University of Nebraska - Lincoln

DigitalCommons@University of Nebraska - Lincoln

CSE Journal Articles

Computer Science and Engineering, Department of

1989

Testability Analysis of Synchronous Sequential Circuits Based On Structural Data

Raghu V. Hudli
University of Nebraska-Lincoln

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

Follow this and additional works at: https://digitalcommons.unl.edu/csearticles

Part of the Computer Sciences Commons

Hudli, Raghu V. and Seth, Sharad C., "Testability Analysis of Synchronous Sequential Circuits Based On Structural Data" (1989). *CSE Journal Articles*. 76.
https://digitalcommons.unl.edu/csearticles/76

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Journal Articles by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

Proceedings. Meeting the Tests of Time., International Test Conference, 1989.
doi: 10.1109/TEST.1989.82320

Testability Analysis of Synchronous Sequential Circuits Based On Structural Data

Raghu V. Hudli and Sharad C. Seth
Department of Computer Science
University of Nebraska
Lincoln NE 68588-0115

Abstract

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.

1 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.

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

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[1] 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.

[1]In this paper, the terms "bound", "upper bound" and "upper bound on the test sequence length" are used interchangeably.

CH2742-5/0000/0364$01.00 © 1989 IEEE

2 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.

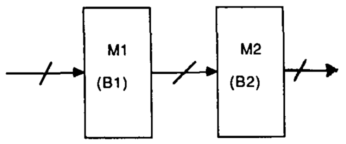

Fig. 1 Series Connection

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.

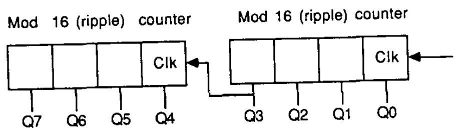

Fig. 2 A ripple counter example

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.

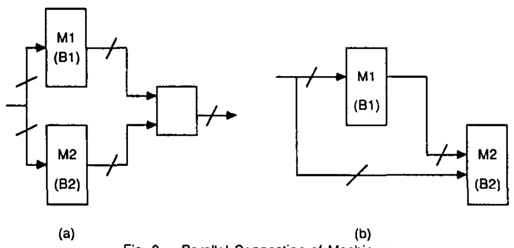

Fig. 3 Parallel Connection of Machines

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 $M2$.

3 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 flip-flops, 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

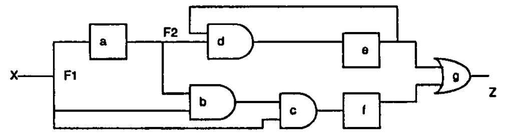

Fig. 4 A Sequential Circuit Example

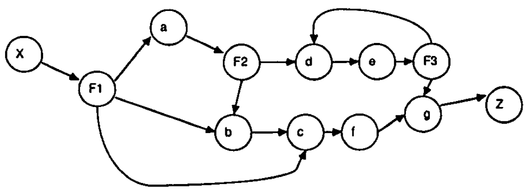

Fig. 5 (a) A directed graph representation of circuit in Fig. 4

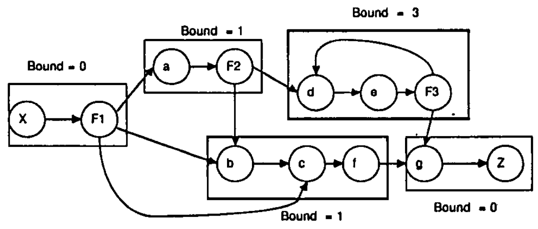

Fig. 5(b) Graph after condensation and collapsing cominational elements

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*. F1 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 $3*3 = 9$.

