CHAPTER 1

Introduction

1.1 INTRODUCTION

Things don't always work as intended. Some devices are manufactured incorrectly, others break or wear out after extensive use. In order to determine if a device was manufactured correctly, or if it continues to function as intended, it must be tested. The test is an evaluation based on a set of requirements. Depending on the complexity of the product, the test may be a mere perusal of the product to determine whether it suits one's personal whims, or it could be a long, exhaustive checkout of a complex system to ensure compliance with many performance and safety criteria. Emphasis may be on speed of performance, accuracy, or reliability.

Consider the automobile. One purchaser may be concerned simply with color and styling, another may be concerned with how fast the automobile accelerates, yet another may be concerned solely with reliability records. The automobile manufacturer must be concerned with two kinds of test. First, the design itself must be tested for factors such as performance, reliability, and serviceability. Second, individual units must be tested to ensure that they comply with design specifications.

Testing will be considered within the context of digital logic. The focus will be on technical issues, but it is important not to lose sight of the economic aspects of the problem. Both the cost of developing tests and the cost of applying tests to individual units will be considered. In some cases it becomes necessary to make trade-offs. For example, some algorithms for testing memories are easy to create; a computer program to generate test vectors can be written in less than 12 hours. However, the set of test vectors thus created may require several millenia to apply to an actual device. Such a test is of no practical value. It becomes necessary to invest more effort into initially creating a test in order to reduce the cost of applying it to individual units.

This chapter begins with a discussion of quality. Once we reach an agreement on the meaning of quality, as it relates to digital products, we shift our attention to the subject of testing. The test will first be defined in a broad, generic sense. Then we put the subject of digital logic testing into perspective by briefly examining the overall design process. Problems related to the testing of digital components and

Digital Logic Testing and Simulation, *Second Edition*, by Alexander Miczo
ISBN 0-471-43995-9

assemblies can be better appreciated when viewed within the context of the overall design process. Within this process we note design stages where testing is required. We then look at design aids that have evolved over the years for designing and testing digital devices. Finally, we examine the economics of testing.

1.2 QUALITY

Quality frequently surfaces as a topic for discussion in trade journals and periodicals. However, it is seldom defined. Rather, it is assumed that the target audience understands the intended meaning in some intuitive way. Unfortunately, intuition can lead to ambiguity or confusion. Consider the previously mentioned automobile. For a prospective buyer it may be deemed to possess quality simply because it has a soft leather interior and an attractive appearance. This concept of quality is clearly subjective: It is based on individual expectations. But expectations are fickle: They may change over time, sometimes going up, sometimes going down. Furthermore, two customers may have entirely different expectations; hence this notion of quality does not form the basis for a rigorous definition.

In order to measure quality quantitatively, a more objective definition is needed. We choose to define quality as the degree to which a product meets its requirements. More precisely, it is the degree to which a device conforms to applicable specifications and workmanship standards.[1] In an integrated circuit (IC) manufacturing environment, such as a wafer fab area, quality is the absence of "drift"—that is, the absence of deviation from product specifications in the production process. For digital devices the following equation, which will be examined in more detail in a later section, is frequently used to quantify quality level:[2]

$$\text{AQL} = Y^{(1-T)} \tag{1.1}$$

In this equation, AQL denotes acceptable quality level, it is a function of Y (product yield) and T (test thoroughness). If no testing is done, AQL is simply the *yield*—that is, the number of good devices divided by the total number of devices made. Conversely, if a complete test were created, then $T = 1$, and all defects are detected so no bad devices are shipped to the customer.

Equation (1.1) tells us that high quality can be realized by improving product yield and/or the thoroughness of the test. In fact, if $Y \geq \text{AQL}$, testing is not required. That is rarely the case, however. In the IC industry a high yield is often an indication that the process is not aggressive enough. It may be more economically rewarding to shrink the geometry, produce more devices, and screen out the defective devices through testing.

1.3 THE TEST

In its most general sense, a test can be viewed as an experiment whose purpose is to confirm or refute a hypothesis or to distinguish between two or more hypotheses.

Figure 1.1 depicts a test configuration in which stimuli are applied to a device-under-test (DUT), and the response is evaluated. If we know what the *expected response* is from the correctly operating device, we can compare it to the response of the DUT to determine if the DUT is responding correctly.

When the DUT is a digital logic device, the stimuli are called *test patterns* or *test vectors*. In this context a *vector* is an ordered *n*-tuple; each bit of the vector is applied to a specific input pin of the DUT. The expected or predicted outcome is usually observed at output pins of the device, although some test configurations permit monitoring of test points within the circuit that are not normally accessible during operation. A tester captures the response at the output pins and compares that response to the expected response determined by applying the stimuli to a known good device and recording the response, or by creating a *model* of the circuit (i.e., a representation or abstraction of selected features of the system[3]) and simulating the input stimuli by means of that model. If the DUT response differs from the expected response, then an *error* is said to have occurred. The error results from a *defect* in the circuit.

The next step in the process depends on the type of test that is to be applied. A taxonomy of test types[4] is shown in Table 1.1. The classifications range from testing die on a bare wafer to tests developed by the designer to verify that the design is correct. In a typical manufacturing environment, where tests are applied to die on a wafer, the most likely response to a failure indication is to halt the test immediately and discard the failing part. This is commonly referred to as a go–nogo test. The object is to identify failing parts as quickly as possible in order to reduce the amount of time spent on the tester.

If several functional test programs were developed for the part, a common practice is to arrange them so that the most effective test program—that is, the one that uncovers the most defective parts—is run first. Ranking the effectiveness of the test programs can be done through the use of a fault simulator, as will be explained in a subsequent chapter. The die that pass the wafer test are packaged and then retested. Bonding a chip to a package has the potential to introduce additional defects into the process, and these must be identified.

Binning is the practice of classifying chips according to the fastest speed at which they can operate. Some chips, such as microprocessors, are priced according to their clock speed. A chip with a 10% performance advantage may bring a 20–50% premium in the marketplace. As a result, chips are likely to first be tested at their maximum rated speed. Those that fail are retested at lower clock speeds until either they pass the test or it is determined that they are truly defective. It is, of course, possible that a chip may run successfully at a clock speed lower than any for which it was tested. However, such chips can be presumed to have no market value.

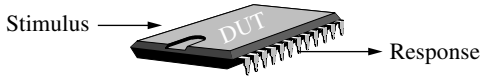

Figure 1.1 Typical test configuration.

TABLE 1.1 Types of Tests

Type of Test	Purpose of Test
Production	Test of manufactured parts to sort out those that are faulty
Wafer Sort or Probe	Test of each die on the wafer.
Final or Package	Test of packaged chips and separation into bins (military, commercial, industrial).
Acceptance	Test to demonstrate the degree of compliance of a device with purchaser's requirements.
Sample	Test of some but not all parts.
Go–no-go	Test to determine whether device meets specifications.
Characterization or engineering	Test to determine actual values of AC and DC parameters and the interaction of parameters. Used to set final specifications and to identify areas to improve process to increase yield.
Stress screening (burn-in)	Test with stress (high temperature, temperature cycling, vibration, etc.) applied to eliminate short life parts.
Reliability (accelerated life)	Test after subjecting the part to extended high temperature to estimate time to failure in normal operation.
Diagnostic (repair)	Test to locate failure site on failed part.
Quality	Test by quality assurance department of a sample of each lot of manufactured parts. More stringent than final test.
On-line or checking	On-line testing to detect errors during system operation.
Design verification	Verify the correctness of a design.

Diagnosis may be called for when there is a yield crash—that is, a sudden, significant drop in the number of devices that pass a test. To aid in investigating the causes, it may be necessary to create additional test vectors specifically for the purpose of isolating the source of the crash. For ICs it may be necessary to resort to an e-beam probe to identify the source. Production diagnostic tests are more likely to be created for a printed circuit board (PCB), since they are often repairable and generally represent a larger manufacturing cost. Tests for memory arrays are thorough and methodical, thus serving both as go–no-go tests and as diagnostic tests. These tests permit substitution of spare rows or columns in order to repair the memory array, thereby significantly improving the yield.

Products tend to be more susceptible to yield problems in the early stages of their existence, since manufacturing processes are new and unfamiliar to employees. As a result, there are likely to be more occasions when it is necessary to investigate problems in order to diagnose causes. For mature products, yield is frequently quite high, and testing may consist of sampling by randomly selecting parts for test. This is also a reasonable strategy for low complexity parts, such as a chip that goes into a wristwatch.

To protect against yield problems, particularly in the early phases of a project, *burn-in* is commonly employed. Burn-in stresses semiconductor products in order to

identify and eliminate marginal performers. The goal is to ensure the shipment of parts having an acceptably low failure rate and to potentially improve product reliability.[5] Products are operated at environmental extremes, with the duration of this operation determined by product history. Manufacturers institute programs, such as Intel's ZOBI (zero hour burn-in), for the purpose of eliminating burn-in and the resulting capital equipment costs.[6]

When stimuli are simulated against the circuit model, the simulator produces a file that contains the input stimuli and expected response. This information goes to the tester, where the stimuli are applied to manufactured parts. However, this information does not provide any indication of just how effective the test is at detecting defects internal to the circuit. Furthermore, if an erroneous response should occur at any of the output pins during testing of manufactured parts, there is no insight into the location of the defect that induced the incorrect response. Further testing may be necessary to distinguish which of several possible defects produced the response. This is accomplished through the use of fault models.

