Scheduling Tests for VLSI Systems Under Power Constraints

Richard M. Chou, Kewal K. Saluja, *Fellow, IEEE,*
and Vishwani D. Agrawal, *Fellow, IEEE*

***Abstract*— This paper considers the problem of testing VLSI integrated circuits in minimum time without exceeding their power ratings during test. We use a resource graph formulation for the test problem. The solution requires finding a power-constrained schedule of tests. Two formulations of this problem are given as follows: 1) scheduling equal length tests with power constraints and 2) scheduling unequal length tests with power constraints. Optimum solutions are obtained for both formulations. Algorithms consist of four basic steps. First, a test compatibility graph is constructed from the resource graph. Second, the test compatibility graph is used to identify a complete set of time compatible tests with power dissipation information associated with each test. Third, from the set of compatible tests, lists of power compatible tests are extracted. And finally, minimum cover table approach is used to find an optimum schedule of power compatible tests.**

***Index Terms*— Built-in self-test (BIST), low-power testing, power-constrained scheduling, test scheduling, VLSI testing.**

I. Introduction

DURING the testing of a complex integrated circuit (IC), digital board, or system, not all tests may be applied at the same time due to resource conflicts. For example, in a given system, suppose subsystem 1 and subsystem 2 share the same inputs, but the test designed for subsystem 1 does not test subsystem 2, and vice versa. Hence, the test for subsystem 2 must be applied at a different time than the test for subsystem 1. The same is true if subsystem 1 and subsystem 2 share outputs, and the test results cannot be observed simultaneously. In general, selected subsets of a complete test set of a system may be applied simultaneously to test different functional blocks provided there is no conflict for resources [1], [5], [9], [12], [13], [18]. The tests that can be applied concurrently are said to be *time compatible* or *compatible* [14]. Each application of time compatible tests is called a *test session*, and the time required for a test session is often referred to as *test length* [5], [13]. In a complete test set, there may be many different subsets of time compatible tests, and these subsets may or may not be disjoint. Therefore, the tests must be scheduled in such a way that suitable time compatible subsets completely test the system while minimizing the total test time. This problem was originally defined by Kime and Saluja [13] and has since been studied by several researchers [1], [5], [9], [12], [18].

The basic approach in test scheduling is to find all time compatible test sessions and determine their ordered sequence to minimize the total test length. The definition of the time compatibility of tests in all previous work [1], [5], [9], [12]–[14], [18] considers only resource conflicts. However, as the device technologies such as multichip modules (MCM's) become available, and larger and denser memory IC's are called for by the high-performance digital systems, the power consumption becomes a critical factor and can no longer be ignored [6], [23], either in normal operation of the system or under testing environment. Absence of resource conflict for a pair of tests does not mean that these two tests can be applied concurrently to parts of a system, because the total power consumption must not exceed the maximum power allowance in order to guarantee proper operating conditions during test.

Consider the modern high-performance memory systems, for example. Memories are organized into blocks of many fixed sizes. Under normal system operation, exactly one block is activated per memory access while other blocks are in the power-down mode to minimize the power consumption. Under testing environment, however, in order to test the memory system in the shortest possible time, it is desirable to concurrently activate as many blocks as possible provided that the power consumption limit of the system is not exceeded. Another example is the testing of MCM's. An attractive approach for testing MCM's is to use built-in self-test (BIST) blocks executing in parallel [11], [21], [23]. Under normal operation, blocks are not simultaneously activated and hence, the inactive blocks do not contribute to power dissipation. However, a concurrent execution of BIST in many blocks will result in high power dissipation which might exceed the maximum power dissipation limit. To ensure the reliability of the system, execution of the self-test blocks must be scheduled in such a way that the maximum power dissipation limit is not exceeded at all times during test. Therefore, the power consumption constraint must be a consideration in the scheduling of tests.

