SRAM Bitmap Shape Recognition and Sorting Using Neural Networks

Article *in* IEEE Transactions on Semiconductor Manufacturing · September 1995

DOI: 10.1109/66.401009 · Source: IEEE Xplore

3 authors, including:

Randy Collica
SAS Institute
14 PUBLICATIONS **82** CITATIONS
SEE PROFILE

Jill Card
Independent Researcher
14 PUBLICATIONS **76** CITATIONS
SEE PROFILE

SRAM Bitmap Shape Recognition and Sorting Using Neural Networks

Randal S. Collica, *Member, IEEE,* Jill P. Card, *Member, IEEE,* and William Martin

Abstract----This paper details the use of neural network technologies in the characterization of bit fail patterns occurring on SRAM chips as an alternative to the more traditional rulebased or knowledge-based approach to fail pattern occurrence and classification analysis. The results of bit fail pattern count analyses are used both for fault analysis post-processing and manufacturing yield improvement methodologies. The move toward neural network implementation comes in response to prohibitively long processing times required for implementation of role-based algorithm on more complex devices and the added flexibility of a neural network to learn new fail types in a more adaptive mode. An unsupervised approach to fail pattern identification was implemented on a 128 K SRAM chip using a two-layer Kohonen Self Organizing Map for identification and concurrence of bit fail pattern categories within SRAM chips. A second network utilized a multilayer perceptron (MLP) architecture with backpropagation of error for prediction of the number of occurrences per bitmap of each of the 34 previously identified shape types. The MLP used the output of a SOM as its input vector to assist in the feature extraction by shape type. Both trained networks out-performed existing role-based algorithms both in ability to identify bit fail pattern types, frequency counts, and speed of processing.

I. INTRODUCTION AND BACKGROUND

SEMICONDUCTOR professionals have used bitmaps of SRAM and DRAM chips for quite some time for process control, defect monitoring. yield improvement, and redundancy analysis [1]-[4]. Typically, the post processing of the bitmap data files are in the form of a rule-based or knowledge-based algorithm type for the sorting of different bit fail patterns on the chips [5]-[7]. When the memory device being bitmapped approaches 1 Mbit and beyond, the computer resources of a rule or knowledge-based algorithm become unmanageable. This is true not because the number of bit failures to classify is frequent, but because the number of different combinations of fail patterns is large. The CPU and elapsed time required for a post processing program to sort the frequency count for each bit fail category on every chip from a single lot could take a hour or more; therefore, this is typically impractical for analyses containing a large numbers of wafers. Also, when a rule or knowledge-based algorithm is used, only patterns programmed into the source code are categorized. Any new patterns will either be misclassified or not counted. This is a serious shortcoming of all rule or knowledge-based programs for bit failure classification.

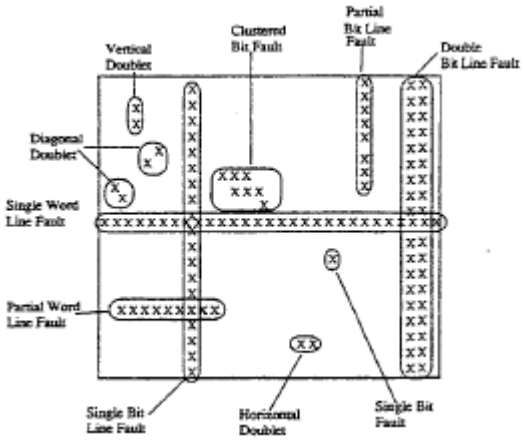

Fig 1. Bitmap of SRAM failure patterns.

This paper proposes a neural network structure for bit fail pattern classification and counting. There has been a great deal of work recently using neural networks for pattern recognition/classification applications such as this, both inside and outside the semiconductor industry. Within the semiconductor industry, neural networks have been used both in the control of the manufacturing process and for the inspection of silicon wafers [15]-[26]. The neural network described herein was developed as a feasibility study on 128 kbit SRAM chips with excellent success. The error of misclassification is small, the time required to process the data is less and the neural network has better resolution at defining categories than the rule-based approach

II. BIT FAIL PATTERN CLASSIFICATIONS

Fig. I shows a typical bit fail map containing the types of patterns analyzed with a rule-based pattern recognition technique. The patterns formed are from a logical 'OR' of several bit map vector tests. The stuck-at-0 and stuck-at-1 information is not usually included due to physical disk space limitations in the data collection. These patterns are typically very useful given that the layout artwork of the SRAM memory cells

lend themselves to a few defect mechanisms for each bit fail pattern or combination of patterns [8]. Analysis of these bit fail patterns can be simulated using an inductive fault analysis technique with Monte Carlo defect simulation as in the VLASIC software package [10]. This simulation results in a probability of failure of a certain defect type causing one of the bit fail patterns.