The process is essentially the same; that is, vectors are simulated against a model of the circuit, except that the computer model is modified to make it appear as though a fault were present. By simulating the correct model and the faulted model, responses from the two models can be compared. Furthermore, by injecting several faults into the model, one at a time, and then simulating, it is possible to compare the response of the DUT to that of the various faulted models in order to determine which faulted model either duplicates or most closely approximates the behavior of the DUT.

If the DUT responds correctly to all applied stimuli, confidence in the DUT increases. However, we cannot conclude that the device is fault-free! We can only conclude that it does not contain any of the faults for which it was tested, but it could contain other faults for which an effective test was not applied.

From the preceding paragraphs it can be seen that there are three major aspects of the test problem:

1. Specification of test stimuli
2. Determination of correct response
3. Evaluation of the effectiveness of the stimuli

Furthermore, this approach to testing can be used both to detect the presence of faults and to distinguish between several faults for repair purposes.

In digital logic, the three phases of the test process listed above are referred to as test pattern generation, logic simulation, and fault simulation. More will be said about these processes in later chapters. For the moment it is sufficient to state that each of these phases ranks equally in importance; they in fact complement one another. Stimuli capable of distinguishing between good circuits and faulted circuits do not become effective until they are simulated so their effects can be determined. Conversely, extremely accurate simulation against very precise models with

ineffective stimuli will not uncover many defects. Hence, measuring the effectiveness of test stimuli, using an accepted metric, is another very important task.

1.4 THE DESIGN PROCESS

Table 1.1 identifies several types of tests, ranging from design verification, whose purpose is to ensure that a design conforms to the designer's intent, to various kinds of tests directed toward identifying units with manufacturing defects, and tests whose purpose is to identify units that develop defects during normal usage. The goal during product design is to develop comprehensive test programs before a design is released to manufacturing. In reality, test programs are not always adequate and may have to be enhanced due to an excessive number of faulty units reaching end users. In order to put test issues into proper perspective, it will be helpful here to take a brief look at the design process, starting with initial product conception.

A digital device begins life as a concept whose eventual goal is to fill a perceived need. The concept may flow from an original idea or it may be the result of market research aimed at obtaining suggestions for enhancements to an existing product. Four distinct product development classifications have been identified:[7]

First of a kind
Me too with a twist
Derivative
Next-generation product

The "first of a kind" is a product that breaks new ground. Considerable innovation is required before it is implemented. The "me too with a twist" product adds incremental improvements to an existing product, perhaps a faster bus speed or a wider data path. The "derivative" is a product that is derived from an existing product. An example would be a product that adds functionality such as video graphics to a core microprocessor. Finally, the "next-generation product" replaces a mature product. A 64-bit microprocessor may subsume op-codes and basic capabilities, but also substantially improve on the performance and capabilities of its 32-bit predecessor.

The category in which a product falls will have a major influence on the design process employed to bring it to market. A "first of a kind" product may require an extensive requirements analysis. This results in a detailed product specification describing the functionality of the product. The object is to maximize the likelihood that the final product will meet performance and functionality requirements at an acceptable price. Then, the behavioral description is prepared. It describes what the product will do. It may be brief, or it may be quite voluminous. For a complex design, the product specification can be expected to be very formal and detailed. Conversely, for a product that is an enhancement to an existing product, documentation may consist of an engineering change notice describing only the proposed changes.

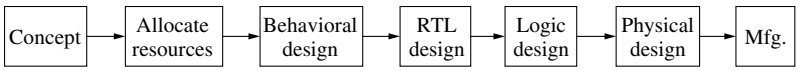

Figure 1.2 Design flow.

After a product has been defined and a decision has been made to manufacture and market the device, a number of activities must occur, as illustrated in Figure 1.2. These activities are shown as occurring sequentially, but frequently the activities overlap because, once a commitment to manufacture has been made, the objective is to get the product out the door and into the marketplace as quickly as possible. Obviously, nothing happens until a development team is put in place. Sometimes the largest single factor influencing the time-to-market is the time required to allocate resources, including staff to implement the project and the necessary tools by which the staff can complete the design and put a manufacturing flow into place. For a device with a given level of performance, time of delivery will frequently determine if the product is competitive; that is, does it fall above or below the performance–time plot illustrated in Figure 1.3?

Once the behavioral specification has been completed, a functional design must be created. This is actually a continuous flow; that is, the behavior is identified, and then, based on available technology, architects identify functional units. At that stage of development an important decision must be made as to whether or not the product can meet the stated performance objectives, given the architecture and technology to be used. If not, alternatives must be examined. During this phase the logic is partitioned into physical units and assigned to specific units such as chips, boards, or cabinets. The partitioning process attempts to minimize I/O pins and cabling between chips, boards, and units. Partitioning may also be used to advantage to simplify such things as test, component placement, and wire routing.

The use of hardware design languages (HDLs) for the design process has become virtually universal.Two popular HDLs, VHDL (VHSIC Hardware Description Language) and Verilog, are used to

Specify an architecture
Partition the architecture into smaller modules
Synthesize an RTL description
Verify that a structural implementation corresponds to the architectural design
Check out microcode and/or diagnostic programs
Serve as documentation

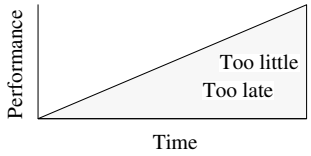

Figure 1.3 Performance–time plot.

A behavioral description specifies what a design must do. There is usually little or no indication as to how it must be done. For example, a large case statement might identify operations to be performed by an ALU in response to different values applied to a control field. The RTL design refines the behavioral description. Operations identified at the behavioral level are elaborated upon in more detail. RTL design is followed by logic design. This stage may be generated by synthesis programs, or it may be created manually, or, more often, some modules are synthesized while others are manually designed or included from a library of predesigned modules, some or all of which may have been purchased from an outside vendor. The use of predesigned, or core, modules may require selecting and/or altering components and specifying the interconnection of these components. At the end of the process, it may be the case that the design will not fit on a piece of silicon, or there may not be enough I/O pins to accommodate the signals, in which case it becomes necessary to reevaluate the design.

Physical design specifies the physical placement of components and the routing of wires between components. Placement may assign circuits to specific areas on a piece of silicon, it may specify the placement of chips on a PCB, or it may specify the assignment of PCBs to a cabinet. The routing task specifies the physical connection of devices after they have been placed. In some applications, only one or two connection layers are permitted. Other applications may permit PCBs with 20 or more interconnection layers, with alternating layers of metal interconnects and insulating material.

The final design is sent to manufacturing, where it is fabricated. Engineering changes must frequently be accommodated due to logic errors or other unexpected problems such as noise, timing, heat buildup, electrical interference, and so on, or inability to mass produce some critical parts.

In these various design stages there is a continuing need for testing. Requirements analysis attempts to determine whether the product will fulfill its objectives, and testing techniques are frequently based on marketing studies. Early attempts to introduce more rigor into this phase included the use of design languages such as PSL/PSA (Problem Statement Language/Problem Statement Analyzer).[8] It provided a way both to rigorously state the problem and to analyze the resulting design. PMS (Processors, Memories, Switches)[9] was another early attempt to introduce rigor into the initial stages of a design project, permitting specification of a design via a set of consistent and systematic rules. It was often used to evaluate architectures at the system level, measuring data throughput and looking for design bottlenecks. Verilog and VHDL have become the standards for expressing designs at all levels of abstraction, although investigation into specification languages continues to be an active area of research. Its importance is seen from such statements as "requirements errors typically comprise over 40% of all errors in a software project"[10] and "the really serious mistakes occur in the first day."[3]

A design expressed in an HDL, at a level of abstraction that describes intended behaviors, can be formally tested. At this level the design is a requirements document that states, in a simulation language, what actions the product must perform. The HDL permits the designer to simulate behavioral expressions with input vectors

chosen to confirm correctness of the design or to expose design errors. The design verification vectors must be sufficient to confirm that the design satisfies the behavior expressed in the product specification. Development of effective test stimuli at this state is highly iterative; a discrepancy between designer intent and simulation results often indicates the need for more stimuli to diagnose the underlying reason for the discrepancy. A growing trend at this level is the use of formal verification techniques (cf. Chapter 12.)

The logic design is tested in a manner similar to the functional design. A major difference is that the circuit description is more detailed; hence thorough analysis requires that simulations be more exhaustive. At the logic level, timing is of greater concern, and stimuli that were effective at the register transfer level (RTL) may not be effective in ferreting out critical timing problems. On the other hand, stimuli that produced correct or expected response from the RTL circuit may, when simulated by a timing simulator, indicate incorrect response or may indicate marginal performance, or the simulator may simply indicate that it cannot predict the correct response.

The testing of physical structure is probably the most formal test level. The test engineer works from a detailed design document to create tests that determine if response of the fabricated device corresponds to response of the design. Studies of fault behavior of the selected circuit family or technology permit the creation of fault models. These fault models are then used to create specific test stimuli that attempt to distinguish between the correctly operating device and a device with the fault.

This last category, which is the most highly developed of the design stages, due to its more formal and well-defined environment, is where we will concentrate our attention. However, many of the techniques that have been developed for structural testing can be applied to design verification at the logic and functional levels.