The power dissipation in a system is not only technology dependent, it is also a function of clock frequencies among other factors [6]. For example, in a synchronous CMOS system, power dissipation is a function of the transistor switching activity which occurs on the system clock edges. The faster the clock frequency is, the more the switching activity over a fixed period of time, and hence higher the power dissipation.

Manuscript received June 28, 1994; revised October 25, 1995 and July 31, 1996. This work was supported in part by the National Science Foundation under Grant MIP-9111886 and by a grant from the AT&T Foundation.

R. M. Chou and K. K. Saluja are with the Department of Electrical and Computer Engineering, The University of Wisconsin, Madison, WI 53706 USA.

V. D. Agrawal is with Bell Labs, Lucent Technologies, Murray Hill, NJ 07974 USA.

Publisher Item Identifier S 1063-8210(97)01950-1.

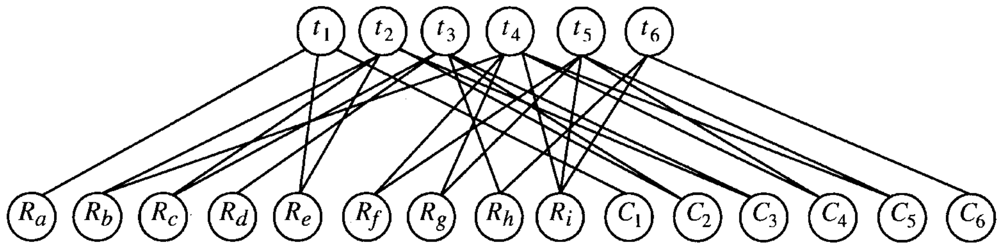

Fig. 1. Resource graph of an example system.

There are several ways in which the power consumption requirements can be satisfied under testing environment. First, clock(s) can be slowed down to reduce the average dynamic power dissipation. However, in order to minimize the total test time, it is desirable to test the system at the highest possible frequency, which renders this method ineffective. Second, the tests can be executed in sequential order such that no two tests are overlapped in time. This method makes the test scheduling problem trivial by excluding the possibilities of exploring parallelism in applying the tests. Again, this method defeats the purpose of minimizing the total test time. These methods are two extremes in the attempt to reduce the power dissipation during the test application at the expense of the total test time. With the clock running at the highest possible frequency, instead of maximizing test parallelism, a better way of scheduling the tests would be to minimize the total test time while satisfying the power constraints. Such considerations have motivated the research discussed in this paper. Note that the method developed in this paper is not only limited to memory testing or BIST environment. It is also applicable to other scenario as well. We formulate the problem with power constraints added to the test scheduling consideration and provide algorithmic solutions.

II. Background and Preliminaries

A digital system can be viewed as a collection of interconnected blocks, each consisting of combinational components and storage registers. A test model for such a system consists of combinational blocks C_i and register blocks R_j, which are jointly called *resources*, for a test t_i. The test problem can be modeled as a bipartite resource graph, hereafter referred to as a resource graph. As shown in Fig. 1, such a graph has been used before [5], [13]. Here, the each test t_i is a sequence of test vectors that can thoroughly test the block C_i. In Fig. 1, the nodes at the top together form a complete set of tests, and the bottom nodes depict the resource set. An edge connecting a top node and a bottom node pairs a test with one of the resources that the test requires.

The resource graph completely describes the covering relationship of the tests and the circuit blocks. In addition, it also contains the information about resource conflicts. For example, in Fig. 1, tests t_3 and t_6 both use resource R_h. This means t_3 and t_6 cannot be executed at the same time. In other words, these two tests are incompatible in time. Only the tests without a resource conflict can be executed concurrently and hence are compatible in time.

