Structural Testability Data Analysis of Synchronous Sequential Circuits

Santhoshini P[1],Vishwa R[2],Thangaselvam R[3],Vasanth P[4] ,Vineeth A[5]
[1]*Assistant Professor, Department of ECE, R.M.D Engineering College, Thiruvallur*
[2,3,4,5]*B.E Student Department of CSE, R.M.D Engineering College, Thiruvallur*

***Abstract*—Industries are increasingly focusing on sophisticated components for the development of multi-purpose machinery and devices, driven by the habitual integration of engineering and technology. Bounds on test sequence length can be used as a testability measure. We give a procedure to compute the upper bound on test sequence length for an arbitrary sequential circuit. We prove that the bound is exact for a certain class of circuits. Three design rules are specified to yield circuits with lower test sequence bounds.**

***Index Terms*—VLSI, Deep Learning, FPGA, Neural Networks.**

I. INTRODUCTION

The automatic generation of test sequences for sequential digital systems has proven to be a hard problem to solve. Unlike combinational circuits for which test generation algorithms exist [8,9,17,18] to mention a few, that use only structural information to generate a test for any fault in the circuit, no complete algorithm is available for sequential circuits. While some recent progress is evident and promising [2,12,14], the best current implementations still spend several CPU hours on circuits of moderate size. Neither does a theoretical basis exist for sequential circuits comparable to the theory of fault detection and diagnosis in combinational circuits. Effective testability analysis techniques have been developed for combinational circuits and testability measures based on controllability/observability considerations have been used to speed up the test generation process. No effective testability measure exists for sequential circuits. Miczo [15] has proved a bound on the synchronizing sequence [11] which may be used as a measure of testability. He has shown that circuits that have synchronizing sequences longer than 3^n-2^n-1, where n is the number of flip-flops in the circuit, are untestable by an ATPG program which uses only structural data. The result however does not tell how circuits can be DESIGNED THAT ARE ATPG-TESTABLE. It is known that sequential circuits may require a very long input sequence to detect a fault in the circuit. Scan design techniques [20] are used to reduce the test sequence length. However, scan designs incur area overhead and speed penalties. Some manufacturers therefore still make chips that have no scan paths. In such a scenario, it is necessary to design circuits so that the length of the longest test is minimal. Consider the following circuits [4] mentioned in the table 1 below. The column labeled "Bound" gives the length of the longest test to detect a fault. The CPU time was obtained on VAX 8650. Even though CHIP-A is three times larger than the Traffic Light Controller(TLC) circuit, it requires one-fifth the time for test generation. TLC has a bound on test sequence length of 243 , which is almost 2.5 times the bound for CHIP-A.In literature, test sequence length is usually used to specify the number of test patterns that need to be applied to achieve a particular fault coverage. Here, we use the phrase in the context of the worst-case fault. It denotes the longest sequence needed to detect a fault in a sequential circuit. Test sequence length is an effective measure of testability of a sequential circuit as demonstrated by the above table. We obtain an upper bound on the test sequence length to detect a fault. We also prove that the upper bound is exact for a certain class of circuits. As a by-product of the bound, we show that our results can also be used to design circuits that require shorter sequences to test. A graph model is used for the circuit to derive the upper bound. We first partition the circuit into subcircuits, each of which is treated as an independent machine. The upper bound for testing the independent machines is computed. We then compute the bound on test sequence length in terms of the bound of the independent machines.

Table 1. Test Generation for Two Sequential Circuits

Circuit	# gates	# FFs	Bound (Experimental)	Test Gen. CPU sec.
TLC	355	21	243	1245.65
CHIP-A	1112	39	102	268.80

II. INTERCONNECTIONS OF SEQUENTIAL MACHINES

In this section, we look at two simple interconnection schemes of sequential machines from a test generation point of view. The series connection and the parallel connection of machines are examined. In a later section, we show that any circuit can be analyzed for upper bound using the analyses carried out in this section. Interconnection of machines has been studied in a different context earlier [11] for behavioral analysis. But here the intention is to find an upper bound for the interconnection in terms of the upper bounds of the constituent machines.

