Yield Analysis of Compiler-based Arrays of Embedded SRAMs

X. Wang, M. Ottavi and F. Lombardi
Northeastern University, Department of Electrical and Computer Engineering
Email:{xiawang,mottavi,lombardi}@ece.neu.edu

Abstract

This paper presents a detailed analysis of the yield of embedded static random access memories (eSRAM) which are generated using a compiler. Defect and fault analysis inclusive of industrial data are presented for these chips by taking into account the design constructs (referred to as kernels) and the physical properties of the layout. The new tool CAYA (Compiler-based Array Yield Analysis) is based on a characterization of the design process which accounts for fault types and the relation between functional and structural faults; a novel empirical model is proposed to facilitate the yield calculation. Industrial data is provided for the analysis of various configurations with different structures and redundancy. The effectiveness and accuracy as provided by CAYA are assessed with respect to industrial designs.

1: 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 percentange of working or fault free chips in a batch) is commonly used as figure of merit for IC manufacturing. Due to increased technology sophistication, 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 memory types remains the static RAM (SRAM). SRAMs are typically embedded in SoC and ASIC chips in large numbers; today, it is not difficult to find large ASIC chips or SoCs with 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 [1]. 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. A systematic method of calculating the yield of compiler-based arrays is proposed. A new tool referred to as CAYA (Compiler-based Array Yield Analysis), has been developed based on this method. A compiler-based memory array is usually made of so-called kernels. Kernels are pre-designed modules (inclusive of layout) which can be integrated onto a chip such as a SoC. Manufaturing of compiler-based memories is a complicated process because it requires to consider the several levels (layers) of the chip. The proposed method analyzes the critical area 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. This together with the

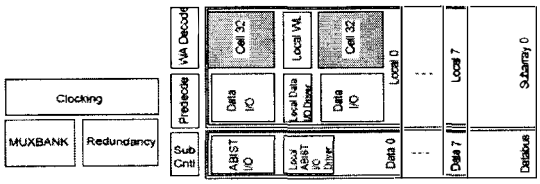

Figure 1. Array structure (SRAM1PR) made of 1 subarray

calculated defect density allows to find the number of different fault types of a configuration of the eSRAM.

2: Embedded SRAMs

In this paper, two memory cells are considered; these cells (denoted as SRAM1PN and SRAM1PR) are one-port compiler-based embedded SRAM (0.13u technology for ASIC). The structures of SRAM1PN and SRAM1PR have many similiarities; SRAM1PR has 4 independent redundant (word) lines while SRAM1PN has no redundancy. The basic structure of a SRAM1PR subarray is shown in Figure 1; this subarray has its own redundant word lines. SRAM1PN has similar structure but no redundancy. A compiler-based array consists of several functional modules. The functional modules of the SRAM1PN and SRAM1PR 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 [2], compiler-based memories have a large number of configurations (for example, SRAM1PN 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. The compiler option refers to the capability of specifying a memory configuration. The compiler option for the two cells considered in this paper is denoted as SRAM1PNwXbDdSsM1 (SRAM1PRwXbDdSsM1). Its detailed description is given in Table 1. Each functional module is built using at least one type of place-

Table 1. Compiler option of SRAM1PN and SRAM1PR

SRAM1PN	standard one-port SRAM with no redundancy.
SRAM1PR	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.
M1	array-clocked timing mode, at present this option is unused.

able kernels. A placeable kernel defines a pre-designed 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, A8F_CELL16_P is the placeable kernel of the cell (either SRAM1PN or SRAM1PR). A8F_CELL16_P has 16 bits cells and A8F identifies the technology (0.13u ASIC technology). Different configurations of compiler-based arrays have different types and numbers of placeable kernels. Table 2 shows the placeable kernels.

