Estimating the Manufacturing Yield of Compiler-Based Embedded SRAMs

Xiaopeng Wang, Marco Ottavi, *Member, IEEE*, Fred J. Meyer, *Member, IEEE*, and Fabrizio Lombardi, *Senior Member, IEEE*

***Abstract*—This paper provides a detailed analysis of the yield of embedded static random access memories (eSRAM) generated using a compiler. Defect and fault analysis inclusive of industrial data are presented for these chips by taking into account the design and the physical properties of the layout. A new tool called compiler-based Array Yield Analysis (CAYA) is introduced. CAYA allows for a characterization of the design process which accounts for fault types and the relation between functional and structural faults; moreover, it also relies on a novel empirical model which facilitates yield calculation. Industrial data is provided for the analysis of various configurations with different structures and redundancy. Architectural considerations, such as array size and line (word and bit) organization are analyzed. Compiler-based features of different kernels (such as required for decoding) are also treated in detail. An extensive evaluation of the provided redundancy (row, column, and combined) is pursued to characterize its impact on the memory yield. Industrial data is used in the evaluation and an industrial ASIC chip (made of multiple eSRAMs) is also considered as design case.**

***Index Terms*—Memory architecture, yield estimation.**

I. Introduction

TODAY'S integrated circuits (ICs) rely on efficient design techniques which allow manufacturing of complex digital systems. For cost-effectiveness, the yield (i.e., the percentage of working or fault free chips in a batch) is commonly used as figure of merit for IC manufacturing. Due to the increased sophistication of current technology, the yield is closely monitored for chips with large production batches, as often encountered in consumer electronics applications which require system-on-chip (SoC), as well as application specific ICs (ASIC). While dynamic array configurations are possible, one of the most common type of memory remains the static RAM (SRAM). SRAMs are typically embedded in SoC and ASIC chips in large numbers; today, large ASIC chips or SoCs can have up to 30 embedded memory arrays (which occupy more than 60% of the chip area). As the most used modules, the yield of eSRAMs is closely monitored because it ultimately affects the overall yield of these chips. For ASIC or SoC, a compiler is commonly employed in the design and organization of the embedded memory module(s); this tool provides flexibility and versatility in design options at both array and chip levels such that a high yield can be retained under different defect distributions. Repair facilities are usually provided to enhance the yield in the presence of defects and faults. The addition of redundancy to replace faulty resources has been proven to be effective in improving the yield of integrated circuits [4]. A yield estimation methodology permits the so-called design for yield (DFY) design approach. This approach mainly impacts the layout design and redundancy assignment of the existing design flow. DFY can help the yield analysis of the manufacturing line and allows to optimize the yield of the layout. For example, given the same schematic a layout with lesser critical area will have a better yield. DFY also optimizes the redundancy assignment during chip level design. For instance, a typical ASIC chip may have more than 100 embedded SRAMs, obviously the highest redundancy for each SRAM would likely provide the highest yield, however, it would be impossible due to the constraint in area overhead. Therefore, in designing ASICs, it is necessary to consider the redundancy and area overhead for a DFY methodology. Finally, a DFY methodology can also estimate the distribution of each fault type of a SRAM based on the manufacturing line. An unusually high percentage of a fault type may allow to detect problems related to manufacturing steps.

The main objective of this paper is to provide a detailed treatment of yield related techniques which contribute to an efficient design of eSRAMs through the utilization of a memory compiler and a detailed assessment of the yield of compiler-based eSRAMs. A systematic method of calculating the yield of compiler-based arrays is proposed. A new tool referred to as compiler-based array yield analysis (CAYA) [2], has been developed based on this method. A compiler-based memory array is usually made of so-called kernels. Kernels are predesigned modules (inclusive of layout) which can be integrated onto a chip, such as a SoC. Manufacturing of compiler-based memories is a complicated process because it requires to consider the several levels (layers) of a chip. The proposed method analyzes the critical area of each kernel of the eSRAMs; from the critical area of the kernels and the chosen compiler option, it is then possible to obtain the total critical area. This, together with the calculated defect density, allows to find the number of different fault types of a configuration of the eSRAM. The yield of the memory is evaluated by considering different design constructs (generally referred to as kernels) that are used in defining the desired architecture through a compiler. Architectural considerations,

Manuscript received September 29, 2004; revised March 29, 2005.

X. Wang is with the IBM Technology and Server Group, Essex Junction, VT 05402 USA.

M. Ottavi and F. Lombardi are with the Department of Electrical and Computer Engineering, Northeastern University, Boston MA 02115 USA (e-mail: lombardi; mottavi@ece.neu.edu).

F. J. Meyer is with the Department of Electrical and Computer Engineering, Wichita State University, Wichita, KS 67260 USA (e-mail: fred.meyer@wichita).

Digital Object Identifier 10.1109/TSM.2005.852108

0894-6507/$20.00 © 2005 IEEE

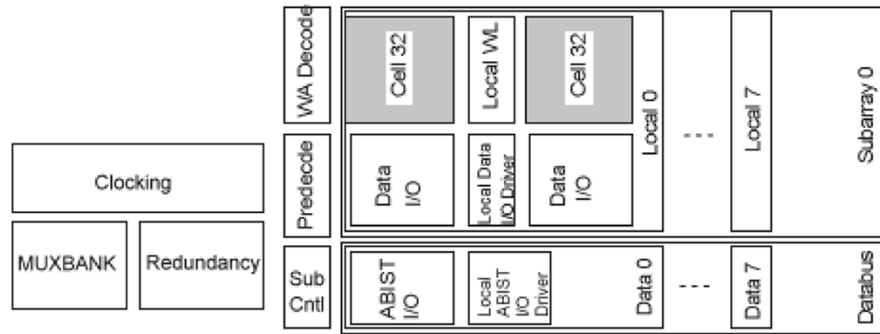


Fig. 1. Array structure (sRamR) made of one subarray.

such as array size and line (word and bit) organization, are analyzed. Compiler-based features for different kernels (such as required for decoding) are also treated in detail. As an eSRAM is generated by using kernels, then different kernels have different impact on the yield. For example, the SRAM with a smaller decoder option will be shown to have a reduced yield due to the additional number of input/output circuitry required in this design.

An extensive evaluation of the provided redundancy (row, column, and combined) is pursued to characterize its impact on the overall memory yield [3]. It will be shown that yield loss is more pronounced for bit lines than word lines (column redundancy requires a more complex implementation than word line redundancy because multiplexers and a sense amplifier along the bit line are needed at the output). Throughout this paper, industrial data is used in the evaluation and an industrial ASIC chip (made of multiple eSRAMs) is also considered. Also, it will be shown that yield loss increases with the size of the eSRAM arrays, i.e., bigger the size of the eSRAM array, a higher improvement in yield is accomplished by using redundancy. For an ASIC chip, this has been accomplished by a greedy assignment of the redundancy to multiple eSRAM arrays, i.e., redundancy is assigned first to the array of largest size.

