1985

Cutting chip-testing costs

Sharad C. Seth
University of Nebraska-Lincoln, seth@cse.unl.edu

Vishwani D. Agrawal
AT&T Bell Laboratories Murray Hill, NJ

Published in: IEEE Spectrum (Volume: 22, Issue: 4, April 1985)
DOI: 10.1109/MSPEC.1985.6370618

Cutting chip-testing costs

Designing VLSI circuits for testability is the most efficient way to reduce the relative costs of assuring high chip reliability

Most engineers would agree that the quality of an integrated circuit depends partly on the ability to test it. But many chips now hold over 10 000 devices, and the cost of testing tends to increase in proportion to the square of the number of devices on the chip. The problem of containing testing costs while ensuring chip quality is one that all semiconductor manufacturers face.

If the line width of a semiconductor device shrinks from 2 micrometers to 1, the number of devices on a die of equal size could quadruple. Thus, the time—and the money—required to develop a computer program to test this chip could increase sixteenfold. Rising costs of chip testing run counter to the recent reductions in the cost of designing and producing chips.

Testing now accounts for 10 percent of the total cost of manufacturing a 1-kilobit random-access-memory chip. For a 64-K RAM chip, the figure rises to 40 percent. New techniques, however, promise help in the struggle to hold down costs, by tackling the circuit-testing problem in the design stage.

Advances on many fronts

The new methods include computer programs that assess during design how easily a circuit can be tested, scan-design techniques for testing sequential circuitry, and ways of partitioning chips into blocks of manageable size for testing. Random testing and built-in self-testing are also employed in some cases to avoid exhaustive testing for every possible fault in a circuit.

In addition, advances in circuit simulation allow engineers to estimate the fault coverage of test programs—that is, the proportion of the possible logic errors that a test will uncover [Fig. 1]. Without this estimate, engineers cannot know how rigorous to make their test programs, and they could overcompensate by making the programs more rigorous than needed—a waste of time and resources.

New approaches to testing have been used successfully with very large-scale integrated (VLSI) chips. They ensure that the cost of testing will increase linearly with circuit complexity—that is, doubling the number of devices on a chip will double, rather than quadruple, the cost of testing it. Computer-aided design has been a prime aid in developing the new methods.

Yet no testing technique is surefire for all kinds of chips; future generations of ICs will certainly require new approaches. In addition, many circuit designs currently pose special problems that no technique or combination of techniques seems to solve entirely. For now, chip manufacturers must live with methods of testing that are inadequate in some cases.

Testing is becoming more closely related to the design and production processes. In the days when the only ICs manufactured had no more than a few hundred devices, circuit-design engineers worked in isolation from test engineers, who became part of the manufacturing cycle only after the design was complete. Now test engineers work closely with design and production engineers to help keep test costs down. And in some places, a single engineer handles both testing and designing.

Sharad C. Seth University of Nebraska
Vishwani D. Agrawal AT&T Bell Laboratories

Measuring the testability

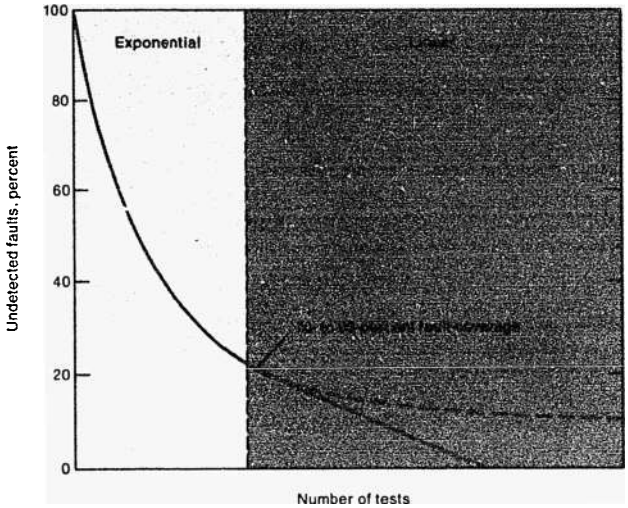
VLSI chips can be much easier to test than their size might indicate. Testability analyses during the design of a VLSI chip are simple ways of measuring how easy it will be to test a circuit. An overall testability measure for a circuit is derived by calculating the difficulty of testing each node in the circuit. Testability-analysis programs are computationally simpler than generating a test for a chip, and thus they can be used relatively quickly while a circuit is being designed on a computer. These rough measures of testability are used for almost all kinds of chips, from semicustom chips—made in small quantities—to custom microprocessors that are mass-produced.

Designers use testability measures to identify portions of a circuit that would be difficult to test. Such inaccessible circuits are said to have poor controllability or observability. Controllability is a rough measure of the ease with which a test engineer can control signals in a circuit from the input pins. Similarly, observability is a rough measure of the ease of determining the behavior of a circuit from the output pins. [See Fig. 2.] After identifying a general section of a chip that has poor controllability or observability, the engineer can then modify the circuit to make it more testable.

Defining terms

Controllability—a rough numerical measure of how easily the values of digital circuit nodes can be controlled from I/O pins.
Fault coverage—the percentage of potential stuck faults in an IC that are uncovered by a set of test vectors; it is usually obtained through computer simulation.
Observability—a rough numerical measure of how easily the values of digital circuit nodes can be determined from I/O pins.
Pattern generator—a circuit that generates a test pattern, usually for built-in testing; it may take any form, with random-number generators and ROMs being the most common.
Sequential circuit—a digital circuit that changes state according to an input signal (normally under clock control); it must be tested with a sequence of signals.
Stuck fault—usually a physical IC fault that results in one input or output of a logic gate improperly remaining either high or low regardless of the behavior of the circuits surrounding it.
Test program—a computer program written in the language of a particular automatic production tester for ICs.
Test vectors (test patterns)—a set of IC inputs and outputs generated for use in test programs.
Testability measure—a rough numerical indication of how easily test vectors can be generated for a particular circuit.

