIZE 1 M (1024 ROWS 1024 COLUMNS) AND $\lambda_0 = 4$ FOR STEP 2

	Method A	Method B	Difference (B-A)
no spare	1.1111111e-01	1.1103700e-01	-7.411e-05
one row	1.8380482e-01	2.1482265e-01	3.101783e-02
two rows	2.6222188e-01	3.0233323e-01	4.011135e-02
four rows	4.0045785e-01	4.1557794e-01	1.512009e-02
eight rows	5.0260367e-01	4.9391717e-01	-8.6865e-03

For higher defect densities, as for the initial stages of a new manufacturing line, method A underestimates the yield even if the results of the first steps are still close. Therefore, the first steps of both computations are mostly equivalent, while by introducing the clustering effect, the results differ at higher values. Thus, to obtain more accurate results for higher values of the average defect rate, the second step of method B should be used.

REFERENCES

[1] "International Roadmap for Semiconductors," SIA, San Jose, CA, 2005. [Online]. Available: http://www.itrs.net/Links/2005ITRS/Home2005.htm
[2] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L. Lewandowski, "Built in self repair for embedded high density SRAM," in *Proc. IEEE Int. Test Conf.*, 1998, pp. 1112–1118.
[3] S. Tsuchida, "Test and repair of non-volatile commodity and embedded memories," in *Proc. IEEE Int. Test Conf.*, 2002, p. 1223.
[4] B. Cowan, O. Farnsworth, P. Jakobsen, S. Oakland, M. R. Oulette, and D. L. Wheater, "On-chip repair and an ATE independent fusing methodology," in *Proc. IEEE Int. Test Conf.*, 2002, pp. 178–186.
[5] Y. Nagura, M. Mullins, A. Sauvageau, Y. Fujiwara, K. Furue, R. Ohmura, T. Komoike, T. Okitaka, T. Tanizaki, K. Dosaka, Y. Koda, and T. Tada, "Test cost reduction by at-speed BISR for embedded DRAMs," in *Proc. IEEE Int. Test Conf.*, 2001, pp. 182–187.
[6] B. T. Murphy, "Cost-size optima of monolithic integrated circuits," *Proc. Inst. Electr. Eng.*, vol. 52, no. 12, pp. 1537–1545, Dec. 1964.
[7] R. B. Seeds, "Yield economics and logistic models for complex digital arrays," in *Proc. IEEE Int. Conv. Rec.*, Oct. 1967, pp. 60–61.
[8] W. G. Ansley, "Computation of integrated circuits yields from the distribution of slice yields for the individual devices," *IEEE Trans. Electron Devices*, vol. ED-15, no. 6, pp. 405–406, Jun. 1968.