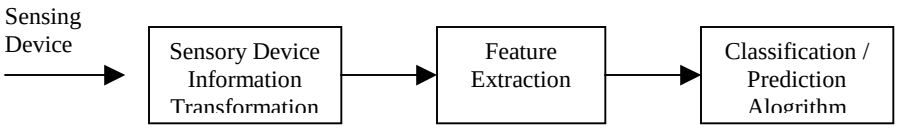

Fig 2. Flow diagram of patter recognition analysis tasking.

SRAM contain 1.024 million bits stored internally to the program for each chip. Each time a pattern is classified, associated bit coordinates are removed from the memory map's virtual array. The program continues classifying patterns in this manner until all patterns have been accounted for on the SRAM chip.

III. USES OF BIT FAILURE PATTERN DATA

The primary use of the bit fail data is for yield improvement. This is accomplished in several ways. Initially, when the wafers are SRAM tested, the bit fail raw data is collected and run through a feature extraction processing program. Following feature extraction, a classification of how many bit fail fault exists on each die capable of classification is then compiled for further analysis. The number of die affected by each pattern fail type is tabulated and running control charts of the proportions of die affected are reviewed. A pareto of bit fail types is used to determine which bit fail pattern to go after first. A combination of chemical deprocessing and analyses is used to determine the root cause of the failures. Then it is eliminated or it's occurrence in manufacturing is reduced to acceptable levels. This process is repeated for the top few bit fail patterns in the pareto chart until the desired level of yield is obtained [8].

IV. OBJECTIVES

The primary objective of this feasibility study was to demonstrate that neural networks can provide an accurate, time-efficient method of classifying and reporting SRAM defect patterns. The time efficiency is a primary concern. As SRAM's increase in their size and complexity, so does the likelihood for more varied and complex combinations of defects. This can greatly increase the time and computing resources necessary to analyze these defects. However, this time efficiency cannot come at the expense of accuracy in the classification and reporting. The specific accuracy goal is that the neural network classifier needs to be at least as accurate as a human trained to analyze these defect patterns. In addition to accurately classifying and reporting the numbers of individual shapes present on each die, we want the neural network to output information regarding the interactions between the different types of shapes. Certain types of processing faults may cause an increase in the frequency with which a particular combination of shapes occur, so tracking interactions between shapes may provide useful information for process fault identification. An additional objective was that the neural network should have the ability to recognize a novel bitmap pattern as being an event never before seen rather than force-fitting the pattern into one of a pre-determined set of categories.

Analysis of the bit fail pattern proportions allows one to perform inductive defect diagnosis [8]-[10]. Using a rule-based approach to classification and counting leads to some very real practical problems. For a rule-based application, bit fail pattern classifications on a 1 Mbit

V. APPROACH

The analysis of a pattern recognition problem generally follows a task flow as in Fig. 2, where physical phenomena are transformed by a sensing device into output data suitable for analysis. The data then undergo a feature extraction transform designed to enhance the pattern information resident within the data for performance of the categorization task. The feature extraction results in a reduction in feature space dimensionality or data compression. The output feature vector maintains the minimal information required to perform the categorization while eliminating all other dimensionality to both speed data handling and increase the probability of correct neural network success in pattern recognition [11]. This reduction in dimensionality is not unique to neural network, however. The feature extraction and shape classification steps are described below.

VI. FEATURE EXTRACTION

The primary task in the feature extraction is to perform data reduction. Each die consists of 128 kbits of binary data (1 = failed bit, 0 = working bit). We seek to reduce the computational complexity of the neural network while maintaming pattern shape information on the die in as few features as possible. The feature extraction must be computationally efficient to maintain speed of final application. The feature extractor operates in three steps.

The first step is to perform a shape analysis on the failed bits on the die. This is based on physical proximity of failed bits to one another using a fixed threshold criteria of distance between the bits. Two failed bits within five bits of each other are considered to be of the same shape. The selection of five as the threshold was based primarily on trial and error. Using an initial threshold of zero, a shape was comprised of all adjacent failed bits. However, in this case, the neural network was unable to detect certain types of partial columns and rows that had one or more good bits inside [Fig. 3(a)]. The other extreme was to set the threshold at sixteen bits. This resulted in groups of visually and fault analytically different shapes being considered a single shape (i.e., two full columns at 8 bits apart are seen as one shape) [Fig. 3(b)]. The choice of five turned out to be a reasonable compromise.

