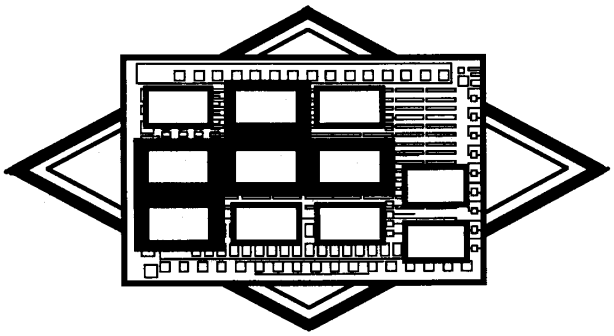

DESIGNING CIRCUITS WITH **PARTIAL SCAN**

VISHWANI D. AGRAWAL
AT&T Bell Laboratories

KWANG-TING CHENG
University of California

DANIEL D. JOHNSON
TONYSHENG LIN
AT&T Bell Laboratories

In this scan design methodology, only selected faults are targeted for detection. These faults are those not detected by the designer's functional vectors. The test generator decides exactly which flip-flops should be scanned using one of two methods. In the first method, all possible tests are generated for each target fault, and the set of tests requiring the fewest flip-flops is selected. In the second method, only one test is generated for each fault, and the use of flip-flops is avoided as much as possible during test generation. Examples of actual VLSI circuits show a savings of at least a 40% in full-scan overhead.

In a full-scan design, all flip-flops of a circuit are chained into one or more shift registers so that the designer can observe and control circuit states directly from primary I/O. As a result, to test a sequential circuit, we can divide it into purely combinational and shift register parts. Since automatic test generation programs are available for combinational circuits, the cost of test generation becomes substantially lower.[1]

Scan methods are not without cost, however. Penalties include extra hardware and the need to adhere to strict logic design rules. About 30% extra area is added in the form of additional circuitry for each D flip-flop and some wiring to form the shift register. Moreover, additional circuitry, such as multiplexers, slows signals and degrades system performance. Adherence to a complete set of logic rules is often too restrictive, although most designers view the use of certain rules, such as synchronous design, as good practice.

In addition, the scan register for a large circuit could be very long, requiring lengthy scan sequences to initialize and read internal states. Only with additional I/O pins can the designer break the scan register into several smaller registers for parallel operation.

An earlier version of this article appeared in the proceedings of the 1987 International Test Conference under the title, "A Complete Solution to the Partial Scan Problem."

0740-7475/88/0400-08$1.00©1988

Partial scan offers an alternative to the all-or-nothing philosophy of scan design, yet preserves the advantages of high fault coverage and automatic test generation. In partial scan, we can also exclude certain flip-flops, such as those that violate scan design rules to meet functional requirements and those in critical timing paths. In general, scan test sequences are shorter than those needed for a full-scan design.

WHY PARTIAL SCAN?

Because not all flip-flops are included in the scan register, area overhead is lower. Since critical paths and their timing margins are easily determined with automated tools,[2] we can minimize unacceptable timing penalties by excluding the flip-flops on critical paths. Also, excluding nonsynchronously controlled flip-flops, when necessary for circuit function, can reduce the overhead of extra hardware.[3]

But what do we lose in partial scan? In full scan, flip-flop faults are covered by shift register tests. In partial scan, the faults in unscanned flip-flops cannot be tested by this method. Also, in generating the test vectors for the combinational portion of the circuit, we must always assume that the I/O from the unscanned flip-flops is in the unknown state. That is, the combinational test generator must not set or observe the I/O corresponding to these flip-flops. As a result, the test generator will not be able to detect some faults in the combinational part of circuit. These faults must be covered by functional vectors, that is, vectors executing normal functions of the circuit.

To minimize the loss of coverage, we should select those flip-flops for the chain that are frequently used by combinational tests and exclude those that are rarely used. In earlier work on partial scan,[4] the authors tried to improve overall circuit testability by selecting scan flip-flops heuristically on the basis of approximate testability measures. However, test generation was still a complex process with no guarantee of good results because tests had to be generated for sequential (often asynchronous) circuits.