0018-9235/85/0400-0038$1.00 ©1985 IEEE



[1] The most difficult task in chip testing is generating a set of inputs and outputs, called test vectors, that will uncover close to 100 percent of all possible chip faults. At first the task goes smoothly, but when 70 to 80 percent of the faults have been detected, a change in test strategy, which entails extensive computer simulation, is required.

To get a rough measure of testability for a block of circuitry, the engineer computes the logarithm of the sum of the controllabilities and observabilities of all the nodes in a circuit. The resulting number, called a testability index, is proportional to the ultimate number of test vectors—inputs and outputs—needed to test a chip.

For example, a 50 000-gate microcontroller chip might have a testability index greater than 6, requiring 100 000 vectors to test for 90 percent of the faults; a programmable logic array with 2000 gates and an index of 5 may require only a few hundred vectors for the same fault coverage. The approximate length of a test can be predicted quite accurately in this way.

The most popular testability measurement program, the Scoap (Sandia Controllability and Analysis Program), calculates six quantities for every node (or signal) in the circuit, based on the effort needed to control and observe the node using a procedure such as the D-algorithm [see "Test design: stuck with the D-algorithm," p. 40].

The way in which a testability-measurement program operates depends on whether a circuit is largely sequential or largely combinational. A combinational circuit is basically a hierarchy of logic gates through which a signal will propagate in a single clock cycle. In such a circuit, controllability and observability are defined in terms of the number of logic gates that a test program must manipulate to either control or observe a node.

Sequential circuits generally have registers that must be clocked to allow signals to propagate. In these circuits, a series of state transitions must be made to control or observe a node. Thus, controllability and observability are defined in terms of the length of the sequence of inputs needed to control or observe a node.

Since the overall complexity of computation in Scoap increases almost linearly with the number of gates, the cost of using Scoap will increase only as quickly as chip area increases. Thus, for the next five years or so, Scoap is attractive because the cost of using it will not increase proportional to the square of chip area, which would be out of proportion to the cost of designing chips.

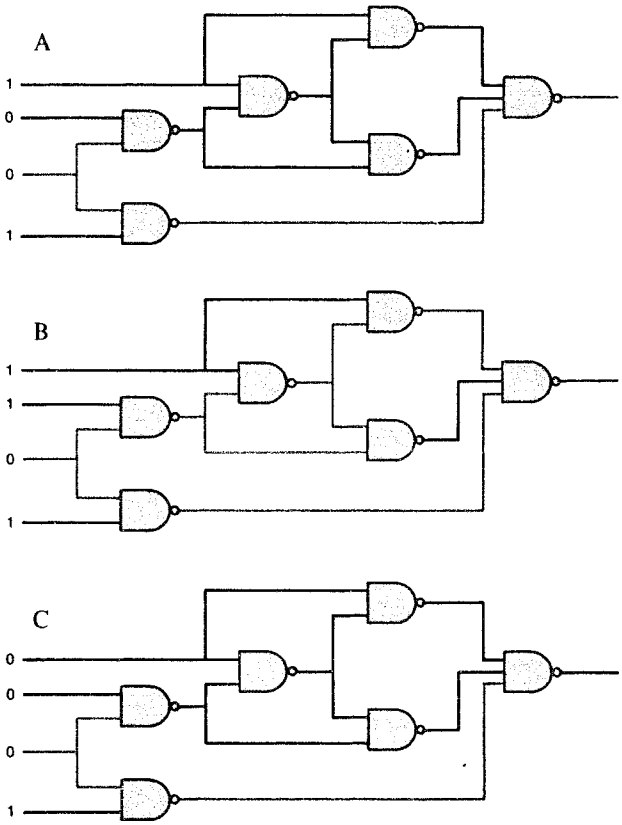
However, the usefulness of testability-measurement programs is limited. Analytical approaches have failed to relate the results of Scoap and other such programs to the fault coverage of particular circuit nodes; a design engineer cannot determine whether any given node in a circuit will be testable. Testability measures are poor predictors of which specific faults in a circuit will remain undetected and which will be detected in a test program. They are good, however, for indicating blocks of circuitry that may be hard to test.

Testability-analysis programs do not work well with sequential circuits because such programs are based on making some approximations when analyzing complex circuits. In combinational circuits, the complexity does not increase proportional to the size as much as in sequential circuits, for which the approximations cause great inaccuracies.

This limits the usefulness of testability-analysis programs, because most chips are a combination of both sequential and combinational circuits. Some circuits, such as microprocessors, are largely sequential, with relatively little combinational circuitry embedded in the chip.

Scan design uses artificial paths

The computation time for test generation and evaluation tends to grow at a rate approximately proportional to the square of the number of gates or the number of transistors in the circuit. Because of the greater complexity of sequential circuits, the cost of testing grows even faster than the circuit size—being proportional to the number of gates cubed. For example, a 4-bit arithmetic logic unit with about 100 gates—a typical combinational circuit—requires 30 test vectors for acceptable fault coverage. By contrast, a typical sequential circuit—a 4-bit multiplier with 350



[2] Testing a node in a circuit requires control and observation from the chip's I/O pins. First, a path to the node must be sensitized by setting the surrounding values so that the value of the test node can be changed or read from the I/O pins. In A and B, the red path has been sensitized by setting the values of three input pins to detect the fault at the node to the right through the remaining pin. In C, the path is not sensitized.

gates—requires 1100 test vectors for the same fault coverage. Even though the multiplier has only 3.5 times as many devices as the combinational circuit, more than 35 times as many vectors are needed to test it.

What can be done to hold down the cost of testing sequential circuits, which do not benefit much from testability measures?

For sequential circuits, chip designers are increasingly employing a technique known as scan design. Scan design gives the engineer access to sequential circuitry through artificial paths built into the circuit especially for that purpose. Through clever placement, the end effect of these artificial pathways is to convert sequential circuitry temporarily into combinational circuitry for testing purposes.

The method is analogous to test-driving automobiles. If an automobile is simple, the best way to test it is to drive it. However, if the automobile is very complex, with all sorts of electronic motor regulators and controls, test-driving alone may not be sufficient. It may be more effective to test the wiring and some of the individual components and then drive the complex automobile for only a short while. Similarly, testing complex sequential circuits requires tremendous effort—perhaps months of writing test vectors by hand.