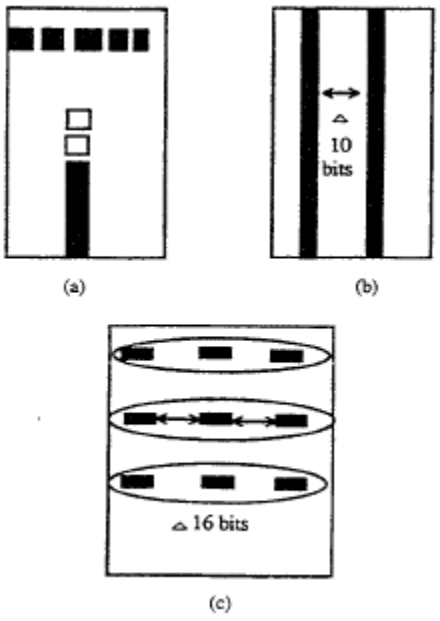

Fig. 3. Shape type identification by bit proximity thresholding and lot row/column coverage. (a) Examples of partial row and partial column bitmap shapes; these distinct shape types are not detected as 1 row and 1 column respectively when a proximity threshold of 0 is used. The row shape would be recorded as 5 distinct shapes, the columns as 3 shapes. (b) Using a proximity threshold of 16 records the above 2 full columns as a singular shape. The large threshold value here clouds the engineering significance of a 2 shape image. (c) Bitmaps exhibiting repeating partial rows where repeat patterns occur 16 bits apart but have a sole fault source should be recorded as a single shape. The above bitmap shows 3 such partial row shapes

Following establishment of shapes within the bitmap, the aspect ratio (ratio of height to width) and the total number of failed bits per shape were computed. The aspect ratio provides information not only on the relationship between the shape's height and width but also on its orientation (vertical or horizontal) which is needed to distinguish rows from columns. The total number of failed bits in the shape provides information on both the size of the shape and its density.

The third and final step in the feature extraction is to bin the shapes based upon these two statistics. The extractor builds a 7 x 6 two-dimensional histogram of aspect ratio and number of failed bits with each cell in the histogram containing the number of shapes on the die which had a particular aspect ratio and number of failed bits. The 7 aspect ratio and 6 total bit count intervals were not evenly spaced. The transition points were based on apparent critical cutoff points for detecting the particular shapes that we knew *a priori* would be present in the data. For example, an aspect ratio of one with one failed bit would have to be a single bit failure and an aspect ratio of 64 with 64 failed bits would have to be a single column failure. These 42 feature vector inputs could not identify a small number of shapes known as repeating rows [Fig. 3(c)], due to the absence of information which relayed the relative positions of the shapes one to another. This issue was resolved by adding two additional components to the feature vector: the number of unique rows and the number of unique columns that contained failed bits. This brought the final length of the feature vector to 44 elements.

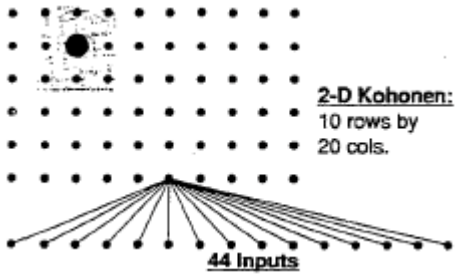

Fig 4. SOM network.

More sophisticated image processing techniques could have been used and have been in similar applications [25], [26), however, the simple technique used above operates very quickly and produces good results.

A. Defect Pattern Classification

Following the feature vector definition two neural networks were designed to perform the pattern classification. The first network architecture used is a Kohonen Self-Organizing Map which is an unsupervised network (i.e., the *correct* answers are not provided to the network by a teacher or supervisor). The network organizes like input vectors together based on a determined distance metric. This network performs a particular type of shape analysis on the input vectors which, after training, permitted examination of the concurrent appearance of several shapes within a individual bitmaps. This network also provides a method for classifying patterns that are unlike those that have been observed in the training set.

The second network architecture developed was a Self-Organizing Map feeding into a backpropagation network. This network, which has both an unsupervised and a supervised portion, is used to output the actual counts of shapes that appeared on a die. For each die, this network outputs the number of single bit failures, the number of single column failures, the number of row failures, etc., present on the die. Both of these networks are discussed in greater detail below.