This paper is organized as follows. Section II introduces the design of eSRAMs by a compiler. In Section III, the tool CAYA is introduced to describe and characterize the yield environment for these chips. Section IV reports defect and critical area analysis which is the basis of the eSRAM yield model proposed in Section V. Section VI describes and evaluates an empirical model for obtaining the yield of a nonredundant array. Section VII reports the results obtained by applying the proposed method to different memory configurations, while in Section VIII the redundancy effects are also analyzed. Finally, a case study is reported in Section IX and conclusions are drawn in Section X.

II. Embedded SRAMs

In this paper, two memory cells are considered; these cells (denoted as sRamN and sRamR) are one-port compiler-based embedded SRAMs. The structures of sRamN and sRamR have many similarities, the main difference is that sRamR has four independent redundant (word) lines, while sRamN has no redundancy. The basic structure of a sRamR subarray is shown in Fig. 1; this subarray has its own redundant word lines. sRamN has a similar structure, but no redundancy.

TABLE I
Compiler Option of sRamN and sRamR

sRamN	standard one-port SRAM with no redundancy.
sRamR	standard one-port SRAM with redundancy.
w	5 digit decimal number, specifying the number of words.
b	3 digit decimal number, specifying the data width of a word in bits.
d	2 digit decimal number, specifying the decoding arrangement.
s	1 digit decimal number, specifying the number of subarrays.

A compiler-based array consists of several functional modules. The functional modules of the sRamN and sRamR arrays are as follows: memory cells, DIO (data IO), AIO (ABIST IO), local word line driver, global word line driver, timing, and decoding control circuits, BIST. ABIST is the array BIST. Unlikely a stand-alone SRAM [5], compiler-based memories have a large number of configurations (for example, sRamN has as many as 14 000 configurations to account for different words, word width and decode options). The memory complier generates the layout of each configuration of the array by utilizing to a so called "compiler option" which allows to specify a custom memory configuration. The compiler option for the two cells considered in this paper is denoted as sRamN w, b, d, s (sRamR w, b, d, s). Its detailed description is given in Table I. Note that d describes how to group the bit lines into a word. The product of the bit line decoder option d and the word width is equal to the number of bit lines. For example, if the word width b is 4, the bit line decoder option d is 2, then number of bit lines is 8, and the 8 bit lines are grouped into two words such that the 0th, 2th, 4th, 6th bit lines are in the first word, while the 1th, 3th, 5th, 7th bit lines are in the second word.

Each functional module is built using at least one type of placeable kernels. A placeable kernel defines a predesigned layout of a circuit for the compiler-based arrays. The memory compiler places the kernels to form the layout of the arrays for a specific configuration. For example, CELL_16 is the placeable kernel of the cell (either sRamN or sRamR) and has 16 bits cells. Different configurations of compiler-based arrays have different types and numbers of placeable kernels.

TABLE II
KERNELS FOR sRamN AND sRAMR ARRAYS

Functional Module	Placeable kernel	Number of kernels from Compiler option
CELL	CELL_16	#cells=wb #CELL_16=wb/16
DIO	D_IO	#DIO=bd/4 #D_IO_L=bd/4/2 #D_IO_R=bd/4/2
LWLD	LWLDVR	#LWLDVR=(bd/64)(w/d/4)
GWLD	GWLDVR	#GWLDVR=(w/d/4)

Table II shows the number of placeable kernels as function of the compiler option values.

III. PROPOSED YIELD ANALYSIS: CAYA

The proposed methodology is based on the following design process as applicable to compiler-based eSRAMs in current industry practice.

1) Specify the desired memory array using the placeable kernels of the modules in the compiler.
2) From the compiler option, design and assemble the configuration of the array. Analyze the critical areas of each placeable kernel and the whole array from the layout of the eSRAM.
3) From the layout, determine the numbers and types of possible defects and faults.
4) If the array has no redundancy, use an appropriate model (such as the negative binomial yield model).
5) If redundancy is provided, calculate the yield according to the proposed yield model (which accounts for redundancy).

A tool named CAYA (compiler-based array yield analysis) has been developed to facilitate the yield analysis procedure. CAYA performs two functionalities: the first functionality consists of calculating the number of each type of faults in an array for a specified configuration and the second functionality calculates the yield of this array. To perform the first functionality, as inputs CAYA is supplied with the critical area of each kernel of the compiler-based array, the defect density and the compiler option for the specified configuration. CAYA generates the number of faults of each type. The critical area of each kernel is already precalculated by a critical area extraction tool (i.e., either INCA or CAA which are critical area analysis tools, the first one is based on shape expansion and the second on Monte Carlo simulation [20]). The defect density is obtained from the manufacturing line, while the compiler option defines the configuration of the array. To perform the second functionality, CAYA utilizes the number of faults of different type and the redundancy as inputs. The output is the yield of the array for the configuration specified by the compiler option, which also specifies whether redundancy is supplied. The yield of a nonredundant array can be computed by CAYA also by means of an empirical based model (as proposed in this paper).

Summarizing, the procedures performed by CAYA to calculate the yield of compiler-based arrays are as follows.

1) *Faults Type Extraction*: Design data is obtained from the layout of the kernels. INCA (or CAA) is executed to find the critical areas. Using the defect density (obtained from the manufacturing line and compiler option for the eSRAM configuration), the number of faults in the eSRAM is established.
2) *Yield Calculation*: Starting from the number of defects and locations, if the eSRAM is not provided with redundancy, then the yield is computed by using an existing yield technique (such as negative binomial model); if the eSRAM has redundancy, the number of faults left unrepaired after the repair process is calculated and then the yield is evaluated.
3) *Empirical Yield Calculation*: if the array is not provided with redundancy CAYA can compute the yield by applying an alternative empirical method based on a linear curve fitting approach of the manufacturing data.

IV. DEFECTS AND CRITICAL AREA IN eSRAMS

The yield loss is caused by chip defects during the manufacturing process. Not all defects lead to faults. Whether a defect leads to a fault depends on its size, location and the layout of the circuit. The critical area is the part of the layout that when the center of the defect fall into this area, then it will lead to a fault [5]. The extraction of the critical area from the IC design database has been discussed in many papers [6], [7]. There are two main types of algorithms to calculate the yield: Monte Carlo dot throwing and Shape expansion. Prior to describing the steps involved in the proposed procedure the models and analysis of the faults and defects in a eSRAM are presented in detail. In the past, Stapper has provided an analysis of the faults in a stand-alone memory chip, as well as a repair process using different spares [5]. The proposed method is similar to Stapper's method, however, it considers fault types and memory configurations. Defects in kernels will cause functional and structural faults; if a defect occurs, then the kernel function will be faulty and the kernel will work incorrectly. Defects are mainly caused by either the absence (missing), or presence (additional or extra) of material during manufacturing; it has been shown that for memory structural level faults can occur according to the usual characterization in seven layers (RX, PC, CA, M1, M2, M3, and V1). Structural faults will lead to functional faults. Industrial experience with compiler-based one-port eSRAMs has shown that nine types of functional faults can occur; the functional faults possible in eSRAMs are single cell (SC), vertical cell pair (VP), horizontal cell pair (HP), single word line (SW), double word line (DW), word and bit line cross (WB), single data column (SD), double data column (DD), chip kill (CK).