1.5 DESIGN AUTOMATION

Many of the activities performed by architects and logic designers were long ago recognized to be tedious, repetitious, error prone, and time-consuming, and hence could and should be automated. The mechanization of tedious design processes reduces the potential for errors caused by human fatigue, boredom, and inattention to mundane details. Early elimination of errors, which once was a desirable objective, has now become a virtual necessity. The market window for new products is sometimes so small that much of that window will have evaporated in the time that it takes to correct an error and push the design through the entire fabrication cycle yet another time.

In addition to the reduction of errors, elimination of tedious and time-consuming tasks enables designers to spend more time on creative endeavors. The designer can experiment with different solutions to a problem before a design becomes frozen in silicon. Various alternatives and trade-offs can be studied. This process of automating various aspects of the design process has come to be known as *electronic design*

*automatio*n (EDA). It does not replace the designer but, rather, enables the designer to be more productive and more creative. In addition, it provides access to IC design for many logic designers who know very little about the intricacies of laying out an IC design. It is one of the major factors responsible for taking cost out of digital products.

Depending on whether it is an IC, a PCB, or a system comprised of several PCBs, a typical EDA system supports some or all of the following capabilities:

- Data management
  - Record data
  - Retrieve data
  - Define relationships
  - Perform rules checks
- Design analysis/verification
  - Evaluate performance/capabilities
  - Simulate
  - Check timing
- Design fabrication
  - Perform placement and routing
  - Create tests for structural defects
  - Identify qualified vendors
- Documentation
  - Extract parts list
  - Create/update product specification

The data management system supports a data base that serves as a central repository for all design data. A data management program accepts data from the designer, formats it, and stores it in the data base. Some validity checks can be performed at this time to spot obvious errors. Programs must be able to retrieve specific records from the data base. Different applications require different records or combinations or records. As an example, one that we will elaborate on in a later chapter, a test program needs information concerning the specific ICs used in the design of a board, it needs information concerning their interconnections, and it needs information concerning their physical location on a board.

A data base should be able to express hierarchical relationships.[11] This is especially true if a facility designs and fabricates both boards and ICs. The ICs are described in terms of logic gates and their interconnections, while the board is described in terms of ICs and their interconnections. A "where used" capability for a part number is useful if a vendor provides notice that a particular part is no longer available. Rules checks can include examination of fan-out from a logic gate to ensure that it does not exceed some specified limit. The total resistive or capacitive loading on an output can be checked. Wire length may also be critical in some applications, and rules checking programs should be able to spot nets that exceed wire length maximums.

The data management system must be able to handle multiple revisions of a design or multiple physical implementations of a single architecture. This is true for manufacturers who build a range of machines all of which implement the same architecture. It may not be necessary to maintain an architectural level copy with each physical implementation. The system must be able to control access and update to a design, both to protect proprietary design information from unauthorized disclosure and to protect the data base from inadvertent damage. A lock-out mechanism is useful to prevent simultaneous updates that could result in one or both of the updates being lost.

Design analysis and verification includes simulation of a design after it is recorded in the data base to verify that it is functionally correct. This may include RTL simulation using a hardware design language and/or simulation at a gate level with a logic simulator. Precise relationships must be satisfied between clock and data paths. After a logic board with many components is built, it is usually still possible to alter the timing of critical paths by inserting delays on the board. On an IC there is no recourse but to redesign the chip. This evaluation of timing can be accomplished by simulating input vectors with a timing simulator, or it can be done by tracing specific paths and summing up the delays of elements along the way.

After a design has stabilized and has been entered into a data base, it can be fabricated. This involves placement either of chips on a board or of circuits on a die and then interconnecting them. This is usually accomplished by placement and routing programs. The process can be fully automated for simple devices, or for complex devices it may require an interactive process whereby computer programs do most of the task, but require the assistance of an engineer to complete the task. Checking programs are used after placement and routing.

Typical checks look for things such as runs too close to one another, and possible opens or shorts between runs. After placement and routing, other kinds of analysis can be performed. This includes such things as computing heat concentration on an IC or PCB and computing the reliability of an assembly based on the reliability of individual components and manufacturing processes. Testing the structure involves creation of test stimuli that can be applied to the manufactured IC or PCB to determine if it has been fabricated correctly.

Documentation includes the extraction of parts lists, the creation of logic diagrams and printing of RTL code. The parts list is used to maintain an inventory of parts in order to fabricate assemblies. The parts list may be compared against a master list that includes information such as preferred vendors, second sources, or alternate parts which may be used if the original part is unavailable. Preferred vendors may be selected based on an evaluation of their timeliness in delivering parts and the quality of parts received from them in the past. Logic diagrams are used by technicians and field engineers to debug faulty circuits as well as by the original designer or another designer who must modify or debug a logic design at some future date.

1.6 ESTIMATING YIELD

We now look at yield analysis, based on various probability distribution functions. But, first, just how important are yield equations? James Cunningham[12] describes a

situation in which a company was invited to submit a bid to manufacture a large CMOS custom logic chip. The chip had already been designed at another company and was to have a die area of 2.3 cm^2. The company had experience making CMOS parts, but never one this large. Hence, they were uncertain as to how to estimate yield for a chip of this size.

When they extrapolated from existing data, using a computer-generated best-fit model, they obtained a yield estimate $Y = 1.4\%$. Using a Poisson model with $D_0 = 2.1$, where D_0 is the average number of defects per unit area A, they obtained an estimate $Y = 0.8\%$. They then calculated the yield using Seeds' model,[13] which gave $Y = 17\%$. That was followed by Murphy's model.[14] It gave $Y = 4\%$. They decided to average Seeds' model and Murphy's model and submit a bid based on 11% die sort yield. A year later they were producing chips with a yield of 6%, even though D_0 had fallen from 2.1 to 1.9 defects/cm^2. The company had started to evaluate the negative binomial yield model $Y = (1 + D_0A/\alpha)^{-\alpha}$. A value of $\alpha = 3$ produced a good fit for their yield data. Unfortunately, the company could not sustain losses on the product and dropped it from production, leaving the customer without a supply of parts.

Probability distribution functions are used to estimate the probability of an event occurring. The *binomial probability distribution* is a discrete distribution, which is expressed as

$$P(k) = \frac{n!}{k!(n-k)!}P^k(1-P)^{n-k} \tag{1.2}$$

If P is the probability of a defect on a die, then $P(k)$ is the probability of k defects on the die, when there are a total of $n = D_0A_w$ defects, where A_w is the area of the wafer. The probability P is $D_0A/D_0A_w = A/A_w$. Substituting into Eq. (1.2) yields

$$P(k) = \frac{n!}{k!(n-k)!}\left(\frac{A}{A_w}\right)^k\left(1-\frac{A}{A_w}\right)^{n-k} \tag{1.3}$$

To derive the equation for a die with no defects, set $k = 0$. This yields

$$P(k=0) = \left[1-\frac{A}{A_w}\right]^{D_0A_w} \tag{1.4}$$

The first distribution that was frequently used to estimate yields was the *Poisson distribution*, which is expressed as

$$P(k) = \frac{e^{-\lambda_0}\lambda_0^{\,k}}{k!} \qquad \text{for } k = 0, 1, 2, \ldots \tag{1.5}$$

where λ_0 is the average number of defects per die. For die with no defects ($k = 0$), the equation becomes $P(0) = e^{-\lambda_0}$. If $\lambda_0 = .5$, the yield is predicted to be .607. In general, the Poisson distribution requires that defects be uniformly and randomly distributed. Hence, it tends to be pessimistic for larger die sizes. Considering again

the binomial distribution, if the number of trials, n, is large, and the probability p of occurrence of an event is close to zero, then the binomial distribution is closely approximated by the Poisson distribution with $\lambda = n \cdot p$.

Another distribution commonly used to estimate yield is the *normal distribution*, also known as the *Gaussian distribution*. It is the familiar bell-shaped curve and is expressed as

$$P(k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(k-\mu)^2/2\sigma^2} \qquad (-\infty < k < \infty) \tag{1.6}$$

The variable μ represents the mean, σ represents the standard deviation, and σ^2 represents the variance. If n is large and if neither p or q is too close to zero, the binomial distribution can be closely approximated by a normal distribution. This can be expressed as

$$\lim_{n \to \infty} P\left(a \le \frac{x - np}{\sqrt{npq}} \le b\right) = \frac{1}{\sqrt{2\pi}} \int_a^b e^{-u^2/2} du \tag{1.7}$$

where np represents the mean for the binomial distribution, $\sqrt{npq}$ is the standard deviation, npq is the variance, and x is the number of successful trials.

When Murphy investigated the yield problem in 1964, he observed that defect and particle densities vary widely among chips, wafers, and runs. Under these circumstances, the Poisson model is likely to underestimate yield, so he chose to use the normalized probability distribution function. To derive a yield equation, Murphy multiplied the probability distribution function with the probability p that the device was good, for a given defect density D, and then summed that over all values of D, that is,

$$Y = \int_0^\infty pf(D)dD \tag{1.8}$$

He substituted $p = e^{-Da}$ for the probability that the device was good. However, he could not integrate the bell-shaped curve, so he approximated it with a triangle function. This gave

$$Y = \left(\frac{1 - e^{-D_0 A}}{D_0 A}\right)^2 \tag{1.9}$$

By substituting other expressions for $f(D)$ in Eq. (1.8), other yield equations result. Seeds used an exponential distribution function $f(D) = e^{-D/D_0}/D_0$. Substituting this into Eq. (1.8), he obtained

$$Y = \frac{1}{1 + D_0 A} \tag{1.10}$$

In 1973 Charles Stapper[15] derived a yield equation that is often referred to as a negative binomial distribution. By substituting $p(x) = e^{-\lambda}\lambda^x/x!$ and the gamma

distribution function $f(\lambda) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}}\lambda^{\alpha-1}e^{-\lambda/\beta}$ into Murphy's equation [Eq. (1.8)] and integrating, he obtained

$$Y = (1 + D_0 A/\alpha)^{-\alpha} \tag{1.11}$$

The mean of the gamma function is given by $\mu = \alpha/\lambda$, whereas the variance is given by α/λ^2. Compare these with the mean and variance of the negative binomial distribution, sometimes referred to as Pascal's distribution: mean = nq/p and variance = nq/p^2.