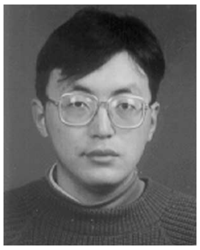
[9] C. H. Stapper, "LSI yield modeling and process monitoring," *IBM J. Res. Develop.*, vol. 20, no. 3, pp. 228–234, 1976.
[10] ——, "Large-area fault clusters and fault tolerance in VLSI circuits: A review," *IBM J. Res. Develop.*, vol. 33, no. 2, pp. 162–173, 1989.
[11] F. J. Meyer and D. K. Pradhan, "Modeling defect spatial distribution," *IEEE Trans. Comput.*, vol. 38, no. 4, pp. 538–546, Apr. 1989.
[12] C. H. Stapper, "Small-area fault clusters and fault tolerance in VLSI circuits," *IBM J. Res. Develop.*, vol. 33, no. 2, pp. 174–177, Mar. 1989.
[13] I. Koren and Z. Koren, "Defect tolerant VLSI circuits: Techniques and yield analysis," *Proc. Inst. Electr. Eng.*, vol. 86, no. 9, pp. 1817–1836, Sep. 1998.
[14] T. Michalka, R. Varshney, and J. Meindl, "A discussion of yield modeling with defect clustering, circuit repair, and circuit redundancy," *IEEE Trans. Semicond. Manuf.*, vol. 3, no. 3, pp. 116–127, Aug. 1990.
[15] I. Koren, Z. Koren, and C. H. Stapper, "A unified negative-binomial distribution for yield analysis of defect-tolerant circuits," *IEEE Trans. Comput.*, vol. 42, no. 6, pp. 724–734, Jun. 1993.
[16] B. Ciciani and G. Iazeolla, "A Markov chain-based yield formula for VLSI fault-tolerant chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 2, pp. 252–259, Feb. 1991.
[17] G. Battaglini and B. Ciciani, "An improved analytical yield evaluation method for redundant RAM's," in *Proc. Int. Workshop Memory Technol., Des., and Testing*, Aug. 24/25, 1998, pp. 117–123.
[18] M. Ottavi, X. Wang, F. J. Meyer, and F. Lombardi, "Simulation of reconfigurable memory core yield," in *Proc. 14th ACM Great Lakes Symp. VLSI*, Boston, MA, Apr. 2004, pp. 136–140.
[19] S.-Y. Kuo and W. K. Fuchs, "Efficient spare allocation in reconfigurable arrays," *IEEE Des. Test*, vol. 4, no. 1, pp. 24–31, 1987.
[20] M. Tarr, D. Boudreau, and R. Murphy, "Defect analysis system speeds test and repair of redundant memories," *Electronics*, vol. 57, no. 1, pp. 175–179, Jan. 1984.
[21] R. C. Evans, "Testing repairable RAMs and mostly good memories," in *Proc. Int. Test Conf.*, Oct. 1981, pp. 49–55.
[22] C. H. Stapper, A. N. Mclaren, and M. Dreckmann, "Yield model for productivity optimization of VLSI memory chips with redundancy and partially good product," *IBM J. Res. Develop.*, vol. 24, no. 3, pp. 309–409, 1980.
[23] J. Day, "A fault-driven comprehensive redundancy algorithm," *IEEE Des. Test*, vol. 2, no. 3, pp. 35–44, Jun. 1985.
[24] K. W. Huang and F. Lombardi, "Approaches for the repair of VLSI/WSIRRAMs by row/column deletion," in *Proc. IEEE FTCS18*, 1988, pp. 342–347.
[25] X. Wang, M. Ottavi, and F. Lombardi, "Yield analysis of compiler-based arrays of embedded SRAMs," in *Proc. 18th IEEE Int. Symp. Defect and Fault Tolerance VLSI Syst.*, Boston, MA, Nov. 2003, pp. 3–10.
[26] A. V. Ferris-Prabhu, "Defect size variations and their effect on the critical area of VLSI devices," *IEEE J. Solid-State Circuits*, vol. SSC-20, no. 4, pp. 878–880, Aug. 1985.
[27] ——, "Modeling the critical area in yield forecasts," *IEEE J. Solid-State Circuits*, vol. SSC-20, no. 4, pp. 874–878, Aug. 1985.
[28] G. A. Allan, "A comparison of efficient dot throwing and shape shifting extra material critical area estimation," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance VLSI Syst.*, Austin, TX, Nov. 1998, pp. 44–52.
[29] C. H. Stapper, "Modeling of integrated circuit defect sensitivities," *IBM J. Res. Develop.*, vol. 27, no. 6, pp. 549–557, 1983.
[30] A. Papoulis, *Probability, Random Variables and Stochastic Processes*. New York: McGraw-Hill, 2002.
[31] *MATLAB User's Guide*, The MathWorks, Inc., Natick, MA, 1994–2006.

Marco Ottavi (M'04) received the Laurea degree in electronic engineering from University of Rome "La Sapienza," Rome, Italy, in 1999 and the Ph.D. degree in microelectronic and telecommunications engineering from University of Rome "TorVergata," Rome, in 2004.

In 2000, he was with the ULISSE Consortium, Rome, as a Designer Engineer of digital systems for space applications. In 2003, he was a Visiting Research Assistant with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA, where, since 2004, he has been a Postdoctoral Research Associate. His research interests include yield and reliability modeling, fault-tolerant architectures, and online testing and design of nanoscale circuits and systems.

Luca Schiano (S'04) was born in Napoli, Italy, in 1975. He received the "Laurea" degree in electronic engineering from University of Bologna, Bologna, Italy, in 2001. He is currently working toward the Ph.D. degree in computer engineering at the Electrical and Computer Engineering Department, Northeastern University, Boston, MA.

His research interests are IC testing, automatic test equipment (ATE), online testing reliability, and nanotechnologies.



Xiaopeng Wang was born in Xuzhou, Jiangsu, China. He received the B.S. degree in computer engineering from China University of Mining and Technology, Jiangsu, in 1992, the M.S. degree in computer science from Fudan University, Shanghai, China, in 1997, and the Ph.D. degree in computer engineering from Northeastern University, Boston, MA, in 2004.