As described previously, CAYAs execution is based on the kernels in the compiler option. These kernels play an important role in defining the critical areas and the number of faults present in the eSRAM. Consider first the critical area. Let the critical area density (CA_D) be defined as

$$CA_D = \frac{C_A}{A_T} \tag{1}$$

TABLE III
$CA_D s$ OF KERNELS OF sRamN AND sRamR VERSUS STRUCTURAL FAULTS

Structural fault	Kernel CELL CA_D	Kernel DIO CA_D	Kernel GWLDVR CA_D	Kernel LWLDVR CA_D	Kernel AIO CA_D	Kernel MASTER CA_D
RX_MISS	0.0228	0.0033	0.0067	0.0096	0.0061	0.0081
RX_EXTRA	0.0113	0.0068	0.0018	0.0081	0.0019	0.0027
PC_MISS	0.0563	0.0169	0.0203	0.2821	0.0204	0.0238
PC_EXTRA	0.0564	0.0241	0.0223	0.0352	0.0236	0.0263
CA_MISS	0.0126	0.0023	0.0015	0.0048	0.0018	0.0016
CA_EXTRA	0.0113	0.0021	0.0014	0.0037	0.0016	0.0015
M1_MISS	0.0265	0.0131	0.0158	0.0213	0.0144	0.0142
M1_EXTRA	0.0445	0.0139	0.0147	0.0212	0.0142	0.0125
V1_MISS	0.0026	0.0010	0.0007	0.0023	0.0009	0.0011
M2_MISS	0.0201	0.0162	0.0095	0.0052	0.0150	0.0112
M2_EXTRA	0.0272	0.0214	0.0142	0.0040	0.0174	0.0152
V2_MISS	0.0016	0.0005	0.0004	0.0010	0.0006	0.0006
M3_MISS	0.0100	0.0046	0.0040	0.0109	0.0047	0.0064
M3_EXTRA	0.0137	0.0070	0.0074	0.0138	0.0061	0.0051

where C_A is the critical area and A_T is the total area. Let the CA_D of kernel k be denoted as $CA_D(k)$. The $CA_D s$ of the cell kernel (denoted as CELL) as well as other kernels (as calculated by CAA) with respect to the possible structural faults are given in Table III (DIO refers to the Data IO; GWLDVR refers to the global word line driver; LWLDVR refers to the local word line driver; AIO refers to the ABIST IO; MASTER refers to the control circuits).

Let the normalized defect density (N_D) of a level be the ratio of the defect density in that level (say L) and the defect density of M1_EXTRA (as obtained from the manufacturing line), i.e.,

$$N_D(L) = \frac{D_L}{D_{M1_\text{extra}}} \tag{2}$$

where D_L is the defect density of level L. Let T_D be the total defect density per unit area. So

$$T_D = \sum N_D(i) CA_D(i) \tag{3}$$

where i denotes the structural level fault (defect). The T_D for each kernel is shown in Table IV as obtained by CAA and INCA.

The results for T_D in Table IV show that there is good agreement between INCA and CAA. DIO, AIO, and GWLDV have similar values of T_D, while CELL accounts for the largest value. The defect density in CELL is three times higher than for the other kernels due to the spacing in the layout of the SRAM cell.

V. eSRAM YIELD MODEL

The yield model of redundant and non redundant memories has been described in many previous works [8], [9], and [13] to [19]. The proposed approach first finds the number of faults left after repair, then it establishes the chip yield based on the characteristics of each eSRAM. Hence, new modeling techniques are introduced to take into account the compiler-based nature of the eSRAM design. In the proposed model, it is assumed that fault types show the same cluster characteristics; Stapper's model can be extended to treat each fault type with different cluster characteristics. Most fault types are caused by defects in multiple layers; hence by assuming that fault types show the same cluster characteristics and adjusting the cluster parameter, it is possible to have a realistic and accurate yield model. It has been shown that on the assumption that all types of faults have the same cluster characteristic, a very good yield model is still possible [16]. This is in accordance with current practices in a manufacturing environment because it is not practical to ascertain the cluster characteristics of each fault type prior to full testing and assembly. The average number of faults can be calculated from the critical areas and the defect density. The sRamN (or sRamR) has 14 structural level faults and nine functional faults. The number of faults present in a eSRAM can be calculated as follows. Let Λ be a 9×1 matrix of the average number for each type of functional faults; let A be a 9×14 critical area matrix, an entry $A_{i,j}$ denotes the critical area of functional fault type j in critical area i; let D be a 14×1 defect density matrix for the 14 structural level faults. Using the defect density matrix obtained from the manufacturing line, the number of faults (of different types) is given as

TABLE IV
T_D OF DIFFERENT KERNELS

	CELL	DIO	AIO	MASTER	LWLDVR	GWLDVR
T_D by CAA	2.72	1.19	1.05	1	1.38	0.95
T_D by INCA	2.67	1.06	0.95	1	1.35	0.96

$$\Lambda^T = (\lambda_{SC}, \lambda_{VP}, \lambda_{HP}, \lambda_{SW}, \lambda_{DW}, \lambda_{WB}, \lambda_{SD}, \lambda_{\text{DD}}, \lambda_{CK}) \tag{4}$$

where SC,VP,HP,SW,DW,WB,SD,DD,CK are the functional defect types as described above, and

$$\Lambda = AD. \tag{5}$$

Let λ be the sum of the average numbers of faults of each type in the eSRAM. So

$$\lambda = \lambda_{SC} + \lambda_{VP} + \lambda_{HP} + \lambda_{SW} + \lambda_{DW} + \lambda_{WB} + \lambda_{SD} + \lambda_{\text{DD}} + \lambda_{CK}. \tag{6}$$

For arrays with no redundancy, the yield is calculated as the probability of having a perfect chip and the negative binomial formula can be used

$$Y = Y_p = P(0) = \left(1 + \frac{\lambda}{\alpha}\right)^{-\alpha} \tag{7}$$

where Y_p is the so called perfect yield, λ is given from (6) and α is a parameter for taking into account the effect of defect clustering. The repair of a redundant memory array is obtained by replacing the defective rows (columns) with spare rows (columns). When both spare rows and columns are provided, the problem of optimal spare allocation is NP complete, many algorithms have been proposed for memory repair [10]–[12].