Our work, on the other hand, tries to achieve the required test coverage with minimum overhead through the use of an automatic combinational test generator.

METHODOLOGY

It is a designer's job to create functional vectors, which are developed primarily for design verification, not the detection of stuck-at faults. Consequently, functional vectors can detect only a certain percentage of faults in a sequential circuit. This partial fault coverage may not meet the standards for manufacturing tests, so additional vectors, called scan vectors, are needed.

The faults in the combinational part of the circuit that functional vectors cannot detect are the target faults for scan vectors. In scan test generation, we can have several alternative tests for each target fault, but if we select one test that minimizes the use of flip-flops, we can reduce the total number of flip-flops in the scan register.

We propose two methods for this selection process—the frequency approach and the distance approach. In both, our goal is to achieve the smallest number of flip-flops in the chain, while keeping fault coverage at an acceptable level. Both methods involve the generation of an option table that provides estimates of total fault coverage for various scan register sizes. After the designer has selected an option, the corresponding test sequences are generated and a list of scan flip-flops is given.

FREQUENCY APPROACH

This approach uses a modified Podem (path-oriented decision-making) test generation program.[5] Instead of generating just one test per fault, the modified version generates all possible tests for each target fault in the combinational part of the circuit. In these tests, only inputs that are essential to detect the fault are set to 0 or 1, leaving all other inputs as "don't care." From these, tests are selected such that target faults are covered by a minimal set of flip-flops. The procedure is shown in Figure 1.

GENERATING AN OPTION TABLE

Since increasing fault coverage and reducing the number of scanned flip-flops are conflicting requirements, we generate an option table so that the designer can make trade-offs. The generation procedure is as follows.

During the generation of all test vectors for each target fault, a record is kept of the flip-flops used by each vector. Table 1 shows the flip-flops used for a circuit containing four flip-flops (A,B,C and D), with four faults to be detected. The entry

"1" under t_{i1} in the table means that the test t_{11} uses no flip-flop.

The objective is to select, for a given number of flip-flops, a set of tests (one per fault) to cover the largest number of faults. For this set coverage problem (an NP-complete problem), the following heuristic is used:

1. Consider the faults with only one test vector and include the flip-flops required to test these faults in the scan chain.
2. Eliminate the faults that can be covered by this set of flip-flops and estimate fault coverage (a procedure for estimating coverage is described later).
3. For any remaining faults, determine the frequency of every flip-flop unscanned thus far. The frequency is the number of faults that use that flip-flop in at least one test.
4. Include the flip-flop having the highest frequency.
5. Go to step 2.

This heuristic can also be implemented using a logic analog. Figure 2 shows this representation for the tests in Table 1. Primary inputs of this two-level AND-OR circuit are named after the flip-flops, A, B, ... The AND gates are named after tests, and the OR gates after faults. This circuit easily determines how the selection of any flip-flops affects fault coverage.

```
Fault simulate entire circuit with functional vectors
Generate list of undetected faults F_un
Isolate combinational portion of circuit
Map F_un onto combinational part to obtain target faults F_comb
Begin do for each f_i ∈ F_comb
    Generate all tests t_ij, j=1,2... for f_i
End
Generate option table: Estimated fault coverage vs. flip-flop usage
Select an option from the option table
In combinational model make unscanned flip-flop I/O inaccessible
Begin do for each vector generated for this option
    Randomly fill unknown bits corresponding to scan flip-flops
    Fault simulate F_comb
    Remove detected faults from F_comb
    Generate scan sequence
End
Add shift register test sequence
```

Figure 1. Partial scan design using the frequency approach.

Suppose we select flip-flop A for scanning. Input A is set to 1, making the output of AND gate t_{31} true. Thus, we know that test t_{31} is possible. Also, f_3 is true, which means that fault f_3 will be covered. Note that the output of gate t_{11} is always true, as is f_1, because this test does not need any flip-flop.

Table 1. Number of flip-flops used in tests.

