[9] S.Naik, F.Agricola, and W.Maly, *Failure analysis of high-density CMOS SRAMs*, IEEE Design and Test of Computers, vol.10, no.2, pp.13-23, June, 1993

[10] J,Khare, W.Maly, S.Griep and D.Schmitt-Landsiedel, *Yield-oriented computer-aided defect diagnosis*, IEEE Trans. on Semc. Manu., vol.8, no.2, pp.195-206, May, 1995

[11] B.C. Peter, JR and H.F. Walker, *The numerical evaluation of the maximum-likelihood estimate of a subset of mixture proportions*, Siam J. Appl. Math., 35, pp. 447-452, 1978

[12] A.P. Dempster, N.M. Laird and D.B. Rubin, *Maximum-likelihood from incomplete data via the EM algorithm*, J. Royal Statist. Soc. Ser. B, 39, pp. 1-38, 1977

[13] A. Jalubowski, W. Marcinial and H.M. Przewlocki, *Diagnostic Measurements in LSI/VLSI Integreted Circuits Production*, World Scientific, 1991

[14] M.Abramovici, M.A. Breuer and A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990

[15] *LOQO an efficient implementation of an interior-point method for large scale linear and/or quadratic programming problems*, available via ftp://elib.zib-berlin.de

[16] P.Y. Papalambros and D.J. Wilde, *Principle of Optimal Design*, Cambridge University Press, 1988

[17] M.H. DeGroot, *Probability and Statistics*, Adderson-Wesley Publishing Company, Second Edition

[18] S.J. Leon, *Linear Algebra with Applications*, Macmillan Publishing Company, 1990

[19] S.R. Searle, *Linear Models*, John Wiley & Sons, Inc., 1971

[20] V.A. Sposito, *Linear and Nonlinear Programming*, The Iowa State University Press, 1975

[21] Jianlin Yu, *Maximum likelihood estimation for failure analysis of SRAM cells using Inductive Fault Analysis*, Master Thesis, University of California Santa Cruz, 1996

[22] B.N. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, 1995

[23] D.W. Hosmer, JR, *A comparison of iterative maximum-likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample*, Biometrics, 29, pp. 761-770, 1973

[24] B.M. Hill, *Information for estimating the proportions in mixtures of exponential and normal distributions*, J. Amer. Statist Assoc., 58, pp. 918-932, 1963

6 Summary

In this paper we have described an iterative ML estimation method for failure analysis. We have numerically compared the ML estimate with other two methods, the least squares and enumeration. The results showed that both the least squares and the enumeration estimators didn't coincide with MLEs. We analyzed the ill-conditioned problem of a DCM, and illustrated some "good" or "bad" DCM structures. The elements of the normalized DCM should be well-separated to guarantee accurate MLEs for defect classes. We also showed how to derive sample size needed to ensure the expected accuracy of ML estimation. Finally, to illustrate the effectiveness of the iterative ML algorithm we have provided an experimental example by performing our ML estimation on a real SRAM cell with the assumption that each electrical fault represents a fault behavior. The ML estimates with such an assumption provide a lower bound on number of samples needed.

References

[1] R.A. Redner and H.F. Walker, *Mixture densities, maximum likelihood, and the EM algorithm*, Siam Review, vol 26, pp. 195-239, 1984

[2] Alvin Jee and F.J. Ferguson, Carafe: *An software tool for failure analysis*, in Proc. of Int. Sym. on Testing and Failure Analysis, pp. 143-149, 1993,

[3] D.P. Helmbold, R.E. Schapire, Y. Singer, M.K. Warmuth, *A comparison of new and old algorithms for a mixture estimation problem*, Appearing in Proc. of the Eighth Annual Conf. on Computational Learning Theory, July 1995

[4] D.Y. Lepejian, J.M. Caywood, A. Kablanian, F.J. Ferguson, and Alvin Jee, *An Automated Failure Analysis (AFA) methodology for repeated structures*, in Proc. of the IEEE VLSI Test Sym., pp. 319-324, 1994

[5] W.Lukaszek, W.Yarbrough, and K.Grambow, *CMOS process problem debugging using complementary defect monitors*, in Proc. of IEEE Int. Test Conf., pp.21-30, 1987

[6] E.M.J.G.Bruls, F.Camerik, H.J.Kretschman and J.A.G.Jess, *A generic method to develop a defect monitoring system for IC processes*, in Proc. of IEEE Int. Test Conf., pp.218-227, 1991

[7] W.Maly, B.Trifilo, R.A.Hughes and A.Miller, *Yield diagnosis through interpretation of tester data*, in Proc. of IEEE Int. Test Conf., pp.10-20, 1987

[8] Y.Kwon and D.M.H.Walker, *Yield learning via functional test data*, in Proc. of IEEE Int. Test Conf., pp.626-635, 1995

To avoid fortuitous phenomena of using only one experiment, we performed 20 more experiments using the same assumption and approach. The results are shown in Table 8. From that

TABLE 8 MLEs for p(d), means of p(d) and average standard deviations of the twenty experiments