The parameter α in Eq. (1.11) is referred to as the cluster parameter. By selecting appropriate values of α, the other yield equations can be approximated by Eq. (1.11). The value of α can be determined through statistical analysis of defect distribution data, permitting an accurate yield model to be obtained.

1.7 MEASURING TEST EFFECTIVENESS

In this chapter the intent has been to survey some of the many approaches to digital logic test. The objective is to illustrate how these approaches fit together to produce a program targeted toward product quality. Hence, we have touched only briefly on many topics that will be covered in greater detail in subsequent chapters. One of the topics examined here is fault modeling. It has been the practice, for over three decades, to resort to the use of stuck-at models to imitate the effects of defects. This model was more realistic when (small-scale integration) (SSI) was predominant. However, the stuck-at model, for practical reasons, is still widely used by commercial tools. Basically put, this model assumes that an input or output of a logic gate (e.g., an inverter, an AND gate, an OR gate, etc.) is stuck to a logic value 0 or 1 and is insensitive to signal changes from the signal that drives it.

With this faulting mechanism the process, in rather general terms, proceeds as follows: Computer models of digital circuits are created, and faults are injected into the model. The fault-free circuit and the faulted circuit are simulated. If there is a difference in response at an observable I/O pin, the fault is classified as detected. After many faults are evaluated in this manner, fault coverage is computed as

Fault coverage = No. faults detected / No.faults modeled

Given a fault coverage number, there are two questions that occur: How accurate is it, and for a given fault coverage, how many defective chips are likely to become tester escapes? Accuracy of fault coverage will depend on the faults selected and the accuracy of the fault model relative to real defect mechanisms. Fault selection requires a statistically meaningful random sample, although it is often the practice to

fault simulate a universal sample of faults, meaning faults applied to all logic elements in a circuit. The fault model, like any model, is an imperfect replica. It is rather simplistic when compared to the various, complex kinds of defects that can occur in a circuit; therefore, predictions of test effectiveness based on the stuck-at model are prone to error and imprecision. The number of tester escapes will depend on the thoroughness of the test—that is, the fault coverage, the accuracy of that fault coverage, and the process yield.

The term *defect level* (DL) is used to denote the fraction of shipped ICs that are bad. It is computed as

$$DL = \text{Number of faulty units shipped} / \text{Total no. units shipped} \tag{1.12}$$

It has also been variously referred to as *field reject rate* and *reject ratio*. In this section we adhere to the terminology used by the original authors in their derivations.

Over the past two decades a number of attempts have been made to quantify the effectiveness of test programs—that is, determine how many defective chips will be detected by the tester and how many will slip through the test process and reach the end user. Different researchers have come up with different equations for computing defect level. The discrepancies are based on the fact that they start with different assumptions about fault distributions. Some of it is a result of basing results on different technologies, and some of it is a result of working with processes that have different quality levels, different failure mechanisms, and/or different defect distributions. We present here a survey of some of the equations that have been derived over the years to compute defect level as a function of process yields and test coverage.

In 1978 Wadsack[16] derived the following equation:

$$yr = (1 - f) \cdot (1 - y) \tag{1.13}$$

where yr denotes the field reject rate—that is, the fraction of defective chips that passed the test and were shipped to the customer. The variable y, $0 \le y \le 1$, denotes the actual yield of the process, and f, $0 \le f \le 1$, denotes the fault coverage. In 1981 Williams and Brown developed the following equation:

$$DL = 1 - Y^{(1-T)} \tag{1.14}$$

In this equation the field reject rate is DL (defect level), the variable Y represents the yield of the manufacturing process, and the variable T represents the test percentage where, as in Eq. (1.13), each of these is a fraction between 0 and 1.

Example If it were possible to test for all defects, then

$$f = 1 \quad \text{and} \quad yr = (1 - 1) \cdot (1 - y) = 0 \qquad \text{from Eq. (1.13)}$$

$$T = 1 \quad \text{and} \quad DL = 1 - Y^{(1-1)} = 0 \qquad \text{from Eq. (1.14)}$$

On the other hand, if no defective units were manufactured, then

$$y = 1 \quad \text{and} \quad yr = (1 - f) \cdot (1 - 1) = 0 \qquad \text{from Eq. (1.13)}$$
$$Y = 1 \quad \text{and} \quad DL = 1 - 1^{(1-T)} = 0 \qquad \text{from Eq. (1.14)}$$

In either situation, no defective units are shipped, regardless of which equation is used. ■■

For either of these equations, if the yield is known, it is possible to find the fault coverage required to achieve a desired defect level. Using Eq. (1.14), the test fraction T is

$$T = 1 - \frac{\log(1 - DL)}{\log(Y)} \tag{1.15}$$

Example Integrated circuits (ICs) are manufactured on wafers—round, thin silicon substrates. After processing, individual ICs are tested. The wafer is diced and the die that tested bad are discarded. If the yield of good die is 60%, and we want a defect level not to exceed 0.1%, what level of testing must we achieve? Using Eq. (1.15), we get

$$T = 1 - \frac{\log(1 - 0.001)}{\log(0.6)} = 1 - 0.001956 = 0.9980$$ ■■

This equation is pessimistic for VLSI. In later paragraphs we will look at other equations that, based on clustering of faults, give more favorable results. Nevertheless, this equation illustrates an important concept. Test cost is not a linear function. Experience indicates that test cost follows the curve illustrated in Figure 1.4.

This curve tells us that we reach a point where substantial expenditures provide only marginal improvement in testability. At some point, additional gains become exorbitantly expensive and may negate any hope for profitability of the product. However, looking again at Eq. (1.14), we see that the defect level is a function of both testability and yield. Therefore, we may be able to achieve a desired defect level by improving yield.

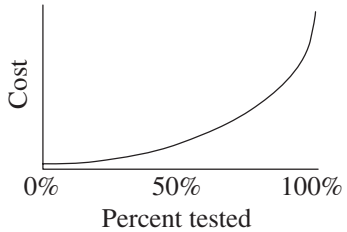

Figure 1.4 Typical cost curve for testing.

Example Yield is improved to $Y = 70\%$; what percentage of testing must be achieved to hold DL below 0.1%?

$$T = 1 - \frac{\log(1 - 0.001)}{\log(0.7)} = 1 - 0.0028 = 0.9972 \qquad \blacksquare\blacksquare$$

Equations (1.13) and (1.14) give the same results at the endpoints, but slightly different results between the endpoints. To understand why, it is necessary to look at the assumptions behind the derivations. Wadsack assumes that $yi = (1 - y)^i$, where yi represents the chips with i faults and y represents the actual functional yield. Williams and Brown assume the existence of n faults, that all faults have equal probability P_n of occurrence, and that the number of chips with i faults is

$$\binom{n}{i}(1 - P_n)^{n-i} P_n^i$$

Working out the derivations from these different starting points results in the different equations. However, regardless of which equation is used, the key point is that, in order to achieve an acceptable quality level AQL ($= 1 - DL$), the fault coverage has to be nearly perfect. In the words of Williams and Brown, the equations are intended to "give estimates for quick calculations." Wadsack, in his paper, points out that even in a circuit with 100% fault coverage, a failure occurred on the tester after the point where the test program had achieved 100% coverage of the faults. But then he points out that, in general, his derivation tends to be pessimistic.

Other authors have found the equations to be pessimistic; that is, even with fault coverage significantly less than that required by the equations, the quality level is better than predicted by the equations. For instance, Wiscombe[17] states that the Williams–Brown model "predicts higher defect levels than seen in practice." Maxwell et al. point out that for a defect level of less than 0.1%, the Williams–Brown equation required fault coverage in excess of 99.6%. However, they were able to realize those defect levels with about 96% fault coverage.[18]

The question of fault coverage versus defect levels was studied by Agrawal et al. in 1982.[19] Their study was motivated by the observation that the defect level equations "produced satisfactory results for chips with high yield (typically, SSI and MSI), but the predictions were too pessimistic for larger chips with lower yield." The authors hypothesize the existence of n faults for a faulty chip, and then examine the consequences of that assumption. They derive the following equation:

$$r(f) = \frac{(1 - f)(1 - y)e^{-(n_0 - 1)f}}{y + (1 - f)(1 - y)e^{-(n_0 - 1)f}} \tag{1.16}$$

In this equation, y is the yield, n_0 is the average number of faults on a faulty chip, f is the fault coverage, and $r(f)$ is the field reject rate for f. If the fault coverage is held fixed, then the defect level goes down as n_0 increases. The papers cited here suggest that the value $n_0 = 3$ appears to give reasonably good results at predicting defect level.

