Data Mining for Manufacturing Control: An application in optimizing IC test

Tony Fountain
San Diego Supercomputing Center
9500 Gilman Drive
University of California, San Diego
La Jolla, CA 92093
fountain@sdsc.edu

Thomas Dietterich
Department of Computer Science
Oregon State University
102 Dearborn Hall
Corvallis, OR 97331
tgd@cs.orst.edu

Bill Sudyka
Hewlett-Packard Co.
1000 NE Circle Boulevard
Corvallis, OR 97330
bjs@hpcvibs.cv.hp.com

Abstract

We describe an application of machine learning and decision analysis to the problem of die-level functional test in integrated circuit manufacturing. Integrated circuits are fabricated on large wafers that can hold hundreds of individual chips ("die"). In current practice, large and expensive machines test each of these die to check that they are functioning properly (die-level functional test; DLFT), and then the wafers are cut up, and the good die are assembled into packages and connected to the package pins. Finally, the resulting packages are tested to ensure that the final product is functioning correctly. The purpose of die-level functional test is to avoid the expense of packaging bad die and to provide rapid feedback to the fabrication process by detecting die failures. The challenge for a decision-theoretic approach is to reduce the amount of DLFT (and the associated costs) while still providing process feedback. We describe a decision-theoretic approach to DLFT in which historical test data is mined to create a probabilistic model of patterns of die failure. This model is combined with greedy value-of-information computations to decide in real time which die to test next and when to stop testing. We report the results of several experiments that demonstrate the ability of this procedure to make good testing decisions, good stopping decisions, and to detect anomalous die. Based on experiments with historical test data from Hewlett-Packard Company, the resulting system has the potential to improve profits on mature IC products.

Keywords: IC test, EM algorithm, unsupervised learning, decision theory, belief networks, real-time control.

1 INTRODUCTION

Modern computer-integrated manufacturing lines provide many opportunities for applying data mining techniques. These lines contain many sensors and computer-controlled devices, so the information needed for intelligent control is available and control decisions can be implemented easily. Furthermore, in most current computer-integrated manufacturing lines, the supply of available sensor data far outstrips the ability of the existing control systems to digest and apply it. Consequently, the combination of machine learning—to analyze data to build and update probabilistic models—and decision-theoretic control—to make decisions based on those models—can have a huge financial impact in reducing costs, increasing throughput, and raising profits.

In this paper, we report our experiences with one such application. We developed a decision-theoretic controller for one phase of the VLSI integrated circuit manufacturing process. The controller manages the die-level functional test (DLFT) process with the goal of maximizing the expected utility of the overall manufacturing process. Our approach is based on combining concepts and methods from decision theory, decision analysis, and data mining.

Our methodology consists of three steps. First, we applied our expertise in VLSI testing to develop an influence diagram for the decision-making process. The influence diagram formalism—which was developed by the decision analysis and belief network research community—allows us to describe the process in terms of three parts: (a) a cost model of manufacturing operations, (b) a probability model of the way that defects are introduced during the manufacturing and testing process, and (c) a model of the points at which decisions must be made.

The second step involves assigning numerical values to the various parameters of the influence diagram. As a result of a related project at Hewlett-Packard Company, the costs of the manufacturing operations had already been determined, so we were able to plug these into the cost model. To acquire the probability model of IC defects, we applied machine learning algorithms to learn these automatically from historical data.

The third step involved implementing a decision-making algorithm that determines which decision would maximize the expected utility of the overall manufacturing process. We implemented a one-step greedy value-of-information algorithm which approximately maximizes the expected utility.

In simulation, our final system achieves substantial profit increases over the current control method. In addition to increasing profits, the system displays several desirable features.

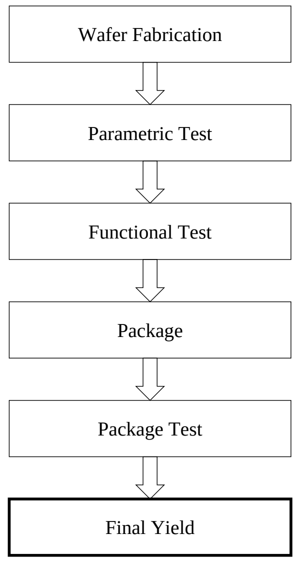

Figure 1 The major steps in IC manufacturing.

First, the system is modular, so that changes to the product under test, or changes to the manufacturing cost, can be integrated without reengineering the system. Second, the system is sensitive to product characteristics. Rather than a fixed testing policy, the system allocates testing resources to products that warrant more complex testing. Finally, the system is robust with respect to changes in the manufacturing costs. As manufacturing costs are updated, the system responds in a predictable, rational, manner.

The remainder of this paper is organized as follows. First, we describe the integrated circuit manufacturing process and the decision-making problem to be solved. Then we present our influence diagram and its probabilistic model of IC failures. The third section of the paper describes the machine learning procedure and our experimental methods. The fourth section reports the results of several experiments to understand and evaluate the behavior of the system and its various components. We summarize our conclusions in the final section.

2 IC MANUFACTURING AND TEST

An integrated circuit (IC) is an electronic circuit in which a number of devices are fabricated and interconnected on a single chip of semiconductor material. According to current manufacturing practice, integrated circuits are produced en masse in the form of processed silicon wafers. While still in wafer form, the ICs are referred to as dice, an individual IC is called a die. The process of cutting the dice from wafers and embedding them into mountable containers is called packaging. Figure 1 is a simplified schematic of the major IC manufacturing steps.

Brief descriptions of the manufacturing steps are provided below.