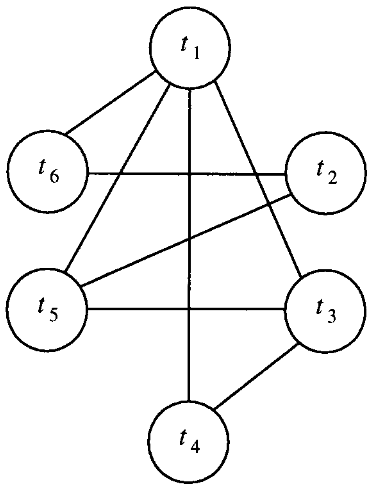

Fig. 2. Test compatibility graph (TCG) of the example system.

From the resource graph, it is possible to obtain the set of all time compatible tests by deriving a *test compatibility graph* (TCG) [5], [13] as shown in Fig. 2. The nodes in the TCG form the complete test set. The edges between the nodes depict the time compatibility of the connecting nodes. For example, in Fig. 2 tests t_1, t_3, and t_5 can be executed concurrently because they are time compatible with each other.

With the TCG based on resource constraints, the test scheduling can be performed by first, finding the *cliques* [10], which are the maximal complete subgraphs of a graph, in the TCG, and then choosing a subset of the cliques according to a cost function as the solution. All nodes in the same clique can be executed concurrently since they are time compatible. Thus, the scheduling problem reduces to that of finding a minimum set of cliques such that all tests are covered. This will give the optimal solution in terms of test time [5], [13]. Alternatively, this problem can also be formulated as a graph coloring problem [5].

When power consumption is also considered in the scheduling of tests, the clique solution is not sufficient. The nodes, or equivalently the tests, in the same clique are time compatible only with respect to the resource constraints. They may not be compatible from the power consumption point of view as executing all tests in the same clique might exceed the maximum power limit imposed by the technology. In such a case, they must not be scheduled in the same test session. For example, consider the clique G1 $= \{t_1, t_3, t_5\}$ in Fig. 2. Assume that t_1 requires two units of power, t_3 requires one unit of power, t_5 requires two units of power, and the