The model that was used to develop Eq. (1.16), referred to as the JSCC model, was subsequently refined using what the authors called the CAD model.[20] A Poisson

distribution is assumed for the faults, and the number of defects is assumed to have a clustered negative binomial distribution. With those assumptions the authors derived a reject ratio $r(f) = [y(f) - y]/y$, where

$$y(f) = [(1 + Ab(1 - e^{-cf})]^{-a} \tag{1.17}$$

In this equation, A is the chip area, f is the fault coverage, and a, b, and c are model parameters that are estimated by fitting $y(f)$ versus f to the experimental data.

In yet another derivation,[21] presented at a workshop in Springfield, Massachusetts, and referred to as the SPR model, the reject ratio $r_n = (y_n - y)/y_n$ is computed as a function of the yield y_n, after n vectors, and the true yield y. The variables y_n and y are computed as a function of the number of chips tested, the number of applied vectors, and the number of chips failing at vector i. The authors point out that the required data are derived from wafer probe. The calculations do not depend on estimated fault coverage of the test vectors. In this same study[21] the authors compare the five models for defect level estimation.

Comparison of the five models was done by gathering statistics on a high-volume chip at Delco Electronics. The chip was a 3-micron digital CMOS IC with 99.7% fault coverage. The test program consisted of 12,188 clock periods, and the cumulative fault coverage was computed after each vector. Of the 72,912 die initially considered, 847 chips that failed parametric test and 7699 chips that failed continuity test were removed from consideration. Of the remaining 64,366 chips, 18,476 failed the functional test. This resulted in an apparent yield of 71.30%. The true yield, using the SPR model, was estimated to be 70.92%. The results of the comparison are presented in Table 1.2.

In most columns the spread between these formulas varies by as much as a factor of two. The one exception is the last column, where the SPR and JSSC models differ by an order of magnitude. The bottom row of the table lists the actual fraction of defects detected at various stages of testing the chips. For the rightmost column, corresponding to a fault coverage of 99.70%, all the vectors had been applied, so no additional defects were found. However, each of the models predicts that additional tester escapes will occur.

TABLE 1.2 Comparing Yield

Model	Fault Coverage						
	20%	50%	80%	91%	95%	98%	99.70%
SPR	0.11291	0.08005	0.03531	0.02160	0.00927	0.00702	0.00532
JSSC	0.21383	0.11373	0.03730	0.01548	0.00834	0.00362	0.00048
CAD	0.21714	0.12439	0.04556	0.01985	0.01090	0.00432	0.00064
Wadsack	0.23267	0.14542	0.05817	0.02617	0.1454	0.00582	0.00087
Williams	0.24038	0.15788	0.06642	0.03046	0.01704	0.00685	0.00103
Actual	0.18440	0.08340	0.02830	0.01330	0.00740	0.00210	0

Although the Williams–Brown model tends to be the least accurate, at least for the data in this experiment, it appears to be the most popular, based on frequency of appearance in the literature. This may be due in large part to its simplicity, which makes it easy for engineers to explain the relationship between quality, process yield, and fault coverage. Perhaps, more significantly, any of these models can tell the user when the fault coverage must be improved. For example, if the user wants no more than 1000 defects per million (DPM), then all of these models convey the message that 98% fault coverage is insufficient.

The SPR model computes tester escapes without benefit of fault simulation. A drawback to this approach is the fact that, without fault coverage estimates for the test program, it could require several iterations on the test floor acquiring data before the test program is adequate. By contrast, when developing a test program with the aid of fault coverage estimates, it is more likely that the test will be at, or near, required coverage levels before it is used on the test floor.

Up to this point, when talking about fault coverage, the number used in the calculations was simply the number of modeled faults that were detected, divided by the total number of modeled faults. It has been assumed, for a given test coverage, that the coverage is uniform across the circuit. However, that may not be the case. Consider the test for a large chip, consisting of several functions. The test program may be a concatenation of several smaller test programs, each of which targets a single function. Suppose there are six clearly identifiable functions on the chip, then there might be six distinct test programs targeting the individual functions. The tests for five of the functions may be near 100%, while the test for the remaining function may be closer to 70%. Gross defects that might be detected in the other functions could escape detection in the function with low coverage.

Maxwell[22] showed that it is necessary to get a uniformly high coverage across the entire area of the chip. Also worth noting is the fact that each function may have some unique characteristics. For example, one function may be sensitive to noise. Another may use unique elements from a standard library, one or more of which are prone to failure. Conceivably a latch or flip-flop, for whatever reason, may have difficulty holding a particular state. These properties may not all be adequately addressed in one or more of the test programs.

Other investigations of defect levels have been performed. McCluskey and Buelow introduce the term *test transparency* (*TT*).[4] It is the fraction of all defects that are not detected by a test procedure:

$$TT = \text{defects not detected} / \text{total no. defects} = 1 - m/n$$

where n is the total number of defects and m is the number of defects detected. They show that, for $DL \leq 0.1\%$ and $Y \geq 90\%$, $DL = TT \cdot (1 - y)$. They state that it is customary to estimate test transparency by the percentage of single-stuck faults that are not detected by the test, $TT \geq 1 - T$, where T is the test coverage. Using $1 - T$ as an estimate for TT gives $DL = (1 - T) \cdot (1 - y)$, which is the Wadsack equation developed in 1978.

1.8 THE ECONOMICS OF TEST

In previous sections we examined some factors that affect the quality of test programs. In this section we examine factors that influence the cost of test. Quality and test costs are related, but they are not inverses of one another. As we shall see, an investment in a higher-quality test often pays dividends during the test cycle.

Test related costs for ICs and PCBs include both time and resource. As pointed out in previous sections, for some products the failure to reach a market window early in the life cycle of the product can cause significant loss of revenue and may in fact be fatal to the future of the product. The dependency table in Figure 1.5 shows test cost broken down into four categories[23]—some of which are one-time, non recurring costs whereas others are recurring costs. Test preparation includes costs related to development of the test program(s) as well as some potential costs incurred during design of the design-for-test (DFT) features. DFT-related costs are directed toward improving access to the basic functionality of the design in order to simplify the creation of test programs.

Many of the factors depicted in Figure 1.5 imply both recurring and nonrecurring costs. Test execution requires personnel and equipment. The tester is amortized over individual units, representing a recurring cost for each unit tested, while costs such as probe cards may represent a one-time, nonrecurring cost. The test-related silicon is a recurring cost, while the design effort required to incorporate testability enhancements, listed under test preparation as DFT design, is a nonrecurring cost.

The category listed as imperfect test quality includes a subcategory labeled as *tester escapes*, which are bad chips that tested good. It would be desirable for tester escapes to fall in the category of nonrecurring costs but, regrettably, tester escapes

		Personnel cost	Test card cost	Probe cost	Probe life	Depreciation	Volume	Tester setup time	Tester capital cost	Wafer radius	Die area	Wafer cost	Defect density
Test preparation	Test generation	*									*		
	Tester program	*											
	DFT design	*									*		
Test execution	Hardware		*	*	*	*	*						
	Tester	*				*	*	*	*				*
Test related silicon										*	*	*	
Imperfect test quality	Escape						*				*		*
	Lost performance						*						
	Lost yield						*						*

Figure 1.5 Cost/benefit dependencies of DFT.

are a fact of life and occur with unwelcome regularity. Lost performance refers to losses caused by increases in die size necessary to accommodate DFT features. The increase in die size may result in fewer die on a wafer; hence a greater number of wafers must be processed to achieve a given throughput. Lost yield is the cost of discarding good die that were judged to be bad by the tester.

The column in Figure 1.5 labeled "Volume" is a critical factor. For a consumer product with large production volumes, more time can be justified in developing a comprehensive test plan because development costs will be amortized over many units. Not only can a more thorough test be justified, but also a more efficient test—that is, one that reduces the amount of time spent in testing each individual unit. In low-volume products, testing becomes a disproportionately large part of total product cost and it may be impossible to justify the cost of refining a test to make it more efficient. However, in critical applications it will still be necessary to prepare test programs that are thorough in their ability to detect defects.

A question frequently raised is, "How much testing is enough?" That may seem to be a rather frivolous question since we would like to test our product so thoroughly that a customer never receives a defective product. When a product is under warranty or is covered by a service contract, it represents an expense to the manufacturer when it fails because it must be repaired or replaced. In addition, there is an immeasurable cost in the loss of customer goodwill, an intangible but very real cost, not reflected in Figure 1.5, that results from shipping defective products.

Unfortunately we are faced with the inescapable fact that testing adds cost to a product. What is sometimes overlooked, however, is the fact that test cost is recovered by virtue of enhanced throughput.[24] Consider the graph in Figure 1.6. The solid line reflects quality level, in terms of defects per million (DPM) for a given process, assuming no test is performed. It is an inverse relationship; the higher the required quality, the fewer the number of die obtainable from the process. This follows from the simple fact that, for a given process, if higher quality (fewer DPM) is required, then feature sizes must be increased. The problem with this manufacturing model is that, if required quality level is too high, feature sizes may be so large that it is impossible to produce die competitively. If the process is made more aggressive, an increasing number of die will be defective, and quality levels will fall. Point A on the graph corresponds to the point where no testing is performed. Any attempt to shrink the process to get more units per wafer will cause quality to fall below the required quality level.

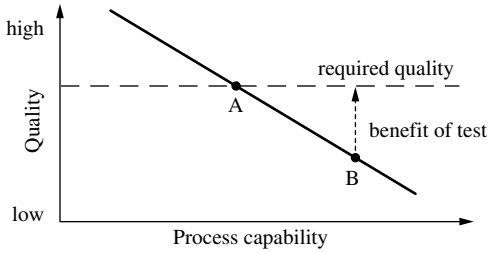

Figure 1.6 The benefits of test.