B. Kohonen Self-Organizing Map

The Self-Organizing Map (SOW) was developed by Teuvo Kohonen and is fully detailed in [12}. The architecture of the Kohonen SOM used for this application is shown in Fig. 4. At the bottom of the figure are 44 input neurons which correspond to the 44-element input vector The input layer feeds into a 200-element Kohonen layer arranged in a 10 row by 20 column two-dimensional grid. Each of the 200 elements in the Kohonen layer have weighted connections to all 44 input nodes.

It is most intuitive to consider these weights as a prototypical input vector to which the neuron will respond. When presented with an input vector, the SOM computes, for each neuron in the Kohonen layer, a Euclidean distance between the input vector and its prototype. The neuron that has the smallest such distance is determined to be the *winner*. The winning neuron outputs a value of one while the remaining Kohonen neurons output a zero value. This is typically referred to as a *winner-takes-all* strategy. During training, input vectors are selected at random from the training set and presented to the SOM and a winning node determined by the method described above.

The winner's prototype vector is adjusted to be closer to the input vector by updating the node weights as follows: Let W be the *j*th element in the weight vector of the *i*th neuron in the Kohonen layer and X_j, be the *j*th element in the input vector. The new value of W_{ij}, W'_{ij}, is given by

$$W'_{ij} = W_{ij} + \alpha(X_j - W_{ij}) \qquad (1)$$

where α is the learning coefficient or step size parameter. Typically α starts out at a moderate value and decreases as the number of training iterations increases.

This simple weight update scheme poses a problem whereby a small number of neurons are repeat winners, the weights adjusted moving them far from the remaining neuron values, further increasing the likelihood of their winning. The remaining neurons are never updated, thereby reducing the overall effectiveness of the SOM. This problem is avoided by introduction of a *conscience* mechanism which prohibits any one neuron from winning too often. Computationally this is accomplished by computing a bias, B_i, for each neuron based upon the winning frequency of the neuron.

$$B_i = \gamma(1/N - F_i) \qquad (2)$$

where F_i is the winning frequency of the *i*th neuron, *N* is the number of neurons, and ◊ is a selectable conscience coefficient This bias is subtracted from the Euclidean distance of the neuron to compute an adjusted distance. This adjusted distance is then used to select the winner rather than the actual distance. If ◊ is selected to be a high value, say 10, then the SOM will spread out rapidly. If ◊ is selected to be a small value then the SOM will tend to spread out more slowly. It is typical in practice to reduce ◊ as the number of training iterations increases.

To compute the bias, the winning frequency of each neuron must be tracked. In practice, the following approximation to the actual winning frequency is used:

For the winning neuron: $F'_i = F_i + \beta(1.0 - F_i)$. (3a)

For all other neurons: $F'_i = F_i - \beta F_i$. (3b)

where F_i, and F_i are the old and new values of the winning frequency respectively. β is a parameter that is selectable during training. Like the other two coefficients, β is typically reduced as the number of training iterations increases.

The SOM also adjusts the prototype vectors of the winner's immediate *neighbors* closer to the input vector using the same algorithm as above. The winner's immediate neighbors can be determined in many ways. For this application, we used a square 3 x 3

neighborhood as depicted in Fig. 4. However the neighbors are determined, adjusting their weight vectors along with that of the winner has two major effects. The first is that neurons in the Kohonen layer which have similar weight vectors tend to be close to one another in the final map while neurons with very dissimilar weight vectors tend to be farther away from one another. This creates topological regions in Kohonen layer. The

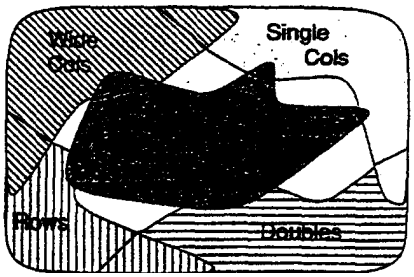

Fig. 5. Initial topographic map for shape type identification.

The second major effect of updating the neighbors' weight vectors is that it creates competition among the neurons. For example, for any given neuron, the left neighbor pulls the weight vector in one direction while the right neighbor pulls the weight vector in another direction. One result of this competition is that neurons who never *win* during the training still have their weight vectors set up to be somewhere between its neighbors. In the final application, this permits the neuron to respond to input patterns that have not been seen by the SOM during the training period.

1) Results: The SOM was trained using approximately 600 die iteratively while adjusting the coefficients α, β ,*and* γ until the neurons in the SOM *spread out* sufficiently so that the SOM classifications were in agreement with the visual shape perception and assumed fault analysis shape patterning. The initial α, β, *and* γ coefficient values were 0.2, 0.0004, and 5, respectively, with incremental decrement through 20 000 iterations to final values of 0.1, 0.0001, and 1.0. A simplified version of the final topographic map is shown in Fig. 5.