	short									break							
	poly-ndiff	poly-pdiff	poly-m1	poly-poly	ndiff-ndiff	m1-m1	ndiff-m1	m1-m2	pdiff-pdiff	ndiff	ndc	m2 or m2c	poly	pdiff	pdc	m1	polyc
p(d)_1	0.052	0.014	0.0377	0.0972	0.0125	0.2648	0.0422	0.1255	0.0042	0.00	0.0360	0.1049	0.00	0.00	0.0781	0.0691	0.0620
p(d)_2.	0.064	0.0080	0.0357	0.0976	0.0114	0.2680	0.0541	0.1176	0.0035	0.00	0.0357	0.1058	0.00	0.00	0.0753	0.0632	0.0600
p(d)_3	0.044	0.0060	0.0611	0.0908	0.0113	0.2601	0.0317	0.1418	0.0031	0.00	0.0352	0.1266	0.00	0.00	0.0741	0.0581	0.0560
p(d)_4	0.060	0.0100	0.0440	0.0789	0.0122	0.2548	0.0527	0.1312	0.0041	0.00	0.0352	0.1186	0.00	0.00	0.0741	0.0581	0.0660
p(d)_5	0.040	0.0120	0.0356	0.1015	0.0248	0.2823	0.0427	0.1259	0.0027	0.00	0.0228	0.1423	0.00	0.00	0.0746	0.0463	0.0420
p(d)_6	0.046	0.0120	0.0439	0.0836	0.0110	0.2592	0.0545	0.1144	0.0033	0.00	0.0417	0.1309	0.00	0.00	0.0799	0.0699	0.0500
p(d)_7	0.042	0.0120	0.0511	0.0828	0.0232	0.2626	0.0470	0.1447	0.0026	0.00	0.0224	0.1092	0.00	0.00	0.0976	0.0409	0.0620
P(d)_8	0.052	0.0120	0.0487	0.0618	0.0130	0.2878	0.0478	0.1352	0.0036	0.00	0.0333	0.1361	0.00	0.00	0.0818	0.0368	0.0500
p(d)_9	0.060	0.0120	0.0729	0.0584	0.0000	0.2337	0.0383	0.1584	0.0043	0.00	0.0290	0.1475	0.00	0.00	0.0693	0.0523	0.0640
p(d)_10	0.048	0.0060	0.0442	0.1060	0.0109	0.2763	0.0366	0.1661	0.0039	0.00	0.0393	0.0992	0.00	0.00	0.0759	0.0436	0.0440
p(d)_11	0.048	0.0140	0.0376	0.0980	0.0106	0.2637	0.0573	0.1284	0.0024	0.00	0.0290	0.1155	0.00	0.00	0.0772	0.0524	0.0660
p(d)_12	0.0660	0.0014	0.0541	0.0738	0.0114	0.2772	0.0483	0.1333	0.0057	0.00	0.0234	0.1186	0.00	0.00	0.0902	0.0578	0.0520
p(d)_13	0.040	0.0040	0.0391	0.1007	0.0102	0.2674	0.0571	0.1178	0.0056	0.00	0.0411	0.1206	0.00	0.00	0.0718	0.0585	0.0680
p(d)_14	0.038	0.0080	0.0493	0.1187	0.0117	0.2569	0.0569	0.1436	0.0017	0.00	0.0351	0.1115	0.00	0.00	0.0598	0.0516	0.0640
p(d)_15	0.044	0.0060	0.0572	0.0958	0.0113	0.2217	0.0552	0.1401	0.0046	0.00	0.0413	0.1407	0.00	0.00	0.0725	0.0575	0.0520
p(d)_16	0.054	0.0040	0.0687	0.0881	0.0117	0.2607	0.0443	0.1235	0.0050	0.00	0.0344	0.1064	0.00	0.00	0.0841	0.0472	0.0680
p(d)_17	0.054	0.0100	0.0428	0.0865	0.0122	0.2518	0.0679	0.1360	0.0047	0.00	0.0178	0.1137	0.00	0.00	0.0813	0.0632	0.0580
p(d)_18	0.048	0.0140	0.0669	0.0982	0.0128	0.2739	0.0538	0.1300	0.0044	0.00	0.0172	0.1183	0.00	0.00	0.0711	0.0454	0.0460
p(d)_19	0.048	0.0060	0.0486	0.0869	0.0119	0.2698	0.0449	0.1307	0.0012	0.00	0.0348	0.1155	0.00	0.00	0.0849	0.0528	0.0640
p(d)_20	0.060	0.0100	0.0530	0.0940	0.0114	0.2716	0.0332	0.1440	0.0048	0.00	0.0345	0.0924	0.00	0.00	0.0883	0.0469	0.0560
mean of p(d)	0.049	0.0097	0.0496	0.0896	0.0122	0.2632	0.0483	0.1344	0.0037	0.00	0.0319	0.1187	0.00	0.00	0.0781	0.0535	0.0575
average stand. dev.	0.0066	0.0029	0.0089	0.0105	0.0033	0.0117	0.0076	0.0101	0.0038	0.00	0.0068	0.0118	0.00	0.00	0.0067	0.0079	0.0069

table, one can easily see that the estimation accuracies in all experiments accord with the expected ones with a sample size of 500.

TABLE 7 MLE for p(d), standard deviation, diagonal elements of inverse of Fisher Information matrix, and sample sizes needed to achieve 0.05 and 0.01 standard deviations

	short									break							
	poly-ndiff	poly-pdiff	poly-m1	poly-poly	ndiff-ndiff	m1-m1	ndiff-m1	m1-m2	pdiff-pdiff	ndiff	ndc	m2 or m2c	poly	pdiff	pdc	m1	polyc
p(d)	0.066	0.012	0.0345	0.0824	0.0111	0.2682	0.0366	0.1176	0.0015	0.00	0.0362	0.1101	0.00	0.00	0.0908	0.0749	0.0580
stnd. dev.	0.016	0.0020	0.0155	0.0076	0.0011	0.0018	0.0134	0.0124	0.0015	0.0000	0.0062	0.0099	0.0000	0.0000	0.0108	0.0249	0.0020
$I^{-1}(p(d))$	0.0660	0.0120	0.4181	0.2655	0.0595	0.6020	0.1117	0.1466	0.0771	0.00	0.1029	0.1422	0.0000	0.0000	0.2654	0.4459	0.0580
sample size (0.05)	30	5	170	110	30	250	50	60	40	1	50	60	1	1	110	180	30
sample size (0.01)	700	130	4200	2700	600	6100	1200	1500	800	10	1100	1500	10	10	2700	4500	600

we assume the probability distribution in Table 6 is the true value for each defect class, the standard deviation of each estimated defect class probability is less than 0.03 (see Table 7). This meets our expectations. Fig. 3 is a histogram of probability distribution of defect classes. It is