Fault f_i	No.of Possible Tests	Flip-Flops Used		
		t_{i1}	t_{i2}	t_{i3}
f_1	2	1	A,B	
f_2	3	A,B	B,C	A,B,D,
f_3	1	A		
f_4	2	A,B,D	A,D	

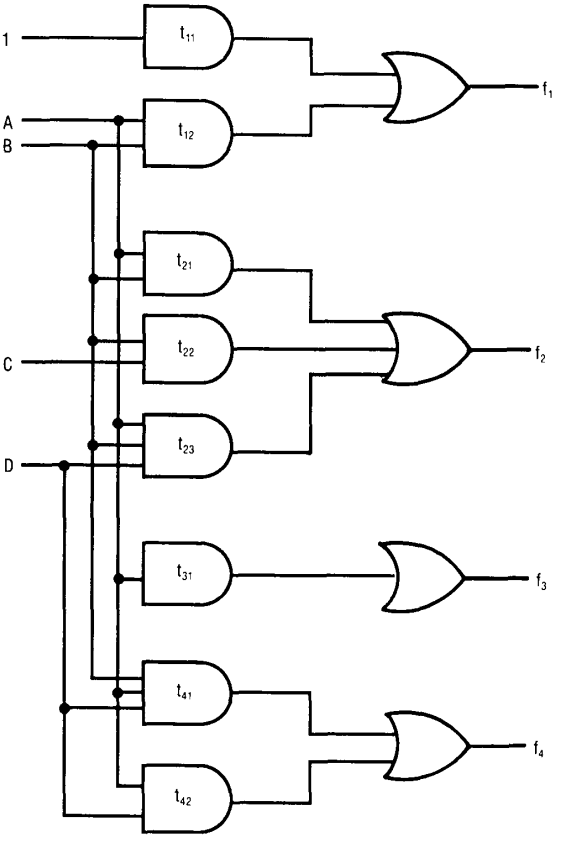

Figure 2. A logical representation of flip-flop use.

Similarly, by setting A, B, and D to 1, all four faults can be covered. We can state this problem in terms of finding the minimum number of 1's at the input of this circuit that will make all outputs 1. We may be able to solve the problem using logic minimization programs.[6]

ESTIMATING FAULT COVERAGE

For a set of flip-flops in the scan register, we can use the procedure just discussed to identify the faults covered in the combinational logic. To find the fault coverage over the entire circuit, we would have to actually implement the partial scan register and perform fault simulation for the tests. To select a partial scan option, however, the designer needs only some idea of the final coverage, and the following estimating method can be used:

$$\text{Estimated Coverage} = \frac{F_{func} + f_{comb} + \alpha F_{ff} + \alpha N_{ff} f_{mux}}{F_{total} + \alpha N_{ff} f_{mux}}$$

where

- F_{func} = the number of faults covered by functional vectors
- F_{comb} = the number of combinational faults not detected by functional vectors
- f_{comb} = the scan coverage of combinational faults $f_{comb} \subseteq F_{comb}$
- F_{ff} = the number of flip-flop faults that functional vectors did not detect
- α = the fraction of flip-flops in the scan register
- N_{ff} = the total number of flip-flops in the circuit
- f_{mux} = the number of faults in the multiplexer of the scan flip-flop
- F_{total} = the total faults in the unscanned circuit, or $F_{func} + F_{comb} + F_{ff}$

In this formula, the symbols of various fault sets represent the cardinality of sets. We assume that scan vectors will completely test all faults in the circuitry that has been added to the scan flip-flops—for example, the multiplexer. Experience supports this assumption.

In addition, the coverage estimate is closer to the correct value unless the coverage F_{func} of functional vectors is too low (typically, below 70%). In that case, the estimate is pessimistic, since for low functional vector coverage, scan vectors also randomly cover additional faults. The examples presented later show that this formula does indeed represent a lower bound of the coverage.