With scan design, however, the engineer is not actually testing the circuitry by operating it as it was intended to operate in the field. Instead, the chips are designed to be put into a mode for testing each logic gate with a much simpler program than would otherwise be possible; an engineer can "check the wires" by way of the artificial pathways. Thus, with the aid of computer programs not available for sequential circuits, the engineer can usually generate test vectors in a few hours. In about one afternoon, scan design can generate the necessary vectors to test a 2000-gate digital demodulator chip at a fault coverage greater than 90 percent. Without scan, this would take a week. The benefit is even greater for more complex circuits.

Even before LSI circuits were made, engineers recognized that the problems in testing were significantly more complex for random sequential circuits than for combinational circuits of comparable size. The detection of a fault in a random sequential circuit often requires a long sequence of inputs to sensitize and observe a faulty node.

To test such circuits, engineers have tried, with little success, to model them from a purely functional rather than a structural viewpoint—in other words, they have tried to test the car simply by driving it. In this approach, sequential machines are represented by a state-transition table, which describes how the values of each node in the circuit change in response to clock cycles and certain test inputs. To model a circuit this way, checking must be done to make sure the test circuit actually behaves as the state-transition table says it should.

However, checking experiments tend to require unreasonably long computations, even for small circuits; the number of state transitions that need to be covered can be enormous. The solution is to modify the original circuit for test purposes so that the sequences of state transitions are short and easily derived. Scan design is the latest and most successful technique of this kind.

Creating 'normal' and 'scan' modes

Scan design requires that the circuit be designed with clocked flip-flops, or latches. When the chip is fabricated, it can be put into either a "normal" or a "scan" mode by way of an input/output pin especially designated for that purpose. In the normal mode, in which the circuit is largely inaccessible, the inputs are interconnected to form a sequential circuit that performs the intended function. However, in the scan mode, the latches are chained together to form shift registers. Digital test vectors are shifted into the register from a scan-in pin of the chip. With the test vectors thus implanted in the circuitry, the circuit can be switched back to the normal mode and tested. The circuit is then switched over to the scan mode, and the resulting values in the shift register are shifted out through a scan-out pin, which, like the mode pin, is added to the circuit for the sole purpose of testing.

Test design: stuck with the D-algorithm

New methods to simplify chip testing are needed partly because the basis of VLSI testing is essentially the same as that developed for the first ICs, which were extremely simple by today's standards. In essence, tests seek only two types of "classical faults" in both bipolar and MOS ICs: stuck-at-1 and stuck-at-0. A stuck fault is an input or an output of a logic gate that remains either a logical "1" or a logical "0" even if its value should change. Stuck faults are usually caused by an error in the fabrication process rather than by design errors, which have presumably been corrected by computer simulation by the time production tests are performed.

Although the model has little relation to the physical behavior of digital circuits and can only represent a subset of possible faults, experience has shown that a test program that uncovers about 95 percent of all possible stuck faults will yield a good-quality product. (Test engineers debate the percentage figure by a few points either way.)

Some engineers argue that the stuck-fault model is doomed to obsolescence because of the difficulty in automating test development for VLSI chips when the model is used. The old algorithms for generating test vectors to cover 95 percent of all stuck faults are unwieldy. For complex random circuits, the D-algorithm—developed by J.P. Roth and his colleagues at IBM Corp. almost 20 years ago—remains the prototype for most commercially feasible algorithms. The algorithm takes its names from the character "D" that is used by test engineers as a variable to show whether a circuit node is affected by a fault ($D = 0$) or not ($D = 1$).

To produce a set of test vectors, engineers reproduce all possible stuck faults on a computer-simulated circuit, and for each fault they invoke the D-algorithm to establish the appropriate value of D. This is repeated until a path is formed from the node of the circuit where the fault exists to an output pin of the chip. The term "sensitized" refers, in this case, to the observation of the test signal at the location of the fault, which the engineer obtains by manipulating the signals to the logic gates extraneous to the signal path. The D-algorithm is a recursive search procedure—advancing one gate at a time and repeating itself until the fault is detected.

As a path is advanced from an input to the output of a gate, values of other inputs to the gate may have to be set to a constant to allow the selected input to control the output—that is, to sensitize the path. These line values may, in turn, change the values of the inputs and outputs of other gates to which they are directly connected. Another recursive step is necessary to take into account all the implications of advancing the sensitized path through one gate. Inconsistencies caused by earlier assignments of gate inputs are discovered at this point. At any juncture, several alternatives might be available, of which the algorithm chooses one arbitrarily and records it in a stack. If the algorithm runs into a dead end, it retraces its steps by reading values off the stack and trying another alternative. This procedure is repeated until a consistent and sensitized path is found from the fault location to an output pin, which constitutes a valid test for one fault.

Clearly, the D-algorithm may involve a great deal of backtracking. In the worst case, it may have to examine all sensitized paths not only one at a time but in all possible combinations as well, because for some single faults it is necessary to sensitize several paths. In practice, the expected number of choices actually examined may be reduced by using heuristics, a method of ordering the choices that the D-algorithm

Scan design makes the generation of tests for sequential circuits easier, and it greatly reduces the number of transitions in the state-transition table that must be verified, thus reducing the task to manageable size. In addition, computer programs have been developed at AT&T Bell Laboratories and elsewhere in the last few years to automate the generation of test vectors for circuits using scan design.

The price that designers pay for using scan design is a requirement for additional logic in a circuit. The precise area needed for the additional circuitry is a matter of dispute, but most estimates fall between 10 and 20 percent of the chip size. This overhead, which degrades performance of the chip somewhat, may seem a high price to pay, but it results in much quicker test generation than most alternative ad hoc methods.

Scan design is finding acceptance in the industry for semicustom circuits such as gate arrays. Since the extensive automation in semicustom-circuit design makes the design process quicker than for handcrafted circuits, the time for generating test vectors must be held to a minimum; scan design can be implemented quickly by computers, and test vectors can be generated automatically. Scan design also eases systems testing for some manufacturers; hierarchical scan design, in which a scan path can be made from the box to the printed-circuit boards and down to individual chips, is not uncommon.

