VLSI Design
1993, Vol. 1, No. 1, pp. 87–97
Reprints available directly from the publisher
Photocopying permitted by license only

Computer-Aided Testing Systems: Evaluation and Benchmark Circuits

SAMIHA MOURAD
Department of Electrical Engineering, Santa Clara University
Santa Clara, California, USA

As the demand on Computer-Aided Testing Systems (CATS)—Automatic Test Pattern Generation (ATPG) and logic and fault simulations as well as testability analysis—increases and the choice becomes more varied, a need to compare the merits of the different systems emerges. *Benchmark circuits* are used to carry out the comparisons.

In this paper, criteria for selecting the benchmark circuits are discussed. These criteria are partly based on the results of experiments carried out to characterize CATS. The focus is particularly on Automatic Test Pattern Generators. The preliminary results show that there is no general agreement on how: 1) fault collapsing is performed, and 2) fault coverage is calculated. In addition, the performance of the ATPGs depends on the circuit representation, topology and size as well as the algorithm. In order to compare the performance of the ATPGs as the circuit under test increases in complexity, it is important to use regular structures that consist of replication of medium size circuits. Practical considerations involved in benchmarking are also examined. Emphasis is on the transfer of circuits between different CATS systems and the use of EDIF as a neutral exchange language.

Key Words: *Testing; CAD Tools; Benchmark; Test patterns; Fault simulation*

With increased device density on a chip, testing cost has become a major proportion of total system development cost. This proportion "has increased from 10% in the mid seventies to 25–50% today" [7]. The increase in testing cost is expected to continue due to the fact that many problems are still encountered with present testing techniques [45]. In the development of test sets as well as in verification of the design for testability techniques, computer-aided design and test equipment have become indispensable. In this paper, the software tools which are generally used in testing—Automatic test pattern generators (ATPG), logic and fault simulators as well as testability analysis programs—will be referred to as Computer-Aided Testing Systems (CATS).

As the demand on CATS systems increases and the choices become more varied, it is rather important to compare the merits of different systems. Twenty-nine systems were reported in a survey conducted by HH&B on behalf of the Navy [39]. Not all of these were equipped with ATPGs. Only a few of these systems survived the fierce competition among developers and were listed in a later survey [47] which reported over 45 systems.

In this paper, no attempt to benchmark systems (in the sense of ranking the systems) will be given. Rather, the intent is to present an approach for the quantification of the attributes of CATS systems for the sake of selecting benchmark circuits. First, the role of benchmarking in other fields will be briefly examined in order to gain more insight about the selection of the benchmark circuits. Then the different functions of a CADS will be reviewed in order to abstract attributes that need to be assessed.

We present some experimental results that will help in the characterization of CATS components. The focus is particularly on Automatic Test Pattern Generators, although many of the results can also be applied to fault simulators. First, the different functions of a CATS system are reviewed in order to abstract the attributes that need to be assessed. The experiments are carried out using three CATS systems and several benchmark circuits. They help assessing: 1) different approaches to fault coverage and collapsing, 2) the behavior of ATPGs—length of the test set and generation time—with different circuit complexity and topology. Criteria for selecting benchmark circuits are then examined. The last

section of the paper deals with practical considerations in exchanging benchmark circuits among different systems.

THE ROLE OF BENCHMARKING

The use of benchmark units to assess products is a well-established practice. Computer programs with specific attributes have been used to evaluate the execution speed of different computers since the early days of computer use [22]. Subsequent efforts in benchmarking computers [4], [17], [20], [41], [51] resulted in the following conclusions: 1) it is not possible to develop one program that includes all the parameters that need to be measured, and 2) as computer technology and software techniques change, different benchmark programs would need to be developed to reflect these changes. Special benchmarks have been devised to evaluate workstations [27], their graphics capabilities [9] and the correctness of floating point operations [24]. Benchmarks can even be tailored to assess a particular operating system such as UNIX [3].

In the digital design and testing community, some circuits have been used as test cases by researchers to demonstrate their methods. The infamous 181 ALU has been used by many researchers [31], [32], [23] as well as by commercial systems [2]. In a study on minimization techniques of PLAs, 56 benchmark circuits were used [8]. In logic synthesis, several suites of benchmarks have been announced [10], [11].

At the 1985 International Symposium on Circuits and Systems (ISCAS) a set of 10 combinational circuits were proposed [10] to demonstrate test pattern generation by several researchers. As soon as these circuits were made public they became very popular and were used in many studies [1], [16], [38], [42], [43], [44], [50], [49]. Sequential benchmark circuits were proposed subsequently, ISCAS 89 [11].

Another indication that benchmark circuits are needed in digital testing—in 1979, a report on benchmarking CATS was published by the Navy [39]. Almost ten years later, the issue was again discussed by Greer [21], and a whole session was devoted to the same topic at COMPCON Spring 88 [14], [18], [29], and 1988 International Test Conference [35].

COMPUTER-AIDED TESTING SYSTEMS

In general, any CATS consists of some combination of the following components: 1) Schematic capture, 2) Timing verification, 3) Logic simulation, 4) Fault simulation, 5) Testability measures, and 6) ATPG.

Most of these programs require a large computer to run efficiently. However, realizing the need to decrease testing cost, the vendors ported their programs (and sometimes a version of these programs) on smaller computers, mostly IBM AT or IBM RT, HP 320 or Microvax. This, of course, comes at the expense of the speed of the mainframes. Some of this loss in speed is compensated by the use of hardware accelerators for these programs which have been developed in order to increase their throughput [5]. CATS can thus be hosted on a variety of hardware configurations. The main interest is to evaluate the capability of the systems to handle digital testing problems. This does not suggest that the generation time is not important but that if a microcomputer-based system is selected, a trade off between speed and cost will be made. The graphics of a workstation is an important feature since it might facilitate the entry of the circuit description. However, it does not affect the quality of the test generation algorithm.

The effectiveness of fault simulators and ATPGs are dependent on the strategy of simulation and test pattern generation as well as the circuit size and complexity. They are also dependent on several common attributes: circuit representation, fault handling—modeling, representation, collapsing and dropping. These attributes will be discussed in the subsequent sections of this paper.