A typical wafer is fabricated through a series of more than 100 process steps (Van Zant 1997; Zorich 1991). Virtually all of the processing is automated, but there are still many potential sources of failure that lead to defective wafers. An entire processing step may fail, in which case all of the die on a wafer will be bad. Many steps involve creating uniform thin layers on the wafer (e.g., by placing a drop of liquid material in the center and then spinning the wafer), and failures in this process can lead to radially-symmetric patterns of failed die (e.g., in the center, in rings around the center, and most commonly, near the edges of the wafer). The wafers can also be scratched during automated handling, which creates linear failure patterns and edge defects. Finally, defects in the silicon substrate and in the applied materials, as well as dust particles, can create spatially uniform patterns of die failures.

Wafer parametric test: Parametric tests measure physical and electrical parameters of the wafer such as electrical conductivity and behavior of individual sample components (transistors, capacitors, resistors). Although the above diagram depicts parametric testing as a distinct stage following wafer fabrication, in reality, parametric tests are performed throughout the fabrication process.

Die-level functional test (DLFT): Functional testing typically occurs once the wafers are completely fabricated and the dice are completely formed and functional. Functional tests measure the operational quality of the individual dice. A large and expensive robotic machine presses electrical probes onto the die contacts, input signals are fed to the circuit, and output signals are measured. Functional testing simulates normal and abnormal operating conditions (e.g., high, normal, and low voltage tests). The conventional approach to DLFT is exhaustive wafer test, i.e., all dice on all wafers undergo DLFT. If a die fails the functional test, an ink dot is placed on it, so that it will not be packaged later. The decision to place the ink dot is called the "inking decision". After inking, the wafers are typically shipped to a separate location for packaging. Note that DLFT is expensive (machines, personnel) and potentially damaging (due to the physical act of probing).

Packaging: To convert the wafers into packaged ICs, the wafers are cut into individual dice by a high-precision diamond saw. All of the resulting (non-inked) chips are mounted into packages, electrical contacts are bonded in place, and then a protective covering is added.

Package test: Once the ICs are packaged, they are tested again to ensure that the packaging process was successful. Package tests usually repeat many of the functional tests that were performed during DLFT. The test results from package testing are used to decide which ICs to sell.

This brief summary shows that Die-Level Functional Test (DLFT) is not essential to the quality of the final product, because any failures will be detected during Package Test. Hence, the main purpose of DLFT is to reduce costs by avoiding packaging defective dice. A secondary purpose of DLFT is to provide rapid feedback to the manufacturing process by detecting and diagnosing faulty manufacturing processes. The challenge for a decision-theoretic approach is to reduce the amount of DLFT (and hence reduce its cost) while not appreciably increasing the costs of packaging and manufacturing.

3 DECISION-THEORETIC WAFER TEST

The goal of our project was to replace the exhaustive test policy with a decision-theoretic policy that decides in real time which die to test and when to stop testing. Because the DLFT is performed by a robotic tester, it can be reprogrammed to test the die in any desired order based on the results of previous tests. The decision-theoretic policy seeks to maximize overall profit by combining a probabilistic model of the spatial distribution of die test results with a utility model of the costs of the IC manufacturing process.

A typical wafer test scenario under the decision-theoretic policy can be described as follows. The system begins with a learned probabilistic model of die test results and a utility model of processing costs. The system chooses the best die to test according to a value-of-information metric that measures the expected increase in profit for the whole wafer that would be obtained by testing each die and chooses the one die whose test result would be most valuable. This die is tested, and the results are incorporated into the system's probabilistic model of the wafer. The system then repeats this test-and-update cycle until the expected value of further testing is non-positive. At that stage, the system makes an individual packaging decision for each of the dice on the wafer, marking those that are rejected. Die that were tested and found to be defective are marked with an ink dot. In addition, die that were not tested are marked based on the *predictions* of the (updated) probabilistic model. Once this is complete, the system determines whether the wafer should be shipped to the packaging facility or should be held for analysis (because it contains a large number of defective die or an unusual pattern of failed die). For each of the wafers shipped to packaging, the unmarked dice are packaged and sent to package test. Those that pass package test are sent to customers.

3.1 The Probabilistic Model

As this scenario indicates, the heart of the testing process is a probabilistic model of the spatial distribution of die failures on the wafer. Figure 2 shows our die failure model. This has the form of a standard Naive Bayes belief network of the kind that have already proven useful in diagnostic systems (Horvitz, Breese, and Henrion 1991) and learning and discovery systems (Cheeseman, Self, Kelly, Taylor, Freeman, and Stutz 1988; Dietterich 1997). Belief networks such as this provide a succinct representation for probabilistic models in the form of directed acyclic graphs (Pearl 1988; Charniak 1991; Jordan 1999). In this representation, nodes represent random variables and arcs between nodes correspond to relationships between variables (often causal relationships). Each node in the graph stores a probability distribution. Hence, the node labeled w stores the distribution $P(w)$, and the nodes labeled f_i store the conditional distribution $P(f_i|w)$. In our case, these distributions are stored as tables indexed by the discrete values of the random variables. Once these probability distributions are known, the belief network fully specifies the joint distribution of the random variables in the network.

Our probabilistic model assumes that the wafers produced during the fabrication process belong to four possible "classes". One class of wafers might be wafers where all of the die are working well (except, perhaps, for randomly distributed failures). A second class of wafers might have virtually all die defective. A third class might have defective die in the center of the wafer, and good die everywhere else, and so on. As we will see below, the actual wafer classes are learned automatically, so these examples should just be considered

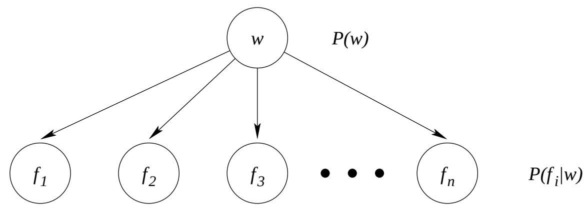

Figure 2 The wafer test belief net.

suggestive. The key idea is that within each wafer class, the die at each given location has a fixed probability of being defective and this probability may differ from one wafer class to another.