We enhanced a Podem test generation program[7] to generate all tests for a given fault. We also incorporated flip-flop selection and coverage estimation in the program to generate the option table. For test generation, the combinational part of the circuit is isolated by removing the flip-flops and setting the flip- flop I/O as primary outputs and primary inputs of the combinational circuit.

Consider, for example, a four-bit multiplier circuit with 10 inputs, 9 outputs, and 645 gates. A set of 28 functional vectors is used to check the functional correctness and the timing performance with no specific consideration to the detection of stuck-at faults. Fault simulation revealed that functional vectors did not detect 47 of 531 faults.

Of these, 34 were in the combinational part of the circuit. Since scan vectors are generated specifically to cover faults in the combinational logic, these 34 faults become the target faults for scan vectors. A program implementing this algorithm produced the following result (Table 2 gives the option table):

1. Total flip-flops: 15
2. Total faults in circuit: 531
3. Functional fault coverage: 91.15%
4. Faults left by functional vectors: 47
5. Combinational target faults in category 4: 34
6. Undetectable faults in category 5: 1
7. Undetected faults in category 5: 0

Table 2. Option table for a four-bit multiplier—frequency approach.

Option	Target Fault Coverage (%)	Estimated Coverage (%)	Flip-Flops in Scan Chain	
			No.	Pctge.
1	97.06	99.84	15	100.00
2	79.41	98.75	14	93.33
3	67.65	97.95	13	86.67
4	61.76	97.46	12	80.00
5	50.00	96.63	11	73.33
6	47.06	96.28	10	66.67
7	41.18	95.56	8	53.33
8	38.24	95.18	7	46.67
9	26.47	94.28	6	40.00
10	23.53	93.64	4	26.67
11	20.59	93.23	3	20.00
12	17.65	92.80	2	13.33
13	5.88	91.80	1	6.67

One undetectable fault in category 6 is due to redundancy in the combinational logic. Target fault coverage in Table 2 is the coverage of 34 target faults as combinational test vectors are processed for scan flip-flop selection. We assume that multiplexers have seven faults.

Suppose we select option 8 of the table, which requires seven flip-flops. In the combinational model of the circuit, the test generator has access to the I/O corresponding to these seven flip-flops. All other flip-flop signals are fixed at an unknown, or X, value. Tests are generated for all faults detectable in this option. If any scanned flip-flop signal is unspecified in a test, it is randomly determined before a fault simulation of the combinational circuit is performed.

Table 3 shows the results of this test compared with the results of full scan. Note that with less than half the flip-flops scanned, the fault coverage is still very close to that achievable with full scan. The number of vectors is only two thirds that of full scan, and the scan design and test generation are still automatic.

Note also that the partial scan fault coverage of 99.3% is higher than the estimated 95.18% in Table 2. We assume in Table 2 that scan vectors will not detect any of the faults in the unscanned flip-flops that functional vectors do not detect. In reality, however, scan sequences will detect some of these faults. For a similar reason, the actual coverage of full scan is slightly lower than that estimated in Table 2. Some flip-flop faults, in this case, are not detectable.

DISTANCE APPROACH

Unlike the frequency method, which requires that all tests be generated for each target fault, the distance approach requires that only one test be generated per fault. It is a good approach for large circuits, where the frequency method can result in unacceptable computation and storage costs. Also, with the distance approach, we do not need to post-process tests.

The distance method uses an enhancement of the distance heuristic of Podem. To generate tests in Podem, a path is sensitized to propagate the state of the faulty line to a primary output. Path sensitizing is done by establishing a series of objectives. The objectives might be the following: set D or $\overline{D}$ at the fault site (for example, D = 1 without a fault and D = 0 with a fault), propagate D or $\overline{D}$ through a gate toward a primary output by setting certain signal values, or justify those signal values from primary inputs.

Podem sets the primary inputs that are nearest to the site of the objective. The distance is simply the number of gates on the path. Similarly, if there are several D's, the one closest to a primary output is propagated first. In the test generator for partial scan (Figure 3), all normal primary inputs and outputs are assigned a distance of 0. Inputs and outputs of combinational logic, derived from flip-flops, are assigned a large distance, say, 100. Once a flip-flop signal is used in a test, its distance becomes 0.