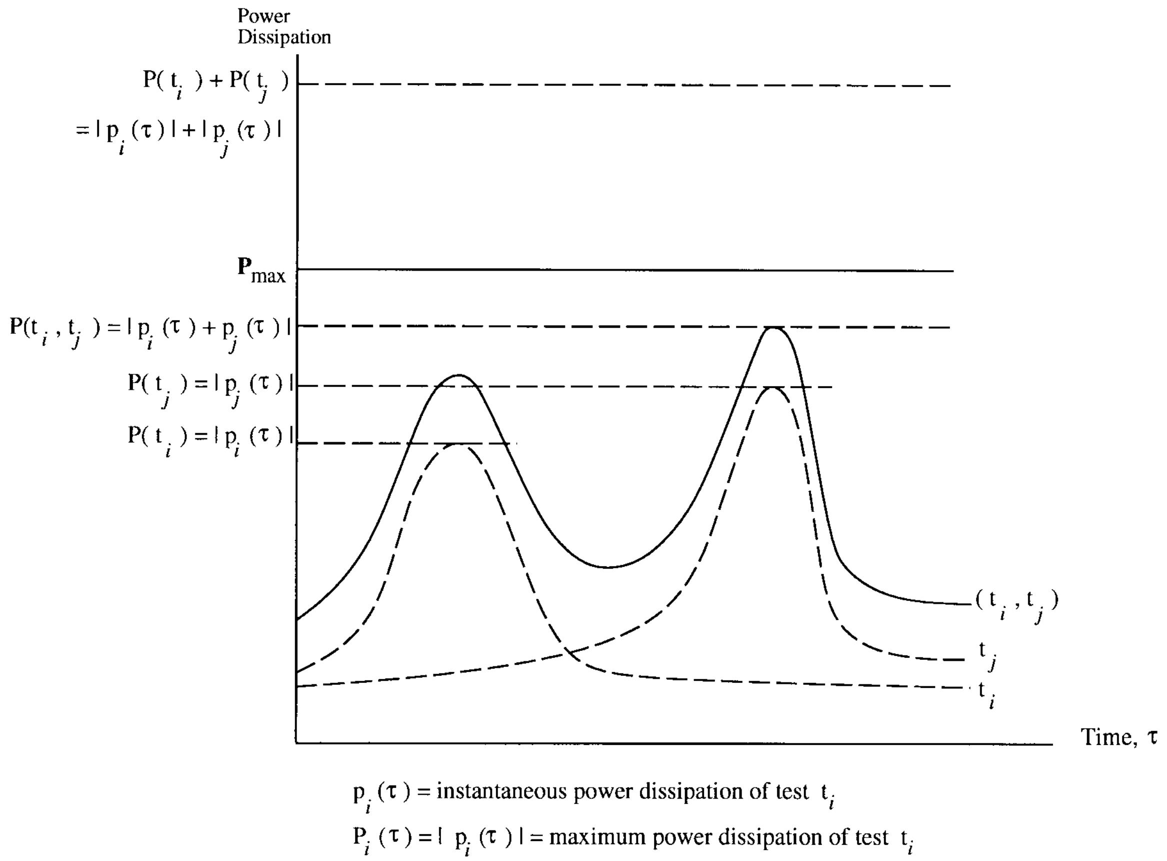

Fig. 3. Power dissipation as a function of time.

maximum power consumption rating of the device is four units. From the resource conflict point of view, all three tests may be executed at the same time. However, executing all three tests concurrently requires five units of power, which exceeds the maximum power limit. Obviously, these three tests must not be scheduled for concurrent execution. This fact and informal statements of results appear in a preliminary version of this paper [4].

III. Problem Formulation

Throughout this paper, the term *test length* will be used as a measure of the real time required to perform the test. Therefore, the terms *test length* and *test time* are used interchangeably. Assume T, which consists of tests t_i, $i \in \{1, 2, \cdots, n\}$, forms a complete set of tests (hereafter, also called "test"), for the device under test. Each test t_i consists of a sequence of test vectors to be applied to the device. A test session s is a subset of T such that all tests in s can be applied concurrently.

Notation:

t_i	test i, $i \in \{1, 2, \cdots, n\}$, $t_i \in T$;
$\ell(t_i)$	test length of t_i, $i \in \{1, 2, \cdots, n\}$;
$P(t_i)$	max power dissipated when test t_i alone is applied to the device;
s_j	test session j, $j \in \{1, 2, \cdots, m\}$, $s_j \subseteq T$;
$\ell(s_j)$	time required for all test vectors in s_j to be completed = Maximum $\{\ell(t_i) \mid t_i \in s_j\}$;
$P_{\max}$	maximum power dissipation allowance of the device.

The power dissipation considered in this work includes both the static and the dynamic power dissipations. Clearly, power dissipation in a circuit or subcircuit is a function of time. Normally, power dissipation at any given time in a circuit depends on the circuit activity. In the case of a CMOS circuit, it is a function of the switching activity for each input applied to the circuit. In testing environment, the power dissipation varies as each test vector is applied to the circuit.

The instantaneous power, $p(\tau)$, is the power dissipation at any time instant τ, i.e., $p(\tau) = v(\tau) \times i(\tau)$ where $v(\tau)$ and $i(\tau)$ are the instantaneous voltage and current in the circuit. Both voltage and current are functions of time. However, in general, the voltage supplied to the circuit does not vary, i.e., $v(\tau) = Vdd$. Therefore, the current waveform will determine the variation of power dissipation. This power dissipation as a function of time may be obtained either from simulation or from a direct measurement carried out on the hardware components. An example of power dissipation waveforms, or equivalently the current waveforms, for a short period of time for tests t_i and t_j are shown in Fig. 3 [3], [8], [22].