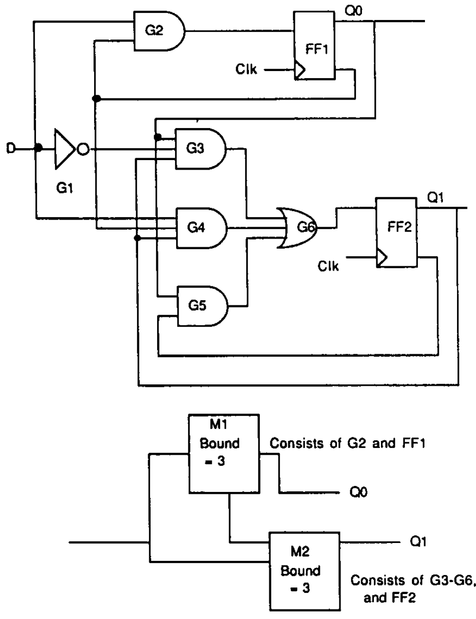

Fig 6. An example sequential circuit and its collapsed schematic corresponding to its condesation graph

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.

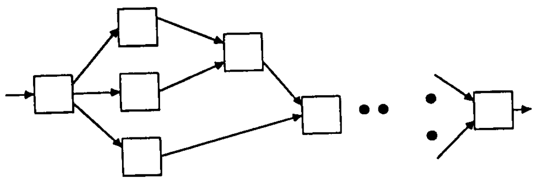

Fig. 7 A series Parallel Structure

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.

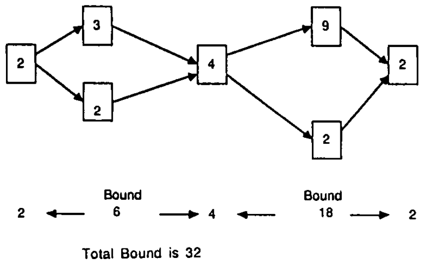

Fig. 8 Computing the bound for a simple series-parallel stem

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, $i+1$, $i+2$, ... 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).

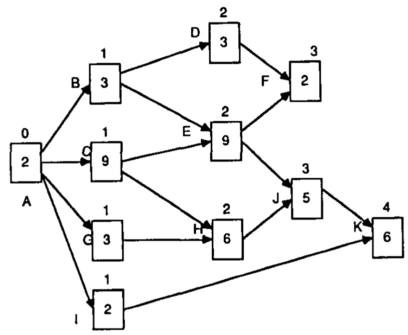

Fig. 9(a) An arbitrary stem region

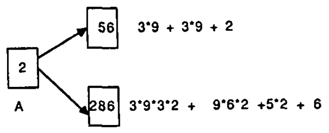

Fig. 9(b) Equivalent Structure

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.

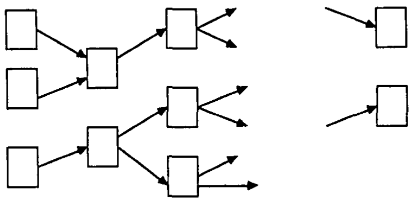

Fig. 10 Equivalent Structure of the overall circuit

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.

1. Find the condensation of the circuit graph. This can be done in $O(n, e)$, where n is the number of nodes and e is the number of edges in the graph.

2. Find the stem regions for the condensed graph. This complexity of this step is $O(n \log n)$.

3. Find the equivalent machines using cutsets. In the worst case, this needs $O(n^2)$ time.

4. From the forest, find the bound for the overall circuit. The time complexity of this step is $O(n^2)$.

The overall complexity of the algorithm is therefore $O(n^2)$.

Theorem: *When the condensation graph is series-parallel, series-parallel reduction and the procedure described for arbitrary circuits give the same bounds.*

Proof: The proof is by induction on the depth of the stem region.

1. Assume the depth is 1, as shown in Fig. 11(a). Without loss of generality, we may assume that there are only two parallel paths. According to the algorithm described, the bound for the stem region is $a + b * c + d$, since there is only one cut-set {B,C}. But also, if we apply series-parallel reduction on the graph, we can collapse the machines B and C into a single machine whose bound is $b * c$. This machine is in series with A and D. Hence the bound for the stem region is $a + b * c + d$, which is also the bound given by procedure.
2. Assume that the procedure gives an identical bound if the stem region is of depth n. Again without any loss of generality, we may assume that the stem region of depth n has the form shown in Fig. 11(b). The equivalent structure is shown in Fig. 11(c). The bound for machine E is

$$\left(\sum_{i=1,3,5,\ldots} b_i * b_{i+1} + \sum_{i=1}^{n-1} c_i \right) a_n + c_n$$

3. We now prove that the procedure gives an identical bound for a stem region of depth $n + 1$. We can get a stem region of depth $n + 1$ from a region of depth n by adding machines A_{n+1} and C_{n+1}, as shown in Fig. 11(d). Using the equivalent structure of Fig. 11(c), we get an alternate structure for Fig. 11(d) as shown in Fig. 11(e). This is similar to the structure shown in Fig. 11(a). Hence the proof. We can also prove by analyzing stem region shown in Fig. 11(d), that the procedure gives an exact bound.