Mainly for economic reasons, scan design has not caught on in so-called commodity circuits, which are manufactured in high volumes for off-the-shelf use. Since commodity circuits, such as advanced microprocessors and other general-purpose chips, are mass-produced, manufacturers are willing to devote considerable resources to handcrafting the design of the chips to get the most yield from their wafers. Unlike semicustom designers, who use computer-aided design (CAD) to automate designs and layouts, the commodity-circuit manufacturers tend to use CAD to aid in hand designing. Since much time and money are spent simulating commodity-circuit designs, the manufacturers generate extensive data about the circuits that are useful for devising tests. They also use many custom techniques to obtain chip tests that cannot be computer-automated.

Furthermore, commodity-circuit manufacturers are entirely unlike semicustom manufacturers in their production test strategy, which does not favor scan-design techniques. Instead, they tend to rely on functional tests. For example, the 30 000-gate 32-bit microprocessor developed at Bell Labs is tested solely with functional techniques that execute the microprocessor instructions instead of finding stuck faults. The lifetime of a commodity chip, which may be five or more years, gives the manufacturers two advantages that negate some of the benefits of scan design: (1) they can afford to spend more time and effort to reduce the overhead for making the chip testable; and (2) they can refine the production test based on the chip failures reported by users of preproduction samples, usually original-equipment manufacturers. Semicustom designers, whose chips have relatively short production lifetimes, cannot rely as heavily on production experience to ensure chip quality.

Divide and conquer

Another design-stage technique for reducing the time and cost of testing large circuits is the "divide and conquer" approach. Often used in conjunction with other techniques, such as scan design, it requires no special circuitry. Rather than being designed as a monolith, a complex circuit is designed as an interconnection of modules, which may be further partitioned into submodules. This method is similar to the structural design of computer programs.

Ideally, a partitioned circuit would be designed with a test mode that would connect the inputs and outputs of each partitioned block to the output pins of the chip, so that the block could be observed. Test vectors would be multiplexed in the test mode through a set of input/output pins. The I/O pins would be used for each block in succession until the entire chip was tested.

Partitioning looks promising. However, the technique has not been widely adopted because the chip under test must be designed with independently testable partitions. So far, partitioning has been used mainly in circuits, such as microprocessors, that have architectures with natural partitions. At present, there is no economical way of imposing partitions on otherwise unstructured circuits.

However, partitioning is one element of two recently developed techniques that are proving quite useful in reducing costs: built-in testing and random testing.

The built-in self-testing (BIST) approach calls for partitioning a circuit into blocks during design; after the chip is fabricated, each block is exhaustively tested with a built-in pattern generator. The response to the pattern from the generator, which may run into millions of bits, is compressed into a "signature" of a relatively small number of bits. A multiple-input linear feedback shift register is used for this purpose, with feedback lines chosen carefully to ensure to a high degree of confidence that the signature is unique. An external control signal is introduced, as in scan design, to put the circuit into a test mode and to start the pattern generator. When the pattern ends, the contents of all the signature registers are compared with signatures stored in a read-only memory (ROM).

After the signatures in the different blocks are scanned with one output pin, the result is an indication that the circuit is either good or faulty.

The scan and BIST design methods complement each other and are often used in conjunction. Scan design solves test problems arising from the sequential nature of a circuit; BIST lessens the burden of generating and storing tests for complex combinational blocks of circuitry.

Random testing is useful for certain exceptional cases in which logic partitioning is not feasible, including gate arrays and other unstructured designs. Exhaustive testing of such circuits is impractical because of the large number of circuit inputs. Consider combinational logic implementing 32-bit multiplication:

makes at each circuit node.

The D-algorithm can be extremely time-consuming for deep circuits—large circuits in which a typical fault path would wind through 15 to 20 gates. In practice, the length of time for running a D-algorithm increases by $n^{1.8}$, where n equals the number of gates in the circuit.

The D-algorithm performs particularly poorly for circuits containing "exclusive-or" gates arranged in a tree structure, which is commonly found in circuits that check the parity of signals. If many bits are fed in parallel into a large exclusive-or gate, the gate will detect the occurrence of an error on any one of the inputs. In practice, a tree of double-input exclusive-or gates is often used instead of multi-input gates. The degradation in performance occurs because a very large number of possibilities may have to be examined for each gate in the recursive step to check the consistency of the process.

This shortcoming is counteracted, however, by other algorithms. The algorithm called Podem (path-oriented decision making) has been designed to minimize backtracking. For a 64-bit arithmetic-logic unit with about 2000 gates, for example, the D-algorithm takes 45 seconds for each test vector on a VAX 11/780 computer. Podem is six times faster. A further enhancement called Fan (for fan-out-oriented test-generation algorithm) is five times faster than Podem.

Even so, the fastest reported algorithms would typically consume 1500 seconds of CPU time on a VAX 11/780 computer to detect all the faults on a 3000-gate arithmetic and logic unit. For VLSI circuits with about 30 000 gates, test-vector-generation algorithms would take about 40 hours, which is not acceptable. —*S.C.S. and V.D.A.*

since a 32-bit multiplier has a total of 64 inputs, exhaustive testing would require 2^{64} test vectors—an astronomical number.

Recent analyses show that very high fault coverage can be attained by nonexhaustive random testing, in which test patterns are random bit patterns. Further, computational algorithms, for which execution times increase linearly with circuit size, can identify those faults not likely to be covered by random testing. Such faults can then be eliminated by redesigning the chip. Alternatively, BIST patterns may be generated and stored in a ROM to catch the remaining faults.

BIST techniques have not yet caught on in many areas of IC manufacturing because of the additional chip area, or overhead, occupied by the partitioning and the logic for internal testing.

The techniques are useful primarily for more complex circuits, such as the 200 000-transistor 68020 microprocessor of the Motorola Corp. Many engineers say that BIST will become more widespread when circuits with at least 100 000 gates become more common, for which about 5 million test vectors would probably be required. Motorola uses built-in testing in its 8-bit 6804P2 microcomputer, which has 17 800 transistors. About 5 percent of the total area of the chip is occupied by a 288-byte ROM to store test programs and a register to detect the signature.