To simplify the analysis, the definition of the power dissipation, $P(t_i)$, for a test t_i, which consists of a sequence of test vectors applied over time, is assigned a fixed value for the test t_i. Let $p_i(\tau)$ and $p_j(\tau)$ be the instantaneous power dissipation of two compatible tests t_i and t_j, respectively, and $P(t_i)$ and $P(t_j)$ be the corresponding maximum power dissipation as shown in Fig. 3. Ideally, if $p_i(\tau)+p_j(\tau)$, which is the sum of the instantaneous powers of tests t_i and t_j, does not exceed the maximum power dissipation limit, $P_{\max}$, then t_i and t_j can be scheduled in the same test session. However, in reality the instantaneous power for each test vector is hard to obtain since it depends, e.g., in a CMOS circuit, on the number of zero-to-one and one-to-zero transitions, which in turn could be dependent on the order of execution of test vectors. Consequently, different test schedules will result in different instantaneous power dissipation profiles for the same test. To simplify the analysis, we assign a fixed value $P(t_i)$ to all test vectors in t_i such that at any time instant when the test t_i is in progress the power dissipation is no more than $P(t_i)$.

There are two ways $P(t_i)$ can be assigned. First, $P(t_i)$ could have been defined as the average power dissipation over all test vectors in t_i. This definition might be overly optimistic in the analysis of power dissipation when many test vectors are applied simultaneously since the average value cannot reflect the instantaneous power dissipation of each test vector. Hence, it might lead to an undesirable test schedule which exceeds the power dissipation allowance of the device at some time instants.

Second, $P(t_i)$ can be defined as the maximum power dissipation over all test vectors in t_i. This is the upper bound power dissipation in t_i. This definition is pessimistic since it disallows two tests t_i and t_j whose peak powers occur at different time instants from being scheduled in the same test session as shown in Fig. 3. However, the test schedule obtained with this definition guarantees the maximum power dissipation allowance of the device to be observed at all time instants. In the test environment, the difference between the average and the maximum power dissipation for each test is often small since the objective is to maximize the circuit activity so that the circuit can be thoroughly tested in the shortest possible time. Therefore, it is reasonable to define $P(t_i)$ to be the maximum power dissipation over all test vectors in t_i. Hence, in the subsequent analysis, $P(t_i)$ is assumed to be the maximum power dissipation of test t_i. Note that the statement of the problem and the constraints given below are independent of the method of assigning values to $P(t_i)$ for a test t_i.

With this definition of $P(t_i)$, the power dissipation $P(s_j)$ for a test session s_j can be defined as follows:

$$P(s_j)=\sum_{t_i\in s_j} P(t_i).$$

Therefore, the power constraint in test scheduling can be defined as follows:

$$P(s_j)\le P_{\max} \qquad \forall j. \tag{3.1}$$

This inequality assures that the total power dissipation due to simultaneous application of all test vectors in test session s_j does not exceed the total device power allowance $P_{\max}$ provided that $P(t_i)\le P_{\max}$ for all $i\in\{1, 2, \cdots, n\}$.

1) Test Scheduling Problem: Minimize $(\sum \ell(s_j),\ j\in\{1,2,\cdots, m\})$ under the power constraint (3.1) such that every test t_i, $i\in\{1, 2, \cdots, n\}$ is executed at least once.

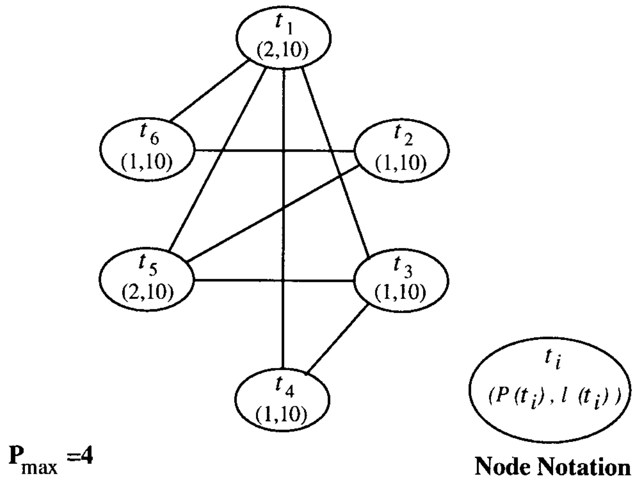

Fig. 4. TCG for power-constrained and equal test length scheduling.