Using series parallel reduction, the bound is

$$b_0 + \left(\left(\sum_{i=1,3,\ldots} b_i * b_{i+1} + \sum_{i=1}^{n} c_i \right) a_n + c_n \right) a_{n+1}$$

$+c_{n+1}$
which can be written as

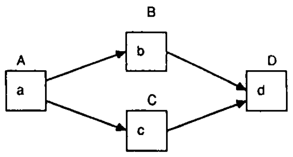

Fig. 11(a) A series-parallel stem of depth 1.

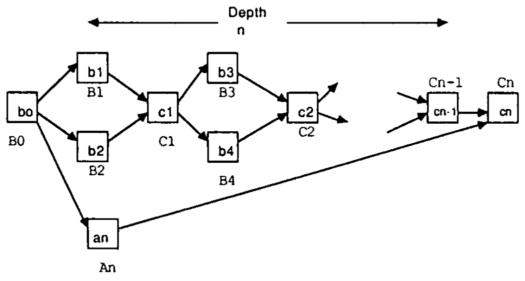

Fig. 11 (b) A series-parallel stem of depth n

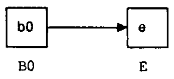

Fig. 11(c) Equivalent structure of fig. 11(b)

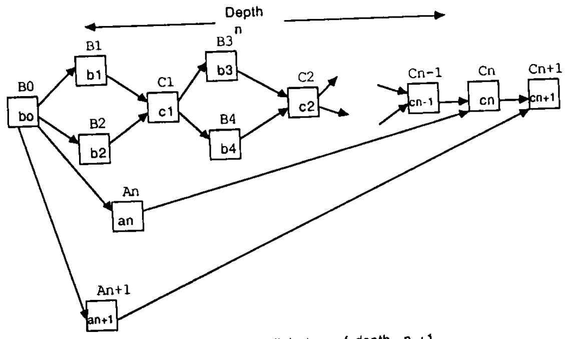

Fig. 11 (d) A series-parallel stem of depth n +1

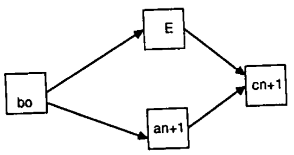

Fig. 11(e) Equivalent structure of fig. 11(d)

$$b_0 + \sum_{i=1,3,5,\ldots} a_n * a_{n+1} * b_i * b_{i+1} +$$

$$\sum_{i=1}^{n} a_n * a_{n+1} * c_i + a_{n+1} * c_n + c_{n+1}.$$

We will now use the procedure described to compute the bound for Fig. 11(d). The cutsets have B's and C's alternately, but always have A_{n+1}. All cutsets up to depth 'n' also have A_n. So the cutsets are

$$\{B_1, B_2, A_n, A_{n+1}\}, \{C_1, A_n, A_{n+1}\},$$

$$\{B_3, B_4, A_n, A_{n+1}\}, \{C_2, A_n, A_{n+1}\},$$
$$\ldots, \{C_n, A_{n+1}\}$$

Taking the product of the bounds of elements in each cutset and summing them, we get

$$b_1*b_2*a_n*a_{n+1}+c_1*a_n*a_{n+1}+b_3*b_4*a_n*a_{n+1}$$

$$+c_2 * a_n * a_{n+1} + \ldots + c_n * a_{n+1}$$

Rearranging, we get,

$$\sum_{i=1,3,5\ldots} a_n*a_{n+1}*b_i*b_{i+1}+\sum_{i=1}^{n} a_n*a_{n+1}*c_i+a_{n+1}*c_n$$