Testing the tests to save time

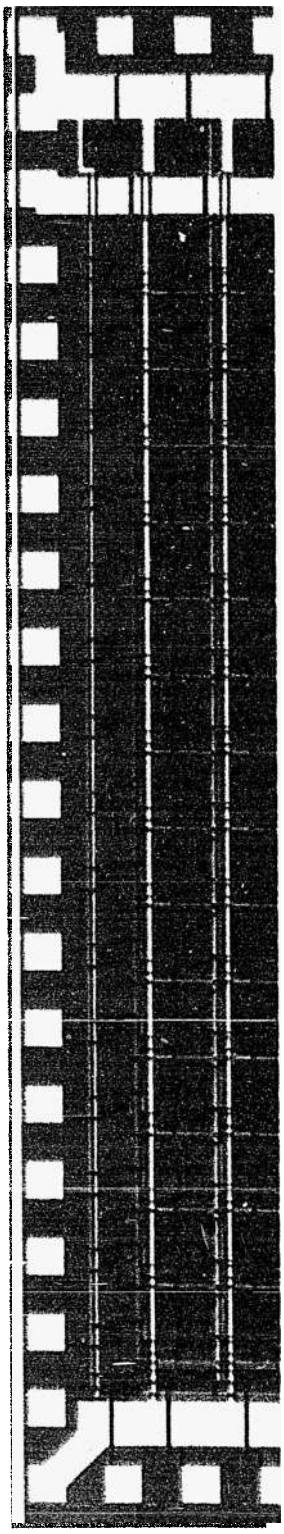
An estimate of the number of faults that will be uncovered by a chip test goes a long way toward reducing the time needed to generate tests. The overall process of generating a test program for a chip is like shooting at a progressively smaller target. The first set of patterns may test, say, 30 percent of the possible stuck faults of a circuit. An additional set of vectors to test another 30 percent of the chip would increase the total coverage to about 50 percent, after allowance is made for a 10 percent overlap. As the coverage increases, the value of each additional test decreases. At roughly 70 to 80 percent fault coverage, the test strategy is usually changed; at this point, specific nodes of the circuit that have not yet been tested are targeted with special algorithms for test generation and fault simulation. They push the fault coverage to 90 or 95 percent.

The key to estimating fault coverage in this way is computer simulation; only by simulating a circuit can the fault coverage of test vectors be evaluated. The test engineer simulates each possible stuck fault in a circuit to determine whether that fault was uncovered by the test. This indicates the fault coverage of the test vectors. Although the basic principles of production-quality fault simulators have been unchanged since the advent of LSI circuits, various techniques have greatly increased the speed of simulators.

The computation time of commercial simulators has been greatly reduced by simulating more than one fault for each input pattern. The simplest and most widely used technique is parallel fault simulation; for each input pattern, each bit of a computer word simulates a different fault.

Deductive fault simulators have further reduced the number of calculations. With such simulators, each line of logic gates is associated with a list of those faults that are sensitized to it—that is, the faults detected up to that point. The simulator refers to the list of faults already sensitized to primary outputs and then computes the fault coverage.

Even these advances in fault simulators have not completely solved the problem of pushing fault coverage to 70 to 80 percent without exorbitant computer simulation costs. The cost of network simulation has been estimated to grow at a rate proportional to the cube of the number of gates in a circuit for parallel fault simulations, and to the square of the number of gates for deductive simulation. The concurrent simulator, a refinement of the deductive simulator, takes 7.4 seconds of central-processing-unit time for each test pattern on a VAX11/780 system for a MOS circuit with 505 transistors. With a quadratic rise in simulation time, a 50 000-transistor circuit would require over 20 hours of simulation time for each test pattern on the same machine.



[3] Four methods of VLSI chip testing are gaining acceptance in the electronics industry. Testability analysis programs (A) allow designers to improve the planning of test strategies during the design of the chip. The other three—scan design (B), partitioning (C), and built-in self-test (D)—entail modifications in the chip design.

Fixed-purpose simulation engines, which have been announced recently, greatly reduce the simulation time for specific circuits by using hardware to concurrently execute different steps of simulation algorithms. What they sacrifice in flexibility, they gain in speed. One such system, the Logic Evaluator made by Zycad Corp., of St. Paul, Minn., has 16 hardware units, which can simulate 60 million active logic gates per second.

By using parallel execution techniques, fixed-purpose systems solve the speed problem for the short term. However, they use algorithms for which the execution times tend to increase as the cube of the number of gates in the circuit being simulated. For this reason, the cost of testing denser chips will increase quickly. Ultimately, test strategies will combine the use of such high-speed simulators with the other techniques described here.

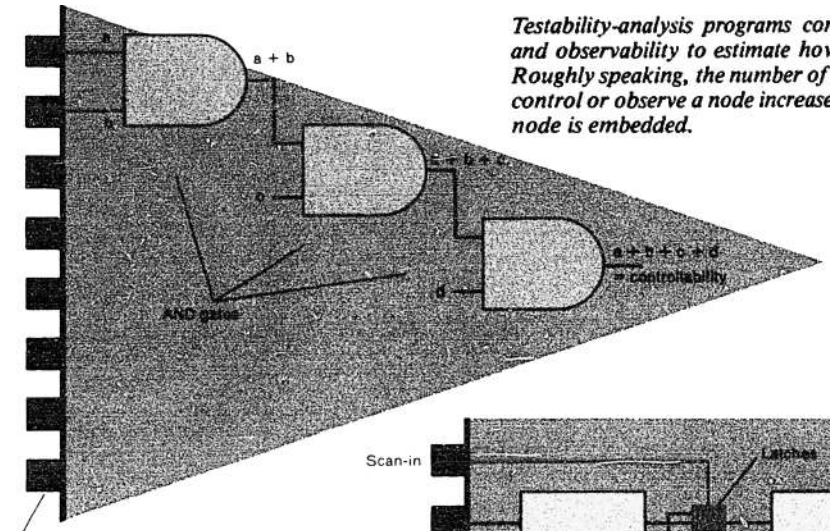
Test engineers have a third way of measuring fault coverage of test vectors before fabrication, in addition to design-for-testability techniques (such as scan design) and special-purpose hardware simulators. The third approach is statistical fault sampling, rather than deterministically checking each one.