Fig. 5 shows the regions of the SOM that respond when a die presented contains either single columns, single bit failures, row failures, etc. The areas of the SOM that respond to combinations of shapes are indicated by the overlapping regions. The areas of the SOM that are outside any of the labeled regions are those that do not respond to any of the training vectors. The neurons in these unlabeled areas are available to respond to input vectors which have not yet been seen.

During the course of training the SOM and examining the resulting topographic map, additional shapes were identified beyond the original eleven that were being recorded by the rule-based analysis. This brought to 34 the total number of shape types to be tracked by the second network.

C. Self-Organizing Map with Back Propagafion Prediction Network

Given the new listing of 34 shape types, a means for counting shape occurrence per bitmap was needed. We opted to design a second supervised neural network rather than a rule-based approach to finding and counting shapes because the classification of some shape types did not fit easily into rule-based definition. For example, Fig. 6 represents a shape classified as a dense cloud. While the human supervision must maintain consistency in classification of this bitmap as a dense cloud, specific density measurement, etc. are not necessary for classification. A neural network appeared well suited to accomplish the necessary feature generalization.

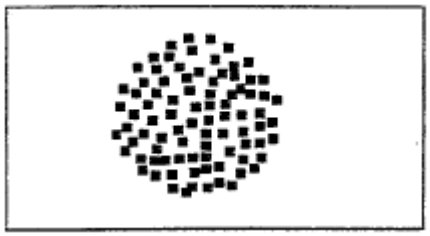

Fig. 6. Dense cloud shape type. A neural network solution generalizes the dense cloud shape accurately.

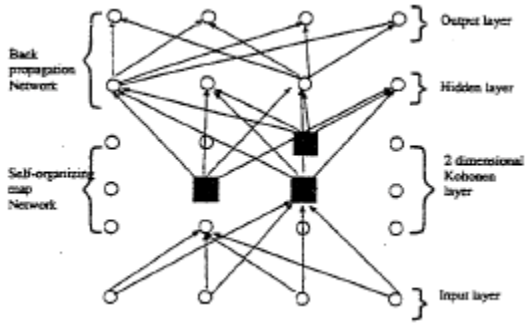

Fig. 7. Self-organizing map with backpropagation prediction network. The three dark graded squares indicate the normalized Kohenen node winners which make up the input into the backpropaganon hidden layer.

A two-tiered network was constructed consisting first of a 2-D 25 x 15 SOM similar to the network previously discussed. The SOM accepted the same 44 element input

vector as previously described. For this application of the SOM, the top three winning nodes were scaled to sum to one and used as input to the second tier network: a three-layer multilayer perceptron (MLP) with training via supervised back propagation of error. Since the SOM forms a topographic map, the input vector is represented in this scheme as a weighted average of the three top winning nodes, i.e., it falls somewhere in the triangle formed by these three nodes. Fig. 7 illustrates the two-tiered network construction. Utilizing a SOM to MLP in this fashion is equivalent to performing feature extraction of the original input vector via SOM neural network, followed by pattern classification and prediction via MLP neural network.

The choice of a combined 2-D Kohonen network followed by an MLP is similar in construction to that of a counter-propagation network. The unidirectional counterpropagation network is comprised of an input layer, a 1-D Kohonen layer and output Grossberg Outstar layer. The principal difference between the two network constructions is the 1-D versus 2-D Kohonen layer for autonomous feature extraction. Choice of the 2-D topology to preserve the order of higher dimensional input data (Fig. 5) was effective in engineering assessment of the mulltidimensional shape space intersection.

Additionally, the 2-D representation worked well with the use of three scaled output nodes from the Kohonen map in providing a physical understanding of the interpolation of the winning neighborhood in both axes for accurate prediction of the output shape count vector. Use of three nodes, rather than four or more, was based on the complexity of shape type intersections per bitmap and subsequently yielded satisfactory prediction for the application space.

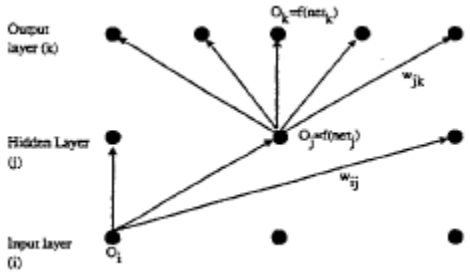

Fig. 8. Multilayer perceptron.