CIRCUIT REPRESENTATION

Accurate modeling of the circuit is an essential factor for proper simulation. There are different parameters that determine circuit modeling. The components of the circuit can be modeled on the functional, gate or switch level as illustrated in Figure 1. On the functional level, the behavior of the component is described by the outputs in terms of the inputs without necessarily specifying any details of the internal structure of the component. The gate level may be considered as a functional description in which all components are elementary gates—AND, OR, and inverters. It is also possible to have a hierarchical structure where any of the components are described on the functional level. Representing a circuit at the gate-level is more suitable for small circuits. Functional-level representation reduces computational complexity. The curves in Figure 2 show simulations on the gate level and the functional level [39]. Beyond a certain gate count, the functional representation is more efficient.

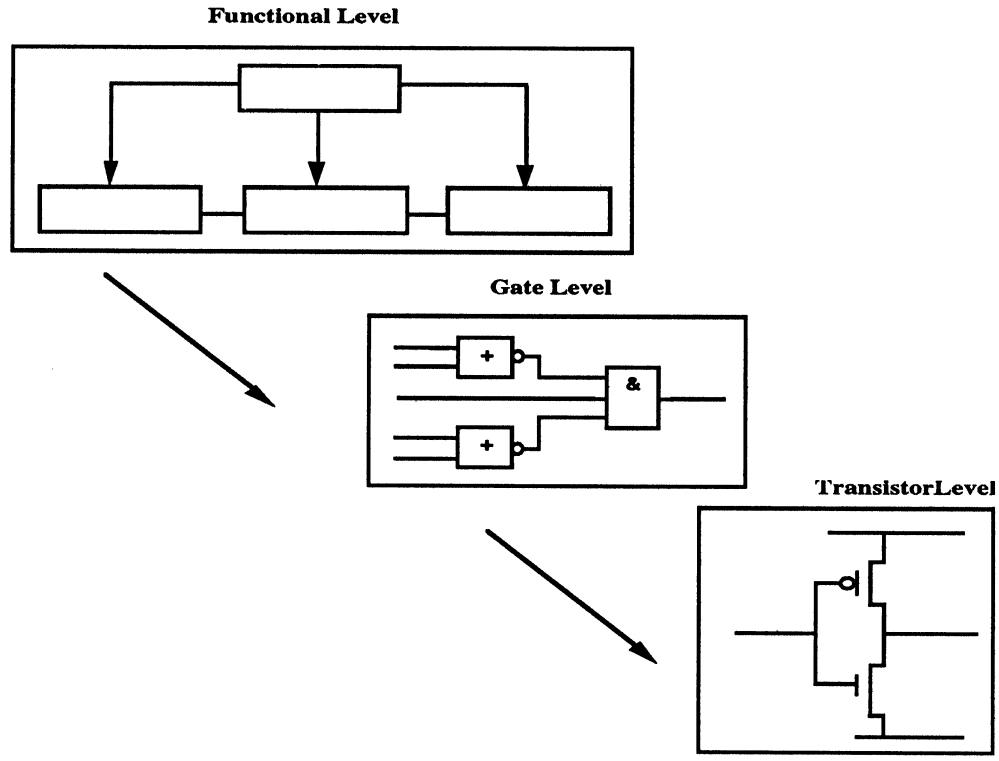


FIGURE 1 Circuit Level Representations.

However, the use of the functional model does not allow evaluation of the internal nodes of the functional blocks. In evaluating these two types of representation, a trade off between the speed of computation and the need for accessing the internal nodes has to be made. If the circuit is amenable to segmentation for pseudo-exhaustive testing [31], [32], then exhaustive testing of the functional parts can be achieved. Mixed level (gate and functional) representation is available in some simulators. How this hybrid representation compares with the other representations is still an open question. When bridging faults in CMOS circuits are studied [33], switch-level representation might be necessary. Another important issue in modeling ICs and circuit representation is special constructs such as tristate, bidirectional pins and wired logic. All other conditions being the same, if in one simulator these constructs are handled implicitly, and in another they are handled by "workaround" techniques, a comparison of the two systems can be considered "fair."

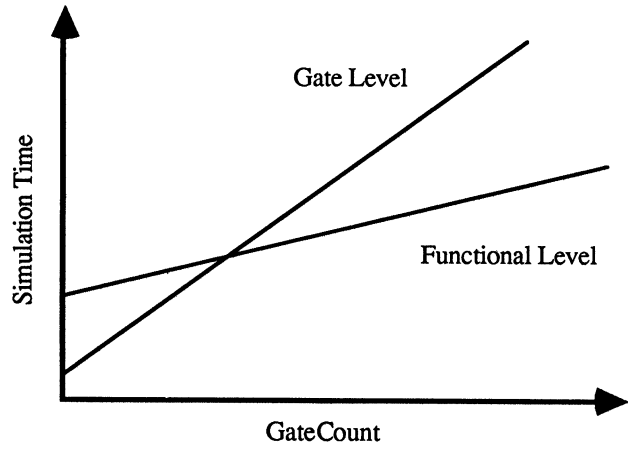


FIGURE 2 Comparing Gate and Functional Representations.

Circuit entry in the system is greatly facilitated by the number of pre-coded primitives and ICs. The size and diversity of the library is another factor to be measured. It is possible to compare libraries by simple counts of the entries. However, depending on the type of the individual units in the library, it is important to make some equivalent gate counts. For example, it is expected that a unit such as an ALU be assigned an equivalent count higher than a simple flip-flop. In addition, it is also important to count unique types of ICs. That is, the SN74163, a mod 16 binary counter, may have several logic family versions. This is a good practice, yet the count of unique ICs is also crucial for diversified modeling.

In the remaining sections of this paper, the presentation will focus on CATS in which: 1) circuit representation is on the gate level, and 2) only single stuck-at faults are considered. Fault modeling, collapsing and coverage will be examined first. Then the performance of the ATPGs will be analyzed using three CATS which are referred to as systems A, B, and C. They run respectively on an Apollo DN550, a Microvax and a Sun 3/160. It is important to reiterate that the intent is not on ranking these systems but on using them as vehicles to formulate guidelines and recommendations for the selection of appropriate benchmark circuits.

FAULT MODELING AND COVERAGE

Fault Modeling

The stuck-at fault is the most widely used fault model. Although it is recommended [19] to consider up to 6 simultaneous faults, none of the commercial CATS attempts to detect multiple faults. *Fault collapsing* and *fault dropping* are other issues that affect the performance of the simulator. According to this latter technique, once the fault is detected it is dropped from the fault list and no further attempts will be made to detect it with other patterns. This is a disadvantage if, for example, the detectability profile for a circuit is to be constructed [28]. Yet, fault dropping could save simulation time and memory requirements.