Table 4 shows the results of applying this procedure to the four-bit multiplier circuit described earlier. Here, scan flip-flops are selected as tests are being generated. Target fault coverage in the option table is simply the percentage of the 34 target faults that either the combinational test generator has processed or the fault simulator has eliminated.

For full scan (option 1), this method produced exactly the same result as the frequency method. With option 6 (seven flip-flops), however, a set of 306 vectors (including 28 functional vectors) covered 97.41% faults—about 2% lower than with the frequency method. The reason is that the frequency method allows the choice of a better optimized test set.

The results of the distance approach are summarized in Table 5. In general, the results of the distance approach depend on the order in which target faults are picked for test generation. In our implementation, this order was purely arbitrary. Also, for a target fault, the first test produced by the test generator was accepted. This may not be the best choice if a global optimization like the one performed in the frequency method is desired.

Thus, for the same number of scanned flip-flops, the fault coverage obtained by the distance approach could be lower than that obtained in the frequency approach. However, as pointed out earlier, the economy of generating just one test for a fault is significant for larger circuits discussed in the next section.

Table 3. Partial scan vs. full scan for four-bit multiplier—frequency approach.

Type of Scan	Flip-Flops in Chain		Total Vectors	Fault Coverage (%)	Total Faults
	No.	Pctge.			
Full (Option 1)	15	100.00	1028	99.69	639
Partial (Option 8)	7	46.67	358	99.30	575

```
Fault simulate entire circuit with functional vectors
Generate list of undetected faults F_un
Isolate combinational portion of circuit
Map F_un onto combinational part to obtain target set F_comb
Set normal I/O distance to 0
Set flip-flop I/O distance to 100
Begin do for each f_i ∈ F_comb
    Generate a test for f_i
    For all flip-flop I/O used by test change distance to 0
    Fault simulate F_comb
    Remove detected faults from F_comb
    Estimate fault coverage
End
Generate option table: Estimated fault coverage vs. flip-flop usage
Select an option from the option table
In combinational model make unscanned flip-flop I/O inaccessible
Initialize all combinational target faults in F_comb
Begin do for each f_i ∈ F_comb
    Generate a combinational test vector
    Randomly fill unknown bits in the test
    Fault simulate F_comb
    Remove detected faults from F_comb
    Generate scan sequence for the combinational test
End
Add shift register test sequence
```

Figure 3. Partial scan using the distance approach.

Table 4. Option table for a four-bit multiplier—distance approach.

Option	Target Fault Coverage (%)	Estimated Coverage (%)	Flip-Flops in Scan Chain	
			No.	Pctge.
1	100.00	99.69	15	100.00
2	50.00	97.01	15	100.00
3	41.18	96.36	14	93.33
4	35.29	95.86	13	86.67
5	17.65	94.86	12	80.00
6	11.76	93.46	7	46.67

Table 5. Partial scan vs. full scan for four-bit multiplier—distance approach.

Type of Scan	Flip-Flops in Chain		Total Vectors	Fault Coverage (%)	Total Faults
	No.	Pctge.			
Full (Option 1)	15	100.00	1028	99.69	639
Partial (Option 6)	7	46.67	306	97.41	575

EXPERIMENTAL RESULTS

Table 6 lists the circuits we used in our partial scan tests. The circuits contained from 1000+ to 13,000 gates, and all use standard-cell design. Circuit A is a random logic block of a large chip. It is mildly sequential and contains deep combinational logic. We used a small set of 221 functional vectors, which covered 49.15% of the faults. The percentage is somewhat lower because the circuit is only a part of the entire chip. Consequently, writing functional vectors is difficult.

Circuit B is an entire chip. It is highly sequential as indicated by 172 flip-flops. A set of just 488 functional vectors covered 82.72% of the faults. As is typical of highly sequential circuits, additional functional vectors only marginally increase the fault coverage.