The optimum solution to this test scheduling problem minimizes the total test length such that the device is completely tested under power constraints. Tests can be divided into two categories: 1) tests with equal length and 2) tests with unequal lengths [5], [13]. With test length and the power dissipation as the two variables to be simultaneously considered in minimization, four cases exist. From the simplest to the most general, these cases are: 1) equal power dissipation, equal test length, 2) unequal power dissipation, equal test length, 3) equal power dissipation, unequal test length, and 4) unequal power dissipation, unequal test length. In the equal test length cases, the test length for each test is assumed to be identical, i.e., $\ell(t_i)=L=constant, \forall i$. Since we use the formulation in which the total power consumed during concurrent execution of tests is equal to the sum of powers consumed by individual tests, the solutions to the above four cases can be considered by grouping cases 1 and 2 together, and by grouping cases 3 and 4 together. This grouping forms two general classes: a) the scheduling of equal length tests with power constraints and b) the scheduling of unequal length tests with power constraints. We now discuss solutions to these two problems. We must point out that case a) is a special case of b), in which all test lengths are set to the same constant L. Therefore, the solution to case a is simpler to construct. An understanding of this simpler scheduling approach will help the reader understand the more complex algorithm presented in Section V.

IV. Scheduling Equal Length Tests with Power Constraints

For the example system whose resource graph is shown in Fig. 1, we construct the corresponding TCG. The power-constrained TCG is shown in Fig. 4. In this TCG, which is topologically similar to Fig. 2, an ordered pair $(P(t_i), \ell(t_i))$ is associated with each node t_i. The first element $P(t_i)$ is the power requirement for executing the test t_i as defined in the previous section, and the second element $\ell(t_i)$ is the corresponding test length. $P_{\max}$, the maximum power consumption allowed by technology, is assumed to be four units in this example.

Since all test lengths are equal (unequal test length are considered in the next section), the objective is to find a power-constrained test schedule that covers every test in at least one test session such that the total time required for testing is minimum. The solution is obtained in two steps: 1) identify the solution space, and 2) search the solution space for an optimum solution. To identify the solution space, we require the following definitions.

Definition 1: A *power compatible set* (PCS) is a set of tests that can be executed concurrently, i.e., they are time compatible and still satisfy the power constraints given by relation (3.1). The execution of the tests in the PCS is equivalent to a test session as defined before.

In Fig. 4, $\{t_1, t_5\}$, $\{t_1, t_3\}$, $\{t_2, t_6\}$, $\{t_1\}$, and $\{t_3\}$ are the power compatible sets. Note that the definition of the power compatible set does not guarantee that the sets are disjoint. Overlaps are possible. In addition, one set can also be completely contained in another set. For example, $\{t_1\}$ is completely contained in $\{t_1, t_5\}$. The time compatibility requirement of tests in a PCS implies that the PCS's must be obtained from the cliques of the TCG. Each PCS represents a test session to be considered in test scheduling. The scheduling algorithm just selects the test sessions in an optimal way such that the total execution time is minimum. Therefore, the solution space for the scheduling algorithm is spanned by all test sessions, or equivalently, all PCS's extracted from all cliques of the TCG. However, if all PCS's so generated are considered by the scheduling algorithm, it is equivalent to an exhaustive search. Fortunately, it is possible to deal with a smaller set of the PCS's which still covers the solution space.

Definition 2: A *maximum PCS* is a PCS to which no compatible test can be added without exceeding the maximum power consumption limit. Thus, no maximum PCS can be completely covered by any other PCS.