Fault Collapsing

For single stuck-at faults, generating a *fault list* seems to be a straightforward technique. The process involves grouping the faults in categories (usually called *fault classes*). Then a representative fault from each category is selected to form a *fault list* that is used in fault simulation and test pattern generation. However, formation of the fault list is not implemented the same way by the different CATS. Figure 3 shows sites where both stuck-at zero and stuck-at one faults were injected by the three different CATS, A, B and C.

However, for fanout free outputs, the SA faults are duplicated as illustrated in Figure 3(a), where the SA fault at the output of gate 9 and the open-to fault at the input of gate 11 are the same fault. To include faults on the branches without duplicating faults on the fanout free outputs, special workaround techniques have to be used for systems B and C, two categories of stuck-at faults are distinguished. The first category includes Stuck-At faults (SA) on the primary inputs and on the outputs of each gate. Thus, in case of fanout, every stem and its branches are considered as one net. The second category of SA faults are known as *open-to* faults. This is defined as a broken wire that is held high (open-to-1) or low (open-to-0) depending on the technology of the circuit. An open fault affects the gates downstream of it but neither the stem nor the other branches are affected. The simulators allow the use of either option or both. When both are used, stuck-at faults on the branches can be included in the fault list. This is the case for the output of gate 8 in Figure 3(a). However, for fan-out free outputs, the SA faults are duplicated as illustrated in Figure 3(a), where the SA fault at the output of gate 9 and the open-to fault at the input of gate 11 are the same fault. To include faults on the branches without duplicating faults on the fan-out free outputs, special work around techniques have to be used in the case of system B, but can be easily implemented for system C.

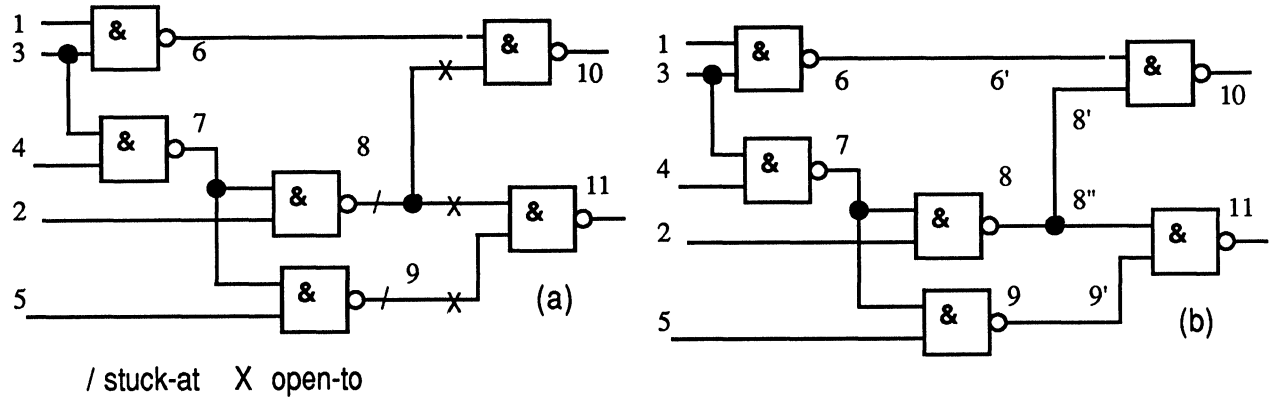


FIGURE 3 Circuit C17: Stuck-At and Open-To Faults: (a) Systems B and C, (b) System A.

TABLE I
Faults and Fault Classes

	System B			System C				
Reference		SA	Open	Both	SA	Open	Both	System A
lines	17	11	12	23	11	12	17	18
faults	34	22	24	45	22	24	34	36
fault classes	22	16	06	22	16	22	22	20

System A considers faults on all inputs and all outputs of each gate, even if the output is input to only one other gate at it is the case for nodes 6 and 9 in Figure 3(b). Thus this technique leads to duplication of the same fault within some fault classes. Also, primary inputs that fanout are dropped when fault collapsing is performed. This practice is only acceptable if no reconverging fanout occurs. The number of faults and fault classes for the circuit shown in Figure 3 are listed in Table I together with the corresponding published data [10] indicated as reference in the table. The total number of faults according to system A is 36 when it cannot exceed 34. Also, the number of fault classes has decreased to 20 because the primary input 3, which has fanout, was removed from the fault list under the assumption that it is dominated by the branches. Another indication of the great divergence in the way fault lists are formed is given in Table II [36].

Here, the number of faults considered by the same commercial CATS and some researchers are tabulated for the 10 combinational circuits announced at ISCAS 85. For any of the circuits, there is a wide range of faults considered by the different references. It is evident that there is a need of standardization of the fault list compilation or a means to find equivalence between the different approaches of collapsing the faults.

Fault Coverage

The *fault coverage* is defined as the ratio of detected faults to the total number of faults. The difference in fault collapsing affects the fault coverage. Systems B and C give the fault coverage, T_F, for all faults considered, but system A calculates the coverage, T_C, for the *fault classes*. In order to appreciate the discrepancies between T_F and T_C, the two coverages will be calculated using the data from system A which is listed in the rightmost column of Table I.

If only one fault class is not detected, then $T_C = 19/20 = 0.95$. The number of faults per class varies from 1 to 4. The corresponding values of T_F are then $35/36 = 0.97$ and $32/36 = 0.89$. For a fair comparison of the performances of the different simulators, a unified fault coverage has to be used.

EVALUATING ATPGS

Test pattern generators are usually based on a certain fault model and a certain level of circuit representation. The stuck-at fault model is the most commonly used model. Although this fault model can be applied for a circuit description on the switch as well

TABLE II
Test Pattern Generation: ISCAS Circuits

Circuit	System A	System B	Murakami	Takamatsu	Rosales
C432	494	282	434	524	738
C499	1092	430	758	758	1126
C880	850	537	942	942	1578
C1355	1508	846	1574	1574	2230
C1908	1813	1300	1876	1876	2935
C2670	2485	1818	2521	2747	4691
C3540	3336	2285	3422	3428	5708
C5315	5100	3356	5314	5350	8708

Murakami et. al. ISCAS 85, p. 678
Takamatsu et. al. ISCAS 85, p. 682
Rosales et. al. ISCAS 85, p. 66t

as on the gate levels, it is usually used with the latter circuit representation.