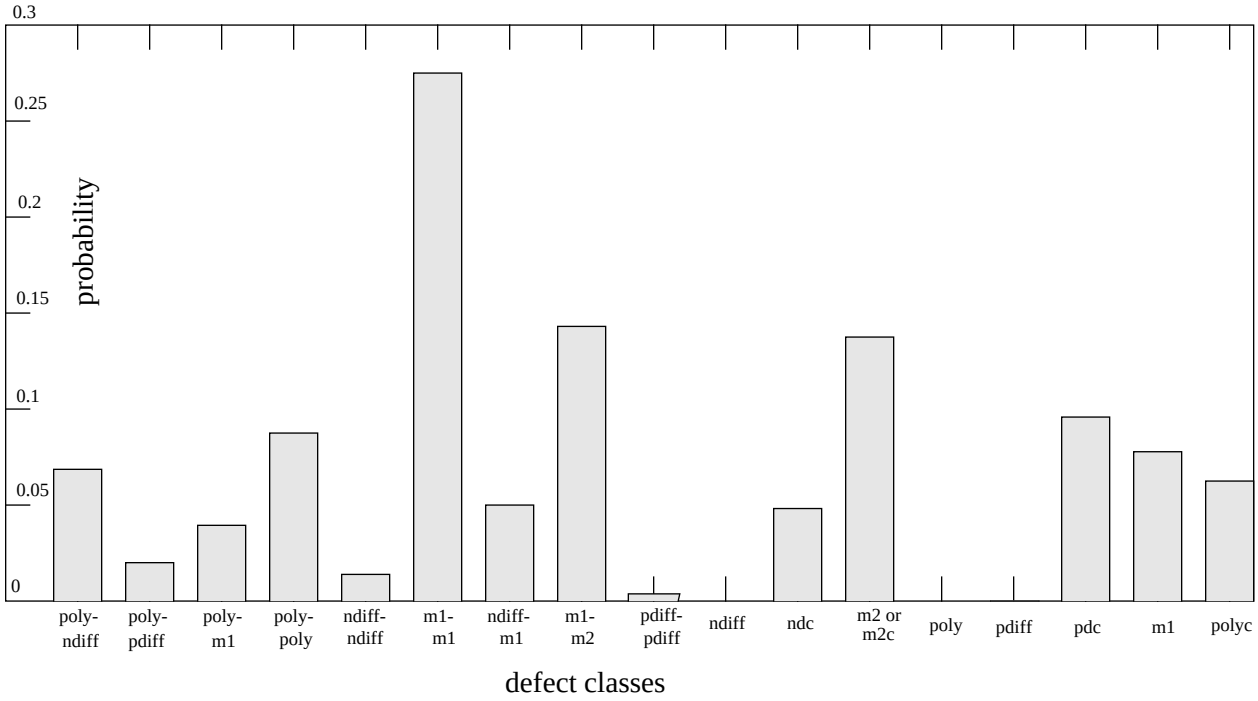

Figure 3. Probability distribution of defect classes

obvious that shorts between two metal1 layers are the most likely cause of faults.

easier to solve since an electrical fault-to-fault mapping is a many-to-one mapping. After combining indistinguishable electrical faults which are caused by the same defect classes and combining indistinguishable defect classes which have the same fault distribution, and then normalizing the conditional probabilities, we get the assumed normalized DCM shown in Table 5. The blank entries in the normalized DCM denote corresponding conditional probabilities being zero. f_1, f_2,..., f_{25} represent the twenty-five different electrical faults in the SRAM cell. Before combining indistinguishable electrical faults there were seventy-one electrical faults. In each experiment, we generated 500 defects using a randomly generated defect class distribution in Table 6. We distributed 500 defects to each defect class according to the defect class distribution in Table 6 to get the number of occurrences of each defect class. Due to using a random number generator, the number of defects in each defect class was seldom exactly the ratio depicted in Table 6. This was to simulate the stochastic nature of which defects occur. Then we distributed the generated defects in each defect class to its corresponding fault behaviors according to the conditional probabilities $p(f_i|d_j)$ in the DCM to get the number of occurrences of a fault given the number of occurrences of a defect class. These were distributed using a random number generator as before to simulate the stochastic nature of which faults occur due to a given defect. Finally we added the numbers of occurrences of the same fault to generate the total number of occurrences of each fault. The last column of Table 5 shows one of the generated number of occurrences of each fault using the above approach with a sample size of 500 defects.

Table 7 shows the results of using the ML estimation on Table 5. We have computed MLE for each defect class using our iterative ML algorithm, and numerically evaluated $I(p(\mathbf{d}))$, its condition number and its inverse for values of $\mathbf{d}$ using MATLAB system. Sample sizes are those needed to ensure that standard deviation of MLE for $p(d_i)$ is 0.05 or less by using the equation (2). From Table 7 one can easily see that the normalized DCM in Table 5 is well-conditioned, and the sample size needed to ensure 0.05 standard deviation is less than 300. Thus the MLEs are reliable. If

TABLE 6 The assumed probability distribution of defect classes

	short									break								
	poly-ndiff	poly-pdiff	poly-m1	poly-poly	ndiff-ndiff	m1-m1	ndiff-m1	m1-m2	pdiff-pdiff	ndiff	ndc	m2	m2c	poly	pdiff	pdc	m1	polyc
p(d)	0.05	0.01	0.05	0.09	0.01	0.27	0.05	0.13	0.00	0.00	0.03	0.02	0.10	0.00	0.00	0.08	0.05	0.06

TABLE 5 The normalized DCM of the SRAM cell and number of occurrence of each fault

	short									break								
	poly-ndiff	poly-pdiff	poly-m1	poly-poly	ndiff-ndiff	m1-m1	ndiff-m1	m1-m2	pdiff-pdiff	ndiff	ndc	m2 or m2c	poly	pdiff	pdc	m1	polyc	**F**
f_1		1.0																6
f_2	1.0																	33
f_3							0.1736											3
f_4								0.6564										39
f_5						0.1956			1.0									27
f_6			0.5957	0.6934	0.0774	0.1177												55
f_7						0.0078	0.0131											2
f_8					0.0559	0.0348												4
f_9						0.2133		0.2311										42
f_{10}					0.1640													1
f_{11}			0.4043		0.5675	0.1582	0.4685											40
f_{12}					0.1353			0.1125										7
f_{13}						0.2726												37
f_{14}				0.3066			0.3448											19
f_{15}										0.2968	0.3319	1.0				0.1321		66
f_{16}											0.3341							6
f_{17}														0.5366	0.7929	0.4127		51
f_{18}										0.7032								0
f_{19}											0.3340					0.2893		17
f_{20}													0.9304					0
f_{21}																0.0296		1
f_{22}																	0.4991	14
f_{23}														0.4634				0
f_{24}															0.2071	0.1363		15
f_{25}													0.0696				0.5009	15

TABLE 4 The condition numbers of normalized DCM, Hessian matrix and Fisher Information matrix, and sample sizes of Example 4

normalized DCM	cond(A)	cond(H)	cond(I)	sample size
A_1	∞	∞	∞	∞
A_2	∞	∞	∞	∞
A_3	3.2151e+16	3.2796e+07	1.2821e+07	1.0e+9
A_4	3.4767	12.6965	12.6964	290

From Table 4 one can easily see that the defect class estimates on A_4 will be more accurate than on other three DCMs. Thus we prefer to have a SRAM with such normalized DCM as A_4 to perform failure analysis.

5 An Experimental Example

In this section, we illustrate the use of the ML method described in the previous sections on a real SRAM cell. The layout of the six-transistor CMOS SRAM cell is drawn for a 0.6μ single poly, double metal CMOS technology. The word line is routed in poly horizontally through the cell. The bit and bit_bar lines are run vertically in the second metal. The Vdd and GND lines are horizontally drawn in the first layer metal. The size of the cell is $14.1 \times 10.5\mu^2$.