Stapper, [4] enumerates the probability of successfully repairing all combinations of fault types using the provided redundancy. By this model, the yield Y_r is computed as the probability of having good or repairable chips, i.e.,

$$Y_r = P_{CK}(0) \cdot \sum_{C_F} P_{SC}(i) \cdot P_{VP}(j) \cdot P_{HP}(k) \cdot \cdot P_{SW}(l) \cdot P_{DW}(m) \cdot P_{WB}(n) \cdot P_{SD}(o) \cdot P_{\mathrm{DD}}(p) \quad (8)$$

where C_F denotes the possible combinations of faults. The proposed model sums the probability of all combinations of faults of different types that can be repaired by the provided redundancy. In (8) the $P_i(k)$ are given by

$$P_i(k) = \frac{e^{-\lambda_i} \lambda_i^{-k}}{k!}. \quad (9)$$

Hence, a Poisson distribution has been considered for computing the probability of having k faults of type i. From Y_r the number of faults left unrepaired after the repair process is computed by inversion, i.e., $\lambda_r = -\ln(Y_r)$. Finally, the yield is computed again with the negative binomial formula

$$Y_r = \left(1 + \frac{\lambda_r}{\alpha}\right)^{-\alpha}. \quad (10)$$

VI. Empirical Model

This section describes an empirical model proposed in the CAYA framework. This model can be used to calculate the yield of a non redundant array at a low computational cost. Instead of using critical area and defect density analysis, the empirical model calculates the yield of the configuration of the eSRAM as generated by the compiler option. This is substantially different from traditional methods; for compiler-based memories, a more practical approach is required due to the large number of configurations possible, as well as the inclusion and interface of CAYA with other design tools. These are important requirements in an industrial environment because IC technology advances (such as low scaling) are reflected in manufacturing through historical monitoring of the line. In this paper, an empirical model based on curve fitting by linear regression is utilized; this empirical model still provides accurate results for the eSRAM configurations. The notation used hereafter must be introduced.

For a given instantiated memory array:

B_L the number of bit lines obtained by the compiler option;

W_L the number of word lines obtained by the compiler option;

F the total number of faults in the array.

The empirical model of the yield of each configuration of the eSRAM is found as follows. From the yield of the configurations of the array, it is well known that for a fixed W_L, F increases linearly with B_L. Let $\hat{B}$ denote the slope of the linear relationship i.e.,

$$F = \hat{B} B_L. \quad (11)$$

$\hat{B}$ increases approximately in linear fashion with W_L, so

$$\hat{B} = B W_L + A. \quad (12)$$

By combining the previous equations

$$F = (B W_L + A) B_L. \quad (13)$$

A and B are empirical parameters obtained from curve fitting of manufacturing data by linear regression for reducing the error. Let the average error of the above empirical model be denoted by e. Define F_i as the actual number of faults in the compiler-based array i; let $\hat{F}_i$ denote the number of faults in the compiler-based array i (as obtained from the empirical model); define N as the total number of configurations (as an example for sRamN, there are 7000 configurations). Then

$$e = \frac{\sum_{i=1}^{N} \sqrt{(F_i - \hat{F}_i)^2}}{N}. \quad (14)$$

The following equations characterize the yield and its calculation from the compiler option. As shown in Table I; $W_L = w/d/s$ and $B_L = bd$. F can be calculated using the linear model of (13). Using this value for the average number of faults, the negative binomial yield formula is therefore given by

$$Y = \left(1 + \frac{F}{\alpha}\right)^{-\alpha} \quad (15)$$

where α is the parameter parameter of the empirical model of Y and is fixed to 2.0.

To validate the empirical model experimental results are presented, they were obtained using CAYA on manufacturing data in an industrial setting. To preserve confidentiality of industrial information only relative yield figures are used, i.e., memory yields as presented in the figures of this paper are normalized with respect to the yield of sRamN32768,032,32,2 (of 1 M bit size) which is fixed to 100%. For a subarray made of sRamN memory cells (no redundancy), assume $Y(\mathrm{sRamN16384, 032, 32, 1}) = 98\%$; using the proposed empirical model, the following values were obtained for the parameters of the linear regression: $A = 0.000\,003\,493\,641$ and $B = 0.000\,000\,023\,269$. The yield values obtained by using the proposed empirical model as well as the fully computed data are shown in Fig. 2. The average error e of the proposed empirical model is 0.001. This is well within an acceptable value for practical applications to CAD software.

VII. Array Parametric Yield Analysis

In this section, different parameters for the configurations generated by the memory compiler are analyzed in detail. These are array-level parameters which are directly related to the ability to improve the yield, while changing the design of the eSRAM. The yield of a 1-M bit sRamN32768,032,32,2 is used as a normalized value of 100.

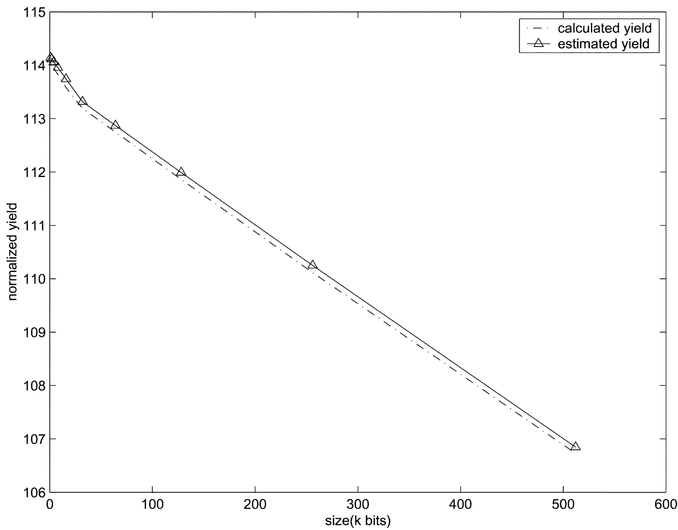


Fig. 2. Yield of sRamN-based memory subarray (fully computed data and proposed empirical model).

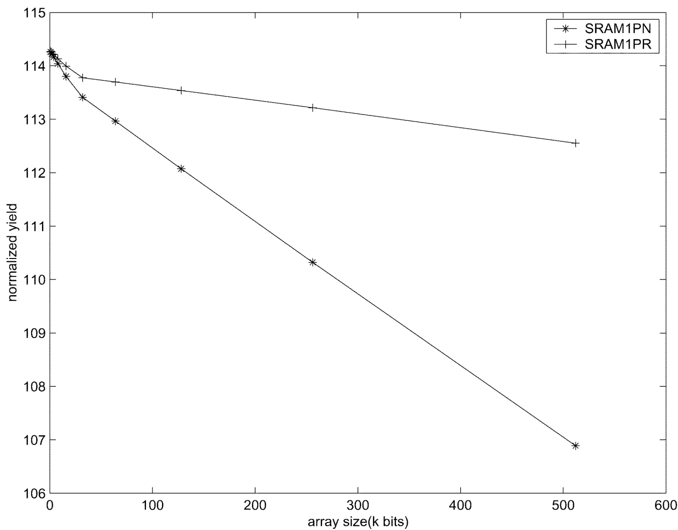


Fig. 3. Normalized yield of one subarray of sRamN and sRamR.