However, if devices are tested, feature sizes can be reduced and more die will fit on each wafer. Even after the die are tested and defective die are discarded, the number of good die per wafer exceeds the number available at the larger feature sizes. The benefit in terms of increasing numbers of good die obtainable from each wafer far outweighs the cost of testing the die in order to identify those that are defective.

Point B on the graph corresponds to a point where process yield is lower than the required quality level. However, testing will identify enough defective units to bring quality back to the required quality level. The horizontal distance from point A to point B on the graph is an indication of the extent to which the process capability can be made more aggressive, while meeting quality goals. The object is to move as far to the right as possible, while remaining competitive. At some point the cost of test will be so great, and the yield of good die so low, that it is not economically feasible to operate to the right of that point on the solid line.

We see therefore that we are caught in a dilemma: Testing adds cost to a product, but failure to test also adds cost. Trade-offs must be carefully examined in order to determine the right amount of testing. The right amount is that amount which minimizes total cost of testing plus cost of servicing or replacing defective components. In other words, we want to reach the point where the cost of additional testing exceeds the benefits derived. Exceptions exist, of course, where public safety or national security interests are involved.

Another useful side effect of testing that should be kept in mind is the information derived from the testing process. This information, if diligently recorded and analyzed, can be used to learn more about failure mechanisms. The kinds of defects and the frequency of occurrence of various defects can be recorded and this information can be used to improve the manufacturing process, focusing attention on those areas where frequency of occurrence of defects is greatest.

This test versus cost dilemma is further complicated by "time to market." Quality is sometimes seen as one leg of a triangle, of which the other two are "time to market" and "product cost." These are sometimes posited as competing goals, with the suggestion that any two of them are attainable.[25] The implication is that quality, while highly desirable, must be kept in perspective. *Business Week* magazine, in a feature article that examined the issue of quality at length, expressed the concern that quality could become an end in itself.[26]

The importance of achieving a low defect level in digital components can be appreciated from just a cursory look at a typical PCB. Suppose, for example, that a PCB is populated with 10 components, and each component has a defect level $DL = 0.999$. The likelihood of getting a defect free board is $(0.999)^{10} = 0.99004$; that is, one of every 100 PCBs will be defective—and that assumes no defects were introduced during the manufacturing process. If several PCBs of comparable quality go into a more complex system, the probability that the system will function correctly goes down even further.

Detecting a defective unit is often only part of the job. Another important aspect of test economics that must be considered is the cost of locating and replacing defective parts. Consider again the board with 10 integrated circuits. If it is found to be defective, then it is necessary to locate the part that has failed, a time-consuming and

error-prone operation. Replacing suspect components that have been soldered onto a PCB can introduce new defects. Each replaced component must be followed by retest to ensure that the component replaced was the actual failing component and that no new defects were introduced during this phase of the operation. This ties up both technician and expensive test equipment. Consequently, a goal of test development must be to create tests capable of not only detecting a faulty operation but to pinpoint, whenever possible, the faulty component. In actual practice, there is often a list of suspected components and the objective must be to shorten, as much as possible, that list.

One solution to the problem of locating faults during the manufacturing process is to detect faulty devices as early as possible. This strategy is an acknowledgment of the so-called *rule-of-ten*. This rule, or guideline, asserts that the cost of locating a defect increases by an order of magnitude at every level of integration. For example, if it cost N dollars to detect a faulty chip at incoming inspection, it may cost $10N$ dollars to detect a defective component after it has been soldered onto a PCB. If the component is not detected at board test, it may cost 100 times as much if the board with the faulty component is placed into a complete system. If the defective system is shipped to a customer and requires that a field engineer make a trip to a customer site, the cost increases by another power of 10. The obvious implication is that there is tremendous economic incentive to find defects as early as possible.

This preoccupation with finding defects early in the manufacturing process also holds for ICs.[27] A wafer will normally contain test circuits in the scribe lanes between adjacent die. Parametric tests are performed on these test circuits. If these tests fail, the wafer is discarded, since these circuits are far less dense than the circuits on the die themselves. The next step is to perform a probe test on individual die before they are cut from the wafer. This is a gross test, but it detects many of the defective die. Those that fail are discarded. After the die are cut from the wafer and packaged, they are tested again with a more thorough functional test. The objective? Avoid further processing, and subsequent packaging, of die that are clearly defective.

1.9 CASE STUDIES

Finally, we present the results of two studies into test thoroughness versus AQL and the consequences of decisions made with respect to test. The first is a classic study published in 1985 that serves to underscore the importance of achieving high fault coverage. The second is a study into the economics of multi-chip modules (MCMs). A model was created and parameters were varied in order to discern their effect on total product cost.

1.9.1 The Effectiveness of Fault Simulation

In this study, the results of which are shown in Figure 1.7, the authors were concerned with the fact that at 96.6% fault coverage they were still getting too many field rejects, and the costs of packaging and test were excessive.[4,28] A decision was made to improve the test program and determine what impact that would have on the defect level.

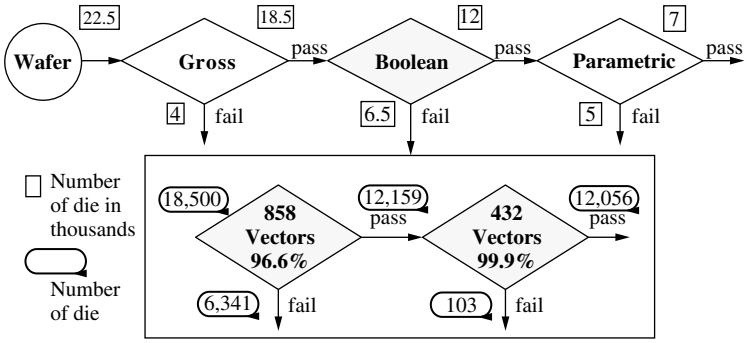

Figure 1.7 Fallout during test.

In their study, investigators analyzed 22,506 die. Of these, 4006 were eliminated at the start of testing because of failures due to gross defects, including opens, shorts, and so on. Then, 18,500 die were subjected to a functional test. The initial test consisted of 858 vectors that provided 96.6% fault coverage. This test identified 6341 failing devices. Over time, the initial test was increased to 992 vectors to address specific field reject problems encountered during production. During this study the test was enhanced by the addition of another 298 vectors to bring the total vector count to 1290. During their experiment, investigators recorded the vector number at which failures occurred. The original 858 vectors uncovered 6341 defective chips. The added 432 vectors uncovered an additional 103 defective chips.

1.9.2 Evaluating Test Decisions

The second study examined test decisions involving (MCMs). The MCM is a hybrid manufacturing technique in which several ICs are placed on an intermediate level of packaging. It can be used to package incompatible technologies such as CMOS and TTL, or it can be used to package digital circuits together with analog circuits that can't tolerate the noise generated by digital circuits. It can also be used to package digital circuits together with memory, such as cache memory, or it can be used to package two digital circuits that are either (a) too big to be placed on a single chip with existing technology or (b) those in which yield of a single, larger chip may be unacceptable. In this last instance, the MCM may be an intermediate phase until manufacturing advances permit the individual digital chips to be integrated onto a single die.

MCMs are often manufactured using known good die (KGD). The KGD is a bare die that has gone through extensive testing. In a normal flow, wafer sort is performed on individual die before they have been cut from the wafer. This is a test whose purpose is to identify, as quickly as possible, those die that are grossly defective. Then, those die that pass the test at wafer sort are packaged and tested more thoroughly. By contrast, KGD must be thoroughly tested on the wafer because they will be sold as

bare die, and the buyer will mount them directly onto the MCM without benefit of an additional layer of packaging. As a consequence of this approach, the MCMs that use these die must be processed in a clean room, which adds to manufacturing cost.

The cost of manufacturing MCMs is affected in significant ways by choices made with regard to test. Some of the factors include: chip yield and the thoroughness of test, the number of chips on the MCM, yield of the interconnect structure, yield of the bonding and assembly processes, and effectiveness of test and rework for detecting, isolating, and repairing defective modules. The High-Level Test Economics Advisor (Hi-TEA) evaluates decisions made with respect to these and other factors, including cost of materials and processes, yield parameters, and test parameters.[29] The metrics used by Hi-TEA are cost and quality: Hi-TEA attempts to optimize one while the other serves as a constraint.

The Hi-TEA user enters many parameters and/or assumptions into the system. Some of these inputs are easily obtained, such as the cost of labor and materials used to package and test the MCMs. Other costs are initially guesses, which can be refined as experience accumulates. In the paper cited here, the authors included several tables contrasting MCM cost versus chip AQL. One of the interesting results brought out was the trade-offs required to compensate for poor quality level of ICs used to populate the MCMs in some of their examples. It was also interesting to note that as AQL for the chips increased from 80% to 99.9%, total cost for MCMs followed a bell-shaped curve, first increasing, then decreasing, so that with 99.9% AQL, it cost less to manufacture MCMs that met a given AQL goal. Another byproduct of higher chip AQL was a significant reduction in the number of defective MCMs shipped to customers.