2.1 Series Connection of Machines

Two machines may be connected in series as shown in Fig. 1. The primary inputs feed $M1$, whose outputs feed M2. Both $M1$ and $M2$ are driven by a single master clock. Let the bounds for machine $M1$ be $B1$ and for $M2$ be $B2$. For the present, one may assume that $B1$ and $B2$ are the number of states in $M1$ and $M2$ respectively.

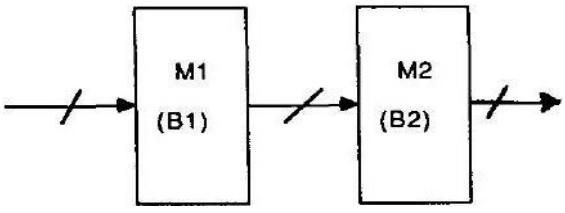


Fig. 1 Series Connection

Claim: The upper bound on test length for series interconnection is $B1 + B2$.

Proof: A fault could be in either $M1$ or $M2$. Consider the case when the fault is in M1. It requires in the worst case $B1$ clock pulses to propagate the effect of fault to the output of $M1$. Once the fault is visible at the output of $M1$ (or equivalently at the input of $M2$), $B2$ is the bound on the number on the clock pulses needed to propagate the effect of the fault to the output of $M2$. Hence we need $B1 + B2$ clock pulses. In other words, we need a test sequence of length $B1 + B2$.

Consider the case when there is a fault in M2. It requires at most $B2$ clock pulses to propagate the effect of the fault to the primary output and set up line justification problems for the input lines of $M2$. The input lines of $M2$ are the output lines of M1. It requires a maximum of $B1$ input vectors to solve the line justification problems at the input of $M2$. Hence to detect a fault in M2, at most $B1 + B2$ input vectors are needed.

Hence for the series connection of two machines, in the worst case $B1 + B2$ input vectors are needed to detect a fault.

A typical example of series connected machines is the shift register. We can think of each flip-flop as a primitive sequential machine, whose upper bound for test generation is 1 , since any fault in the flip-flop can be detected by applying one test vector. Only input/output faults are being considered here. A shift register consists of several flip-flops serially connected. The upper bound on the length of test sequence is the sum of the upper bounds of each flip-flop. Hence the upper bound is equal to the number of flip-flops in the register.

Note however that the number of states of the equivalent machine of a series interconnection, is equal to the product of the states of each machine [11]. For Fig. 1, we would have $B1 * B2$ states. But we do not have to visit all the $B1 * B2$ states to detect a fault.

Also, consider the circuit shown in Fig. 2, which is the circuit for a mod 256 counter constructed from two mod 16 counters. Each counter is a ripple counter. Since the interconnection has two asynchronous machines, our analysis does not apply. For the ripple counter, 256 clock pulses are needed to test for a fault.

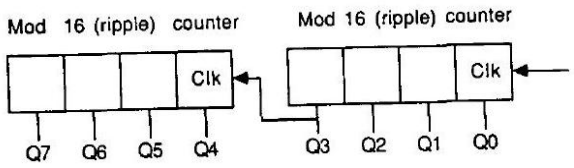
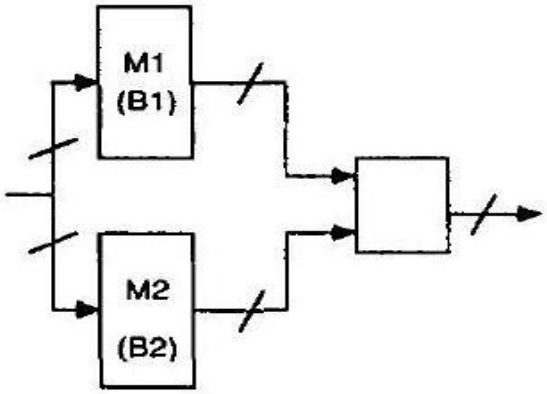