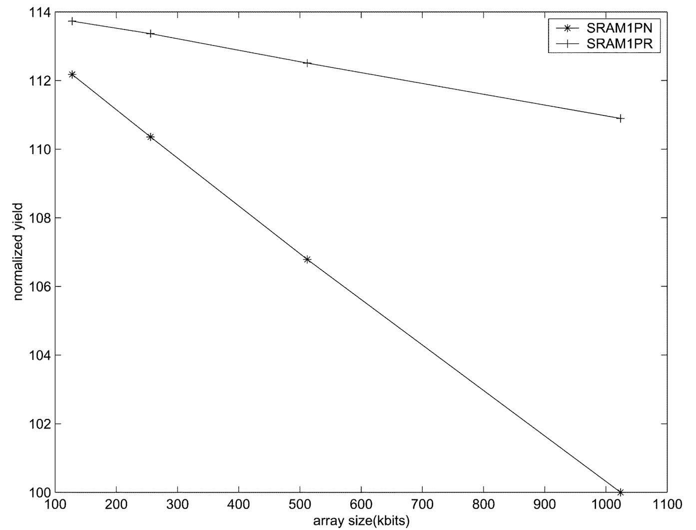


Fig. 4. Normalized yield of sRamN and sRamR arrays made of two subarrays.

TABLE V
YIELD IMPACT OF 512 kBIT sRam DECODER OPTION

Configuration	#words	word width	decoder (d)	#word lines	# bit lines	Norm. yield
Non redundant	16384	32	32	512	1024	106.89
Non redundant	8192	64	16	512	1024	106.93
Non redundant	4096	128	8	512	1024	106.96
4 spare rows	16384	32	32	512	1024	112.58
4 spare rows	8192	64	16	512	1024	112.60
4 spare rows	4096	128	8	512	1024	112.62

A. Array Size

There can be as many as hundreds of eSRAM arrays in an ASIC chip, so redundancy allocation must be carefully allocated to reduce penalties involved in additional area and increased test time. Designers need to determine accurately the levels of redundancy in each eSRAM array, so that the overall chip can be manufactured at no significant increase in cost and die size. The addition of redundancy significantly influences memory design. For example, sRamR has four redundant word lines per subarray, while sRam2R has two redundant word lines per subarray. Normalized array yields versus size of sRamN and sRamR arrays are plotted in Figs. 3 and 4. Not surprisingly, by analyzing these figures, yield loss increases with array size and number of subarrays. The yield difference between arrays (as direct benefit of the provided redundancy) increases also with array size. It has been found that a simple greedy algorithm can be utilized to assign the redundancy to the eSRAM arrays of an ASIC chip, i.e., within the limitation of chip size, assign first the redundancy to the eSRAM of largest size and so on. The detailed description of this algorithm is beyond the scope of this paper.

B. Decoder Option

In the compiler, the word width defines the number of output bits of a memory. The decoder option defines the number of bit lines to be decoded into one bit of the output word. For example in sRamN16384,032,16,1, the word width is 32 and the decoder option is 16, so 16 bit lines will be decoded into one bit (the total number of bit lines is 16×32).

The decoder option in the compiler also affects the yield; Table V shows the impact of this option on sRamN and sRamR arrays of fixed size (i.e., 512 kbits). An increase in the decoder option (i.e., d) results in a decrease of the yield, i.e., if two eSRAM arrays have the same number of word and bit lines, then the SRAM array with a larger decoder has also a larger number of AIO's. So if two SRAM arrays have the same critical areas for the common kernels, then the eSRAM array with a larger decoder will account for more critical areas due to the additional AIO's and therefore, a smaller yield will be accomplished.

C. Word and Bit Lines

The numbers of word and bit lines have different impact on the yield of eSRAMs. To quantify the yield loss due to bit lines, the number of word lines was fixed to 512 and the decoder option was fixed to 32; the relationship for the yield by increasing

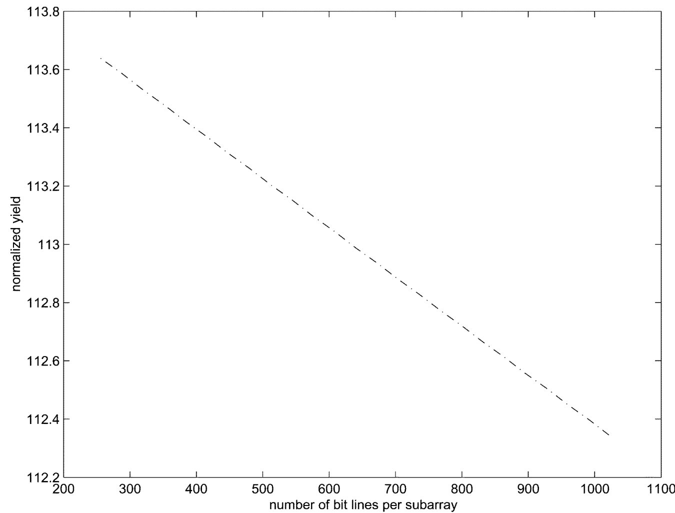


Fig. 5. Normalized yield versus bit lines of sRamN.

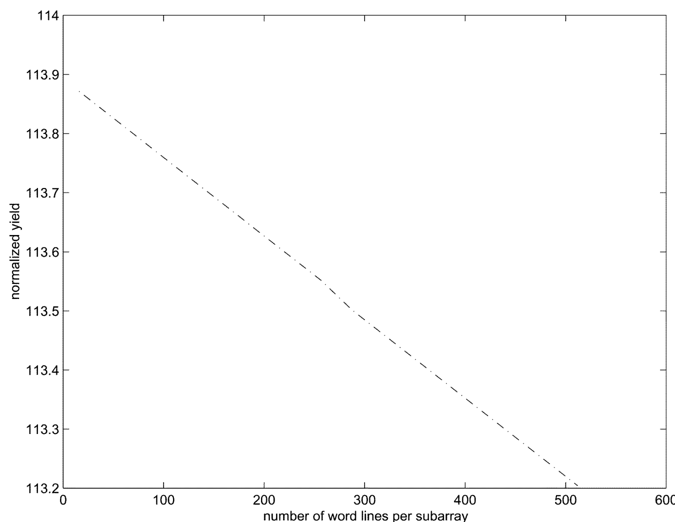


Fig. 6. Normalized yield versus word lines of sRamN.