The SOM portion of the network is trained independently of the back propagation. Following training, the SOM weight structure is frozen and a supervised backpropagation of error training regime is employed for the MLP. The MLP is then trained until a convergence criteria for the error function is met.

A three layer MLP is depicted in Fig. 8, where o_i represent the values of the three input nodes to the MLP (the scaled SOM winners), the o_j represent the transformed values of the hidden layer nodes, and the o_k represent the transformed values of the output nodes. Each layer is fully connected to the next layer by weights. The w_{ij} weights connect the input to hidden layer nodes, and the w_{jk} weights connect the hidden to output layer nodes.

Computation of the hidden layer node values is as follows:

$$o_j = f(net_j) \tag{4}$$

where

$$net_j = \sum o_i w_{ji} \tag{5}$$

and

$$f(z) = (e^z - e^{-z})/(e^z + e^{-z}) \tag{6}$$

is the hyperbolic tangent function, where o_i is the ith input node, o_j is the jth hidden layer node, w_{ij} is the weight connecting the ith to the jth hidden node.

The output layer nodes are computed using the same equations.

When the original feature extracted 44 element vector is presented to the initialized SOM, a three node vector is computed and fed into the MLP. All MLP weights are initialized at random and a 34 node output vector ($\boldsymbol{o_k}$) is computed. Each node of the output represents the total count of the number of times a particular shape appears in the original bitmap. The backpropagation algorithm, introduced by Rumelbart [14] (with a similar algorithm appearing in the same timeframe by Porter [15]), was used to train the MLP network. A 34 node supervision vector of shape counts $\boldsymbol{d_k}$ is compared to the computed output vector $\boldsymbol{o_k}$ and a global error term computed as follows:

$$E = 0.5 \left(\sum_{k} (\mathbf{d_k} - \mathbf{o_k})^2 \right).$$

We seek to minimize the global error with respect to the hidden-to-output layer weights (w_{jk}) and with respect to the input-to-hidden layer weights *(w_{ij}).*

The change in a single weight Δw_{jk} or Δw_{ij} is determined by calculation of the gradient on the global error surface with respect to the weight ($\partial E/\partial w_{jk}$ or $\partial E/\partial w_{ij}$) and moving a constant step, referred to as the learning coefficient *(1coef),* along the opposite of that

gradient to minimize *E*. *A* momentum term *(mcoef)* is used to facilitate learning speed without resorting to use of larger *1coef* values, which assume larger local error surfaces

$\Delta w_{jk} = -1coef\,(\partial E/\partial w_{jk}) + mcoef(previous\ \Delta\ w_{jk})$. (8)

Following update of all weights w_{ij} and w_{jk}, another input vector is presented to the network, supervised output compared to computed output vector, and the update procedure repeated. The training process continues until the global error *E* meets a minimum convergence criteria $\in$, where $E \leq \in$. See Zurada [11] for details on basic backpropagation and parameter enhancements.

1) Results: The original approximately 600 die sample was split into training and test sets of 470 die and 130 die, respectively. Of the 34 defined shape types, 25 were present in the training set. Consequently, all results presented are for this category subset.

The training set was presented to the SOM. The α, β, and γ initial parameter settings in (1)-(3) were 1.2, 0.1, and 2, respectively, with incremental parameter decrements to 0.1, 0.05, and 1 at 40 000 iterations. Following SOM training, the data set was presented to the MLP via the SOM along with the known 34 node supervision vector of shape counts per bitmap. Several networks were trained, varying the hidden layer node size from 5 to 100 nodes, keeping the input and output layers fixed at 3 and 34 nodes respectively. The final network contained 25 hidden layer nodes. The *lcoef* value (8) for the hidden layer ranged from 0.4 down to 0.1 at 60 000 iterations. The output layer *lcoef* values ranged from 0.09 to 0.03. *A* constant value of 0.4 was used throughout the MLP training for *mcoef* in (8).

Fig. 9 displays the train and test set results in terms of the ±2 standard deviation (95% confidence interval) of the error between the predicted count/bitmap minus the actual count/bitmap for each of the 25 different shape categories. Results indicate that for all shape categories the predicted count differs from the actual count by less than 1 in absolute magnitude 95% of the time. In addition, with exception of the first two shape types (single bits and horizontal doublets) all other shape types show ±2 standard deviations of the (predicted - actual) count of less than 0.5. Given that the tallies of shape counts to be used as indicators of process fault types, these results are well within the precision requirements needed to provide value added.