Fig. 2 A ripple counter example

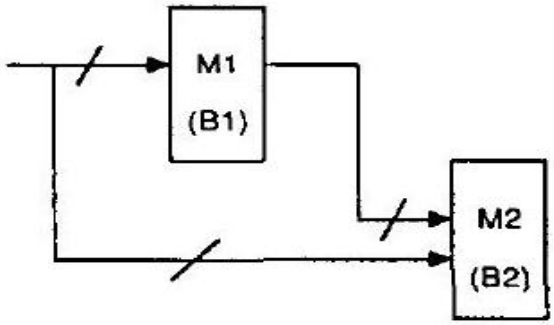
2.2 Parallel Connection of Machines

We consider the connections shown in Fig. 3 as parallel connections of machines. In parallel

connections of machines, there are some inputs that fan out to more than one machine and there is a reconvergence of the inputs. Let the bounds on the test length for $M1$ and $M2$ be $B1$ and $B2$ respectively.



(a)



(b)

Fig. 3 Paraltel Connection of Machines

Claim The upper bound for the parallel interconnection is $B1^{*}B2$.

Proof: In Fig. 3(a), consider the case when there is a fault in $M1$. In order for the fault to be detectable, the effect of the fault has to be propagated to the output of $M1$ and also the output of $M2$ has to have propagating values. Since the two constraints have to be solved simultaneously, by a common input, in the worst case we have to visit all the states of the equivalent machine. Hence the upper bound for detecting the fault will be $B1^{*}B2$. The case in which there is a fault in $M2$ is identical to the case we have discussed.

Similarly in Fig. 3(b), since the input can simultaneously change the state of $M1$ and $M2$, we require in the worst case $B1 * B2$ vectors to detect a fault in either $M1$ or $M2$.

III. BOUND FOR AN ARBITRARY CIRCUIT

In this section, we describe a scheme for computing the upper bound on the test sequence length for an arbitrary circuit. In [16], a bound for the search space is obtained. The search space bound is 2^{m+n}, if there are m latches in the circuit and n inputs. Implicitly, the bound on the test sequence length is 2^{m}. However, if ATPG is used and the initial state is assumed to be unknown then the bound will have to be modified as 3^{m}, since ATPG uses three logical values viz. 0,1 , and X. This is a very pessimistic bound. Consider for example a 4 -bit shift register. The bound on the test length given by the above formula would be 3^{4}, but since the shift register is a series connection of 4 flipflops, each of which has a bound of unity, the upper bound on test sequence length would be 4 and not 3^{4}. We give a tighter bound on test sequence length and prove that the bound is exact for a certain class of circuits. From the previous section, it is obvious that our method gives an exact bound on shift registers and the class of synchronous circuits which can be recursively decomposed it to a series or parallel connection of sub-machines.

The circuit is represented as a directed graph. There is an edge for each line in the circuit. The primary inputs, gates, flip-flops and fanout stems are represented as nodes in the circuit graph. The graph for a general sequential circuit is a cyclic graph because of feedback lines in the circuit. Fig. 4 shows an example sequential circuit and its graph representation is shown in Fig. 5. As a first step in computing the bound we partition the circuit into strongly connected components, that is, within each component every node is reachable from any other node. Each strongly connected component is collapsed into a single supernode. The supernode represents the submachine. The graph thus transformed will be an