Table 2. Kernels for SRAM1PN and SRAM1PR arrays

Kernel	Placeable kernel	Number of kernels
CELL	A8F_CELL16_P	#cells=wb #A8F_CELL16_P=$wb/16$
DIO	A8F_DIO_L_P and A8F_DIO_R_P	#DIO=$bd/4$ #A8F_DIO_L_P=$bd/4/2$ #A8F_DIO_R_P=$bd/4/2$
LWLD	A8F_LWLDVR_P	#A8F_LWLDVR_P=$(bd/64)(w/d/4)$
GWLD	A8F_WLDVR_P	#A8F_LWLDVR_P=$(w/d/4)$

3: Proposed Yield Procedure

The proposed methodology is based on the following design process as applicable to compiler-based eSRAMs in today's industry:

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 the negative binomial yield model.
5. If redundancy is provided, calculate the yield according to a new 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; the second functionality calculates the yield of this array. To perform the first functionality, CAYA is supplied as inputs 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 pre-calculated by a critical area extraction tool (i.e. either INCA or CAA which are critical area analysis tools, the first one based on shape expansion and the second on monte-carlo simulation [9]). 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 with the specified configuration. Redundancy is also supplied through the compiler option. The yield is found by CAYA using a novel empirical based model as proposed in this paper. The procedure to calculate the yield of compiler-based arrays as applicable using CAYA utilizes the following steps:

- *Step 1:* 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. If the eSRAM has no redundancy, use existing yield techniques (such as the negative binomial model).
- *Step 2:* If the eSRAM has redundancy, then repair of the eSRAM is simulated. Also, the number of faults left unrepaired is calculated. Fault types are also taken into account.
- *Step 3:* CAYA is executed to calculate the yield of the eSRAM based on a new model using a linear curve fitting by regression.

Table 3. 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

4: Defects and Faults in eSRAMs

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 spare resources [2]. The proposed method is similiar 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 SC (single cell), VP (vertical cell pair), HP (horizontal cell pair), SW (single word line), DW (double word line), WB (word and bir line cross), SD (single data column), DD (double data column), CK (chip kill).

5: Critical Areas and Defect Analysis

As described previously, CAYA's execution is based on the kernels in the compiler option. These kernels play an important role in defining the critial 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

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

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_Ds of the cell kernel (denoted as CELL) as well as the other kernels (as calculated by CAA) with respect to the possible structural faults for the eSRAMs considered in this paper can be obtained. 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.

$$N_D(L) = \frac{D_L}{D_{M1\_extra}} \tag{2}$$

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 3 as obtained by CAA and INCA. The results for the T_D in Table 3 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.

Table 4. Area and fault ratios

Ratio	CELL	LWLDVR	GWLDVR	DIO and AIO	Master and Support Circuits
Area	0.8	0.06	0.03	0.1	0.01
Fault	0.91	0.034	0.01	0.042	0.004

6: eSRAM Yield Model

The yield model of redundant and non redundant memories has been described in many previous works [3] [4] and [5] to [8]. The proposed approach first establishes 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 by assuming that all types of faults show the same cluster characteristic, a very good yield model is still possible [8]. 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. SRAM1PN (or SRAM1PR) has 14 structural level faults and 9 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 structral level faults. Using the defect density matrix obtained from the manufacturing line, the number of faults (of different types) is given as

$$\Lambda^T = \begin{pmatrix} \lambda_{SC} & \lambda_{VP} & \lambda_{HP} & \lambda_{SW} & \lambda_{DW} & \lambda_{WB} & \lambda_{SD} & \lambda_{DD} & \lambda_{CK} \end{pmatrix} \tag{4}$$

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,

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

Let Y_r be the yield of the eSRAM after repairing the memory using the provided redundancy; repair effectively translates into a process by which hopefully all faults can be corrected. Let the number of faults left unrepaired be λ_r and Y_p be the so-called perfect yield which is the probability that there is no fault left unrepaired (i.e. the number of faults to be repaired is λ). Hence,

$$Y_p = P(0) = (1 + \lambda/\alpha)^{-\alpha} \tag{7}$$

A model from Stapper, [1] enumerates the probability of successfully repairing all combinations of fault types using the provided redundancy. Let C_F denote the possible combinations of faults. The proposed model also enumerates the probability of all combinations of faults of different types that can be repaired by the provided redundancy (C_F denotes such possible combinations). Assume a Poisson distribution for the faults; for k faults of type i,

$$P_i(k) = \frac{e^{-\lambda_i} \lambda_i^{-k}}{k!} \tag{8}$$

So,

$$Y_r = P_{CK}(0) \sum_{C_F} P_{SC}(i) P_{VP}(j) P_{HP}(k) P_{SW}(l) P_{DW}(m) P_{WB}(n) P_{SD}(o) P_{DD}(p) \tag{9}$$