Figure 1.8 provides a summary of test cost versus quality trade-offs for several different test and DFT strategies. The test vehicle for this study was an MCM that contained a CPU, a coprocessor, and ten 4-Mbit SRAM chips. The clock speed for this MCM was faster than that of any existing workstations at the time of the design. It was assumed that there would be three defects per square inch for the CMOS CPU and coprocessors, and six defects per square inch for the BICMOS SRAM wafers. It

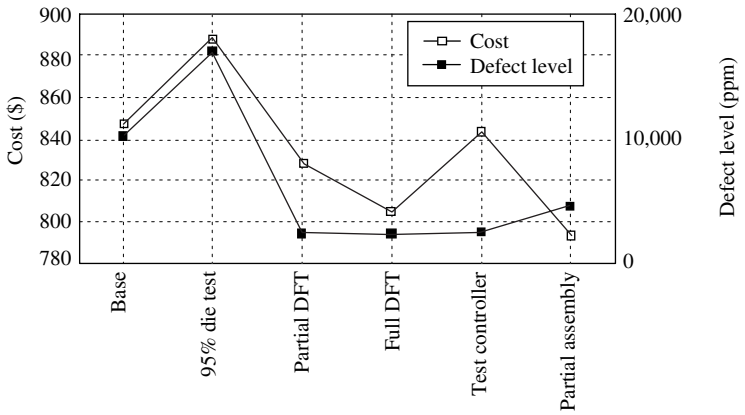

Figure 1.8 Cost/quality trade-offs for various test/DFT strategies.

was also assumed that 10% of the die would fail during burn-in. Test coverage at wafer probe was 80%, and coverage at the die level was 99%. Substrate yield was 99.999% and test coverage for MCM test was 95% of all possible defects, including faulty die, assembly errors, and so on.

From the base test, the next case reduced by half the test time for the die. As a result, the fault coverage for the die decreased from 99% to 95%. From Figure 1.8 it can be seen that, compared to the base case, final product cost increased by about 5% and defect level went up by almost 70%.

The next objective was to study the impact of DFT and built-in self-test (BIST) on the cost and quality of the MCMs. The first experiment involved adding DFT and BIST to the CPU and coprocessor. Compared to the base case, the use of partial DFT reduced defect level from 10,000 to about 3000 ppm while reducing cost from $845 to about $830. For the full DFT case the defect level remained about the same as with the partial DFT case, but cost fell to about $805. An advantage that did not get factored into these computations is the availability of the DFT features at higher levels of integration, such as systems test.

The use of a test controller on the MCM was intended to evaluate the situation where the manufacturer has no control over the ICs used in the design. In this scenario, the test controller provides greater access to the individual chips on the MCM. The cost of the additional test controller chip added $60 to the cost of the MCM, but its presence helped to reduce the overall test cost slightly when compared to the base case. The defect level was reduced by almost 80% relative to the base case.

The final scenario considered testing the MCM after the SRAMs were attached. If defects were encountered, they were repaired and the MCM retested. Then, when the partial assembly passed the test, the CPU and coprocessor were mounted and the MCM was retested. In this scenario the SRAMs can be considered hardcore (cf. Section 9.7.1) and used to test the remaining logic on the MCM. Because diagnosis is improved, it is less expensive to isolate defects and make repairs. Special fixtures can be created to improve access to test points on the MCM. Note that this case provides the lowest overall cost of the MCM, although the defect level is slightly higher than when DFT is used.

1.10 SUMMARY

During the past three decades a great deal of research has gone into the various facets of IC design, including system architectures, equipment used to create digital circuits with ever-shrinking feature sizes, and EDA tools used to facilitate the migration from concept to digital product. Along the way, quality has benefited from a better understanding of defect mechanisms, the development of better test methods to identify and diagnose the causes of defects, and a better understanding of the technical and economic trade-offs required to achieve desired quality levels.

Product reliability is another beneficiary as digital products have migrated from SSI (small-scale integration), through very-large-scale integration (VLSI), into deep

submicron (DSM). Greater integration has resulted in fewer assembly steps and fewer soldering joints. As far back as 1979 it was reported that, based on five billion device hours of experience, LSI devices with 70 to 100 gates per chip experienced twice the failure rate of SSI devices with four to eight gates per chip. Put another way, LSI devices experienced one-seventh the failure rate of SSI devices, on a per-gate basis.[30] CMOS technology, running at much lower power levels than equivalent circuits implemented in previous technologies (ECL, TTL, etc.), has contributed to improved reliability.

As the IC industry matures, and engineers gain a better understanding of the many factors that contribute to yield loss, they are able to apply this new-found knowledge to reduce both the sizes and the numbers of defects that occur in a given die area, with the result that yields increase. This is all the more remarkable in view of the fact that feature sizes continue to shrink and chip complexity continues to increase. A relationship between complexity and minimum defect size is suggested in Figure 1.9, where trends are projected to the year 2010.[31]

The incentive to shrink die size is motivated by a rather basic imperative, improved profitability.[32] Consider a wafer with N die and a yield Y. There will be $Y \times N$ good die on the wafer. Each of these will be sold for Z dollars, producing an income of $Y \times N \times Z$. This income must exceed the cost of designing, manufacturing, packaging, testing, and marketing the chips. If die size is reduced, there will be more die on each wafer, but the number of bad die may increase. If shrinking the die size causes a disproportionately larger increase in the number of good die, then income increases, assuming production costs do not go up disproportionately. Given a fixed selling price, then, the object is to find die size and yield that maximize the product term $Y \times N \times Z$.

A simplistic analysis could lead to the conclusion that the number of good die must increase disproportionately. Consider the following: If there were simply a fixed number of point defects on a wafer, and they caused $(1 - Y)$ die to fail, then doubling the number of die on a wafer would produce $N + (1 - Y) \times N$ good die. In effect, the overall yield increases. However, it is not quite that simple.

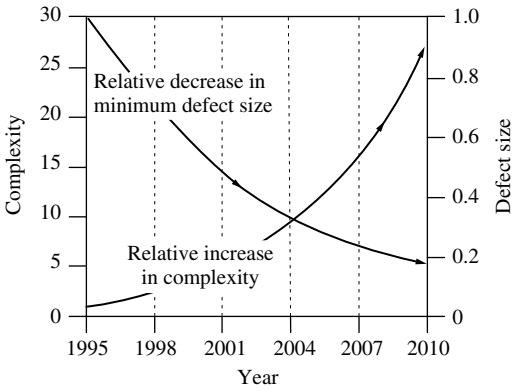

Figure 1.9 Complexity versus defect size.

As feature sizes shrink, supply voltages are reduced. This reduces power consumption, heat dissipation, and failures caused by electric fields greater than the circuit can tolerate. But, reducing the supply voltage increases gate delay and thus reduces the maximum clock rate. To compensate for this, the threshold voltage (the voltage at which the transistor turns on) is reduced. If the threshold voltage is reduced too far, leakage current becomes excessive. It is estimated that for every 60 mV that the threshold is lowered, leakage current increases by an order of magnitude.[33] New failure mechanisms may be introduced into the process. Lower operating voltages imply less noise margin. Traces on the die are closer together, resulting in greater potential for crosstalk. Greater capacitive coupling exists. Also, some point defects on the wafer that may not have been problems at larger feature sizes may become problems as feature sizes are reduced.

In summary, processes are improving, but as long as the universe is subject to entropy, defects will continue to occur. The existence of defects implies a need for test programs capable of detecting them, whether it be for reducing field rejects or to help debug first silicon. The existence of chips with larger gate count implies a need to develop more efficient test programs. The emergence of new fault mechanisms implies a need for new test algorithms targeting those fault mechanisms. Furthermore, the ability to accurately compute defect level is important because it tells us that, given levels of testability and yield beyond which we cannot hope to improve (economically), we must expect a certain percentage of defective units shipped and plan our business strategy accordingly, whether it be to stock more spare parts or to improve our service department.

Another factor that has grown in importance in recent years is end-user expectations. In 1994, when a floating point problem was encountered in early Pentium processors, the first inclination by Intel Corp. was to downplay the significance of the problem, asserting that a typical user might only encounter an incorrect calculation once every 27 years. The outcry far exceeded anything that was anticipated by Intel. They found that in order to maintain a favorable public image, it was necessary to establish a generous return policy for anyone with a Pentium based microprocessor system. The resulting message from this experience is that, with electronic products more pervasive than ever in many different end-user products, there is a less forgiving public unwilling to understand or tolerate defective products. One slip by a major vendor, and there will be another company waiting in the wings, ready to step in and exploit the opportunity.

It is interesting to note that the delivery of correct and reliable computing is influenced by factors that can be classified as nontechnical. For example, IBM's Server Group claims that the mean time between critical failures (MTBCF) of its System/390 mainframe is 20 to 30 years, where MTBCF is the average time between failures that force a reboot and initial program load.[34] A large part of the reason for this is because the core software is extremely stable, a change is implemented only if it is determined beyond all doubt that a bug exists. Of course, the hardware must also be stable.

One of the design parameters for a new system being developed is mean time before failure (MTBF). The goal is to keep a system up and running as long as possible. However, another parameter that often must be considered when developing a new

system is mean time to repair (MTTR). While it is desired not to have a system fail, in some circumstances it may be even more desirable to be able to get a system up and running again after it has failed. This may necessitate the inclusion of hardware whose sole purpose is to help diagnose and isolate failure to a field replaceable unit (FRU). Design-for-test or built-in self-test may be vitally necessary to achieve MTTR goals.

Change, and an urge for novelty, are key aspects of human existence, but sometimes these urges must be resisted. This ability to resist the urge to make changes unless it is absolutely necessary to do so is cited as a major reason for Intel's success. In an article in the *San Jose Mercury News*, the story is told of a drop in yield at one of Intel's foundries.[35] An investigation revealed that a processing change caused wafers to move more quickly from one station to the next. As a result, the temperature of the wafers as they arrived at the next station deviated from what it had previously been, and the deviation was enough to adversely affect the yield of the die on those wafers.