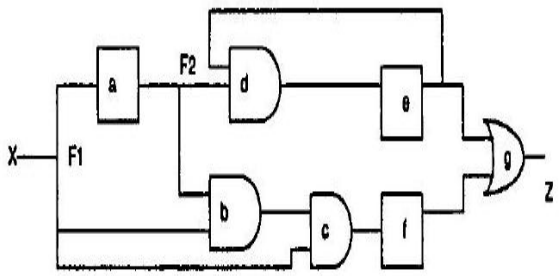


Fig. 4 A Sequential Circuit Example

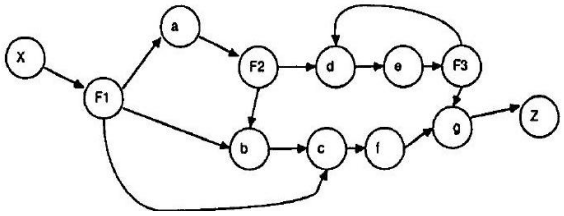


Fig. 5 (a) A directed graph representation of circuit in Fig. 4

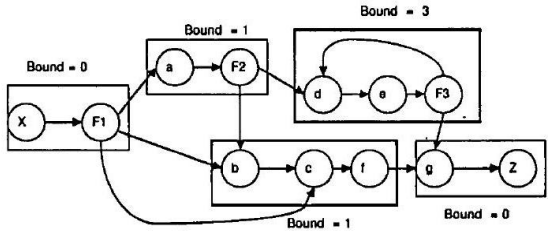


Fig. 5(b) Graph after condensation and collapsing

combinational elements acyclic graph representing connections between independent submachines. Finding the strongly connected components and transforming the graph into an acyclic graph is the standard problem of finding the condensation of a cyclic graph [6,10] . All the strongly connected components of a graph can be found in polynomial time. A linear algorithm exists [19], which uses depth-first search on the graph for finding the strongly connected components. It can be proved that the condensation of a cyclic graph is unique.

After the condensation graph is found, the following collapsing is done for combinational elements. Combinational elements that form a fanout free region [1] are collapsed into a single combinational element with bound of zero and merged into a sequential machine that is fed by the combinational gates. If no sequential machine is driven by the fanout free region, then the region is left as is, with a bound of zero. Fanout nodes are merged into machines that feed the fanout nodes. This processing is illustrated in Fig. 5(c). F2 is merged into machine a, b and c are merged into f. F 1 is merged into X .

The upper bound for each submachine in the graph is 3^n , where n is the number of latches in the submachine. Since each flip-flop influences every other flip-flop - in a strongly connected in the circuit graph - in order to generate a test sequence, in the worst case, we have to go through all the states of the machine. There are 3^n possible states, since each flip-flop can be either in the 0,1 or X (unknown or uninitialized) state. Flip-flops by themselves not contained in any machine have a bound of 1. In addition, we do series collapsing of sub-machines; if two sequential machines are in series, we combine them as one and add the bounds of the two machines.

An example of this graph transformation is shown in Fig. 6. The circuit [14] is an implementation of a sequential machine [15] where it is claimed to pose a formidable task for an ATPG program. In Fig. 6, each submachine has an upper bound of 3 . Since the transformed graph has a parallel connection of machines, it is clear from our discussion in the previous section that the upper bound for the entire circuit is $3^*3 = 9$.

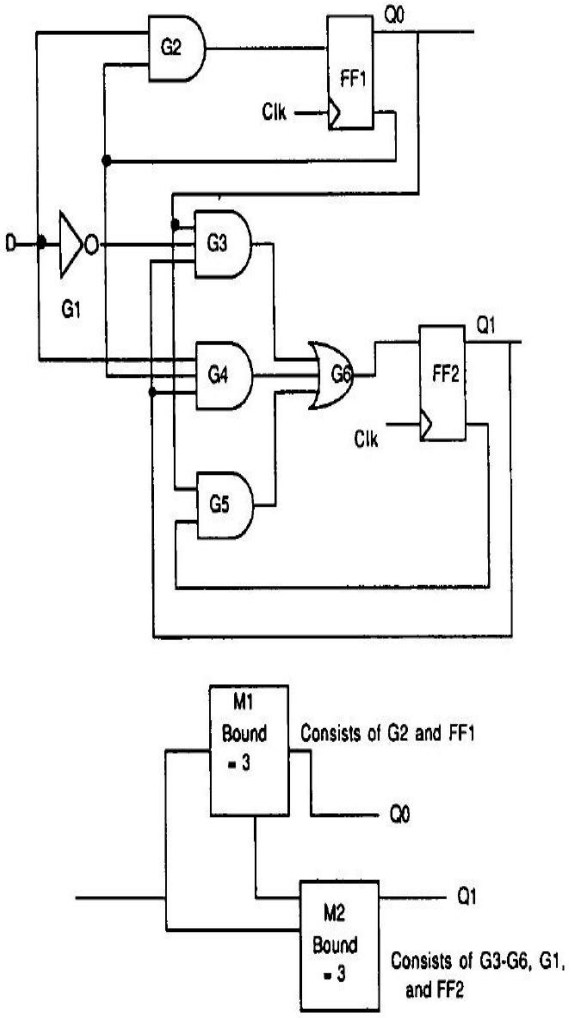


Fig 6. An example sequential circuit and its collapsed schematic corresponding to its condesation graph