In equation (9), Y_r is the repairable yield for $(i+j+k+l+m+n+o+p)$ faults of different types. Let λ_r denote the number of faults left unrepaired; then by inversion, for an eSRAM array with no redundancy, let the yield after repair is given by

$$Y_r = (1 + \lambda_r/\alpha)^{-\alpha} \tag{10}$$

The next step in CAYA (following the analysis of the critical areas and the defect density) is to calculate 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 (less computational intensive) 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 provides a method that can be incorporated within CAYA at reduced execution complexity, while still providing accurate results for the eSRAM configurations. The notation used hereafter must be introduced. (1) W_{LPerSA}: number of word lines per subarray of the eSRAM. W_{LPerSA} is obtained from the compiler option. (2) B_{LPerSA}: number of word lines per subarray of the eSRAM. B_{LPerSA} is also found through the compiler option. (3) Y: yield of the eSRAM. (4) Y_{PerSA}: yield of a subarray. (5) F: number of faults of the eSRAM. (6) A: parameter of the empirical model of Y_{PerSA}. A is found from curve fitting by linear regression (7) B: parameter of the empirical model of Y_{PerSA}. B is found from curve fitting by linear regression (8) α: parameter of the empirical model of Y_{PerSA}, α is fixed to 2.0. 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_{LPerSA}, F increases linearly with B_{LPerSA}. Let $\hat{B}$ denote the slope of the linear relationship i.e. $F = \hat{B} B_{LPerSA}$. As $\hat{B}$ increases approximately in linear fashion with W_{LPerSA}, then $\hat{B} = B W_{LPerSA} + A$. So

$$F = (B W_{LPerSA} + A) B_{LPerSA} \tag{11}$$

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 SRAM1PN, there are 7000 configurations). then,

$$e = \frac{\sum_{i=1}^{N} \sqrt{(F_i - \hat{F}_i)^2}}{N} \tag{12}$$

The following equations characterize the yield and its calculation from its compiler option. As by Table 1: $W_{LPerSA} = w/d/s$ and $B_{LPerSA} = bd$ then using the linear model of (11), F can be calculated so that the yield is: $Y = (1 + F/\alpha)^{-\alpha}$

7: Evaluation

This section presents some experimental results 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 SRAM1PN32768X032D32S2M1 (of 1M bit size) which is fixed to 100%. For a subarry made of SRAM1PN memory cells (no redundancy), assume Y(SRAM1PN16384X032D32S1M1)=

98% ; using the proposed empirical model of the previous section, the following values were obtained for the paramers in the linear regression: A=0.000003493641, and B=0.000000023269. The yield values obtained by using the proposed empirical model as well as the fully computed data are shown in Figure 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. This model for example can be used to compare the yield of different eSRAM configurations with and without redundancy. Moreover the following results are reported for the proposed empirical model versus the number

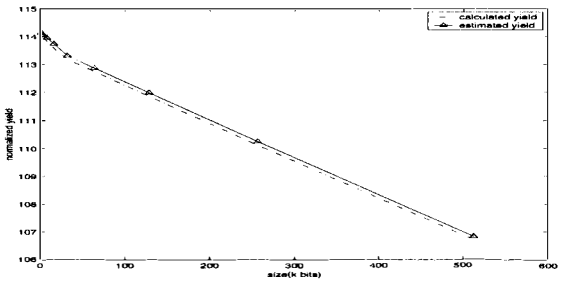

Figure 2. Yield of SRAM1PN-based memory subarray (fully computed data and proposed empirical model)

of bit lines in a subarray.