He did an intern job on the yield modeling of embedded memories in 2001 and 2002 with the ASIC Department of IBM Corporation, Essex Junction, VT, and officially joined the ASIC Department of IBM in 2003 as an embedded SRAM designer. His research interests include very large-scale integration (VLSI) design, VLSI testing, algorithm analysis, and computer architecture.

Yong-Bin Kim (S'88–M'88–SM'00) was born in Seoul, Korea, in 1960. He received the B.S. degree in electrical engineering from Sogang University, Seoul, in 1982, the M.S. degree in computer engineering from New Jersey Institute of Technology, Newark, in 1989, and the Ph.D. degree in computer engineering from Colorado State University, Fort Collins, in 1996.

From 1982 to 1987, he was with the Electronics and Telecommunications Research Institute, Seoul, as a member of the technical staff. From 1990 to 1993, he was with Intel Corporation as a Senior Design Engineer and was involved in microcontroller chip design and Intel P6 microprocessor chip design. From 1993 to 1996, he was with Hewlett Packard Company, Fort Collins, as a member of the technical staff and was involved in the HP PA-8000 RISC microprocessor chip design. From 1996 to 1998, he was with Sun Microsystems, Palo Alto, CA, as an individual contributor and was involved in the 1.5-GHz Ultra Sparc5 CPU chip design. From 1998 to 2000, he was an Assistant Professor with the Department of Electrical Engineering, University of Utah, Salt Lake City. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA. His research focuses on high-speed low-power VLSI circuit design and methodology.

Fred J. Meyer (S'86–M'91) received the Ph.D. degree from the Electrical and Computer Engineering Department, University of Massachusetts, Amherst, in 1991.

He is currently an Assistant Professor of electrical and computer engineering with the Wichita State University, Wichita, KS. His research interests include integrated-circuit (IC) yield modeling, fault-tolerant architectures, and digital design and test.

Dr. Meyer is a member of the IEEE Societies on Computers and Reliability and the IEEE Council on Test Technology.

Fabrizio Lombardi (M'82–SM'02) received the B.Sc. degree (Hons.) in electronic engineering from the University of Essex, Essex, U.K., in 1977, the Master in microwaves and modern optics degree from University College London, London, U.K., in 1978, and the Diploma in microwave engineering and the Ph.D. degree from University of London in 1978 and 1982, respectively.

In 1977, he joined the Microwave Research Unit at University College London. He is currently the holder of the International Test Conference (ITC) Endowed Professorship at Northeastern University, Boston, MA. At the same institution during the period 1998–2004, he served as the Chair of the Department of Electrical and Computer Engineering. Prior to Northeastern University, he was a faculty member at Texas Tech University, Lubhock, the University of Colorado-Boulder, and Texas A&M University, College Station. His research interests are testing and design of digital systems, quantum and nanocomputing, ATE systems, configurable/network computing, defect tolerance, and CAD VLSI. He has extensively published in these areas and has edited six books.

Dr. Lombardi has been involved in organizing many international symposia, conferences, and workshops sponsored by professional organizations as well as guest editor of special issues in archival journals and magazines such as the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, IEEE MICRO MAGAZINE, and IEEE DESIGN & TEST MAGAZINE. He is the Founding General Chair of the IEEE Symposium on Network Computing and Applications. He was an Associate Editor (1996–2000) of the IEEE TRANSACTIONS ON COMPUTERS and a distinguished visitor of the IEEE-CS (1990–1993 and 2001–2004). Since 2000, he has been the Associate Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTERS and an Associate Editor of the IEEE DESIGN AND TEST MAGAZINE. Since 2004, he has been the Chair of the Committee on "Nanotechnology Devices and Systems" of the Test Technology Technical Council of the IEEE. He has received many professional awards: the Visiting Fellowship at the British Columbia Advanced System Institute, University of Victoria, Victoria, BC, Canada (1988), the Texas Experimental Engineering Station Research Fellowship (1991–1992 and 1997–1998), the Halliburton Professorship (1995), the Outstanding Engineering Research Award at Northeastern University (2004), and an International Research Award from the Ministry of Science and Education of Japan (1993–1999). He was the recipient of the 1985/86 Research Initiation Award from the IEEE/Engineering Foundation and of a Silver Quill Award from Motorola-Austin (1996).