It is highly improbable that all circuit graphs reduce to one of the three forms discussed in the previous section. Some circuits may reduce to series-parallel structure as shown in Fig. 7.

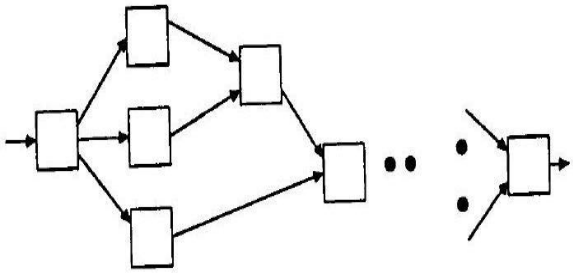


Fig. 7 A series Parallel Structure

Applying the reduction rules of the previous section, i.e., adding bounds of machines in series and multiplying bounds of machines in parallel, we will be able to compute the bound for the overall circuit. For example the circuit corresponding to Fig. 8, which is series-parallel graph has a bound of 32 . We now give a general procedure for finding the bound for a circuit whose condensation graph is arbitrary. We prove that the procedure gives exact bound for circuits whose condensation is series-parallel. The problem we are faced with in a non-series parallel graph is the arbitrary reconvergence structure of submachines whose outputs fanout to more than one sub-machine.

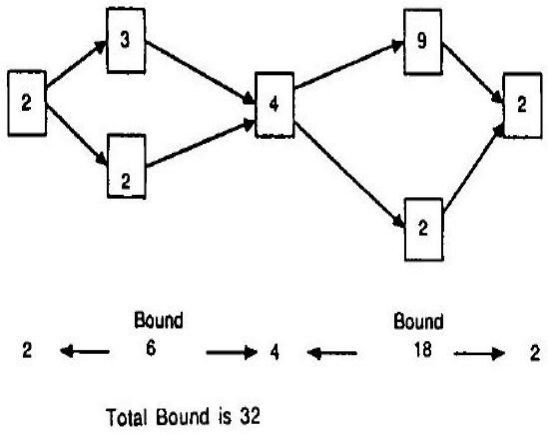


Fig. 8 Computing the bound for a simple series-parallel stem

We use the idea of stem regions [13] to analyze such reconvergence structures. A reconvergent point that is not driven by any other reconvergent point is called a closing reconvergent point of a stem. We are concerned only with closing reconvergences of stems for finding the bound. The stem regions can be found in $O(n\log(n))$ time[7]. The region of a stem X, which lies in the region of stem Y is properly contained in the stem region of stem Y.

In the condensation graph, we identify each node with a level. Primary inputs are at level 0 . A node is at level $i+1$, where $i = \max\{$ level of predecessor nodes $\}$. We maintain a list of stems, ordered by level. Within a stem region of stem s, we define the (relative) depth of each node as the difference between the levels of the node and the stem s. The depth of a stem region corresponding to a closing reconvergence is defined as the difference in the levels of the reconvergent node and the stem. The number of submachines in any path from the stem to its reconvergence is at most equal to the depth of the stem

region corresponding to that reconvergence node. The procedure to find the bound for an arbitrary stem region is describe below.