As mentioned before, we use Carafe to analyze this cell and extract a list of electrical faults with probability of the occurrence of each electrical fault. For the experiment in this paper, we ran Carafe using a defect radius of 3.25 microns. We used a uniform scaling factor between layers to find the critical area for each defect. The ratio of each defect critical area to the total critical area is the occurrence probability of an electrical fault if that defect occurs. For the purpose of illustrating the essential idea behind the ML method, here we assume that each electrical fault has a unique fault behavior. That is, we use defect-to-electrical fault mapping information instead of defect-to-fault mapping information to construct the DCM. This assumption makes the problem

That is, if the normalized DCM is well-conditioned, the performance of ML estimation method is very good in the sense that it converges rapidly to the unique maximum and the rate of convergence is not sensitive to the initial estimates. On the other hand, for the ill-conditioned matrix, slopes can be steep, thus much larger sample size needed to guarantee the accuracy of the ML estimation.

The simple example below illustrates the conditions of several DCM structures.

<u>Example 4.</u> Let's suppose there are three possible faults and three defect classes with probabilities in each DCM A'_i for $i = 1,2,3,4$. Matrix A_i is the normalized version of matrix A'_i.

$$A'_1 = \begin{bmatrix} \frac{1}{18} & \frac{2}{18} & \frac{3}{18} \\ \frac{1}{18} & \frac{2}{18} & \frac{3}{18} \\ \frac{1}{18} & \frac{2}{18} & \frac{3}{18} \end{bmatrix} \quad \Rightarrow \quad A_1 = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

$$A'_2 = \begin{bmatrix} \frac{1}{18} & \frac{1}{18} & \frac{1}{18} \\ \frac{2}{18} & \frac{2}{18} & \frac{2}{18} \\ \frac{3}{18} & \frac{3}{18} & \frac{3}{18} \end{bmatrix} \quad \Rightarrow \quad A_2 = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

$$A'_3 = \begin{bmatrix} \frac{1}{45} & \frac{2}{45} & \frac{3}{45} \\ \frac{4}{45} & \frac{5}{45} & \frac{6}{45} \\ \frac{7}{45} & \frac{8}{45} & \frac{9}{45} \end{bmatrix} \quad \Rightarrow \quad A_3 = \begin{bmatrix} \frac{1}{12} & \frac{2}{15} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{7}{12} & \frac{8}{15} & \frac{1}{2} \end{bmatrix}$$

$$A'_4 = \begin{bmatrix} \frac{1}{45} & \frac{9}{45} & \frac{6}{45} \\ \frac{4}{45} & \frac{2}{45} & \frac{8}{45} \\ \frac{7}{45} & \frac{5}{45} & \frac{3}{45} \end{bmatrix} \quad \Rightarrow \quad A_4 = \begin{bmatrix} \frac{1}{12} & \frac{9}{16} & \frac{6}{17} \\ \frac{1}{3} & \frac{1}{8} & \frac{8}{17} \\ \frac{7}{12} & \frac{5}{16} & \frac{3}{17} \end{bmatrix}$$

The computed condition numbers for each normalized DCM are shown in Table 4. The condition number of a matrix is computed using interactive MATLAB program. Assume that the values of $p(d_1)$, $p(d_2)$, $p(d_3)$ for A_3 and A_4 are *0, 0, 1* and *0.28, 0.21, 0.51* respectively. Sample sizes are those needed to ensure that the standard deviation of each MLE of $p(d_i)$ is 0.1 or less. They are determined using the largest sample size among the components of probability vector $p(\boldsymbol{d})$.

$$= \sum_{i=1}^{m} \nabla\nabla^T p(F_i) \log p(f_i)$$

where $\nabla\nabla^T = \left(\frac{\partial^2}{\partial p(d_i)\, \partial p(d_j)} \right)$. It is an $n \times n$ symmetric matrix. If we use 2-norm the condition number, *cond(**H**)*, of the Hessian matrix is

$$cond(\boldsymbol{H}) = \|H(p(\boldsymbol{d}))\|_2 \left\| H^{-1}(p(\boldsymbol{d})) \right\|_2 = \frac{\sigma_1 \text{ (the largest singular value)}}{\sigma_n \text{ (the smallest singular value)}}$$

We can also use Fisher Information to estimate the condition of the log function. Recall that we have mixture model defined as $p(f_i) = \sum_{j=1}^{n} p(d_j)\, p(f_i | d_j)$. The Fisher Information matrix $I(p(\boldsymbol{d}))$ is defined in [17] as:

$$I(p(\boldsymbol{d})) = -E\left[\nabla\nabla^T \log p(f_i) \right]$$

Since $p(f_i)$ is a probability function, we have

$$I(p(\boldsymbol{d})) = -\sum_{i=1}^{m} \left(\nabla\nabla^T \log p(f_i) \right) p(f_i)$$

Since the condition number of $I(p(\boldsymbol{d}))$ can reflect the limits of accuracy of the condition number of Hessian matrix [1], we can measure the ill-conditioning of the log function by computing Fisher Information matrix instead of Hessian matrix.

By using Fisher Information, we can also derive Cramer-Rao lower bound for the variances of the MLEs for $p(\boldsymbol{d})$ as follows [17]:

$$var(p(d_j)) \geq \frac{1}{\sum_{i=1}^{m} F_i} I^{-1}(p(\boldsymbol{d}))_{jj} \tag{2}$$

where $I^{-1}(p(\boldsymbol{d}))_{jj}$ is the *j*th diagonal element of the inverse of the Fisher Information matrix. Thus the variances of the MLEs for $p(\boldsymbol{d})$ cannot be smaller than the reciprocal of the Fisher information in the sample. From equation (2) we can derive a lower bound on sample size needed for a given standard deviation.

The performance of our estimates largely depends on the condition of the Fisher Information matrix. Since the condition number of the normalized DCM is directly proportional to that of the Fisher Information matrix, the condition of the inherent normalized DCM will play an important role in determining the accuracy of the MLEs of the probability distribution of defect classes.

$$A = \begin{bmatrix} 0.37 & 0.002 & 0.183 \\ 0.105 & 0.087 & 0.333 \\ 0.05 & 0.444 & 0.147 \\ 0.235 & 0.301 & 0.003 \\ 0.24 & 0.166 & 0.334 \end{bmatrix} \qquad \boldsymbol{F} = \begin{bmatrix} 10 \\ 10 \\ 6 \\ 2 \\ 2 \end{bmatrix}$$

The results are shown in Table 3.