The class of a wafer is represented by a random variable w with four values. The boolean random variable f_i indicates whether the die at location i is functioning correctly or not. The probabilistic model is represented by the probability distributions $P(w)$ and $P(f_i|w)$. The best way to understand this probabilistic model is to view it as a simulation of the process of manufacturing a wafer. According to this simulation, the fabrication process begins by "choosing" what class of wafer to generate. This is modeled as choosing the value of w at random according to a probability distribution $P(w)$. Then, once w is chosen, the fabrication process "decides" whether each die should be defective or not. This is modeled by choosing for each physical position i on the wafer, the value f_i at random according to $P(f_i|w)$.

Although we can view the model as a simulation of the fabrication process, the model will be used in a very different way. During testing, the system begins not knowing the values of any of the random variables. Each time it performs a test (e.g., of die i), it learns the value of one of the variables f_i. Given the observed values of some of the f_i, probabilistic inference algorithms can compute the probability of the other random variables. Hence, given the result of testing dice 1, 5, 12, and 15, these algorithms can estimate the probability that die 23 is defective. They do this by first computing the probability that the wafer belongs to class $w = 1$, $w = 2$, etc., and then multiplying this by the probability $P(f_{23}|w)$ that die 23 is defective.

This kind of probabilistic model is also known as a *mixture model*, because the probability that a particular die is defective is a weighted sum (i.e., a mixture) of the probability that it is defective in each of the wafer classes. Specifically, $P(f_i) = \sum_w P(w)P(f_i|w)$.

Note that this application of Naive Bayes is different from two other ways in which Naive Bayes networks have been applied previously. When Naive Bayes networks are employed in supervised learning, the true class w of each wafer would be known during training, and the goal during testing would be to determine the class w of the wafer. In our application, the true classes are never known. Indeed, the notion of a "wafer class" here is just a mathematical abstraction, and the wafer class variable w is a hidden variable. The learning algorithm has the freedom to adjust the probability distributions $P(w)$ and $P(f_i|w)$ to best fit the historical wafer testing data.

Another application of Naive Bayes networks is to find "clusters" in data (e.g., Autoclass; Cheeseman, et al., 1988). These clustering algorithms also fit a Naive Bayes network to the data, but the goal in those applications is to examine the resulting learned classes to gain insight into patterns that may exist in the data and to determine the "true" number of such classes. For our testing application, we simply view the mixture model as a convenient representation of the joint distribution $P(f_1, f_2, \ldots, f_n)$ of failures of the die on the wafers. The virtue of this representation is that it allows us to easily compute the probability of failure of all of the untested dice given the test results for the dice that have been tested so far. Nonetheless, we can visualize the learned wafer classes and attempt to relate these to aspects of the manufacturing process. This is an interesting and potentially promising application. However, it is not considered in this paper.

3.2 Learning the Probabilistic Model via Expectation Maximization

In order to employ this probabilistic model to guide the testing of wafers, we must first learn the probability distributions $P(w)$ and $P(f_i|w)$. This is challenging, because the historical wafer test data only contain the values of the variables f_i—the wafer class variable is *latent* (i.e., unobserved). Fortunately, this is a well-understood learning problem that can be easily solved via the Expectation Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977; McLachlan and Krishnan 1997; Cheeseman, Self, Kelly, Taylor, Freeman, and Stutz 1988). The EM algorithm is really an "algorithm schema" that can be instantiated in many ways. Rather than giving an abstract description of the algorithm, we will describe how it is instantiated in our wafer testing problem.

Consider the following: If we knew the value of w for each wafer in our historical data, then it would be trivial to estimate $P(w)$ and $P(f_i|w)$. In particular, $P(w = 1)$ can be estimated as the fraction of the wafers for which $w = 1$, and $P(f_{23}|w = 1)$ can be estimated as the fraction of wafers from class $w = 1$ in which die 23 was defective. Hence, if we could somehow *guess* the value of w for each wafer, then we could compute the probabilistic model. The basic idea of EM is to iteratively guess the value of w (based on the probabilistic model from the previous iteration or random guesses in the first iteration) and then update the probabilistic model based on these guesses. The process of guessing the w values is called the E-step, and the process of updating the probabilistic model based on those guesses is called the M-step. These two steps give the EM algorithm its name.

In EM, the E-step does not guess a *single value* for w. Instead, it hypothesizes a *probability distribution* over the possible values of w. Hence, for a particular wafer, EM might guess that $w = 1$ with probability 0.20, $w = 2$ with probability 0.60, $w = 3$ with probability 0.04, and $w = 4$ with probability 0.16. In the M-step, it then estimates the overall $P(w = 1)$ as the average value of the (hypothesized) probability that $w = 1$ for each wafer. Similarly, it estimates $P(f_i|w = 1)$ by identifying all wafers where f_i is true, adding up the hypothesized probability that $w = 1$ for each of these wafers, and dividing by the total hypothesized probability that $w = 1$.