1. Consider the stem at the lowest level that is still unprocessed.
2. $i = 1$. For all closing reconvergent points do steps 3 - 6
3. Starting at depth i, find the minimum set of machines (nodes) that when removed from the circuit, will isolate the stem and the reconvergent point i.e., find the cutset for the two nodes in the graph. The level of any machine has to be at most i, also it has to be as close to i as possible. If any machine is a stem, mark it as processed. The stem region of this machine is enclosed in the region of the stem in question.
4. The set of machines found in step 3, are machines that are in different paths from the stem to its reconvergence point. In other words, they are in parallel and can therefore be reduced to a single machine whose bound is equal to the product of the bounds of each machine.
5. If reconvergence is not reached, increment i and goto step 3.
6. We get an equivalent machine at each depth following the processing described in steps 3 and 4 . The equivalent machines at depth $i, i+1, i+2, \ldots$ are in series. The bounds of the equivalent machine at each depth are added. Finally the bound of the reconvergence is added.

For combinational elements in the condensation graph, the following processing is done. If a stem region has only combinational elements, the bound is 0 . If a combinational element occurs in a cutset, the bound is considered to be 1. If the combinational element occurs by itself, then the bound is considered to be 0 .

Consider Fig. 9(a), which shows a stem region of arbitrary structure. The stem A has two closing reconvergence points F and K. Each box is a submachine. The depths are indicated above the boxes. The bound for each submachine is indicated in the corresponding box. Consider the reconvergence F. The stem region has a depth of 2. At level 1 , the cutset is $\{B, C\}$ with the bound of 27 . At level 2, the bound is again 27, with D and E forming the cutset. The overall bound is therefore $27 + 27 = 54$, to which we add the bound of F to get 56 . This equivalent connection is shown in Fig. 9(b). Similarly, we compute the bound for the region corresponding to K as the reconvergence. Note in this case I is an element of the cutset at depths 1,2 , and 3 . The equivalent structure is shown in Fig. 9(b).

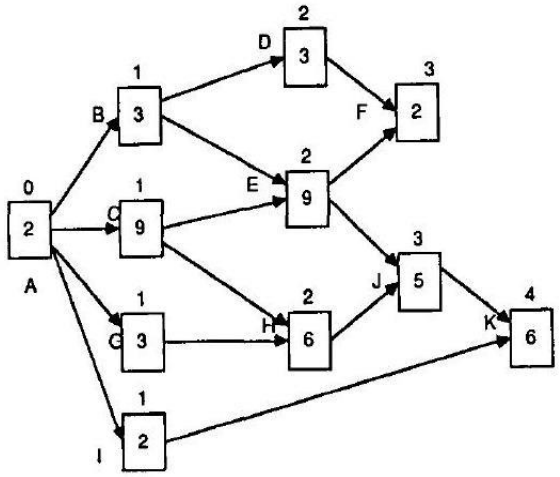


Fig. 9(a) An arbitrary stem region

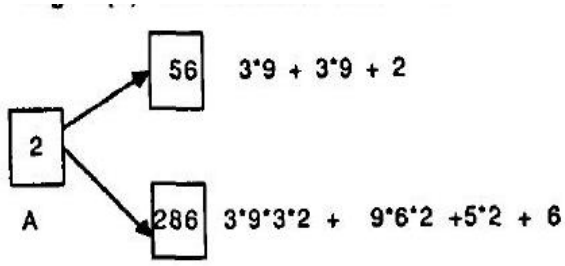


Fig. 9(b) Equivalent Structure

After the above processing is done, the corresponding circuit graph becomes a forest. Some of the nodes may be shared between two or more trees as shown in Fig. 10. The overall bound can be computed by finding the path with the most weight, where the bound of each node is considered as the weight. This can be done using depth first traversal for each tree in the forest.

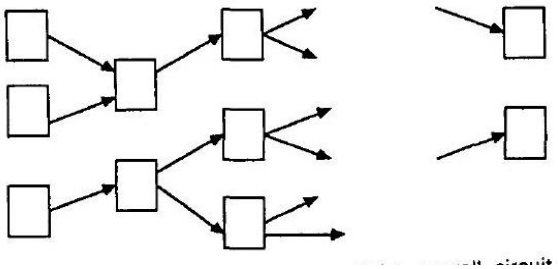


Fig. 10 Equivalent Structure of the overall circuit

To summarize the procedure for finding the upper bound on test sequence length for an arbitrary circuit, we restate the steps involved and give the complexity of each step.