TABLE 3 The computational results of Least Squares Estimate (LSE), Enumerative Estimate (EE) and Maximum Likelihood Estimate (MLE) for Example 3

Method	$p(d_1)$	$p(d_2)$	$p(d_3)$	value of likelihood function
LSE	0.3543	0.0823	0.5634	1.482e-21
EE	0.2667	0.0666	0.6667	1.648e-21
MLE	0.2710	0.0770	0.6520	1.655e-21

From above examples, one can easily see that the solutions of both constrained least squares and enumerative methods are not always coincide with MLEs which are our expected solutions. Thus these two methods are not ML estimation methods for $p(\boldsymbol{d})$. The reason why the least squares is not ML estimation may be that linear model $A\boldsymbol{D} = \boldsymbol{F} + \varepsilon$ (ε are random measurement errors) does not reflect stochastic nature of $\boldsymbol{D}$ and $\boldsymbol{F}$, and improperly compensates for this by adding a measurement error vector ε. Both the least squares and the enumeration methods provide estiamtes of $\boldsymbol{D}$, not those of the underlying defect distribution causing $\boldsymbol{D}$, which are $p(\boldsymbol{d})$.

4 Conditioning of the Maximum Likelihood Estimation

The term *conditioning* and *condition* are used to indicate how sensitive the solution of a problem may be to small perturbations in the input data. A problem is ill conditioned if relatively slight perturbations in the data can produce large changes in the solution. The ill-conditioning of an optimization problem is usually measured by the condition number of Hessian matrix of the function to be optimized[1][16]. In our case, the Hessian is given by

$$H(p(\boldsymbol{d})) = \nabla\nabla^T L(p(\boldsymbol{d}))$$

The results are shown in Table 1[1].

TABLE 1 The computational results of Least Squares Estimate (LSE), Enumerative Estimate (EE) and Maximum Likelihood Estimate (MLE) for Example 1

Method	$p(d_1)$	$p(d_2)$	$p(d_3)$	value of likelihood function
LSE	0.9693	0	0.0307	1.112e-10
EE	0.8571	0	0.1429	1.316e-10
MLE	0.8839	0	0.1161	1.338e-10

Example 2. Given the normalized DCM A and vector $\boldsymbol{F}$ as follows:

$$A = \begin{bmatrix} 0.35 & 0.41 & 0.1 \\ 0.4 & 0.33 & 0.74 \\ 0.25 & 0.26 & 0.16 \end{bmatrix} \qquad \boldsymbol{F} = \begin{bmatrix} 10 \\ 15 \\ 5 \end{bmatrix}$$

The results are shown in Table 2.

TABLE 2 The computational results of Least Squares Estimate (LSE), Enumerative Estimate (EE) and Maximum Likelihood Estimate (MLE) for Example 2

Method	$p(d_1)$	$p(d_2)$	$p(d_3)$	value of likelihood function
LSE	0	0.6250	0.3750	4.850e-14
EE	0	0.6333	0.3667	4.850e-14
MLE	0.0003	0.6287	0.3710	4.851e-14

Example 3. Given the normalized DCM A and vector $\boldsymbol{F}$ as follows:

1. The LSE's results were obtained by using a public domain software package LOQO[15]. The LSE and EE's results D_j were converted into $p(d_j)$ by using the approximate formula $p(d_j) = D_j / \sum_{i=1}^{m} F_i$.

1) Compute the probability for each fault

$$\hat{p}(f_i) = \sum_{j=1}^{n} p^{(\Upsilon)}(d_j) p(f_i | d_j)$$

2) Recompute the probability for each defect class

$$p^{(\Upsilon+1)}(d_j) = p^{(\Upsilon)}(d_j) \sum_{i=1}^{m} \frac{p(F_i)\, p(f_i | d_j)}{\hat{p}(f_i)}$$

This iterative procedure should always converge to a global maximum since the Hessian matrix of the log-likelihood function $L(p(\boldsymbol{d}))$ is always negative definite. The convergence proof is the same as for other maximum-likelihood estimate problems [1]. If there exist several points that fall into the same maximum, the starting estimates of defect class probabilities will play an important role in deciding the solutions. Note that the initial estimates of mixture proportions $p^{(0)}(\boldsymbol{d})$ cannot be zero in our algorithm, because otherwise $p(\boldsymbol{d})$ will always be zero in the iteration.

If the sample size in the defect estimate is large, a modified version of EM called EM_η-update algorithm can be used to speed up the convergence with small extra computational cost. The new update uses the following formula:

$$p^{(\Upsilon+1)}(d_j) = p^{(\Upsilon)}(d_j)\left(\eta\left(\nabla L\left(p^{(\Upsilon)}(\boldsymbol{d})\right)_j - 1\right) + 1\right)$$

where the parameter η is called the learning rate. When $\eta = 1$ this gives the standard EM update. In practice, the EM_η-update algorithm can use either fixed scheduling for η e.g. $\eta = 2.5$ or line-searches to find the best choice of η on each iteration. For the ideas behind the EM_η-update algorithm please refer to Helmbold[3].

The following three small examples compare the numerical results of the ML method with other two methods, the least squares estimate (LSE) and the enumeration estimate (EE). For the detailed description of these two methods please see [21].

<u>Example 1</u>. Given the normalized DCM A and vector $\boldsymbol{F}$ as follows:

$$A = \begin{bmatrix} 0.6 & 0.1 & 0.2 \\ 0.2 & 0.3 & 0.8 \\ 0.2 & 0.6 & 0.0 \end{bmatrix} \qquad \boldsymbol{F} = \begin{bmatrix} 20 \\ 7 \\ 1 \end{bmatrix}$$

$$L'(p(\boldsymbol{d})) = \prod_{i=1}^{m} p(f_i)^{F_i}$$

where F_i is the number of occurrences of fault f_i. We define the log-likelihood function

$$L(p(\boldsymbol{d})) = \frac{1}{\sum_{i=1}^{m} F_i} \log L'(p(\boldsymbol{d}))$$

$$= \frac{1}{\sum_{i=1}^{m} F_i} \sum_{i=1}^{m} F_i \log p(f_i)$$

$$= \sum_{i=1}^{m} p(F_i) \log p(f_i)$$

where $p(F_i) = F_i / \sum_{i=1}^{m} F_i$. Since maximizing the log-likelihood function maximizes the likelihood function, our objective can be to find the mixture coefficients $p(d_1)$, $p(d_2)$,..., $p(d_n)$ that maximize either the likelihood or the log-likelihood function.