Once $P(w)$ and $P(f_i|w)$ have been estimated in this fashion, the EM algorithm can compute new hypothesized distributions for the value of w for each wafer. These are computed by treating the wafer class w as unknown, and applying standard probabilistic inference algorithms (Jordan 1999; Pearl 1988) to propagate the evidence concerning each f_i to determine the probability distribution over w. Algebraically, this process involves

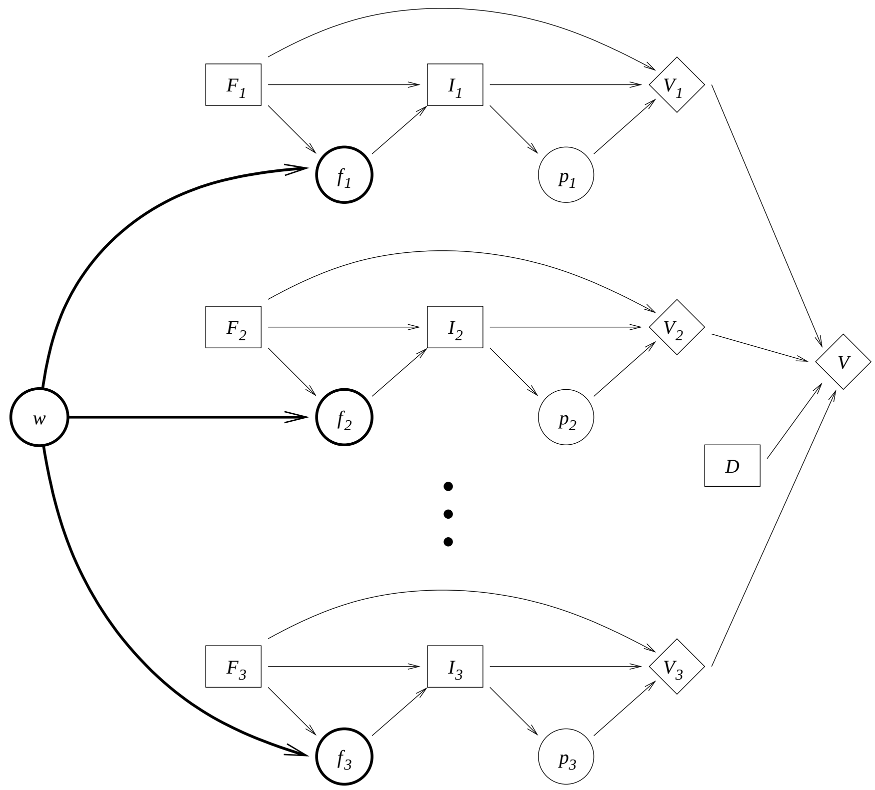

Figure 3 Influence diagram of the wafer test process.

computing for each wafer the normalized product of the current $P(W)$ with probability for each observed test result, $P(f_i = r|W)$ (where r is the test result for die i).

In each iteration, EM is guaranteed to improve its estimates of $P(w)$ and $P(f_i|w)$ until it reaches a local maximum in the likelihood of the training data. In our experiments, the algorithm typically converged in fewer than 50 iterations.

3.3 An Influence Diagram for Wafer Testing

Once the probabilistic model is learned, it is incorporated into a much larger model of the decision-making process called an *influence diagram*. Figure 3 shows the influence diagram that we constructed for the DLFT problem. The probabilistic model described in the previous section is shown here using slightly heavier lines than the rest of the influence diagram.

This influence diagram includes several kinds of nodes that were not present in the simple probabilistic model of Figure 2. Specifically, it includes additional *random variables* (p_i), *decision nodes* (F_i, I_i, and D), and *utility nodes* (V_i and V). We explain each of these in turn.

Additional Random Variables: The random variable p_i represents the result of the final package test for die i. Its probability distribution, $P(p_i|f_i)$ represents the probability that the die will fail package test depending on the results of the functional test. Rather than learning this distribution from historical data, we acquired it from domain experts. If $f_i = 1$ (the die is defective), then $p_i = 1$ (the die fails final package test) as well. But if $f_i = 0$ (the die passes functional test), the die may still later fail package test (with a low probability) because of problems introduced during the packaging process. The reason for relying on human estimates for the conditional package test probablity values was that the actual data was unavailable for this study. In our application, packaging occurred at a remote facility, and although yield rates were reported, specific die-level package results were not available. However, in principle, these parameters could also be derived from historical data. It seems reasonable to assume that parameters derived from such historical data would yield more accurate predictions of die failure and thereby improve wafer test decisions.

Decision Nodes: A decision node is a variable whose value can be chosen by the decision-making procedure. In Figure 3, the boolean variable F_i represents the decision to test die i, and the boolean variable I_i represents the decision to ink the die i. The boolean variable D represents the wafer disposition decision: should the wafer be sent to packaging or retained for analysis by the fabrication engineers? In the diagram, decision nodes are represented by rectangles.

Utility Nodes: A utility node represents the cost or benefit of some combination of other nodes. Specifically, the node V_i represents the utility of various combinations of testing F_i, inking I_i, and final package test results p_i. For example, if $F_i = 0$, $I_i = 0$, and $p_i = 0$, this corresponds to a die that was not tested and was not packaged (and therefore, we consider it to have failed the package test, so $p_i = 0$). The utility of this die is zero, because we have not spent any money testing or packaging it, but, of course, we can't sell it for a profit either. On the other hand, if $F_i = 1$, $I_i = 1$, and $p_i = 1$, then the utility of this combination, $V_i(1, 1, 1) = v_k - c_f - c_p$, which is the sales value of the die, v_k, less the cost of the die-level test, c_f, and the cost of packaging and testing, c_p. The utility node V represents the net profit obtained from the wafer, so it depends on the utilities of each individual die (represented by arcs from the V_i nodes) and also on the wafer disposition decision D. In the diagram, utility nodes are represented by diamonds. The details of the utility model are described in Table 1.

If a decision node has an incoming arc from another node, this means that the value of that other node will be known at the time the decision must be made. Hence, before making the inking decision I_i, we will know whether die was tested (F_i) and the result of that test (f_i). To keep the diagram simple, however, it does not include all of the arcs connecting test results to decision variables. In particular, the diagram does not show that at the time that we are deciding to test die i (i.e., by choosing a value for F_i), we will know the values of f_j for any other die j that have already been tested. Similarly, the diagram does not show that at the time we are making the wafer disposition decision D, the values of all of the inking decisions I_i will be known (but not the results of the package

Table 1 The wafer test utility model.