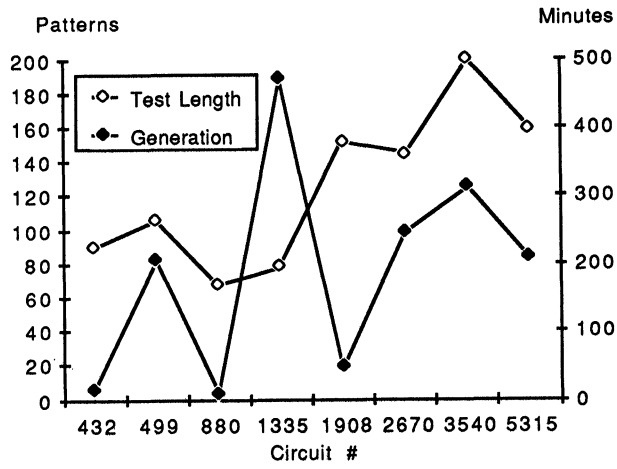
Given a stuck-at fault model and a gate level representation, there are several approaches to test pattern generation: *exhaustive, pseudo-exhaustive, random, pseudo-random, and deterministic*. It is possible to combine more than one approach. For example, an ATPG may start with a random test set and switch to an algorithm test to generate patterns for those faults which were not detected by the random test set. The complexity of the algorithm will affect the speed of pattern generation. However, this speed is not the only factor determining the performance of the ATPG. The length of the test set and its effectiveness (fault coverage) in detecting the faults are also important factors. Hence, it is useful to formulate a relation among the length of the test set, its fault coverage, and its generation time in order to compare different ATPGs. Toward this end, some experiments are performed using the ATPGs of the commercial systems A, B and C to generate patterns for the ISCAS 85 suite as well as other circuits.

Experiment I The speed of operation is affected by the level of circuit representation, the number of states, the fault model as well as the strategy of fault or test pattern generation. For example, in order to assess the performance of an ATPG, test sets for circuits of different sizes (line count) are generated and their lengths and generation times are plotted versus the gate count or the number of faults. An experiment was carried out using System A's ATPG to generate test sets for some of the ISCAS 85 circuits [10]. The results of this experiment are given in Table III, where the number of nets (the number associated with the circuit's name), the gate count, the fault classes, the test length, the test generation time and the fault coverage are listed. The change in test length and generation time with the number of wires is shown in Figure 4. These results indicate that there is no correlation between the number of patterns generated and the size of the circuit. Also, no correlation exists with the number of fault classes. The test generation time depends neither on the number of fault classes nor on the test length.

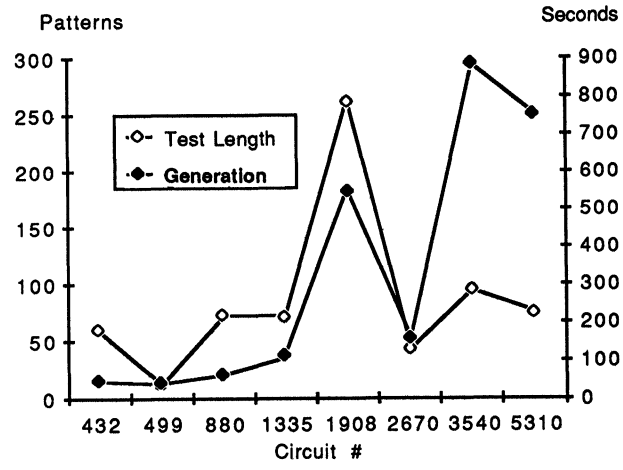
TABLE III
Test Set Length and Generation Time: ISCAS Circuits

Circuit Name	Gate Count	Fault Classes	Test Set Length	Generation Time	Fault* Coverage
C432	160	494	89	14.00	99.00
C499	202	1092	106	165.33	94.00
C880	383	850	67	8.50	100.00
C1355	546	1508	78	470.60	94.13
C1908	880	1813	151	48.22	99.88
C2670	1193	2485	144	246.00	98.01
C3540	1669	3336	199	312.31	99.63
C5315	2307	5100	159	207.30	99.46

*Based on detectable faults only.



(a)



(b)

FIGURE 4 Test Length and Generation Time for System A.

The experiment has also been carried out on system B. The generation times for the two systems A and B are plotted in Figure 5(a). Both curves are normalized to the time registered for the smallest circuit, C432. System B seems to have a better performance. The longest generation time does not exceed 20 times that of the smallest circuit. However, as indicated in Figure 5(b), the fault coverage of system A is better than that of system B for all circuits except C499.

Since for a given system, the same algorithm is used on all circuits, the profiles shown in Figures 4 and 5 reflect mostly the characteristics of the circuits

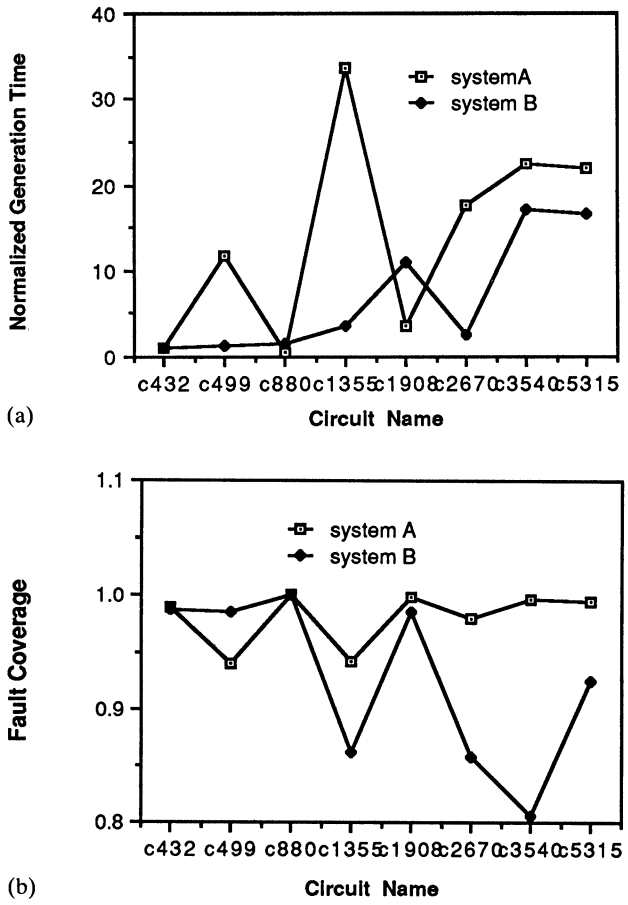


FIGURE 5 Comparison of Systems A and B (a) Test Generation Time, (b) Fault Coverage.