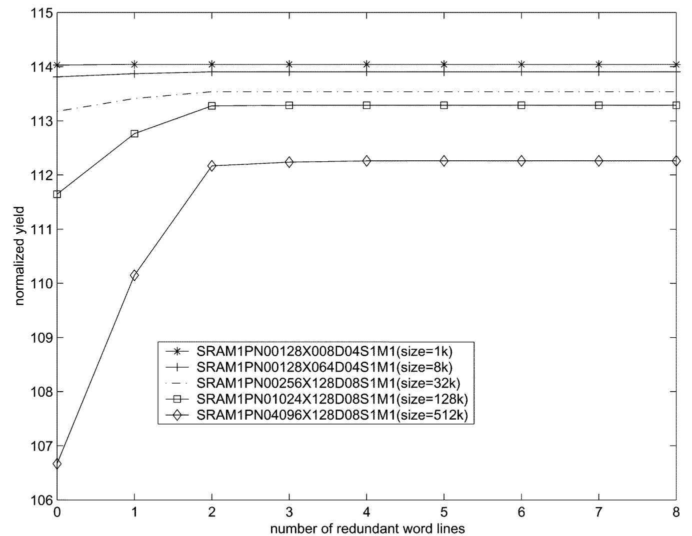


Fig. 7. Yield of eSRAMs versus number of redundant word lines.

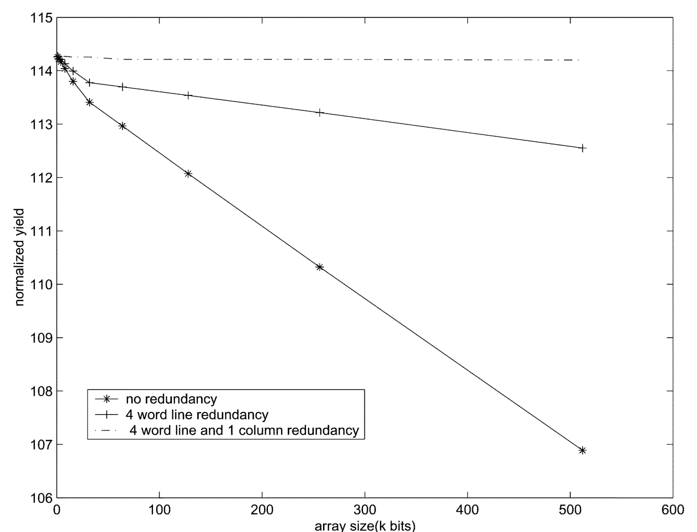


Fig. 8. Normalized yield of a sRamN subarray with different redundancy schemes.

the number of bit lines is shown in Fig. 5. The slope of this line (S_{bl}) is given by $-0.000\,016\,9$ (Fig. 5). For eSRAMs with a fixed number of bit lines (i.e., 512) and decoder (fixed to 32), Fig. 6 shows the plot of the yield as function of increasing the number of word lines (the slope of the line S_{wl} is $-0.000\,013\,3$).

In Figs. 5 and 6, the value of S_{bl} is smaller than S_{wl}, i.e., the yield loss due to bit lines is larger than the yield loss due to word lines. This occurs because the critical areas along bit lines are larger than the critical areas along word lines, i.e., in the layout, word lines provide a better source of redundancy.

VIII. Redundancy Analysis

This section provides new results for quantify the effect of redundancy on the yield of eSRAMs.

A. Redundant Word Lines

As described previously, sRamR has word lines as redundancy, so initially this type of redundant resource will be analyzed. Fig. 7 shows the impact of different numbers of redundant word lines on the yield of eSRAMs. Redundant word lines have more impact on the yield of large size eSRAMs; two redundant word lines account for the most significant increase in yield due to repair. Redundancy of three or more lines seems to be an excessive amount as saturation in yield occurs for all eSRAMs.

B. Column Redundancy

In this section, column redundancy (as a different redundancy resource) is analyzed. As discussed in a previous section, many faults (such as the faults in SD, DD, and WB) can not be repaired by word line redundancy. The addition of column redundancy is more difficult than word line redundancy because a column includes bit lines, as well as DIO. The normalized yield of three arrangements is shown in Fig. 8 versus array size.

As calculated by CAYA, different types of faults require different redundancy: for a 1024 kbits sRamN array, 76% of the structural faults are SC, HP, SW, and DW types and can be fixed by word line redundancy; an additional 21% of the structural faults are SD and DD types and can be fixed by column redundancy. 2% of the structural faults occur in the support circuits

TABLE VI
AREA OVERHEAD OF REDUNDANCY SCHEMES FOR A sRamN ARRAY

RWL	RCOL	Area overhead (percentage)
4	0	1
3	1	1.3
2	2	1.5
1	3	1.8
0	4	2

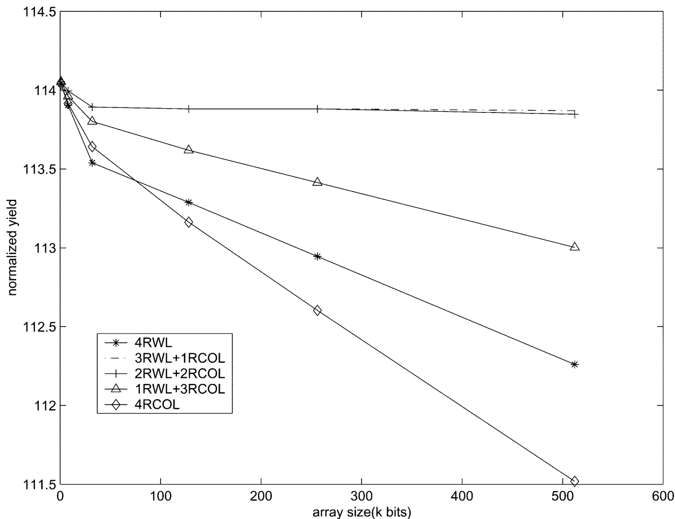


Fig. 9. Redundancy schemes for a sRamN array.

and they are not repairable. Assume equal density and distribution in the faults and kernels; under this assumption, a 4 to 1 assignment in the number of row/column (word and bit lines) lines provides the best results.

C. Combined (Row and Column) Redundancy

In this section, the impact of a combined row and column redundancy is evaluated with respect to the yield by considering the fault density and its relationship to area and kernels. The array yield is computed by employing the redundant schemes of Table VI. In the considered industrial systems there is no column redundancy, so only the area overhead of the redundant column and word lines is accounted in the analysis. The area overhead has been calculated for sRamN04096,128,08,1, i.e., a memory array (single subarray) of 512 kbits made of sRamN cells. Note RWL (RCOL) denotes the number of redundant word (column) lines.

The characteristics of the eSRAM arrays whose yield is evaluated in Fig. 9 are given in Table VII.