Maximum likelihood estimation based on such a mixture model puts our defect class estimate within the framework of standard statistical theory. The gradient ascent iterative schemes[1][11][16] and Expectation-Maximization (EM) algorithm[12] are two commonly used techniques to solve this mixture proportion estimate problem. The iterative algorithm for our defect class estimate problem is based on standard EM method. That is, on the γth iteration, we choose the new update $p^{(\Upsilon+1)}(d_j)$ as

$$p^{(\Upsilon+1)}(d_j) = p^{(\Upsilon)}(d_j) \nabla L\left(p^{(\Upsilon)}(\boldsymbol{d})\right)_j$$

where $\nabla L\left(p^{(\Upsilon)}(\boldsymbol{d})\right)_j$ is the *j*th element of the row vector $\nabla L(p(\boldsymbol{d}))$ at probability vector $p^{(\Upsilon)}(\boldsymbol{d})$, and $\nabla L(p(\boldsymbol{d}))$ is the gradient of partial derivatives with respect to the elements of $p(\boldsymbol{d})$. That is

$$\nabla L(p(\boldsymbol{d})) = \left(\frac{\partial}{\partial p(d_1)} L(p(\boldsymbol{d})), \frac{\partial}{\partial p(d_2)} L(p(\boldsymbol{d})), \ldots, \frac{\partial}{\partial p(d_n)} L(p(\boldsymbol{d}))\right)$$

$$= \left(\sum_{i=1}^{m} \frac{p(F_i)\, p(f_i|d_1)}{p(f_i)}, \sum_{i=1}^{m} \frac{p(F_i)\, p(f_i|d_2)}{p(f_i)}, \ldots, \sum_{i=1}^{m} \frac{p(F_i)\, p(f_i|d_n)}{p(f_i)}\right)$$

Our ML estimation algorithm starts with the initial estimates of the probabilities of defect classes $p^{(0)}(d_j)$ for *j=1, 2,..., n*, and then updates the $p(d_j)$ estimates by iterating the following two steps:

problem by using a mixture model. Mixtures of distributions, in particular normal distributions, have been used extensively as models in a wide variety of important practical situations where data can be viewed as arising from two or more populations mixed in varying proportions. The mixture model is given by[1]

$$f(x) = \sum_{i=1}^{t} p_i f_i(x)$$

where $f(x)$ is a mixture probability density or distribution function, $f_1(x)$, $f_2(x)$, ..., $f_t(x)$ refer to t component probability density or distribution functions, and p_1, p_2, ..., p_t are t proportions or coefficients with constraints $p_i \geq 0$ and $\sum_{i=1}^{t} p_i = 1$. The usual mixture estimation method is the resolution of a mixture into its separate components on the basis of sample data.

Now we use the above mixture model to describe our defect estimate problem. First let $p(f_i)$, $i = 1, 2,..., m$, denote the probability of a fault f_i, $p(d_j)$, $j = 1, 2,..., n$, denote the probability of a defect class d_j. Then we have the mixture model

$$p(f_i) = \sum_{j=1}^{n} p(d_j)\, p(f_i|d_j)$$

in which $p(f_i)$ refers to the mixture's probability distribution, $p(f_i|d_j)$ refers to component probability distribution, which is known to us, and $p(d_j)$ refer to the coefficients, Thus $p(d_j) \geq 0$ and $\sum_{j=1}^{n} p(d_j)$=1. Recall that $\sum_{i=1}^{m} p(f_i|d_j)$ =1, thus we have

$$\begin{aligned}\sum_{i=1}^{m} p(f_i) &= \sum_{i=1}^{m}\sum_{j=1}^{n} p(d_j)\, p(f_i|d_j) \\ &= \sum_{j=1}^{n}\left(p(d_j) \sum_{i=1}^{m} p(f_i|d_j) \right) \\ &= \sum_{j=1}^{n} p(d_j) = 1\end{aligned}$$

Hence in the above mixture model we have component probability distributions given and only the mixture coefficients are unknown. Our goal is to find the mixture coefficients $p(d_1)$, $p(d_2)$,..., $p(d_n)$ which maximize the likelihood of the sample under the mixture distribution $\sum_{j=1}^{n} p(d_j)\, p(f_i|d_j)$. To get this, we denote $p(\boldsymbol{d}) = (p(d_1), p(d_2), \ldots, p(d_n))^T$ and define the likelihood function L' to be

$$p(f_i|d_j) = \frac{p(f_i, d_j)}{p(d_j)} = \frac{p(f_i, d_j)}{\sum_{k=1}^{m} p(f_k, d_j)}$$

Thus we have $\sum_{i=1}^{m} p(f_i|d_j) = 1$. Note that $p(f_i|d_j)$ is the probability of fault f_i if a *critical* defect in defect class d_j occurs.

Our goal in this work is to find the most likely distribution of spot defects in the fabrication line based on the observed fault behaviors. That is, the maximum likelihood estimators (MLE) of the underlying probability distribution that cause ***D***, given the DCM and the occurrence numbers ***F***. This is similar to finding the MLEs for ***D***. Please note that ***D*** can be used to estimate defect densities. If we let A be the normalized DCM, that is

$$A = \begin{bmatrix} p(f_1|d_1) & p(f_1|d_2) & \dots & p(f_1|d_n) \\ p(f_2|d_1) & p(f_2|d_2) & \dots & p(f_2|d_n) \\ \dots & \dots & \dots & \dots \\ p(f_m|d_1) & p(f_m|d_2) & \dots & p(f_m|d_n) \end{bmatrix}$$

then we can express the resulting stochastic process in terms of ***D*** by using matrix notation as followings:

$$A\boldsymbol{D} = \mathrm{E}[\boldsymbol{F}] \tag{1}$$

That is, if given matrix A and vector ***D***, it is expected to produce vector ***F***, the number of occurrences of the observable fault behaviors ***f***. E[***F***] denotes the expected values of ***F***. Note that some elements in matrix A may be zero. That is, if existence of a defect class d_j cannot cause occurrence of a fault f_i, then the corresponding $p(f_i|d_j) = 0$. Further note that we are assuming that only one defect is present in a single cell. That is, $d_1, d_2, \dots, d_n$ are mutually exclusive in the presence of a fault behavior. This is a reasonable assumption for small cells such as SRAM cells. Thus we have

$$\sum_{i=1}^{m} F_i = \sum_{j=1}^{n} D_j$$

Based on the above formulation of the problem, we propose a statistical method to solve the estimation problem in the next section.

3 Maximum Likelihood Estimation

In this section, we introduce an ML estimation method to solve our defect class estimation

These defect classes are indistinguishable using their fault behaviors. This reduces diagnosis resolution. Those faults that caused by the same defect classes are combined into the same row. This has no affect on diagnosis ability. The resulting reduced matrix is called *defect classification matrix* (DCM). If *n* different defect classes may cause *m* different fault behaviors, the corresponding DCM is shown in Fig.2.

$$\begin{array}{c} \\ f_1 \\ f_2 \\ \cdots \\ f_m \end{array} \begin{array}{c} \begin{array}{cccc} d_1 & d_2 & \cdots & d_n \end{array} \\ \begin{bmatrix} p(f_1, d_1) & p(f_1, d_2) & \ldots & p(f_1, d_n) \\ p(f_2, d_1) & p(f_2, d_2) & \ldots & p(f_2, d_n) \\ \ldots & \ldots & \ldots & \ldots \\ p(f_m, d_1) & p(f_m, d_2) & \ldots & p(f_m, d_n) \end{bmatrix} \end{array}$$

Figure 2. The defect classification matrix (DCM)

Our goal in this work is to determine which defect classes are the most likely causes of the observed faults given the DCM and the number of occurrences of each fault in a wafer or series of wafers. That is, we must translate the results of observed fault behaviors into the most likely probability distribution of the defect classes. Due to the random nature of structural defects in an IC and the fact that multiple defect classes can cause the same fault, the problem of finding the most likely distribution of defect classes must be solved as a statistical optimization problem.