used in the experiments. For example, circuit C1355 is functionally equivalent to circuit C499 except that all XOR gates in the latter circuit are expanded into 4 NAND gates in the former [10]. In system B, for which the XOR function is represented as a gate, the number of faults for C499 is expected to be smaller than for C1355. However, this is not the case for system A which does not include among its primitives an XOR gate. Hence, each XOR is expanded into AND and OR and inverter gates. The result obtained so far confirms that the performance of ATPGs depends on the circuits' topology and size as well as the algorithm. It is necessary to relate the performance to the fault coverage which measures the effectiveness of the test set generated. The generation per fault detected may be considered as the cost per fault. However, an ATPG can attain a certain fault coverage using random patterns and hence for the same cost per fault the test length may be excessively long. A relation that also includes the number of test patterns generated is now under investigation.

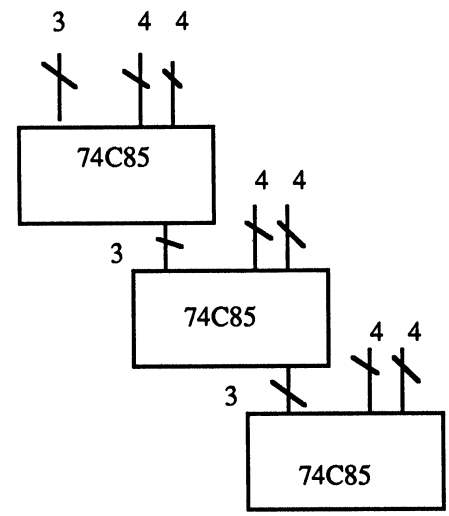


FIGURE 6 Cascaded 4-Bit Comparators.

Experiment II In order to minimize the effect of the circuit in evaluating the performance of an ATPG, replication of a certain circuit is used to obtain larger circuits. For example several 4-bit comparators (National Semiconductor's 74C85) have been cascaded, as shown in Figure 6, to form larger comparators [37].

In this experiment, the fault classes are assumed to be proportional to the size of the circuits. By replicating the same structure to form larger circuits, the curve representing the test generation time versus the circuit size characterizes the ATPG itself. Plots of the test length and the generation time for system A are shown in Figure 7. In contrast to Figures 4 and 5, here the curves are monotonically increasing functions of the number of faults.

The experiment for the comparators was repeated for balanced parity trees with primary inputs varying

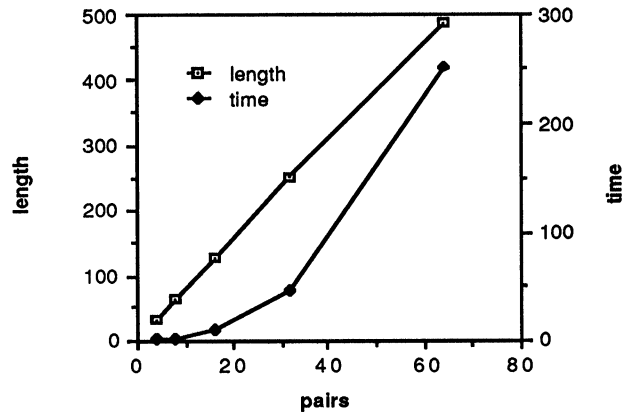
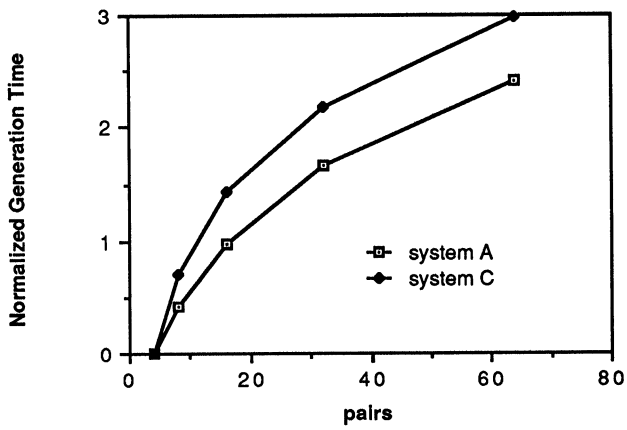


FIGURE 7 Test Pattern Generation for Comparator Circuits.

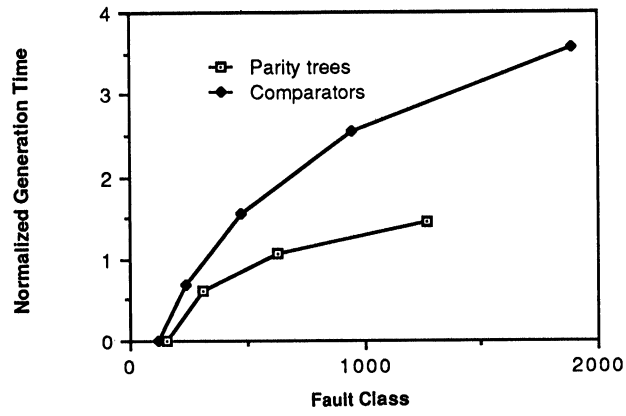
from 4 to 64. The test generation times for the comparators (and the parity trees) on systems A and C are shown in Figure 8. The values are normalized to the generation time for the smallest circuit (a 4-pairs comparator and a 4-input parity tree). In order to emphasize the difference between the two curves for the small size circuits, a logarithmic scale is used.

The curves for the comparators indicate that the two ATPGs behave exponentially but the rate of increase is higher for system C than for system A. For the parity trees, however, the curve for system A rises at a higher rate than that for system C. The different behavior of the two systems can be explained as follows: system C generates only 4 patterns for any size parity tree (a correct optimum test set [6]) while for system A, the number of test patterns increases with increased input pairs.