$W_j \equiv \text{wafer}_j$	$c_k \equiv$ cost of packaging a single die
$N_d(W_j) \equiv \#$ of dice on wafer_j	$c_h \equiv$ single wafer handling cost
$N_f(W_j) \equiv \#$ of functional tests on wafer_j	$v_k \equiv$ value of a single good package
$N_p(W_j) \equiv \#$ of package tests on wafer_j	value of package yield $\equiv V(Y(W_j)) = Y(W_j) \cdot v_k$
$N_k(W_j) \equiv \#$ of dice packaged from wafer_j	total cost of functional tests $\equiv C_f = N_f(W_j) \cdot c_f$
$Y(W_j) \equiv$ package yield of wafer_j	total cost of package tests $\equiv C_p = N_p(W_j) \cdot c_p$
$c_f \equiv$ cost of single functional test	total cost of packaging $\equiv C_k = (N_k(W_j) \cdot c_k) + c_h$
$c_p \equiv$ cost of single package test	wafer profit $\equiv V(Y(W_j)) - C_f - C_p - C_k$

tests, p_i). Although these arcs are omitted from the diagram, they are all present in the computational model.

If a utility node has incoming arcs from other variables, these show that the utility depends on the values of these other variables. The goal of inference algorithms for influence diagrams is to identify values of the decision variables that maximize the expected value of the value nodes. In our case, the goal is to choose values for the testing decisions F_i, inking decisions I_i, and wafer disposition decision D in order to maximize the expected utility of the entire wafer represented by V.

Our cost model does not capture three important costs. First, the cost (or benefit) of detecting fabrication problems is ignored. A reduction in die testing may reduce packaging costs but increase fabrication costs, because fabrication problems are not diagnosed as quickly. Second, the capital costs of purchasing and maintaining the die testing machines is ignored. If die-level testing can be significantly reduced, then fewer machines are needed, and these capital costs can be saved. Third, for some IC products, the current exhaustive testing policy becomes a rate-limiting bottleneck. A decrease in the number of dice tested per wafer means that more wafers can be tested, and therefore, wafer starts, throughput, and profits per unit time can increase.

3.4 Evaluating the Decision Model

Exact evaluation of the decision model to determine the decisions that maximize expected utility is infeasible, because it requires expanding a search tree consisting of all combinations of all testing decisions and test results. Hence, approximate methods must be employed.

One approximate method that has been employed previously in automated decision making is known as the one-step value of information (VOI) heuristic. According to this heuristic, we decide which die to test next by comparing two courses of action. One course of action is to stop testing immediately and make the inking and wafer disposition decisions. The other course of action is to test *one more die* and then proceed to the inking and wafer disposition decisions. The one-step VOI heuristic computes the expected value of these two courses of action and chooses the one with the larger expected value. Note

Table 2 Statistics summarizing the test set wafers in the two data sets. The yield is defined as the fraction of good dice out of the total number of dice.

	Data Set 1	Data Set 2
Total number of good dice	102288	101516
Total number of dice	125400	125400
Yield	0.8157	0.8095

that the heuristic does not consider testing two more die, three more die, etc., and hence, it avoids the combinatorial explosion of possible testing policies.

In more detail, the VOI computation proceeds as follows. Initially, all of the functional test decision nodes F_i are set to 0 (no test). The die inking decisions, I_i, and the wafer disposition decision, D, are then made to maximize expected utility. The computed utility of this course of action is called the "termination utility", U_{term}, because it is the utility of terminating functional testing and carrying out inking and packaging. The computation then computes the expected utility U_i of testing die i and then making the inking and wafer disposition decisions. If $\max_i U_i > U_{term}$, then test i is performed, f_i is observed, and the termination utility U_{term} is recomputed. Otherwise, testing terminates, and the inking and disposition decisions are made.

Because it is an approximate method, one-step greedy VOI does not in general yield the optimal policy for a sequential decision problem. However, we expect it to perform very well in this particular problem, because we are assuming that each testing action does not alter the wafer (e.g., by causing other die to fail). In addition, the inking decisions can be made independently, and the total utility is additive. Below, we will test experimentally how well greedy VOI works.

4 METHODS

Hewlett-Packard Company provided a data set for a mature IC product. The data consisted of the test results from 2400 wafers. We split this data into two separate data sets. Set 1 was used during model development and debugging. It consisted of 1200 wafers: 600 for training and 600 for testing. Set 2 was used for a final test of the system, and it also consisted of 1200 wafers: 600 for training and 600 for testing. The wafers are grouped into "lots" of 24 wafers, which are kept together (in a cassette) during the manufacturing process. Because there is a strong possibility that wafers within a lot share the same defects, we divided the wafers according to entire lots, so each training data consisted of 25 lots and each testing set of 25 lots. The lots were kept in chronological order.

Tables 2 provides summary statistics for these two data sets. Each wafer contains 209 dice.

We applied the EM algorithm to train the Naive Bayes probabilistic model. To determine the number of values of the latent variable w, we fit models with 1, 2, 4, 8, 12, 16, 20, and 24 values to the training data from Data Set 1 and measured the log likelihood of the wafers in the corresponding validation set. The log likelihood is a measure of how well the model predicts (or explains) the data. The best validation set log likelihood was achieved

Table 3 Comparison of the performance of four die testing policy on Data Set 1. The "Exhaustive" policy tests all die and packages those found to be defective. The "Package All" policy tests no die and packages them all. The "DT" policy is our decision-theoretic VOI policy. The "Oracle" tests no die and only packages the good ones. The total profit is the total profit obtained by that policy on the 600 wafers in the test set of Data Set 1.