Circuit C is also an entire chip, but is much larger. A set of 10,795 functional vectors covered 88.08% of the faults, with 522 target faults remaining in the combinational logic. We made no attempt to isolate any possible redundant faults.

We generated option tables for each circuit and for each partial scan method—frequency and distance. Using a design goal of 95% fault coverage, we selected options from these tables. Results for these options as well as

Table 6. Circuits used in experiment.

Circuit	Primary I/O	Logic Gates	Flip-Flops	Functional Tests		No. of Faults		
				No. of Vectors	Coverage (%)	Total	Undetected	Target
A	25/46	1532	42	221	49.15	1084	551	183
B	38/16	1656	172	488	82.72	4871	842	426
C	21/27	13,726	351	10,795	88.08	10,094	1203	522

Table 7. Partial scan with a design goal of 95% fault coverage.

Circuit	Type of Scan	Flip-Flops in Chain No.	Flip-Flops in Chain Pctge.	Total Vectors	Fault Coverage (%)	Total Faults
A	Full	42	100.00	3477	99.48	1336
	Partial (Frequency Option 3)	27	64.29	1665	97.35	1246
	Partial (Distance Option 6)	31	73.81	2285	98.89	1266
B	Full	172	100.00	17,568	98.59	5905
	Partial (Frequency Option 2)	110	63.95	9992	96.29	5531
	Partial (Distance Option 4)	111	64.53	10,308	96.22	5533
C	Full	351	100.00	92,889	98.75	12,222
	Partial (Frequency Option 8)	176	50.14	25,347	96.37	11,150
	Partial (Distance Option 12)	209	59.54	40,457	98.14	11,374

results for full scan are given in Table 7 for each circuit. We obtained fault coverages by fault simulation of sequential circuits in which a scan register was implemented. On average, with partial scan, about 40% of the flip-flops are excluded from the scan register. In general, the distance method tends to use more flip-flops. However, in circuits A and C, where more flip-flops are used, the final fault coverage is also higher.

In each case, we exceeded our design goal of 95% coverage. The reason, as we said earlier, is that the estimation is pessimistic. It assumes that scan vectors will not detect any of the faults in the unscanned flip-flops that are not detected by functional vectors—which is not always true. The test sequence in all cases is significantly shorter than that for full scan.

As we have mentioned, the result of the distance approach is sensitive to the order in which the test generator processes the target faults. In our experiments, the order was arbitrary. It is encouraging to see that the influence of ordering does not appear significant for larger circuits.

Consider the sample four-bit multiplier described earlier, which contains 645 gates. Coverage with the distance approach was about 2% lower than that using the frequency approach. For circuits A, B, and C, which are considerably larger, the coverage obtained by either method seems to depend only on the number of scanned flip-flops.

In our partial scan methodology, designer's functional vectors play an important role. Since scan vectors are generated only for the faults that the functional vectors do not detect, simulation of the sequential circuit with all faults is required. Such simulation for large circuits can be very expensive because we cannot use established methods of statistical fault sampling.[8]

A 40% savings in scan overhead is still attractive, however. Although we never excluded any flip-flops because they violated scan design rules or critical timing, the combinational test generator can easily exclude any primary I/O contributed by such flip-flops from test generation. Excluded primary inputs are permanently set to the unknown state, while the excluded primary outputs are simply eliminated from the list of fault detection points. We can also use this capability during test generation and fault simulation after selecting a partial scan option.

We have presented two methods to automatically select a near-optimal set of scan flip-flops. According to the experimental results on actual VLSI circuits, we can include less than 65% of the flip-flops in the scan chain and yet obtain a fault coverage higher than 95%. The advantage of partial scan is that better functional testability produces low scan overhead. Full scan does not take advantage of this testability.

Another advantage of our partial scan methodology is that it is completely automatic. Test generation can be further enhanced, since some unscanned flip-flops may have set and clear signals that can be controlled from primary inputs. If we take advantage of this characteristic, we can use unscanned flip-flops in a restricted manner.