IV. CONCLUSIONS

Test sequence length is an effective measure of testability of a sequential circuit. The lower the bound on the length, the more testable the circuit is. In this paper we have used graph theoretic approach to compute the bound on test sequence length for any sequential circuit. We first found the condensation of the graph, by collapsing the strongly connected components into single nodes. Analyzing each stem region, we can compute the bound on test sequence length for the entire circuit. The time complexity of the procedure is $O(n^2)$ where n is the number of nodes in the circuit graph. The bounds of the individual sub-machines can be used in test generation, scan design and built in self test (BIST) design. Since the test sequence length indicates the testability, it is important to design circuits with lower test length bounds. We have given three design rules that will yield circuits whose test sequence bounds are lower.

REFERENCES

[1] Abromovici, M., Menon, P., and Miller, D. Critical Path Tracing: An Alternative to Fault Simulation. IEEE Design and Test of Computers 1, 1 (Feb. 1984), 83-93.

[2] Agrawal, V. D., Cheng, K. T., and Agrawal, P. CONTEST: A Concurrent Test Generator For Sequential Circuits. In Proc. ACM/IEEE Design Automation Conference (1988), pp. 84-89.

[3] Brglez, f., Bryan, D., and Koźmiński, K. Combinational Profiles of Sequential Circuits. In

Proc. of the Intl. Symposium on Circuits and Systems (1989).

[4] Cheng, K., and Agrawal, V. An Economical Scan Design for Sequential Logic Test Generation. In Proc. of the 19th Fault Tolerant Computing Symposium (FTCS-19) (1989).

[5] Cheng, K., and Agrawal, V. Concurrent Test Generation and Design For Testability. In Proc. of the Intl. Symposium on Circuits and Systems (1989).

[6] Deo, N. Graph Theory with Applications to Engineering and Computer Science. Prentice Hall, 1974.

[7] Friedman, M., Harel, D., Maamari, F., and Rajski, J. A Dominators View of Stem Regions in Combinational Logic and its application to Fault Simulation. Tech. Rep. CR 87-50, CRL Tektronix Laboratories, 1987.

[8] Fujiwara, H., and Shimono, T. On the Acceleration of Test Generation Algorithms. IEEE Trans. Comput. (Dec. 1983), 1137-1144.

[9] Goel, P. An Implicit Enumeration Algorith to Generate Tests for Combinational Circuits. IEEE Trans. Comput. (Mar. 1981), 215-222.

[10] Harary. Graph Theory. John Wiley, 1972.

[11] Hennie, F. Finite-State Models for Logical Machines. John Wiley & Sons, Inc., 1968.

[12] Hudli, R., and Seth, S. Temporal Logic Based Test Generation for Sequential Circuits. In Proc. of IFIP Conference on CAD Systems Using AI Techniques (1989).

[13] Maamari, F., and Rajski, J. Reconvergent Fanout Analysis and Fault Simulation Complexity of Combinational Circuits. Tech. Rep. TR-87-3R, VLSI Design Lab, McGill University, 1987.

[14] Marlett, R. An Effective Test Generation System for Squential Circuits. In Proc. ACM/IEEE Design Automation Conference (1986), pp. 250-256.

[15] Miczo, A. The Sequential ATPG: A Theoretical Limit. In Proc. Intl. Test Conference (1983), pp. 143-147.

[16] Miczo, A. Digital Logic Testing and Simulation. Harper Row, 1986.

[17] Roth, J. Diagnosis of Automata Failures. IBM Journal of Research and Development (July 1968), 278-291.

[18] Schultz, M., Trischler, T., and Starfet, T. SOCRATES: A Highly Efficient Automatic Test Pattern Generation System. In Proc. ACM/IEEE Design Automation Conference (1987), pp. 1016-1025.

[19] Tarjan, R. Depth First Search and Linear Graph Algorithms . SIAM Journal of Computing (June - 1972), 146-160.