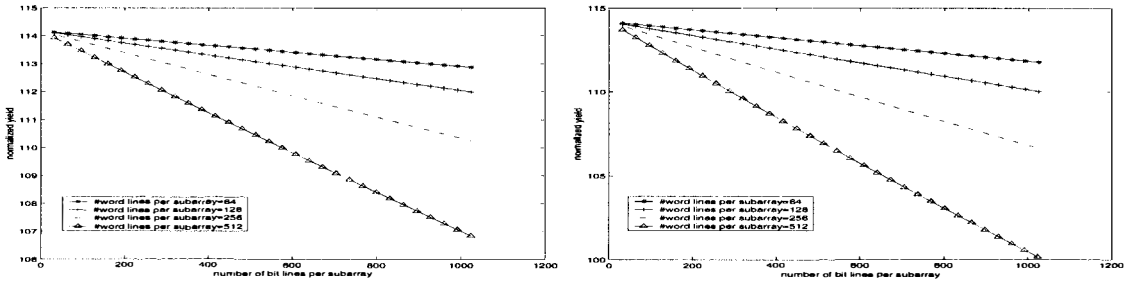

Figure 3. Yield of SRAM1PN-based memory:(a) 1 subarray (b) 2 subarrays

- Yield of SRAM1PN-based memory subarray in Figure 3 (a).
- Yield of SRAM1PN-based memory made of 2 subarrays in Figure 3 (b).
- Yield of SRAM1PR-based memory subarray in Figure 4 (a).
- Yield of SRAM1PR-based memory made of 2 subarrays in Figure 4 (b).

The following conclusions can be drawn from these figures: (1) CAYA provides excellent computational accuracy (the error between the empirical model as proposed in this paper and the computed data is very small). (2) A substantial improvement in yield can be accomplished using the compiler option in the design of a eSRAM. (3) The increase in yield is a function of the memory organization: it increases by reducing the number of wordlines while it decreases by increasing the number of bitlines (i.e. for a fixed memory size).

8: Conclusions

This paper has introduced a new tool (denoted as CAYA) for the yield of compiler-based embedded SRAMs (eSRAMs). CAYA is based on a novel characterization of the yield within a compiler

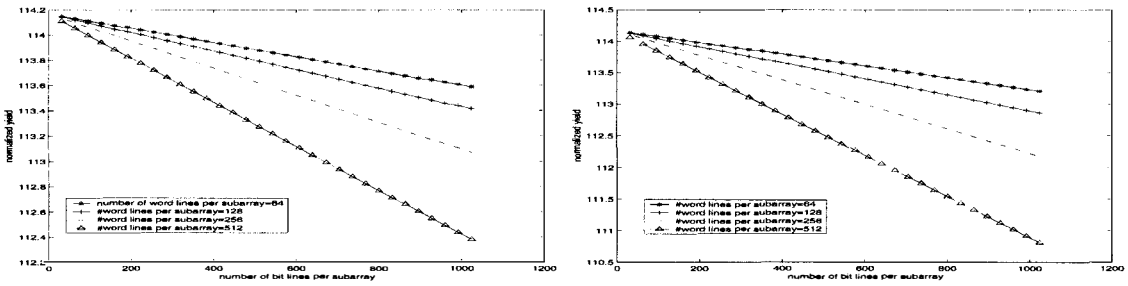

Figure 4. Yield of SRAM1PR-based memory with 4 redundant lines:(a) 1 subarray (b) 2 subarrays

design environment. The memory compiler utilizes pre-defined 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 (0.13u technology for ASIC). CAYA (Compiler-based Array Yield Analysis) 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 and the 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. 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 effectiveness and accuracy as provided by CAYA have been assessed with respect to industrial designs.

References

[1] 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.

[2] C. H. Stapper, " Modeling of Integrated Circuit defect sensitivities". *IBM J. Res. Develop.*, vol.27, no.6, pp.549-557, Nov. 1983.

[3] R. M. Warner, "Applying a Composite Model to the IC Yield Problem". *IEEE Journal of Solid State Circuits.*, vol.SC-9, no.3, pp.86-95, June 1974.

[4] C. H. Stapper, "On Murphy's Yield Integral". *IEEE Trans. Semiconductor Manufacturing*, vol.4, no.4, pp.294-297, Nov. 1991.

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

[6] C. H. Stapper, "Large-Area Fault Clusters and Fault Tolerance in VLSI Ciruits" *IBM J. Res. Develop.*, vol.33, no.2, pp.162-173, March 1989.

[7] C. H. Stapper, "Small-Area Fault Clusters and Fault Tolerance in VLSI Ciruits" *IBM J. Res. Develop.*, vol.33, no.2, pp.174-177, March 1989.

[8] C. H. Stapper, "Improved Yield Model for fault-Tolerant Memory Chips". *IEEE Tran. on Computers,*vol.42, no.7, pp.872-881, July 1993.

[9] G A Allen, "A Comparison of Efficient Dot Throwing and Shape Shifting Extra Material Critical Area Estimation," *Proc. of IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems* , 1998, pp. 4452.