In the TCG of Fig. 4, the PCS's generated from clique $G1 = \{t_1, t_3, t_5\}$ are $\{t_1, t_3\}$, $\{t_1, t_5\}$, $\{t_3, t_5\}$, $\{t_1\}$, $\{t_3\}$, and $\{t_5\}$, when the maximum allowable power is 4. However, the maximum PCS's are $\{t_1, t_3\}$, $\{t_1, t_5\}$, and $\{t_3, t_5\}$.

The set of maximum PCS's (test sessions) generated from all cliques of the TCG forms the minimum solution space for the test scheduling problem under power constraints. If one of the maximum PCS's is selected for inclusion in the final schedule, then all subsets of that maximum PCS are automatically covered and, hence, the completeness of the solution is justified. Next, in the search step, we find a set with the smallest number of the maximum PCS's that covers all tests.

Note that in this case, all tests have the same length. Consequently, the execution time of any test session is a constant. Furthermore, each maximum PCS in the solution space can be conceptually viewed as a prime implicant of a logic function. Thus, any covering table minimization technique, as is normally used to minimize the number of prime implicants of a logic function [2], [7], [16], will minimize the number of maximum PCS's required, and hence, minimize the number of test sessions. Since each test session executes in constant time, the total test time is also minimized. In forming the maximum PCS's from the cliques of TCG, power constraints are considered. It is not necessary to distinguish the cases of equal and unequal power consumption by each block since the equal power consumption case is just a special case of the unequal consumption case. Hence, both cases can be solved using the same scheduling algorithm. The scheduling algorithm that provides an optimal solution is shown in Fig. 5.

```
Procedure PCELTSA ()

{  Obtain the test compatibility graph (TCG) from the resource graph;

   Obtain all cliques of the TCG;

   For each clique

       Obtain the maximum power compatible set (PCS) of tests;

   Apply covering table procedures to the maximum PCSs;

   Return (the minimum cost cover);      // The optimum test schedule.

}
```

Fig. 5. Power-constrained equal length test scheduling algorithm.

For the example of Fig. 4, the steps of the scheduling algorithm of Fig. 5 are as follows.

1) The TCG is given in Fig. 4.
2) All possible cliques of the TCG are

$$\begin{aligned} G1 &= \{t_1, t_3, t_5\} \\ G2 &= \{t_1, t_3, t_4\} \\ G3 &= \{t_1, t_6\} \\ G4 &= \{t_2, t_5\} \\ G5 &= \{t_2, t_6\}. \end{aligned}$$

3) All possible maximum PCS's ($P_{\max} = 4$) are

$\{t_1, t_3\}$, $\{t_1, t_5\}$, $\{t_3, t_5\}$	obtained from G1
$\{t_1, t_3, t_4\}$	obtained from G2
$\{t_1, t_6\}$	obtained from G3
$\{t_2, t_5\}$	obtained from G4
$\{t_2, t_6\}$	obtained from G5

$\{t_1, t_3\}$ is not a maximum PCS since it is contained in $\{t_1, t_3, t_4\}$, and hence, it is excluded from the set of maximum PCS's.

4) Use of the covering table minimization procedure is shown in Table I. In this table, each "x" indicates the cover of a test by a maximum PCS. The covering table minimization procedure results into the following optimum schedule:

$\{t_1, t_3, t_4\}$	Test session length required $= 10$
$\{t_2, t_5\}$	Test session length required $= 10$
$\{t_2, t_6\}$	Test session length required $= 10$
	Total test length required $= 30$.

It can be easily verified that the solution obtained is indeed an optimal schedule.