Fig. 9 shows that three redundant word lines and one redundant column (3RWL + 1RCOL), or 2 redundant word lines and two redundant columns (2RWL + 2RCOL) are the arrangements that result in the highest yield. However, column redundancy occupies more area than word line redundancy and is more difficult to implement on a chip; hence, the best arrangement once area complexity is also taken into account, is (3RWL + 1RCOL). This is a function of array size too. From Fig. 9, for eSRAMs of small size, a redundancy scheme made of 4RCOL

TABLE VII
CHARACTERISTICS OF THE eSRAM ARRAYS

Array size (k bits)	Option configuration	#word lines	#bit lines
1	sRamN00128,008,04,1	32	32
8	sRamN00128,064,04,1	32	256
32	sRamN00256,128,08,1	32	1024
128	sRamN01024,128,08,1	128	1024
256	sRamN02048,128,08,1	256	1024
512	sRamN04096,128,08,1	512	1024

TABLE VIII
NON REDUNDANT AND ROW REDUNDANT CONFIGURATIONS IN THE eSRAM ARRAYS OF THE ASIC CHIP

Number of Arrays	Non Redundant Configuration	Redundant Configuration	Redundant Word Lines
20	sRamN00128,128,04,1	sRamR00128,128,04,1	4
8	sRamN00192,107,04,1	sRamR00192,107,04,1	4
2	sRamN00256,015,04,1	sRamR00256,015,04,1	4
4	sRamN00512,009,04,1	sRamR00512,009,04,1	4
2	sRamN01024,009,04,1	sRamR01024,009,04,1	4
1	sRamN01664,128,04,1	sRamR01664,128,04,1	4
2	sRamN02048,064,04,1	sRamR02048,064,04,1	4
4	sRamN02048,071,04,1	sRamR02048,071,04,1	4
2	sRamN02048,072,04,1	sRamR02048,072,04,1	4
2	sRamN02048,128,04,1	sRamR02048,128,04,1	4
1	sRamN04096,008,08,1	sRamR04096,008,08,1	4
6	sRamN04096,064,16,1	sRamR04096,064,16,1	4
2	sRamN04096,128,08,1	sRamR04096,128,08,1	4
2	sRam2D0256,128,04,1	sRam2DR0256,128,04,1	2
1	sRam2D0384,064,04,1	sRam2DR0384,064,04,1	2
10	sRam2D0640,128,04,1	sRam2DR0640,128,04,1	2
1	sRam2D3840,008,16,1	sRam2DR3840,008,16,1	2

results in a higher yield than the 4RWL redundancy scheme; however for eSRAMs of large size, 4RCOL results in less yield than 4RWL. The reason for this result is that by increasing the size of the eSRAM, the number of word lines (as shown in Table VII) and the percentage of faults occurring on the word lines are also increased. Hence, the provision of word line redundancy greatly affects the yield in a more significant manner than column redundancy.

IX. CASE STUDY: AN INDUSTRIAL ASIC CHIP

In today's electronic systems, an ASIC may integrate more than 100 eSRAM arrays; if there is no redundancy, then a single bit fault in an eSRAM array will cause the whole chip to fail. The addition of redundancy will increase the area of the chip; CAYA can be used to guide a designer in allocating redundancy at chip level. An industrial ASIC chip is considered; this chip has 68 eSRAMs of different configurations as listed in the first column in Table VIII. The second column of the table reports the non redundant configurations, for example, sRam2D denotes a two-port eSRAM with no redundancy, while the third column

reports the corresponding redundant arrangement, for example, sRam2DR is a two-port eSRAM with two redundant word lines.

Similarly to previous Tables, the yield of sRamN32768,032,32,2 (size of 1 Mbits) is used at a normalized value of 100%. The ASIC chip prior to introducing redundancy had a normalized yield of 47% with an array layout area consisting of 16 482 093 cells (where the cell is the basic unit for this technology). If row redundancy is used in all arrays (with the number of redundant lines shown in the fourth column of the Table), the normalized yield is increased to 80.7% while the total area is given by 17 651 363 cells. This corresponds to an increase of 72% in normalized yield at a 7.1% increase in area. Currently, the considered ASIC technology does not permit in the compiler to have column redundancy, therefore an estimate was derived. The addition of a redundant column to the redundancy of Table VIII results in a 219% normalized yield and a 10% increase in additional area (both values compared with the case of no redundancy). This shows that the provision of redundancy results in significant benefits in yield of ASIC chips at a modest area overhead.

X. Discussion and Conclusion

This paper has introduced a new tool (denoted as CAYA) for the yield of compiler-based eSRAMs. CAYA is based on a novel characterization of the yield within a compiler design environment. The memory compiler utilizes predefined modules (referred to as kernels) to design the organization of the eSRAM through a so-called memory option. A detailed defect and fault analysis (inclusive of industrial data) has been presented for these chips by taking into account the design constructs (referred to as kernels) and the physical properties of the layout. In this paper, two compiler-based cells (with and without redundancy) are utilized for designing arrays and calculating the yield. These are one-port compiler-based embedded SRAM. CAYA is based on a characterization of the design process which accounts for fault types and the relation between functional and structural faults. This paper has provided a systematic method which is based on critical area and defect analysis for calculating the yield of the compiler-based array. The proposed method analyzes the critical area of the layout of each kernel of the eSRAMs; from the critical area of the kernels and the compiler option, it is then possible to obtain the total critical area. Together with the calculated defect density, this allows to find the number of different fault types of a configuration of the eSRAM. A novel empirical model has been proposed to facilitate the yield calculation. Industrial data has been provided for the analysis of various configurations with different structures and redundancy.

The following features of redundancy and its implications on eSRAMs can be ascertained from the results of the yield analysis sections.

1) Yield loss increases with the size of the eSRAM arrays, i.e., bigger the size of the eSRAM array, the most improvement in yield is accomplished by using redundancy. For an ASIC chip, this has been accomplished by a greedy assignment of the redundancy to multiple eSRAM arrays, i.e., redundancy is assigned first to the array of largest size.
2) The choice of a compiler option has also a significant impact on yield. The decoder option has been analyzed in detail. For two SRAM arrays with the same number of bit and word lines, it has been shown that the SRAM with a smaller decoder option will have less yield due to the additional number of AIOs required.
3) In the proposed design, the compiler-based array is generated by the kernels. The yield model is a function of the critical areas (as per definition of the kernels). Therefore, different kernels have different impact on the yield. For example, consider again the decoder option; it has been shown in this paper that for small eSRAM arrays, the decoder option is not significant. The decoder option becomes very important for large arrays because memory designs are significantly affected by this feature once the number of (bit and word) lines is increased.
4) It has been shown that yield loss is more pronounced for bit lines than word lines. This occurs because along bit lines, there are DIO and AIO of larger size and more critical area than for the GWLDVR and the LWLDVR.
5) Column redundancy requires a more complex implementation than word line redundancy because multiplexers and a sense amplifier along each bit line are needed at the output. However, column redundancy is necessary for some configurations of eSRAM arrays. For a 1-Mbits memory array sRamN32768,032,32,2 (with 1024 word lines and 1024 bit lines), faults along bit lines account for more than 25% of the density; these faults can not be repaired by word line redundancy. As an extreme case, consider the memory array sRamN00064,256,04,1 (with 16 word lines and 1024 bit lines): faults along bit lines account for 80% of the fault density, hence column redundancy is a necessary and also in this case, CAYA provides excellent facilities to select an appropriate assignment of redundant resources.