This drop in yield was notable because Intel reportedly practices a policy called "Copy Exactly." This practice involves building a fabrication plant as part of the research and development process for a new product. The R&D process involves not just the designers of a next generation chip, but also the people in manufacturing who must fabricate and test it. Once a manufacturing process is put into place, changes are not made until after considerable debate and considerable examination of the data. This is basically an implementation of *concurrent engineering*, which is defined as "a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support."[36]

An appreciation for the relationship between test cost, yield, and reject rate can be gained by considering an analogous situation in the field of communications. When communicating through a noisy medium, communications can be made more reliable by increasing transmission power. However, Shannon's theorem for communications in a noisy channel tells us that it is possible to make the transmission error rate arbitrarily small by resorting to error correcting codes (ECC). The most economic solution is found by factoring in both the cost of transmission power and the cost of employing ECC circuitry to find a solution that allows the most reliable communication at the highest possible rate, at the lowest possible cost.

Consider that the objective, when processing wafers, is to ship only good die. If field reject rate is too high, it could be improved by resorting to larger feature sizes. However, it can also be improved by employing a more thorough test that identifies more of the defective die before they are shipped to customers. The most economic solution is a complex function of process yield and test coverage.

PROBLEMS

1.1 For a semiconductor process with a yield $Y = 0.7$, compute the defect level DL by means of Eqs. (1.13) and (1.14) for values of T equal to 0.7, 0.8, 0.9, and 0.975. Repeat using Eq. (1.16), with values of n_0 equal to 1 and 3. Repeat all calculations for $Y = 0.9$.

1.2 Assume that the relative cost, C_d, of diagnosing and repairing defects, expressed as a function of the percentage t of faults tested, is $C_d = 100 - 0.7t$. Furthermore, assume that the cost C_p of achieving a particular test percentage t is $C_p = \frac{t}{100 - t}$. What value of t will minimize total cost?

1.3 Using Eq. (1.14), draw a graph of defect level versus fault coverage using each of the following values of yield as a parameter: $Y = \{.40, .50, .70, .90, .95\}$.

1.4 Using Eq. (1.5), calculate $P(0)$ for $\lambda_0 = \{.25, .5, .75, 1.0, 2.0\}$. Repeat using Eq. (1.10) and assume $D_0A = \{.25, .5, .75, 1.0, 2.0\}$. Repeat using Eq. (1.11), for $\alpha = 2$ and for $\alpha = 4$.

1.5 Assume two randomly distributed defects per square inch, and assume that each defect only affects one die. If there are four die on each square inch of wafer, what is the yield? If feature sizes are shrunk so that there are nine die per square inch, what is the yield?

1.6 Assume that the maximum allowable reject rate for a particular IC is 500 ppm. Use Eq. (1.5) to draw a graph of yield versus fault coverage for values of $n_0 = 0, 1, 2, 3, 4, 5$.

1.7 Given an MCM with 20 die, each of which has an AQL of 99.5%, what is the probability of a fault-free MCM?

REFERENCES

1. Doyle, E. A. Jr., How Parts Fail, *IEEE Spectrum*, October 1981, pp. 36–43.
2. Williams, T. W., and N. C. Brown, Defect Level as a Function of Fault Coverage, *IEEE Trans. Comput.*, Vol. C-30, No. 12, December 1981, pp. 987–988.
3. Rechtin, Eberhardt, The Synthesis of Complex Systems, *IEEE Spectrum*, July 1997, Vol. 34, No. 7, pp. 51–55.
4. McCluskey, E. J. and F. Buelow, IC Quality and Test Transparency, *Proc. Int. Test Conf.*, 1988, pp. 295–301.
5. Donlin, Noel E., Is Burn-in Burned Out?, *Proc. Int. Test Conf.*, 1991, p. 1114.
6. Henry, T. R., and Thomas Soo, Burn-in Elimination of a High Volume Microprocessor Using I_{DDQ}, *Proc. IEEE Int. Test Conf.*, 1996, pp. 242–249.
7. Weber, Samuel, Exploring the Time to Market Myths, *ASIC Technol. News*, Vol. 3, No. 5, September 1991, p. 1.
8. Teichrow, D., and E. A. Hershey, III, PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems, *IEEE Trans. Software Eng.*, Vol. SE-3, No. 1, January 1977, pp. 41–48.
9. Bell, C. G., and A. Newell, *Computer Structures: Readings and Examples*, McGraw-Hill, New York, 1971.

10. Davis, A. M., and D. A. Leffingwell, Using Requirements Management to Speed Delivery of Higher Quality Applications, Technical Report 0001, Requisite, Inc., http://www.requirement.com/requisite.
11. Sanborn, J. L., Evolution of the Engineering Design System Data Base, *Proc. 19th D.A. Conf.*, 1982, pp. 214–218.
12. Cunningham, J. A., The Use and Evaluation of Yield Models in Integrated Circuit Manufacturing, *IEEE Trans. Semicond. Mfg.*, Vol. 3, No. 2, May 1990, pp. 60–71.
13. Seeds, R. B., Yield and Cost Analysis of Bipolar LSI, *Proc. IEEE IEDM*, Washington, D.C., October 1967.
14. Murphy, B. T., Cost-Size Optima of Monolithic Integrated Circuits, *Proc. IEEE*, Vol. 52, December 1964, pp. 1537–1545.
15. Stapper, C. H., Defect Density Distribution for LSI Yield Calculations, *IEEE Trans. Electron Devices*, Vol. ED-20, July 1973, pp. 655–657.
16. Wadsack, R. L., Fault Coverage in Digital Integrated Circuits, *Bell Syst. Tech. J.*, May–June 1978, pp. 1475–1488.
17. Wiscombe, Paul C., A Comparison of Stuck-at Fault Coverage and I_{DDQ} Testing on Defect Levels, *Proc. Int. Test Conf.*, 1993, pp. 293–299.
18. Maxwell, P. C., R. C. Aitken, V. Johansen, and I. Chiang, The Effect of Different Test Sets on Quality Level Prediction: When Is 80% Better than 90%?, *Proc. Int. Test Conf.*, 1991, pp. 358–364.
19. Agrawal, V. D., S. C. Seth, and P. Agrawal, Fault Coverage Requirement in Production Testing of LSI Circuits, *IEEE J. Solid-State Circuits*, Vol. SC-17, No. 1, February 1982, pp. 57–61.
20. Das, D. V., S. C. Seth, P. T. Wagner, J. C.Anderson, and V. D. Agrawal, An Experimental Study on Reject Ratio Prediction for VLSI Circuits: Kokomo Revisited, *Proc. 1990 Int. Test Conf.*, pp. 712–720.
21. Seth, S. C. and V. D. Agrawal, On the Probability of Fault Occurrence, in *Defect and Fault Tolerance in VLSI Systems*, ed. I. Koren, pp. 47–52, Plenum, New York, 1989.
22. Maxwell, Peter C., Reductions in Quality Caused by Uneven Fault Coverage of Different Areas of an Integrated Circuit, *IEEE Trans. CAD*, Vol. 14, No. 5, May 1995, pp. 603–607.
23. Wei, S., P. K. Nag, R. D. Blanton, A. Gattiker, and W. Maly, To DFT or Not to DFT?, *Proc. Int. Test Conf.*, 1997, pp. 557–566.
24. Aitken, R. C., R. K. Scudder, and P. C. Maxwell, Never Mind the Cost of Test—Look at the Value!, Test Cost Reduction Workshop, *SEMI 1997*, pp. D1–D5.
25. Young, Lewis H., *Electronic Business Today*, October 1995, p. 50.
26. *Business Week*, August 8, 1994.
27. Thompson, Tom, How to Make the World's Fastest CPUs, *Byte Magazine*, Vol. 22, No. 2, February 1997, pp. bona3–bona12.
28. Daniels, R. G., and W. C. Bruce, Built-In Self-Test Trends in Motorola Microprocessors, *IEEE Des. Test*, *Comput.,* April 1985, Vol. 2, No. 2, pp. 64–71.
29. Abadir, M. S., et al., Analyzing Multichip Module Testing Strategies, *IEEE Des. Test Comput.*, Spring 1994, Vol. 11, No. 1, pp. 40–52.
30. Slana, Matthew F., Workshop Report: Computer Elements for the 80's, *IEEE Comput.*, Vol. 12, No. 4, April 1979, p. 102.
31. Vallett, David P., IC Failure Analysis: The Importance of Test and Diagnostics, *IEEE Des. Test*, July–September 1997, Vol. 14, No. 3, pp. 76–82.

32. Oldham, William G., The Fabrication of Microelectronic Circuits, *Sci. Am.*, September 1977, Vol. 237, No. 3, pp. 111–128.
33. Pountain, Dick, Amending Moore's Law, *Byte Magazine*, March 1998, pp. 91–95.
34. Halfhill, Tom R., Crash-Proof Computing, *Byte Magazine*, April 1998, pp. 60–74.
35. Gillmor, Dan, Curb on Tweaking Made Intel Strong, *San Jose Mercury News*, August 18, 1997, p. 1E.
36. Carter, Donald E., and B. S. Baker, Concurrent Engineering: The Product Development Environment for the 1990s, *Addison-Wesley, Reading, MA*, 1992.