The behavior of systems C and A for the two types of circuits is shown in Figure 9. In this case the time is plotted versus the fault classes. It is clear that for system A, handling XOR gates is a problem. This explains why the two circuits that contain XOR constructs, namely C499 and C1355, need longer test generation time. The observations made in this section help formulating preference in selecting the benchmark circuits and raise some practical considerations in benchmarking CATS.

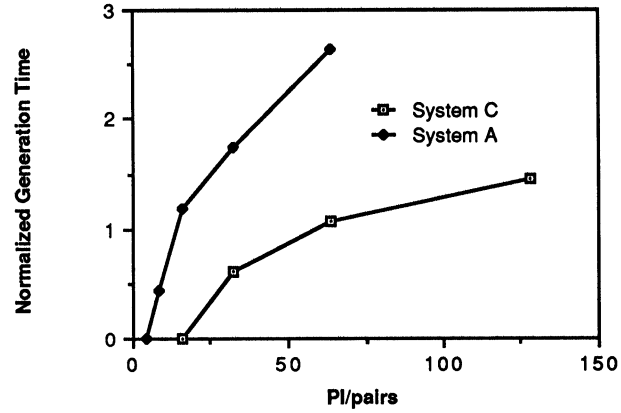


(a) Comparators

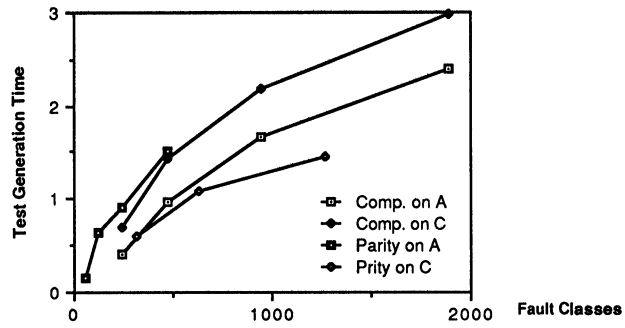


(b) Parity Trees

FIGURE 8 Test Generation Times on Systems A and C.



(a) System C



(b) Systems A and C

FIGURE 9 Test Generation Times for the Comparators and Parity Trees.

BENCHMARK CIRCUITS

The several experiments described in earlier sections lead to the following recommendations regarding the approach in evaluating and comparing ATPGs as well as the selections of benchmark circuits.

The fault coverage is an important parameter to be used as a measure of the capabilities of the ATPG to generate a complete test set. The fault coverage may be measured in terms of the faults classes or the individual faults. Hence, the same measure needs to be used. However, since there is no uniform way to

generate fault lists, the fault coverage comparison is not meaningful. A starting point for comparing the fault coverage is then to have the same fault list.

Different ATPGs do not use the same test pattern generation algorithm and they do not handle different circuit constructs in the same fashion. For example the handling of XOR trees by system A is more efficient than by system B. But the reverse is true when the comparators are used. Hence, a variety of circuits types need to be available to allow comparison of the different aspects of the ATPGs.

In evaluating the performances of the ATPGs as the circuit complexity increases (number of lines), it is evident the diversity of the ISCAS 85 benchmarks did not permit an assessment of the performances. However, when circuits consisted of replication of the same construct, each ATPG was represented by a characteristic curve. Such curves for different ATPGs can be easily compared.

Finally, there is the inclusion of different special circuit constructs: tristate, bidirectional, transmission gates and special design for test constructs such as scan path or self-checking circuits. Ideally, the profile of a benchmark circuit should include the characteristics listed in Table IV.

There are some suites of benchmark circuits that have been announced, among them are ISCAS 85 [10], ISCAS 89 [11], HLS 89 [52]. To complement these suites, benchmarks *regular* circuits that are formed by cascading MSI-size of circuits have been developed [40]. This last suite consists of comparators, parity trees, ALUs, funnel shifters, counters, and other circuits.

TABLE IV
Characteristics of Benchmark Circuits

Type	Combinational Versus Sequential
Technology	Bipolar, CMOS, etc
Number of	Gate
	States
	Primary inputs
	Primary outputs
	Lines
Fanout	Average and maximum
Fanin:	Average and maximum
Special Structures:	
	Tristate
	Transmission gate
	Scan path

PRACTICAL CONSIDERATIONS

When benchmark circuits are selected to exercise the different functions of the CATS, it is important to facilitate the transfer of the circuits from one system to another. Each of these systems has its own circuit description language (or schematic capture), it is thus necessary to transfer the circuit descriptions to different systems. It is always possible to enter the benchmark circuits in one system, the *Source System,* SS, and write several translators to convert the circuits to the languages of other systems, *Target Systems* (TS). This type of exchange is illustrated in Figure 10, where $n - 1$ translators are needed but there is no flexibility in passing from the TS systems to SS. For each system to swap data with all others, $n(n - 1)$ translators are needed. The problem of moving data among different systems has made the standardization of data exchange essential. Many vendors have been adopting and supporting standard neutral languages or formats such as VHDL [46] and EDIF [13], among others [12], [48]. It is thus natural to explore the possibility of exchanging the circuit descriptions through these intermediate languages. This is illustrated in Figure 11. Here each system is both a source and a target. It is then mandatory to have a translator from each system's language to the neutral language, *a writer,* and vice versa, a *reader* [13]. Thus only $2n$ translators would be sufficient for full communication among the systems.

Circuit description may be entered in the CATS either in a netlist form or in a schematic form. In either case, the circuit may be modeled at the switch, gate or functional level. A netlist may be considered as a schematic without graphical symbols to represent

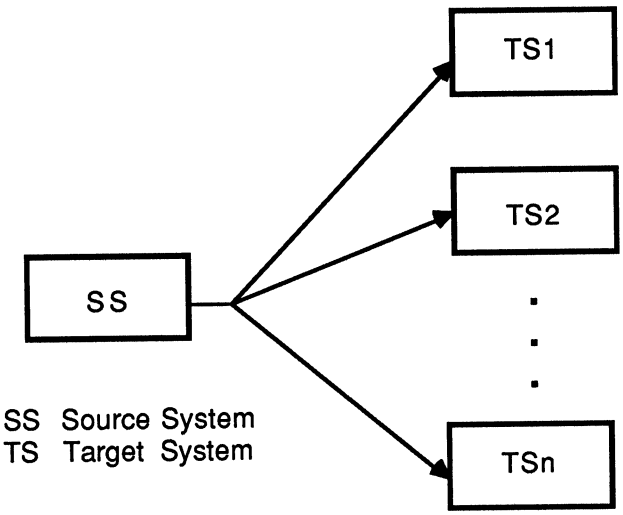


FIGURE 10 Translation from a Source System (SS) to Many Target Systems (TS).