Only a fraction of the possible faults are simulated in a statistical sampling technique to estimate fault coverage. The method is analogous to public opinion polls: randomly sampled faults are simulated, and the percentage of these faults that are detected by the set of test vectors is used as an estimate of the overall fault coverage.

The confidence range of these estimates gets narrower as the estimate of the fault coverage approaches 100 percent. For a sample of 1000 faults, an estimate of 50 percent fault coverage is accurate to within ± 5 percent, whereas an estimate of 95 percent is accurate to within ± 2 percent. Nevertheless, statistical sampling can effectively estimate any fault coverage.

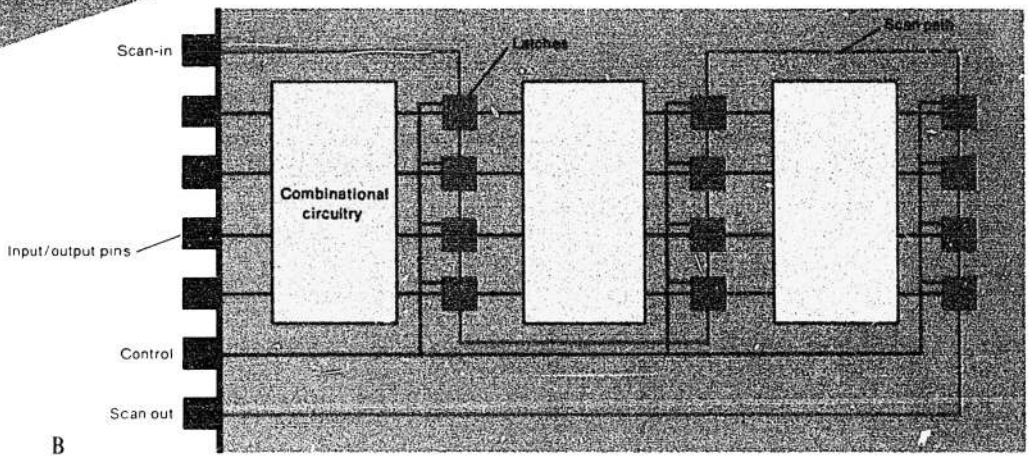
Not perfect

Regardless of whether stuck faults are simulated exhaustively or statistically sampled, the estimated fault coverage is only an



A

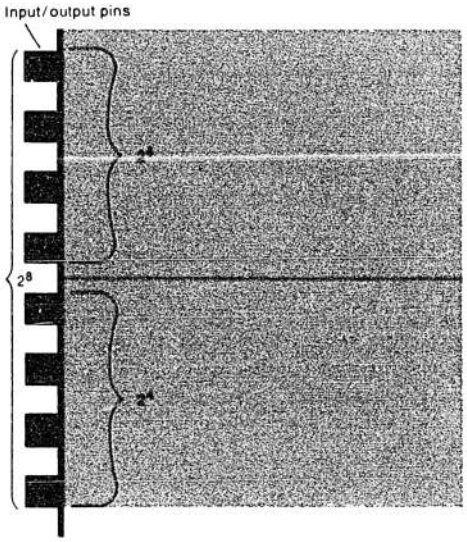
Testability-analysis programs compute the indexes of controllability and observability to estimate how easily a circuit can be tested (left). Roughly speaking, the number of I/O pins that must be manipulated to control or observe a node increases according to the depth to which the node is embedded.



B

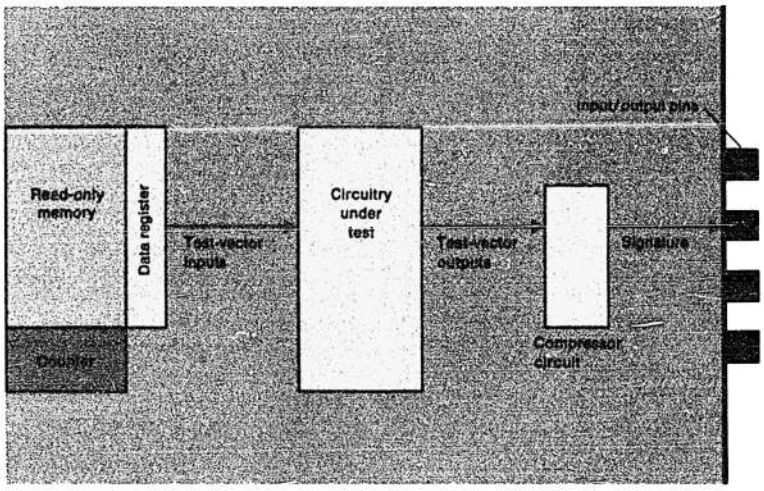
Scan design simplifies the testing of sequential circuits by breaking them into blocks of combinational circuitry (above). When the circuit is placed in a "scan" mode, D flip-flops are linked so that test vectors can be inserted into embedded circuitry through the I/O pins.

C



Partitioning simplifies the testing of large blocks of circuitry by dividing them into smaller blocks that can be tested independently. For example, if 2^8 patterns must be generated to test one block of circuitry, only 2^4 patterns are needed if the block is divided in half.

D



For built-in self-testing, a test-program generator is embedded in a circuit. The common type of generator here uses a ROM to store test vectors and a simple circuit for compressing the response into a signature, which is read through one or more I/O pins. Other types include random-number generators.

Testing chips early in the game

Integrated-circuit chips undergo a series of tests that add to the costs. The final production test is the one that causes the greatest increase in chip costs, mainly because it is the most thorough. It caps a long and complicated test process that varies widely from one type of circuit to another and from one manufacturer to another. However, because of the high number of internal devices on current VLSI chips, production test programs are becoming too time-consuming and too costly to devise and run.

Chip testing begins when the chips are still part of a wafer. Parametric tests—which determine electrical properties such as gate threshold, polysilicon resistance, and diffusion resistance—are usually performed on four specially designed chips on a wafer. The results are used in one of two ways: (1) if the measured values are not within satisfactory ranges, the wafer is either scrapped or recycled with no further testing; or (2) changes in the fabrication process to maintain its integrity may be indicated. Such tests are essentially the same as when LSI chips were introduced.