	Total Profit	**Number Tested**	**Number Packaged**
Exhaustive	1184550	125400	102288
Package All	1226598	0	125400
DT	1229531	8210	121560
Oracle	1278600	0	102288

with a model containing 4 classes. For this model, the EM algorithm converged after no more than 50 iterations, which required less than three minutes of CPU time. To obtain our final probability model for the decision-theoretic tester, we trained a 4-class model on the training data from Data Set 2.

5 RESULTS

We performed four experiments to address the following questions:

- How well does the decision-theoretic approach perform compared to exhaustive testing, no testing, and optimal testing?
- How well does greedy value of information (VOI) perform in deciding when to stop testing?
- How does the system respond to abnormal wafers? Is the approach sensitive to process problems?
- How robust is the system with respect to changes in the utility parameters? Do changes in utility parameters result in rational responses from the system?

Each of these questions is addressed in turn.

5.1 Performance

We compared the performance of the decision-theoretic testing policy ("DT") with three other policies: (a) the current exhaustive test approach ("Exhaustive"), (b) a policy that performed no tests and packaged all die ("Package All"), and (c) an optimal testing policy ("Oracle") that performs no testing but packages only those die that would have passed the functional test. The oracle policy provides an upper bound on the best that any implementable policy could do.

The results from the model tests are presented in Tables 3 and 4 (for the test data in Data Set 1 and Data Set 2, respectively). Data Set 1 was employed in the original development and testing of the models. Data Set 2 was an independent data set that was used to get

Table 4 Comparison of the performance of four test policies on Data Set 2. See Table 3 for an explanation of the policies.

	Total Profit	Number Tested	Number Packaged
Exhaustive	1174900	125400	101516
Package All	1215211	0	125400
DT	1218309	5194	122292
Oracle	1268950	0	101516

some measure of how well the methods generalize. Both data sets were from the same wafer production and testing lines. The results show that the current exhaustive testing policy is the worst, and the DT policy is the best, of the three implementable policies. Indeed, the DT policy achieves 96% of the profit that can be realized by the Oracle, and it gives a 3.8% improvement in profit over exhaustive testing.

The Package All policy produces almost as much profit as the DT policy, so a reasonable question is what advantage the DT policy has over Package All. The answer is that on wafers with high yield there may be little benefit. The problem with Package All is that process problems will not be detected until package test results become available. This can be a problem, because often packaging is performed at a location (e.g., Asia) far from the wafer fabrication plant (e.g., US). In these cases the delay in feedback and the costs of shipping and handling make the Package All policy risky. Furthermore, after packaging, the physical position of each die on the wafer is no longer known, so the spatial information provided by die-level test is lost. This spatial information is valuable for diagnosing fabrication problems. One of the benefits of the DT policy is that for good wafers it can produce profits comparable to those produced with a Package All policy, but for bad wafers, it can detect process problems while the wafers are still at the fabrication plant

5.2 Evaluation of the greedy VOI stopping criteria

The decision-theoretic policy relies on a greedy value of information (VOI) computation to decide when to terminate testing. To determine how well this heuristic works, experiments were performed in which greedy VOI stopping was compared to the optimal stopping point. To determine the optimal stopping point, the decision-theoretic policy was modified to continue testing past the past the point of non-positive VOI until all dice were tested. Then the history of testing decisions was analyzed to find the moment at which the net profit would have been maximized had the system stopped then. Net profit includes the costs for functional tests up to that point and the sales obtained by making package decisions at that point.

Table 5 summarizes the results of this experiment on the test data from Data Set 1. The table shows that greedy VOI Stopping tests less than Optimal Stopping (performing only about 73% as many tests) and packages more (about 2% more packages). So Optimal Stopping spends a bit more on functional testing in order to reduce the number of bad

Table 5 An evaluation of how well the one-step VOI heuristic determines the optimal point to stop testing. "Exhaustive" tests all die; "Optimal stop" tests die in the same sequential order as the VOI heuristic, but uses an oracle to determine the optimal stopping point.

	Total Profit	**Number Tested**	**Number Packaged**
Exhaustive	1184550	125400	102288
VOI	1229531	8210	121560
Optimal stop	1231970	11206	119264

Table 6 Statistics describing six wafers selected from the test set of Data Set 1. Wafers 1 and 2 have good yield, wafers 3 and 4 have poor yield, and wafers 5 and 6 have mediocre yield.

Wafer ID Number	**Number Good Die**	**Yield**
1	178	0.85
2	180	0.86
3	17	0.08
4	2	0.01
5	138	0.66
6	118	0.56

dice packaged. Despite this difference, greedy VOI Stopping performs very well and realizes over 99% of the profit achieved by Optimal Stopping.

5.3 Detecting process problems

An interesting question for any testing policy is how it responds to abnormal wafers. Although the decision-theoretic approach was targeted towards a stable mature product, process problems are common, and abnormal wafer test results are often the first symptoms of such problems. An important question for a testing policy is how well can it recognize abnormally bad wafers?

To explore this issue, we discuss six wafers from Data Set 1 in detail. The first two wafers are typical "good" wafers with yields over 80%. The next two wafers are "bad", with yields of less than 10%. The final two wafers are "mediocre", with yields of 66% and 65%. The test wafers are described in Table 6. The simulation results are presented in Table 7.

In Table 7, exhaustive test profit is the profit realized under the exhaustive test policy. VOI profit is the profit realized under the decision-theoretic policy.

Table 7 Summary of the behavior of the VOI heuristic on six selected wafers and a comparison of the profit obtained on each wafer by VOI to the profit obtained by exhaustive testing. Note that the number of die tested is directly related to the true yield of the wafer (see Table 6).