Adding to this the bounds of B_0 and C_{n+1}, we get

$$b_0 + \sum_{i=1,3,5,\ldots} a_n * a_{n+1} * b_i * b_{i+1} +$$

$$\sum_{i=1}^{n} a_n * a_{n+1} * c_i + a_{n+1} * c_n + c_{n+1}$$

Q.E.D

3.1 Results

In Table 2 the results of the bound calculations for the various benchmark circuits [3] are given. Also shown in the table are the number of sub-machines into which the circuit can be partitioned and the maximum and minimum bounds of the sub-machines. In a recent paper [5] cycle analysis of the benchmark circuits was proposed for testability assessment. The result of that analysis is the product of the number of cycles in a circuit with the average cycle length. These values are reproduced in the last column of Table 2. Our bounds calculations correlate very well with the cycle analysis of Cheng and Agrawal. But s526 is an anomalous case where the cycle analysis measure grows at a much greater rate than the bounds. It is worth noting that s420 and s526 require about the same time for *test generation per fault*. But the jump in the cycle measure for the two circuits is significant. The bounds are however not that apart. The large numbers for s641 and s5378 are due to the presence of large connected components. The bounds and the cycle measure are very high; they are not very indicative of the test generation effort. The reason may be because the worst case scenario is considered for calculating the bounds. This simplistic approach may not suffice when the components are very large. A more detailed structural analysis of the individual sub-machines may yield a better bound.

Apart from the testability measure, the bounds have application in test generation. The bounds of individual sub-machines can be used for choosing the elements through which we want to propagate the fault effect or justify a line value. A similar approach based on temporal logic parameters has been used in a new test generation algorithm for sequential circuits [12]. Also the bounds of the individual sub-machines can be used to determine which components (sub-machines) should be chosen to have scan flip-flops. Components that have higher bounds should have scan flip-flops. Cheng and Agrawal have proposed an algorithm that uses heuristics to choose which flip-flops should be made scan flip-flops [4]. Once, the component is chosen, their analysis can be used to choose flip-flops using their method.

Table 2 Structural Profile and Testability Measures of Benchmark Circuits

Circuit	Number of Flip-flops	Number of sub-machines	Max. sub-machine Bound	Min. sub-machine Bound	Overall Bound	Cycle Analysis
s208	8	8	3	3	24	8
s420	16	16	3	3	48	16
s510	6	1	729	729	729	4283
s526	21	15	27	3	93	21972
s641	19	5	14348907	1	14348908	401152
s820	19	1	243	243	243	659
s953	29	24	729	1	730	4794
s1196	18	18	1	1	5	0
s1488	6	1	729	729	729	4827
s5378	179	54	$1.4 * 10^{16}$	1	$1.4 * 10^{16}$	$1.1 * 10^{9}$

4 Design Rules

We can now state the *design rules* for minimizing the upperbound on the test sequence length. The design rules can be obtained based on the procedure described in the previous section. The design rules are intuitive and obvious from the procedure.

The bound of each independent submachine grows exponentially with the number of flip-flops it has. Therefore the first step should be to minimize the bound on the independent submachines. Hence the following design rule.

Design Rule 1:
Minimize the number of latches in strongly connected components in the circuits.

In other words, the components should be small, and the feed back chains should be short. This reduces the bound for each submachine which in turn reduces the bound on test length for the entire circuit.

The bound on the stem can be minimized if circuits are designed using the following two rules. These rules are based on the *girth* of stem regions, which is defined as the size of the largest cutset for the stem region, and the depth of stem regions, which was defined in the previous section.

Design Rule 2:
The girth of the stem region should be as narrow as possible.

If the girth is wide, there will be more machines in each cutset, which will increase the bound for test sequence length.

Design Rule 3: The depth of the stem region should be as small as possible.

A single machine that connects the reconvergent point to the stem will be present in cutsets at all depths. Hence the bound of this machine multiplies the bounds of other machines at different depths. If the depth is minimized, it is clear that the overall bound will be minimized.

5 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* 19^{th} *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.

[20] WILLIAMS, T., AND PARKER, K. Design For Testability — A Survey. *IEEE Transactions on Computers* (Jan. 1982), 12–30.