Next is the wafer-sort test, which is usually inexpensive compared with the final production test; it is designed to screen out chips that have gross faults before they undergo the rather expensive packaging operation. However, in recent years the wafer sort has taken on some of the burden of the final test. A computer-controlled tester applies a series of functional tests to each chip on the wafer. These tests are similar to the final test in that they check the internal gates of a circuit using a set of test vectors that cover the inputs and outputs of a properly working chip. In some cases, the wafer sort can achieve a fault coverage as high as 80 percent—meaning that 80 percent of the possible faults are detected. The final test, which typically has 95 percent fault coverage, often uses the same vectors as the functional test in the wafer sort. However, these vectors are applied under wider variations in conditions such as temperature, voltage, and speed.

Other tests following the wafer sort are as important as the functional tests in ensuring quality. In recent years, tests have become much more rigorous in such areas as checking the contacts between test probes and chip pads, applying heat stress to accelerate latent defects, and measuring the power dissipation. —*S.C.S. and V.D.A.*

imperfect measure of the effectiveness of a set of test vectors. For this reason, VLSI test engineers rely to a great extent on their experience to determine what fault coverage is adequate.

A fault simulator cannot evaluate the coverage of physical faults that are not covered in the stuck-fault model, such as short circuits or open circuits in metal, diffusion, or polysilicon; shorts between semiconductor layers; or parametric irregularities. The simultaneous occurrence of two or more faults is also not simulated because of the very large number of possible fault combinations, even though a processing defect is quite likely to lead to multiple faults, especially with small circuit geometries.

The real value of estimating the fault coverage of a set of test vectors may also depend on the simulator used, since simulators incorporate criteria for detecting specific faults other than stuck ones—fault-induced races and oscillations, for example.

Redundant circuits can also throw off the accuracy of fault-coverage simulation. They give rise to faults that are not detected in a test because the faults do not cause a circuit to work improperly. In addition, simulators have no way of distinguishing between faults hidden by redundant circuits and valid faults that the test program simply cannot identify.

Redundant faults are not identified by simulators because the computer time required to do the job would increase with the complexity of the circuit at a rate that is always greater than a polynomial. This means that even if a test method were devised to keep the cost of testing the circuit linearly proportional to the circuit size, the cost of locating redundant faults would still increase at a faster rate.

At present, test engineers have no way of knowing the extent to which redundant faults influence any given estimate of fault coverage. Time is often wasted trying to raise fault coverage a few percentage points above 90 when perhaps 5 percent of the possible circuit faults are redundant. In such a case, a test program might in fact have 95 percent fault coverage—usually considered adequate for most chips—although the simulator would show only 90 percent fault coverage.

Despite the drawbacks of the current measures of fault coverage by simulators, this method continues to be relied upon as the figure of merit for a test vector set. One may rightfully ask how this figure of merit relates to the quality of the tested chips. A quantitative answer can be given, based on a model of the fault distribution on the chip. It is assumed in such a model that a random number of logical faults are caused by each physical defect on a chip. Since the physical defects themselves are randomly distributed, a compound distribution can be used to describe the occurrence of logical faults.

The model of the fault distribution predicts that for denser chips, a lower fault coverage is needed to obtain the same quality level. In smaller geometries, a defect caused by a dust particle, for example, will damage more gates, because the particle will be larger relative to the gates. Since more gates will thus be affected by a single particle, there will be more faults to flag the effects it causes. Although other problems will certainly arise in testing even more complex chips, this is at least one encouraging sign, especially in view of the disproportionately high cost of increasing fault coverage.

To probe further

Recent analyses showing that a high fault coverage can be obtained by nonexhaustive random testing are reported in "On random test," a paper given at the International Test Conference in 1983 and available in the proceedings of that conference. Another paper, "When to use random testing," in the November 1978 issue of *IEEE Transactions on Computers*, pp. 1054–55, also discusses this issue.

Several conferences now deal with one or more aspects of VLSI chip testing, including computer software for implementing many of the techniques described here. The International Test Conference 1985 will be held in October in Philadelphia, Pa. The Design Automation Conference, which has dealt increasingly with testing-related topics, will be held this June in Albuquerque, N.M. Registration information for both conferences may be obtained by writing to the IEEE Computer Society, 1109 Spring St., Suite 300, Silver Springs, Md. 20910; telephone 301-589-8142. To order the proceedings of last year's conferences on VLSI chip testing, write to the IEEE Order Dept., 445 Hoes Lane, Piscataway, N.J. 08854.

The IEEE Custom Integrated Circuits Conference 1985 will be held May 20–23 in Portland, Ore. For registration information, write to Laura Silzars, 6900 South Canyon Drive, Portland, Ore. 97225; telephone 503-292-6374. To order last year's proceedings, write to the IEEE Order Dept. at the address above.

The eighth annual Design for Testability Workshop will be held April 23–25 in Beaver Creek, Colo. For information, write to Thomas Williams, IBM Corp., P.O. Box 1900, Boulder, Colo. 80302.

The IEEE Instrumentation and Measurement Society sponsored its second annual Instrumentation and Measurement Tech-

nology Conference last March in Tampa, Florida. For a copy of the proceedings, write to Robert Myers, Conference Coordinator, 1700 Westwood Blvd., Los Angeles, Calif. 90024; telephone 213-475-4571.

Many of the topics covered in this article were touched upon in "The one-month chip: testing," by Fred Guterl, which appeared in the September 1984 issue of *Spectrum*, p. 40, as part of a five-part report. Another article on printed-circuit-board testing, dealing with issues similar to those discussed in this article, is "Automated board testing: coping with complex circuits," by Rodham E. Tulloss, in the July 1983 issue of *Spectrum*, p. 38.

A variety of automatic test systems for test applications are described by W.G. Fee in *Tutorial—LSI Testing*, second edition, IEEE Computer Society, Long Beach, Calif. (catalog no. EHO 122.2).