Let f_i denote a fault behavior, d_j denote a defect class, F_i denote the number of occurrences of the fault f_i, and D_j denote the number of occurrences of the defect class d_j. Suppose there are *m* different fault behaviors $\boldsymbol{f} = (f_1, f_2, \ldots, f_m)^T$, and the number of occurrences of $\boldsymbol{f}$ is $\boldsymbol{F} = (F_1, F_2, \cdots, F_m)^T$. Throughout this paper we assume that a vector is a column vector. $(\ldots)^T$ denotes the transpose of a vector, and we use bold letters to denote vectors. Let $\boldsymbol{d} = (d_1, d_2, \ldots, d_n)^T$ denote the *n* different defect classes that may cause the *m* faults. The number of occurrences of defect classes $\boldsymbol{d}$ is $\boldsymbol{D} = (D_1, D_2, \ldots, D_n)^T$. The entry in the DCM, $p(f_i, d_j)$, for $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, is the joint probability of a fault f_i and a defect in defect class d_j. We can obtain this information by means of the Inductive Fault Analysis tool Carafe and circuit simulator tool SPICE introduced in the last section. By using Bayes' formula, we get the conditional probability

fault is called a *critical defect*. Throughout this paper the term defect implies a critical defect. The term *fault behavior*, or *fault*, is used to describe the behavior of a circuit with an electrical fault. Stuck-at faults, transition faults, state coupling faults, data retention faults, high quiescent power supply current, or any combination are examples of faults in a SRAM. A fault occurs as a result of one or more defects. For example, a defect that bridges two metal layers and causes an electrical short may cause a fault behavior of excess quiescent power supply current when the two metal layers have different voltages applied to them in the fault-free circuit. It is clear that usually more than one class of defect may cause the same electrical fault and hence the same fault behavior.

To correctly perform defect diagnosis, we first need mapping information between defect to fault behavior. Carafe[2], an IFA software package, is an ideal tool to obtain defect-to-electrical fault mapping information. It simulates the results of spot defects on the physical layout of the IC to generate a list of electrical faults with the probability of the occurrence of each electrical fault, based on the manufacturing process specifications and defect attributes. A circuit simulation of each electrical fault is run using SPICE to determine the fault behavior caused by each electrical fault. This program flow is shown in Fig.1.

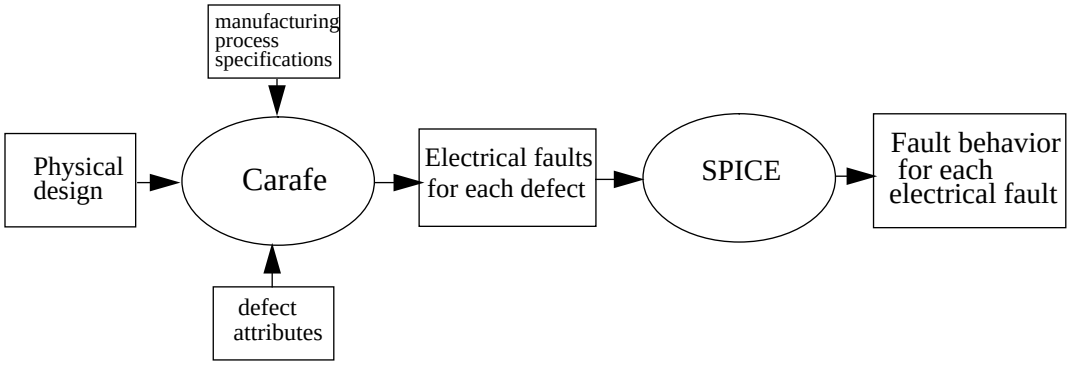

Figure 1. Program flow from physical design of circuit to fault behaviors

The result of above program flow is defect-to-fault mapping information and the probability that a fault occurs due to the presence of each defect class. These can be organized into a matrix where each row represents one of the faults and each column one of the defect classes. The entry in the *i*th row and the *j*th column represents the joint probability of a fault *i* and a defect from defect class *j*. We combine defect classes with the same fault distribution into the same column.

The main drawback of the above mentioned approaches is that they perform defect diagnosis based on the assumption that a fault behavior is caused only by single defect type, or that the defect type with the greatest critical area causes the observed fault behavior. The defect diagnosis method described in this paper can be applied to the situation in which a single fault behavior can be caused by multiple defect types. The remaining part of this paper is organized as follows: The problem is formulated in section 2. Section 3 describes the ML estimation procedure. Section 4 considers the conditioning of the ML estimation. An experimental example is illustrated in section 5. Finally, section 6 gives a summary.

2 Problem Formulation

In this paper we use the following terminology. *Defects* are deviations from the ICs' physical design caused by perturbations or contaminants in the manufacturing process. We assume that the IC design is correct. According to Jalubowski et al. [13] three types of IC chip defects affect the yield: 1) Structural defects, 2) Parametric defects, and 3) Gross manufacturing errors. The gross manufacturing errors result from serious operator's errors, or from major equipment break downs. Such errors are easily diagnosed and will not be discussed here. Parametric defects are errors in manufacturing process or improper silicon substrate parameters. The presence of this type of defect affects all chips in a large area of the silicon wafer. Parametric defects are usually easily detected and diagnosed since they occur to a large portion of the IC. Structural defects, also called random defects, are changes in the physical properties of an IC in a small physical area. They may be caused by particulate contamination on silicon wafers and photomasks, or local defects of silicon substrates. The number of structural defects may be large even in a stable and well organized manufacturing environment. These defects may cause shorts between conducting regions or across the dielectric layers, breaks in the conducting layer, excessive leakage currents of p-n junctions, or excessive resistance of the contact between conducting layers. In this paper, we only consider structural defects. The term *defect class* in this paper will be used to refer to a group of same type of defect. For example, all shorts between metal1 and metal2 are caused by defects in a single defect class and all shorts between metal1 and poly are caused by defects in another defect class.