References

[1] I. Koren and Z. Koren, "Defect tolerant VLSI circuits: Techniques and yield analysis," *Proc. IEEE*, vol. 86, pp. 1817–1836, Sep. 1998.

[2] X. Wang, M. Ottavi, and F. Lombardi, "Yield analysis of compiler-based arrays of embedded SRAMs," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2003, pp. 3–10.

[3] X. Wang, M. Ottavi, F. J. Meyer, and F. Lombardi, "On the yield of compiler-based eSRAMs," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2004, pp. 11–19.

[4] 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. 398–409, 1980.

[5] C. H. Stapper, "Modeling of integrated circuit defect sensitivities," *IBM J. Res. Develop.*, vol. 27, no. 6, pp. 549–557, Nov. 1983.

[6] P. K. Nag and W. Maly, "Hierarchical extraction of critical area for shorts in very large ICs," in *Proc. IEEE Int. Workshop Defect Fault Tolerance VLSI Syst.*, 1995, pp. 10–18.

[7] I. Bubel, W. Maly, T. Waas, P. K. Nag, H. Hartmann, D. Schmitt-Landsiedel, and S. Griep, "AFFCCA: A tool for critical area analysis with circular defects and lithography deformed layout," in *Proc. IEEE Int. Workshop Defect Fault Tolerance VLSI Syst.*, 1995, pp. 19–27.

[8] R. M. Warner, "Applying a composite model to the IC yield problem," *IEEE J. Solid State Circuits*, vol. SC-9, no. 3, pp. 86–95, Jun. 1974.

[9] C. H. Stapper, "On murphy's yield integral," *IEEE Trans. Semicond. Manuf.*, vol. 4, no. 4, pp. 294–297, Nov. 1991.

[10] R. C. Evans, “Testing repairable RAM’s and mostly good memories,” in *Proc. Intl. Test Conf.*, Oct. 1981, pp. 49–55.

[11] S.-Y. Kuo and W. K. Fuchs, “Efficient spare allocation in reconfigurable arrays,” *IEEE Design Test*, vol. 4, no. 1, pp. 24–31, 1987.

[12] K. W. Huang and F. Lombardi, “Approaches for the repair of VLSI/WSI RRAM’s by row/column deletion,” in *Proc. IEEE FTCS18*, 1988, pp. 342–347.

[13] C. H. Stapper, “On yield, fault distributions and clustering of particles,” *IBM J. Res. Develop.*, vol. 30, no. 3, pp. 326–338, May 1986.

[14] ——, “Large-area fault clusters and fault tolerance in VLSI ciruits,” *IBM J. Res. Develop.*, vol. 33, no. 2, pp. 162–173, Mar. 1989.

[15] ——, “Small-area fault clusters and fault tolerance in VLSI ciruits,” *IBM J. Res. Develop.*, vol. 33, no. 2, pp. 174–177, Mar. 1989.

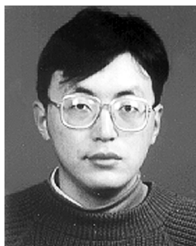
[16] ——, “Improved yield model for fault-Tolerant memory chips,” *IEEE Trans. Comput.*, vol. 42, no. 7, pp. 872–881, Jul. 1993.

[17] J. Khare *et al.*, “Accurate estimation of defect-related yield loss in reconfigurable VLSI circuits,” *IEEE J. Solid-State Circuits*, vol. 28, no. 2, pp. 146–156, Feb. 1993.

[18] T. Yamagata, H. Sato, K. Fujita, Y. Nishmura, and K. Anami, “A distributed globally replaceable redundancy scheme for sub-half-micron ULSI memories and beyond,” *IEEE J. Solid-State Circuits*, vol. 31, no. 2, pp. 195–201, Feb. 1996.

[19] J. E. Rondey, Y. Tellier, and S. Borri, “A silicon-based yield gain evaluation methodology for embedded-SRAM’s with different redundancy scenarios,” in *Proc. 2002 IEEE Int. Workshop Memory Technol. Design Testing*, Nov. 2002, pp. 57–61.

[20] G. A. Allen, “A comparison of efficient dot throwing and shape shifting extra material critical area estimation,” in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 1998, pp. 4452–4452.



Xiaopeng Wang was born in Xuzhou, Jiangsu, China. He received B.S. degree in computer engineering from China University of Mining and Technology, Jiangsu, China, 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 held an internship job on the yield modeling of embedded memories in 2001 and 2002 in 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 interest includes VLSI design, VLSI testing, algoritm analysis, and computer architecture.

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 microelectronics and telecommunications from the University of Rome “Tor Vergata,” Rome, Italy, in 2004.

In 2000, he was with ULISSE Consortium, Rome, as designer of digital systems for space applications. In 2003, he was a Visiting Research Assistant in the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, where he is currently a Postdoctoral Research Associate. His research interests include yield and reliability modeling, fault-tolerant architectures, and online testing and design of nano scale circuits and systems.

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

He is currently Assistant Professor of Electrical Computer Engineering at Wichita State University, Wichita, KS. His research interests include integrated circuit yield modeling, fault-tolerant architectures, and digital design and test.

Dr. Meyer is a Member of the IEEE Council on Test Technology.

Fabrizio Lombardi (M’81–SM’02) received the B.Sc. degree (Hons.) in electronic engineering from the University of Essex, Essex, U.K., in 1977, and the M.S. degree in microwaves and modern optics, the Diploma in microwave engineering, and the Ph.D. degree from the Microwave Research Unit, University College London, London, U.K., in 1978, 1978, and 1982, respectively.

He is currently the holder of the International Test Conference (ITC) Endowed Professorship at Northeastern University, Boston, MA, where he served as Chair of the Department of Electrical and Computer Engineering from 1998 to 2004. Prior to joining Northeastern University, he was a Faculty Member at Texas Tech University, University of Colorado-Boulder, and Texas A&M University. His research interests are testing and design of digital systems, quantum and nano computing, ATE systems, configurable/network computing, defect tolerance and CAD VLSI. He has extensively published in these areas and edited six books.

Dr. Lombardi has received many professional awards including the Visiting Fellowship at the British Columbia Advanced System Institute, University of Victoria, Canada (1988), twice the Texas Experimental Engineering Station Research Fellowship (1991–1992, 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 a Silver Quill Award from Motorola-Austin (1996). He was an Associate Editor (1996–2000) of 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 IEEE TRANSACTIONS ON COMPUTERS and an Associate Editor of the IEEE DESIGN AND TEST MAGAZINE. Since 2004, he has served as the Chair of the Committee on “Nanotechnology Devices and Systems” of the Test Technology Technical Council of the IEEE. He 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, the IEEE Micro Magazine and the IEEE DESIGN AND TEST MAGAZINE. He is the Founding General Chair of the IEEE Symposium on Network Computing and Applications.