For information on stuck-type and nonclassical faults in MOS circuits, see "A Fault Simulator for MOS LSI Circuits," A.K. Bose et al., *Proceedings* of the 19th Design Automation Conference, pp. 400–409, June 1982. Nonclassical faults are also described in "Fault Modeling and Simulation of CMOS and MOS Integrated Circuits," R.L. Wadsack, *Bell System Technical Journal*, Vol. 57, pp. 1449–1472, May–June 1977. The inadequacy of stuck-type faults at the gate level for MOS circuits is discussed in "Physical vs. Logical Fault Models of MOS LSI Circuits: Impact on Their Testability," J. Galiay, Y. Crouzet, and M. Vergniault, *IEEE Transactions on Computing*, C-29, pp. 527–31, June 1980.

The D-algorithm is described in "Programmed Algorithms to Compute Tests and to Detect and Distinguish Between Failures in Logic Circuits," J.P. Roth, W.G. Bouricius, and P.R. Schneider, *IEEE Transactions on Electrical Components*, EC-19, pp. 567–80, October 1967. The Podem enhanced algorithm for test generation is described in "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," P. Goel, *IEEE Transactions on Computing*, C-30, pp. 215–22, March 1981. The FAN enhanced algorithm is examined in "On the Acceleration of Test Generation Algorithms," H. Fujiwara and T. Shimono, *IEEE Transactions on Computing*, C-32, pp. 1137-44, December 1983.

The Scoap testability measure of L.H. Goldstein is the subject of "Controllability/Observability Analysis of Digital Circuits," *IEEE Transactions on Circuit and Systems*, CAS-26, pp. 685–93, September 1979. For applications of testability measures and their accuracy, a useful work is "Testability Measures—What Do They Tell Us?" V.D. Agrawal and M.R. Mercer, *Digest of Papers*, International Test Conference, pp. 391–96, 1982.

A scan-design method is described in "Logic Structure for LSI Testability," E.B. Eichelberger and T.W. Williams, *Journal of Design Automation and Fault Tolerant Computing*, Vol. 2, pp. 165–78, May 1978. A proposal for logic partitioning and multiplexer logic for observing internal logic blocks is made by E.J. McCluskey and S. Bozorgui-Nesbat in "Design for Autonomous Test," *IEEE Transactions on Computing*, C-30, pp. 866–75, November 1981.

Built-in self-testing (BIST) is an active area of research. A primary forum for exchange of ideas is the annual BIST Workshop, the third of which was held in March 1985 at Kiawah Island (Charleston), S.C., under the chairmanship of Richard M. Sedmak, Self-Test Services, Maple Glenn, Pa. At the 1984 International Test Conference in Philadelphia, a tutorial on BIST was organized by P.H. Bardell of IBM Corp., Poughkeepsie, N.Y. 12602.

Methods of fault simulation are described by M.A. Breuer and A.D. Friedman in *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, Rockville, Md., 1976. Test-generation and fault-simulation costs are the subject of "Test Generation Costs Analysis and Projection," P. Goel, *Proceedings* of the 17th Design Automation Conference, pp. 77–84, June 1980.

The use of sampling techniques in evaluating fault coverage is discussed by V.D. Agrawal in "Sampling Techniques for Determining Fault Coverage in LSI Circuits," *Journal of Digital Systems*, Vol. 5, pp. 189–202, 1981. A relationship between fault coverage and product quality is derived by V.D. Agrawal, S.C. Seth, and P. Agrawal in "LSI Product Quality and Fault Coverage," *Proceedings* of the 18th Design Automation Conference, pp. 196–203, 1981. This is further refined by S.C. Seth and V.D. Agrawal in "Characterizing the LSI Yield Equation from Wafer Test Data," *IEEE Transactions on Computer Aided Design*, CAD-3, April 1984.

A grab bag of methods for designing testable chips

Method	Advantages	Disadvantages
Testability analysis: computer programs for evaluating how easily a circuit design can be tested	Easy to use; executes quickly; useful for all kinds of circuits	Inaccurate for particular nodes
Scan design: a method of converting sequential circuitry into combinational circuitry for testing	Allows automation of test generation; simplifies test program; good for semicustom circuits (fast turnaround)	Overhead in chip area degrades performance
Partitioning: dividing a circuit into sections that can be tested independently	Moderate overhead	Largely experimental; used only in conjunction with other methods
Built-in self-test: execution of test programs by circuits built into the circuit that is subject to testing	Reduces task of automatic production testers	Gives only a 'go-no go' indication; poor for diagnostics
Random testing: use of random signals to yield a probable fault coverage	Less time-consuming than exhaustive tests; easy to implement on the test chip (as with built-in self-test)	Can't predict coverage of particular faults

About the authors

Shared C. Seth (SM) is a professor of computer science at the University of Nebraska in Lincoln, where he joined the faculty in 1970. He held visiting positions at the Indian Institute of Technology in Kanpur, India, in 1974–75 and 1982–83 and worked at the AT&T Bell Laboratories in Murray Hill, N.J., during the summers of 1980 and 1982. His current research interests are in the areas of VLSI testing and design and reliability analysis of fault-tolerant systems. He holds a bachelor of engineering degree in electronics and telecommunications from Jabalpur University in India, a master of technology degree in electrical engineering from the Indian Institute of Technology in Kanpur, and the Ph.D. in electrical engineering from the University of Illinois in Urbana.

Vishwani D. Agrawal (SM) is supervisor of the test-aids group at Bell Laboratories in Murray Hill. Before joining Bell Labs, he worked at TRW Defense and Space Systems Group in California and as assistant professor at the Indian Institute of Technology in New Delhi. He holds a bachelor's degree in telecommunication engineering from the University of Roorkee in India, a master's degree in engineering from the Indian Institute of Science in Bangalore, and a Ph.D. from the University of Illinois in Urbana. The author of more than 60 papers, he won the best-paper award at the 1982 International Test Conference in Philadelphia. He is vice chairman of the Design Automation Standards Subcommittee of the IEEE and is a member of the editorial board of *IEEE Design and Test Magazine*. He is a fellow of the Institution of Electronics and Telecommunication Engineers of India.