Wafer ID Number	# Die Tested	# Die Packaged	Exhaustive Test Profit	VOI Test Profit
1	8	205	2068.25	2158.25
2	8	206	2093.25	2185.50
3	201	25	55.75	43.75
4	201	10	–131.75	–143.75
5	57	182	1568.25	1583.25
6	12	202	1318.25	1277.00

To visualize the testing behavior, wafer test maps are presented in Figure 4. These maps show the good dice, the dice that were tested according to the selective test policy, and the dice that were packaged according to this policy. In each map, white encodes true, so white represents good dice, tested dice, and packaged dice. Black represents bad dice, untested dice, and unpackaged dice. Wafers 1 and 2 are good wafers; wafers 3 and 4 are bad wafers; and wafers 5 and 6 are mediocre wafers.

The analysis of these six wafers shows that the decision-theoretic policy is responsive to abnormal wafers. There is a direct relationship between wafer yield and the number of functional tests performed. The higher the yield, the fewer tests. This means that, for a stable and mature product, the system tests only a small fraction of the total dice. However, the system is sensitive to abnormal yields, which indicate process problems. On such wafers, the system tests more thoroughly. For wafers with extremely low yield, the system responds by testing almost all dice. This means that a minimal amount of resources are expended on good wafers, yet bad wafers are detected.

The reasons for this highly-desirable behavior are not entirely clear. One explanation is that because our IC product was a mature, high-yield product, the learned stochastic model is expecting to see good wafers. So when it encounters a bad wafer, its predictions concerning the untested dice become uncertain (near 0.5), and it must do more testing to make good inking decisions. If our product had been one where half of the wafers were good and the other half very bad, then a bad wafer would not have been surprising, and the decision-theoretic policy would only perform enough tests to be confident of which kind of wafer it had. Then it would proceed to the inking decisions.

This analysis suggests that the output from the decision-theoretic approach could be fed into statistical process control (SPC) methods that routinely monitor for process problems. A straightforward extension to the current SPC system would be to replace actual test measures with predicted measures. So, for example, rather than setting control limits around the actual functional test results, the control limits could be set around the predicted functional test results. Thus, the decision-theoretic approach provides dual

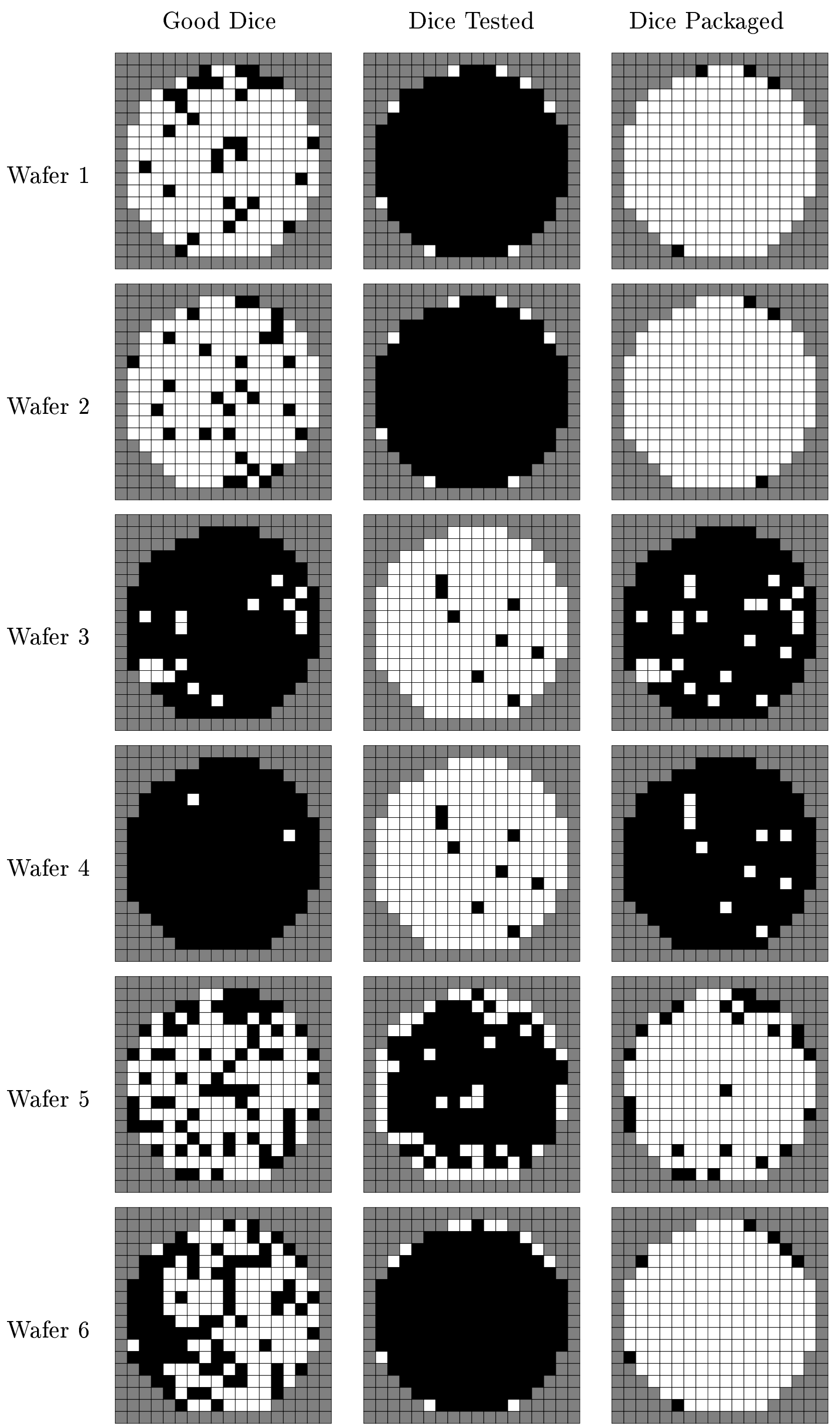

Figure 4 Detecting Process Problems.