An *electrical fault* is the change in the electrical description of the circuit due to a defect, such as a short circuit, an open circuit, or missing transistor. A defect that causes an electrical

answering a sequence of questions that is supposed to eliminate unlikely reasons. An important feature of this classification is that nodes (that is, reasons for yield losses) at the higher levels of this hierarchy are easy to classified while nodes at the lower levels require special analysis to distinguish between the various groups of reasons which can provide more detailed information than those at the higher levels. This approach can diagnose all categories of defect and process instabilities. The problem is that it requires considerable certainty or experience to use this methodology.

Kwon and Walker [8] presented a method of estimating defect densities through the use of production functional test data. Their method first generates a POF (probability of failure) matrix using defect and fault simulations with a stop criterion such as failing test pattern number, then collects failure patterns (the combinations of failing test patterns) and their counts of occurrences from production functional testing with the same stopping criterion. The resulting defect densities are derived using least squares fit. This procedure is similar to the method we present in this paper. The advantage of their method is that it can use the product itself as a defect monitor, thus saving silicon area. The disadvantages are the sample size (the number of unique failure patterns) depends on the complexity of the product, and the results derived by using a least squares fit may be not accurate as we show later.

S.Naik et al. [9] and J.Khare et al. [10] developed a failure analysis method for CMOS SRAMs using realistic defect modeling and results of functional and I_{DDQ} testing. Their method first constructed a defect-to-bitmap signature vocabulary or dictionary (a bitmap signature is actually a pattern of failing bits in the memory core) using simulation techniques, then the signature derived from SRAM testing was used to match a predetermined pattern in the developed defect-to-signature vocabulary that corresponds to the defect type causing the fault. This methodology was shown to be efficient in their experiments. However there are some problems. First, there is great possibility that a signature generated during actual diagnostic testing may not find a matching signature in the pre-stored vocabulary. Second, it is inevitable that more than one defect may have the same signature in this limited vocabulary. Third, their method excluded from the vocabulary all faults which affect single cells since they have the same bitmap signature. This discarded source of information would otherwise be very useful for revealing the most likely defect types causing yield loss if the tests were supplemented in the ways described in this paper.

to physically deprocess the SRAM chip to find the defect that causes the SRAM to fail. This can be done by determining the possible electrical behaviors of the SRAM in the presence of classes of defects so that for these defects it is unnecessary to physically deprocess the chip if there is only a single defect type that can cause the observed electrical behavior. How to use IFA software to do this can be found in Lepejian et al. [4]. The approach described in this paper extends this methodology to cases in which a single fault behavior may have been caused by multiple defect types. That is, by using the maximum likelihood (ML) estimation, it can be determined which defect class is most likely to have caused a fault when several defect classes are possible candidates. The presence of these defects are therefore more quickly diagnosed so that the yield can be improved more cost effectively.

The topic of process diagnosis has been discussed in a number of papers. Here we briefly review recent research on structural defect diagnosis that is pertinent to this paper.

W.Lukaszek et al. [5] and E.M.J.G.Bruls et al. [6] described how to design test structures for the purposes of CMOS process diagnosis and monitoring in the development or pilot production stage of product evolution. Test structures are specially designed process monitors which are fabricated by the same process used to fabricate IC products. Two types of test structures are commonly used in the industry: comb structure, for detecting shorts, and string structure, for detecting opens. Usually these two structures are combined to detect both short and open circuits. Test structures can be arranged in several ways based on their purpose. They may be implemented and collected in test chips which then replace some of the production chips on a wafer, or in the development phase of process, they are implemented on process validation wafers which are consist of test chips only. Defect diagnosis using test structures can be easy and quick due to the simple regular structures of these test vehicles. The main drawback of using test structures for defect diagnosis is that one type of test structures can be used to analyze only one specific defect class. Thus a set of test structures is needed for the whole process flow with each type of test structure used for a specific step in the process flow. Therefore, large silicon area is used to implement test structures which is undesirable in a manufacturing process, especially in a mature phase.

W.Maly et al. [7] used a hierarchical classification of the reasons causing yield losses to conduct yield diagnosis. The classification has the form of a tree in which each node represents a group of reasons with certain common characteristics. The yield diagnosis was performed by

Maximum Likelihood Estimation for Failure Analysis of SRAM Cells Using Inductive Fault Analysis

F. Joel Ferguson Jianlin Yu
fjf@cse.ucsc.edu jianlin@cse.ucsc.edu
Department of Computer Engineering
University of California
Santa Cruz, CA 95064

1 Introduction and Review of Previous Work

The yield of an Integrated Circuit (IC) fabrication line is the percentage of the manufactured ICs that are good. It is the most comprehensive measure of quality of the IC manufacturing process. One of the major goals of semiconductor manufacturers is to increase the yield in their fab lines. This increases their profit by increasing the revenue to manufacturing-cost ratio.

Due to imperfections of the manufacturing process, various defects are present in ICs. To increase the factory's yield, engineers must know in which phase of the manufacture of the ICs the defects are being introduced so that the causes of the defects can be eliminated or reduced. This is traditionally done by isolating the defects' location on the IC and destructively removing portions of the chip in that location until the defect is found. This is called *deprocessing* the chip. After the defect is found it can be determined which process step it was introduced and which machines need to be adjusted or cleaned to raise the yield. Since defects are often smaller than a micron (1/1000 of a millimeter) in diameter this is a very time consuming and costly process, and sometimes the defect is not found during deprocessing.

Static Random Access Memories (SRAMs) are often used as process monitors since the location of the defect can usually be isolated to a row or column in the IC, or to a single SRAM cell. This reduces the time spent looking for the defect considerably and thus reduces the cost of failure analysis. But it doesn't totally eliminate the disadvantages of deprocessing. Inductive Fault Analysis (IFA) software coupled with other analysis software can reduce the frequency of having

Contents

Maximum Likelihood Estimation for Failure Analysis of SRAM Cells Using Inductive Fault Analysis

F.Joel Ferguson
Jianlin Yu

UCSC-CRL-96-08
March 5, 1996

Board of Studies in Computer Engineering
University of California at Santa Cruz
Santa Cruz, CA 95064

ABSTRACT

This paper presents an iterative maximum likelihood (ML) estimation method for statistical analysis of yield loss. By means of Inductive Fault Analysis (IFA) and circuit simulation, the map between defects and corresponding fault behaviors can be constructed for process-monitor SRAMs. Using the data from a tester describing the number of times each fault behavior occurs, the most likely causes of low yield can be identified automatically using the approaches presented in this paper without the need for physically deprocessing the defective SRAMs. To our knowledge the application of ML method using a mixture model on yield diagnosis has not appeared before in the literature.

Keywords: Maximum likelihood (ML) estimation, Inductive Fault Analysis (IFA), defect, fault, probability distribution, mixture model, Hessian matrix, Fisher Information matrix