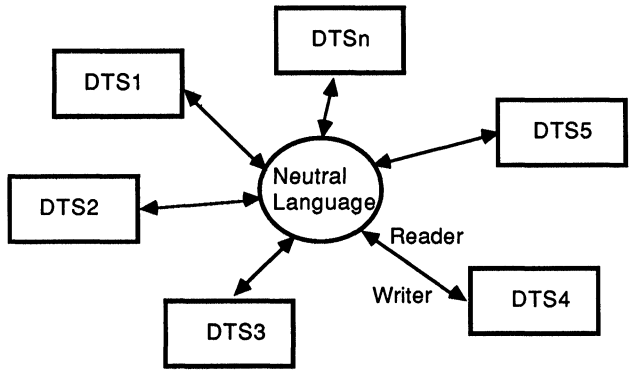


FIGURE 11 Neutral Language with Several Writers and Readers.

the functional units or the gates. However, other properties (orientation, names, size of fonts) associated with the graphical symbols make the translations of schematics more complex than netlists.

The circuits may be described via VHDL or EDIF. VHDL which is an ADA-like language is better suited for high level behavioral description [48]. While EDIF, a LISP-like language, can migrate netlist descriptions to schematic designs directly [14]. Both languages can be used in simulation modeling [25].

In this study, EDIF was selected as the neutral language because of the following reasons. First, this standard has been initiated and supported by the industry. At the 26th Design Automation Conference several vendors collaborated and exchanged a circuit in EDIF format and displayed it on their corresponding systems [30]. Also, a netlist in EDIF can easily be put into a schematic form and vice versa. The translation of the NSF 89 benchmarks into EDIF is reported at EDIF World 89 [26].

SUMMARY AND CONCLUSIONS

Computer-Aided Testing Systems as computer software packages need to be assessed and compared. As benchmark programs are written to evaluate computers, digital circuits need to be selected to evaluate testing systems. Like computer systems, CATS require several types of benchmarks, since no single benchmark can suffice to measure all attributes of a system.

The focus in this study was on Automatic Test Pattern Generators although many of the results can also be applied to fault simulators. Experiments were carried out using three CATS and several benchmark circuits. Only combinational circuits were considered. Further investigation will make use of sequential circuits for which test pattern generation presents more complexity [34]. The preliminary results show that there is no general agreement on how: 1) fault collapsing is performed, and 2) fault coverage is calculated. Hence in comparing ATPGs, a unified fault list and fault coverage need to be used. In addition, the performance of the ATPGs is a function of the circuit representation and size as well as the test generation algorithm. In order to compare the performance of the ATPGs as the circuit under test increases in complexity, it is important to use regular structures that consist of replication of medium size circuits. Practical considerations involved in benchmarking are also examined. Emphasis is on the transfer of circuits between different CATS and the use of EDIF as a neutral exchange language.

Acknowledgments

The author would like to thank Cendrine Bitard, Zan Matsumoto, and Atilio Canessa for running the simulation experiments. This work was supported in part by grants from the National Science Foundation, NSF #MIP8796340, and the Clare Boothe Luce Foundation.

References

[1] V. Agrawal, "Probabilistically Guided Test Generation," *IEEE Int'l Symp. on Circuits and Systems (ISCAS)*, Kyoto, Japan, 687–690, June 1985.
[2] AIDA Corp., "User's Manual," Santa Clara, CA.
[3] AIM Technology, "Evaluating UNIX Computers Through Benchmarking," Press Release, Santa Clara, CA, January 1985.
[4] N. Benwell, *Benchmarking: Computer Evaluation and Measurement*. Washington, DC: Hemisphere Publishing Corp., 1975.
[5] T. Blank, "Survey of Hardware Accelerators Used in Computer-Aided-Design," *IEEE Design & Test of Computers*, 1(3), 21–39, August 1984.
[6] D.C. Bossen, D.L. Ostapko, and A.M. Patel, "Optimum Test Patterns for Parity Networks," *Proc. AFIPS Fall 1970 Joint Computer Conference*, Houston, Texas, 37, 63–83, November 1970.
[7] G.H. Bowers Jr. and B.B. Pratt, "Low Cost Testers," *Proc., IEEE Int'l Test Conf.*, Philadelphia, PA, 40–49, November 1984.
[8] R.K. Brayton et al., *Logic Minimization Algorithms for VLSI Synthesis*. Boston: Kluwer Academic Publishers, 1984.
[9] B.E. Brown and R.L. Judd, "Benchmarking Graphics Workstations," *IEEE Int'l Conf. on Computer Workstations*, 116–121, 1985.
[10] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator into FORTRAN," *IEEE Int'l Symp. on Circuits and Systems (ISCAS)*, Kyoto, Japan, 659–662, June 1985.
[11] K.T. Cheng, "Concurrent Test Generation and Design for Testability," *IEEE International Symposium on Circuits and Systems (ISCAS)*, Kyoto, Japan, 1935–1941, June 1989.
[12] J.D. Crawford, "EDIF: A Mechanism for the Exchange of Design Information," *IEEE Design and Test*, 63–69, February 1985.