Table 8 Robustness Tests: Changes to Package Cost

	# Die Tested	# Die Packaged	# False Positives	# True Negatives
$0.5c_k$	240	9854	1658	178
c_k	796	9672	1386	360
$2c_k$	2484	9312	1026	720

benefits. First, it greatly reduces the requirement for testing resources. Second, it satisfies the requirement for prompt detection of process problems.

5.4 Robustness to changes in utility parameters

One benefit of decision-theoretic methods is that changes in utility parameters should result in rational changes in performance without the need to manually re-engineer the learned models or the control structures. To verify this, we experimented with changes to two of the utility parameters:

- Cost of performing a single functional test and
- Cost of packaging a single die.

For each of these parameters, a series of tests was performed in which the parameter of interest was swept through a range of values and performance on a testing scenario was measured. For these tests, we applied the stochastic model trained on the 600 training wafers from Data Set 1 to test 48 test-set wafers. Performance was measured by the number of functional tests performed, the number of dice packaged, the number of false positives (i.e., bad die packaged), and the number of true negatives (i.e., bad die not packaged).

5.4.1 Changes to Package Cost

In the first set of tests, the cost to package a single die was manipulated. Let c_k represent the normal package cost. Then consider the effects of cutting the package cost in half ($0.5c_k$) or doubling it ($2c_k$). The results are summarized in Table 8.

The results show that when it is relatively inexpensive to package dice, the system packages more and tests less. As the package cost increases, false positives become more expensive, so more tests are performed to reduce this risk. Thus, with respect to changes in package cost, the system performs rationally by adjusting its testing and package decisions to maximize expected profits.

5.4.2 Changes to Functional Test Cost

A second set of tests was performed in which the functional test cost was manipulated. Let c_f represent the current cost of a single functional test. Then consider the effects of setting the functional test cost at $0.1c_f$, $0.67c_f$, c_f, $1.33c_f$, $1.67c_f$, $2c_f$, and $10c_f$. The results are summarized in Table 9.

Table 9 Robustness Tests: Results of Changes to Functional Test Costs

	# Die Tested	**# Die Packaged**	**# False Positives**	**# True Negatives**
$0.1c_f$	10032	8286	0	1746
$0.67c_f$	1509	9485	1199	547
c_f	796	9672	1386	360
$1.33c_f$	321	9835	1549	197
$1.67c_f$	240	9854	1568	178
$2.0c_f$	169	9912	1626	120
$10c_f$	0	10032	1746	0

This table shows that the system behaves rationally by adjusting its testing and packaging decisions to reflect changes in cost parameters. When the package cost is increased, the system tests more to avoid wasting resources by packaging bad dice. On the other hand, when the functional test cost is increased, the system tests less and packages more. If the test cost is set sufficiently low, then the system tests all dice. If the test cost is set sufficiently high, then the system tests none of the dice. In none of the test scenarios was it profitable to miss a good die, so the number of true positives was always equal to the total number of good dice, and the number of false negatives was always zero. This behavior is the result of two factors. First, given the quality of the wafers in the training set, all dice had a reasonable prior probability of being good. Second, the value of a good package was sufficient to justify packaging all dice based on these prior probabilities.

6 CONCLUDING REMARKS

The experiments in this paper have demonstrated that manufacturing data can be mined to produce effective decision-theoretic control methods. Furthermore, these methods can produce substantial improvements in the die-level functional test stage of VLSI IC manufacturing. Specifically, the experiments have shown the following:

1. The decision-theoretic policy produced more net profit than either the exhaustive test policy or the policy of performing no die-level testing.
2. The greedy VOI stopping criterion produced near-optimal stopping behavior.
3. The decision-theoretic policy is able to detect abnormal wafers, and it responds by testing them more thoroughly. Hence, although the cost model does not reflect the value of die-level test for detecting fabrication problems, the decision-theoretic policy still detects these problems well. Existing process control statistics could be based on the predictions of the stochastic model.
4. The decision-theoretic policy is robust to changes in testing costs and packaging costs.

These experiments demonstrate that the decision-theoretic approach to die-level functional test has the potential to reduce testing costs, increase wafer starts, and improve the bottom line. In addition, the method is easy to implement—the EM training was straightforward and efficient, and the cost model can easily be changed to reflect changes in market conditions. As market competition forces IC manufacturers to look for methods to reduce production costs, intelligent wafer test is one promising area to explore. Although the current study performed simulations of historical test scenarios, we are optimistic that future work will lead to a production implementation of these methods. We believe that in this and many other computer-integrated manufacturing applications, data mining and decision-theoretic methods have an important role to play.

Acknowledgements

The authors thank Hewlett-Packard Company for providing the wafer test data for this project. T. Dietterich acknowledges the support of the National Science Foundation under grant number 9626584-IRI and the support of the Air Force Office of Scientific Research under contract F49620-98-1-0375.

References

Charniak, E. (1991). Bayesian networks without tears. *AI Magazine 12*(4), 50–63.

Cheeseman, P., M. Self, J. Kelly, W. Taylor, D. Freeman, and J. Stutz (1988). Bayesian classification. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Cambridge, MA, pp. 607–611. AAAI Press/MIT Press.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B39*, 1–38.

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine 18*(4), 97–136.

Horvitz, E., J. Breese, and M. Henrion (1991). Decision analysis and expert systems. *AI Magazine 12*(4), 64–91.

Jordan, M. L. (1999). *Learning in graphical models*. Cambridge, MA: MIT Press.

McLachlan, G. and T. Krishnan (1997). *The EM algorithm and extensions*. New York: Wiley.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. San Francisco, CA: Morgan Kaufmann.

Van Zant, P. (1997). *Microchip fabrication: A practical guide to semiconductor processing (3rd ed.)*. New York: McGraw-Hill.

Zorich, R. (1991). *Handbook of Quality Integrated Circuit Manufacturing*. San Diego, CA: Academic Press.