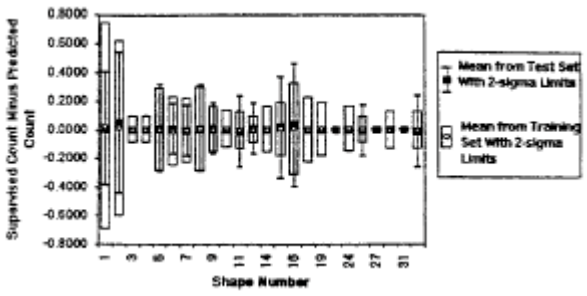

Fig. 9. Summary of test and training results.

VII. CONCLUSION

We have presented two neural networks involving the use of SOM's for identification of shape types exhibited by the 128 K SRAM bitmap. The first network organizes shapes into a topographic map allowing for the investigation of concurrent shape appearances as well as discrimination by unique shape type. Using the initial topographic map for shape type identification, we construct a second SOM which feeds a three layer MLP designed to count the occurrence of each shape type within a given bitmap for subsequent statistical rollup and analysis. The use of neural networks in this application assists the engineer in the discovery of shape types and relieves her/him of the need to classify shapes by rigid if-then-else rule-based algorithms. Finally, the use of self organization maps enable flagging for investigation of any bitmap shape-type occurring following training which had not previously been seen before.

Currently this two-network approach is being developed for 1 Mbit devices, for which the total shape types are larger and different than those of the 128 K SRAM, making a rule-based shape search difficult and inefficient. In addition, a rule-based approach on a devices of this size would be prohibitively time consuming to process compared to the SOM and MLP processing speeds of <1 s. Given the ability of this combined network approach to 1) identify new shape types by device quickly and accurately, 2) provide update on additional shapes not previously encountered, 3) provide all standard shape count rollup information to within ±0.1 deviation accurately, and 4) to provide the above in real time, we conclude that this technique represents an innovative and useful approach to engineering fault analysis in an age of both increased device size and complexity.

REFERENCES

[1] J. Khare, D. B. I. Feltham, and W. Maly, "Accurate estimations of defect-related yield loss in reconfigurable VLSI circuits," *IEEE J Solid-State Circuits*, vol. 28, no. 2, pp. 146-156, Feb. 1993.

[2] S. Kikuda, H. Miyamoto, S. Mori, M. Niiro, and M. Yamada, "Optimized redundancy selection based on failure-related yield model for 64-Mbit DRAM and beyond," *IEEE J. Solid-State Circuits* vol. 26, no. 11, pp. 1550-1555. Nov. 1991.

[3] R. Mayer, S. Lopez, and D. Bakker, "Correlating defects to bit map failures using automated patterned