[13] Electronic Industries Association (EIA), EDIF version 200, EIA Intrim Standard No. 44, Washington, DC, April 1987.
[14] J. Eurich, "A Tutorial Introduction to the Electronic Design Interchange Format," *Proc. 23rd Design Automation Conference,* 327–333, June 1986.
[15] J. Eurich, "EDIF: Test Generation and Fault Simulation," Digest of Papers, *Spring COMPCON 88,* San Francisco, CA, 342–349, February–March 1988.
[16] H. Fujiwara, "FAN: A Fanout-Oriented Test Pattern Generation Algorithm," *IEEE Int'l Symp. on Circuits and Systems (ISCAS),* Kyoto, Japan, 671–674, June 1985.
[17] S.H. Fuller et al., "Evaluation of Computer Architecture via Test Programs," *National Computer Conference,* 1977.
[18] P. Goel, "Benchmarking Digital Testing Systems," Digest, *Spring '88 COMPCON,* San Francisco, CA, March 1988.
[19] L.H. Goldstein, "A Probabilistic Analysis of Multiple Faults in LSI Circuits," *IEEE Computer Society Repository* R-77, 1977.
[20] R.D. Grappel and J.E. Hemenway, "A Tale of Four μPs: Benchmarks Quantify Performance," *EDN,* 179–185, April 1, 1981.
[21] D.L. Greer, "The Quick Simulator Benchmark," *VLSI Systems Design,* 8(12), 41–57, November 1987.
[22] H.A. Grosch, "High Speed Arithmetic: The Digital Computer as a Research Tool," *J. Opt. Soc. Am* (4), April 1953.
[23] J.A.L. Hughes, S. Mourad, and E.J. McCluskey, "An Experimental Study Comparing 74LS181 Test Sets," Digest, *Spring '85 COMPCON,* San Francisco, CA, 384–387, March 1985.
[24] R. Karpinski, "Paranoia: A Floating Point Benchmark," *Byte,* 10(2), 223–235, February 1985.
[25] E.C. Klass and M. Wayne, "IBM's Plans for VHDL and EDIF," Digest of the *Fourth EDIF User Group Workshop,* Colorado Springs, CO, 141–151, September 1988.
[26] P. Lee et al., "EDIF in Benchmarking Digital Testing Systems," *EDIF World,* September 1989.
[27] M.A. Linton, "Benchmarking Engineering Workstations," *IEEE 1985 Int'l Conf. on Computer Workstations,* 108–115, 1985.
[28] Y. Malaiya and S. Yang, "The Coverage Problem for Tandem Testing," *Proc. Int'l. Test Conf.,* Philadelphia, PA, 237–245, October 1984.
[29] F.A. Mann, "IEEE Standard Logic Symbols, Why Use Them?," Digest, *Spring '88 COMPCON,* San Francisco, CA, March 1988.
[30] V.F. McCord, "Trials and Tribulations of EDIF 200 Schematic Transfer," the *Fourth EDIF World,* Colorado Springs, CO, 105–111, September 1988.
[31] E.J. McCluskey and S. Bozorgui-Nesbat, "Design for Autonomous Test," *IEEE Trans. on Computers,* C-30(11), 866–875, November 1981.
[32] E.J. McCluskey, "Verification Testing—A Pseudo-Exhaustive Test Technique," *IEEE Trans. on Computers,* C-33(6), 541–546, June 1984.
[33] S. Millman, "Using LASAR for CMOS Bridging Fault Simulation," TR #88-2, Center for Reliable Computing, Stanford University, CA, March 1988.
[34] A. Miczo, "The Sequential ATPG: A Theoretical Limit," *IEEE Int'l Test Conf.* 143–147, September 1983.
[35] S. Mourad and E.J. McCluskey, "On Benchmarking Digital Testing Systems," A Poster Session, *Proc. Int'l Test Conference,* Washington, DC, 997, September 1988.
[36] S. Mourad, "Practical Considerations in Benchmarking Digital Testing Systems," Poster Session, *Int'l Test Conf.,* September 1989.
[37] S. Mourad, M. Martonosi, and E.J. McCluskey, "Benchmarking Magnitude Comparators," *New Directions in VLSI Design Workshop,* Victoria, Canada, October 1989.
[38] M. Murakami and H. Kikuchihera, "Test Generation for LSI Circuits Using Extended Nine-Valued Method," *IEEE Int'l Symp. on Circuits and Systems (ISCAS),* Kyoto, Japan, 659–662, 675–678, June 1985.
[39] "Selection Guide for Digital Rest Program Generation Systems," Prepared for Chief of Naval Material by Test and Monitoring System Office, NAVAMATINST 3960,9A, 27 March 1979.
[40] "NSF 89 Suite of Benchmark Circuits," Santa Clara University, CA.
[41] W. Patstone, "16-Bit Benchmarks: An Update with Experimentation," *EDN,* 169–175, September 16, 1981.
[42] B. Rosales and P. Goel, "Results from Application of a Commercial ATG System to Large-Scale Combinational Networks," *IEEE Int'l Symp. on Circuits and Systems (ISCAS),* Kyoto, Japan, 667–670, June 1985.
[43] Y. Takamatsu and K. Konoshita, "An Efficient Test Generation Method" *IEEE Int'l Symp. on Circuits and Systems (ISCAS),* Kyoto, Japan, 679–682, June 1985.
[44] E. Trischler and M. Schultz, "Application of Testability Analysis to ATG: Methods and Experimental Results," *IEEE Int'l Symp. on Circuits and Systems (ISCAS),* Kyoto, Japan, 691–694, June 1985.
[45] P. Varma, A.P. Ambler, and K. Baker, "An Analysis of the Economics of Self-Test," *Proc. Int'l. Test Conf.,* Philadelphia, PA, 95–106, October 1984.
[46] VHDL Reference Manual, Versions 5.0, 7.0, & 7.2, Air Force System Command, Wright Patterson Air Force Base, Ohio, July 1984, January 1985, and August 1985.
[47] VLSI "Survey of CAE Systems," *VLSI Systems Design,* 8(6), 85–105, June 1986.
[48] R. Waxman, "The Many Languages of Electronic Computer-Aided Design," *Proc. IEEE Int'l Conf. on Computer Design,* 74–77, 1984.
[49] J.A. Waicukauski and E. Lindbloom, "Transition Fault Simulation by Parallel Pattern Fault Propagation," *Proc., Int'l. Test Conf.,* Washington, DC, 542–549, October 1986.
[50] R.S. Wei and A. Sangiovanni-Vincentelli, "New Front End and Line Justification Algorithm for Automatic Test Generation," *Proc. Int'l. Test Conf.,* Washington, DC, 122–128, October 1986.
[51] R. Weicker, "Dhrystone: A Synthetic Systems Programming Benchmark," Communication of the ACM, 27(10), 1013–1030, March 1986.

Biographies

SAMIHA MOURAD is the Clare Boothe Luce Professor at Santa Clara University, California, where she teaches and conducts research in digital design and testing. Before joining the faculty of Santa Clara, she was at Fordham University in New York and the Bendix Corporation in Teterboro, New Jersey. Professor Mourad is a member of IEEE, ACM and Sigma Xi.