Acknowledgments

We are indebted to S.C. Seth for several technical suggestions and to those who developed the Podem program at the University of Nebraska. We also thank E.S. Kuh and T.G. Szymanski.

K-T Cheng's research was supported in part by the National Science Foundation under grant ECS-85-06901 and by the State of California Microelectronics Innovation and Computer Research Opportunities Program.

REFERENCES

1. V.D. Agrawal, S.K. Jain, and D.M. Singer, "Automation in Design for Testability," *Proc. Custom Integrated Circuits Conf.*, May 1984, pp. 159-163.
2. V.D. Agrawal, "Synchronous Path Analysis in MOS Circuit Simulator," *Proc. Design Automation Conf.*, June 1982, pp. 629-635.
3. M.R. Mercer, V.D. Agrawal, and C.M. Roman, "Test Generation for Highly Sequential Scan-Testable Circuits Through Logic Transformation," *Proc. Int'l Test Conf.*, Oct. 1981, pp. 561-565.
4. E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," *Proc. Int'l Test Conf.*, Nov. 1980, pp. 153-162.
5. P. Goel, "An Implicit Enumeration Algorithm To Generate Tests for Combinational Logic Circuits," *IEEE Trans. Computers*, Mar. 1981, pp. 215-222.
6. R.K. Brayton et al., *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, 1984.
7. T. Lin and V.D. Agrawal, "A Test Generator for Scan-Design VLSI Circuits," *Proc. AT&T Conf. Electronic Testing*, Oct. 1986, pp. 23.1-23.7.
8. V.D. Agrawal, "Sampling Techniques for Determining Fault Coverage in LSI Circuits," *J. Digital Systems*, 1981, pp. 189-202.

Vishwani D. Agrawal is conducting research on VLSI testing at AT&T Bell Laboratories in Murray Hill, New Jersey. Formerly, as supervisor of the Test and Design Verification Group, he was responsible for the development of VLSI CAD methods and tools for scan design, built-in self-test, testability audits, testability measures, static timing analysis, formal logic verification, connectivity verification, hardware-assisted simulation, fault simulation, and vector generation. He recently headed a task force to study built-in self-test in AT&T's design environment.

Agrawal holds a PhD from the University of Illinois at Urbana-Champaign. He has published more than 80 papers, with four Best Paper awards, and holds a patent on a scan-testable circuit. He is a fellow of the IEEE, a fellow of the IETE (India), and a member of the ACM. He was the editor-in-chief of *IEEE Design & Test of Computers* from mid-1985 to 1987 and is currently serving on the program committees of the Design Automation Conference and the International Test Conference.

Kwang-Ting Cheng is with the Electronics Research Laboratory at the University of California, Berkeley, where he is also working toward a PhD in electrical engineering and computer science. His areas of interest are VLSI testing and IC layout. Previously, Cheng served in the Chinese Air Force as a systems engineer. He has also worked at AT&T Bell Laboratories, Murray Hill. He holds a BS in electrical engineering from the National Taiwan University, Taipei.

Daniel D. Johnson is a senior technical associate in the Test and Design Verification Group of AT&T Bell Laboratories, Allentown. His interests are computer-aided design, test vector generation, and design for testability. His previous work includes the testing of AT&T's microprocessors.

Johnson holds an associate degree in electrical engineering from the DeVry Institute of Technology, Chicago, and is currently working toward a BS in computer science at Moravian College in Bethlehem, Pennsylvania.

Tonysheng Lin joined AT&T Bell Laboratories, Murray Hill, in 1985 as a member of the Test and Design Verification Group. He has worked on VLSI testing, testable design, and fault simulation. Previously, he was a full-time lecturer in the National Chiao Tung University, Taiwan. His current interests include computer-aided design and fault-tolerant computing.

Lin holds a BS and an MS in computer engineering from the National Chiao Tung University, and a PhD in computer science from the State University of New York at Binghamton.

Questions about this article can be directed to V. Agrawal, AT&T Bell Laboratories, Rm. 2C-476, 600 Mountain Ave., Murray Hill, NJ 07974-2070.