wafer inspection systems," presented at the UltraClean Manufacturing Symposium, Feb. 1992
[4] C. H. Strapper, "On yield, fault distributions, and clustering of particles," *IBM J. Res. Dev.*, vol. 30, no. 3, pp. 326-338, May 1986.
[5] T. Viacroze and M. Lequeux, "Memory failure analysis using expert system techniques," *Microelectronics Manufact. Technol*., pp. 27-37, Feb. 1991.
[6] B. B. Sindahl, "Interactive graphical analysis of bit-fail map data using interactive pattern recognitions," in *Proc. Int. Test Conf*., 1987, pp 687-695.
[7] P. Mazumder and J. K. Patel. "Parallel testing for pattern-sensitive faults in semiconductor random-access memories," *IEEE Trans. Computers*, vol. 38. no. 3, pp. 394-407. Mar. 1999.
[8] R. S. Collica et al,. "A yield enchancement methodology for custom VLSI manufacturing," *Digital Tech. J*., vol. 4, no. 2, Spring 1992.
[9] P. Gangatirkar, R. Pression. and L. Rosner, "Test/characterization procedures for high density silicon RAMs," in *Proc. ISSCC*, 1982,
[10] D. M. H. Walker, *Yield Simulation for Integrated Circuits*. Boston, MA.: Kluwer Academic Publishers, 1987.
[11] J. M. Zurada, *Introduction to Artificial Neural System.* New York: West Publishing, 1992, pp. 95-96.
[12] T.Kohoncn, *Self-Organization and Associative Memory*, 2nd ed. New York: Springer Verlag**,** 1988.
[13] D. DeSieno, "Adding a conscience to competitive learning," in *Proc. Int. Conf. Neural Networks*, 1988. pp. 117-124.
[14] D. E. Rumelhart and J. L. McClelland, Eds., "Parallel distributed processing: Explorations in the microstructure of cognition," in *Foundations* (vol 1). Cambridge, MA: MIT Press. 1986.
[15] D. B. Parker. "Learning logic," MIT Center for Computational Research in Economics and Management Science, Cambridge, MA. Rep. TR-47, 1995.
[16] F. Nadi, A. M. Agogino, and D. A. Hodges, "Use of influence diagrams and neural networks in modeling semiconductor manufacturing processes," *IEEE Trans. Semiconduct. Manufact*.. vol. 4, no. 1. pp. 52-58, Feb. 1991.
[17] L. Liu and G. H. Liu. "Neural network computing applied in design for manufacturability," in *IEEE/SEMI Int. Semiconduct. Manufact Sci.* Symp., 1990.
[18] E.Rietman and E. R. Lory, "Use of neural networks in modeling semiconductor manufacturing processes: An example for plasma etch modeling," *IEEE Trans. Semiconduct. Manufact.* vol. 6, no. 4, pp. 343-347. Nov. 1993.
[19] C. B. Bose and H. A- Lord, "Neural network models in wafer fabrication." *Applicat. Artificial Neurals IV*. 1993.
[20] W. Zhang and L Milor, "A neural network based approach for surveillance and diagnosis of statistical parameters in IC manufacturing process," in *IEEE/SEMI Int. Semiconduct. Manufact. Sci*. *Symp*.. 1993.
[21] E. A. Rietman, R. C. Frye, E.R. Lory, and T. R. Harry, "Active neural network control of wafer attributes in a plasma etch process." *J. Vacuum Sci. Technol*., B, vol. 11, no. 4, 1993.
[22] C. D. Himmel and G. S. May, "Advantages of plasma etch modeling using neural networks over statistical techniques," *IEEE Trans. Semiconduct*. *Manufact.*, vol. 6. no. 2. May 1993.
[23] Y. Dai, "An application of an artificial neural network to reliability screen classification from noise measurement," *Microelectron. Reliabil.,* vol. 33. no. 4. pp. 451-453. Mar. 1993.
[24] S.S. Chen, "Intelligent control of semiconductor manufacturing processes," in *IEEE Int. Conf. Fuzzy Syst*., 1992.
[25] G. T. Stubbendieck and W. J. B. Oldham, "A fractal dimension feature extraction technique for detecting flaws in silicon wafers," in *IJCNN Int. Joint Conf. Neural Networks*, 1992.
[26] D.L. Sikka, "Two dimensional curve shape primitives for detecting line defects in silicon wafers," in IJCNN Int. Joint Conf. Neural Networks, 1992.

Randall S. Collica (S'79-M'82) received the B.S. degree in electronic engineering from Northern Aruziba University, in 1982.

He joined Mirco-Rel, a Division of Medtonic, Tempe, Arizona, in 1982, as a device engineer and then worked as a characterization engineer at Analog Devices Semiconductor, Wilmington, MA. He has been with Digital since 1987, as a principal yield engineer. His primary interests are in SRAM fault tolerance, and experimental design and analysis. In his assignments he established the use of yield models for redundancy on

advanced processor chips containing onboard cache RAM's. These yield models were used for line control and productivity optimization for current and future VLSI products. He has also had assignments in failure analysis, test chip design and analysis, and I process experimentation.

Jill P. Card (M'89) received the M.S. degree in theoretical and applied statistics from Florida State University and the B.S. degree in biology from Cornell University, Ithaca, NY, in 1975.

She has worked for the past 15 years as a statistician and applied mathematician in a broad spectrum of fields, including: medical research, biological research, reliability and quality control analysis, and most recently neural network analysis for semiconductor engineering and manufacturing. Her experience includes the following positions: Principal Engineer for Digital Equipment Corporation, Member of Technical Staff for AT&T Bell Laboratories, and Principal Engineer for Wang Laboratories, Biostatistician for Tufts University School of Medicine, and Statistics Consultant for Florida State University. She is currently with Digital Equipment Corporation, Hudson. MA.

William Martin received the B.S. degree in mathematics and the M.S. degree in computer science from Columbia University, NY, in 1980 and 1982, respectively.

He worked at the Quality Assurance Center at Bell Telephone Laboratories and Bell Communications Research from 1980 to 1988, where he built software tools for reliability modeling and analysis of telephone switching systems. Since 1988, he has worked at the Quality and Reliability department of Digital Semiconductor, where he has the responsibility for the development and introduction of statistical/mathematical techniques for q&r data analysis and decision-making.