TABLE I
COVERING TABLE FOR EQUAL TEST LENGTH CASE

Max. PCSs	t_1	t_2	t_3	t_4	t_5	t_6
$\{t_1, t_5\}$	x				x	
$\{t_3, t_5\}$			x		x	
$\{t_1, t_3, t_4\}$	x		x	x		
$\{t_1, t_6\}$	x					x
$\{t_2, t_5\}$		x			x	
$\{t_2, t_6\}$		x				x

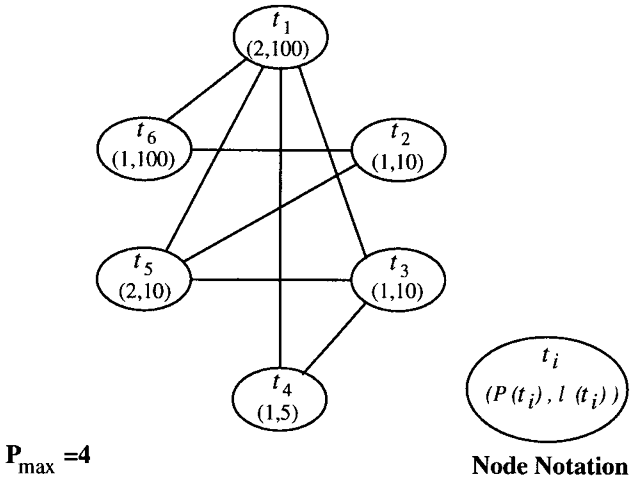

Fig. 6. TCG for power-constrained and unequal test length scheduling.

V. Scheduling Unequal Length Tests with Power Constraints

In general, tests for various blocks of a system can have different lengths. If two blocks with different test lengths are tested concurrently, then testing of one block will complete before the other block is fully tested. We assume that all tests scheduled in one session for concurrent execution are fully executed before tests from the next sessions can be scheduled. Even though some tests in a session may finish earlier, releasing resources, no new tests from the next session can be initiated until all tests of the current session are finished. One way to achieve this may be to let the completed test continue, and thus apply some redundant tests. A practical reason for this restriction is that interruption of a session to initiate new tests can increase the complexity of the test controller which will add hardware overhead. To examine this assumption further, let us consider an example system with multiple BIST blocks. Suppose, results of BIST are held in signature registers, and all pattern generators and signature registers form a single scan chain. The same scan chain is used to initialize pattern generators and to read out signatures. Also, all scan flip-flops are controlled by a common scan mode signal. For testing such a system the following procedure is used: 1) scan in the initialization pattern, 2) apply one or more system clocks, and 3) scan out results. Consider a system with three BIST blocks. The three BIST tests t_1, t_2, and t_3 have lengths $\ell(t_1) = 5$, $\ell(t_2) = 10$, and $\ell(t_3) = 8$. Further, the compatible test sets are $\{t_1, t_2\}$ and $\{t_2, t_3\}$ and tests t_1 and t_3 are incompatible because they share a signature register. If t_1 and t_2 are scheduled in one session, after five units of time, t_1 will complete while t_2 is still running. One may argue that t_3 can be initiated such that t_2 and t_3 will run concurrently for the next five units of time. The difficulty arises because of the common scan mode signal. The signature of t_1 must be scanned out before the register can be used by t_3. However, a scan out will interrupt t_2. In principle, a scan out is possible if the state of the t_2 signature was saved and restored. Clearly, an implementation will require a complex test control strategy. The scenario described in this example is somewhat similar to the testing strategy used in the chip ASIC Z [23].

The algorithm of the previous section cannot provide an optimum schedule for the unequal test length case. Consider the example given in Fig. 6. This is the same system under test as before, except that the test length of each node is different. A schedule obtained for this example using the algorithm for the equal test length case is given below.

Test schedule using the equal test length algorithm:

$\{t_1, t_3, t_4\}$	Test session length required $= 100$
$\{t_2, t_5\}$	Test session length required $= 10$
$\{t_2, t_6\}$	Test session length required $= 100$
	Total test length required $= 210$.

However, this is not an optimal solution because a better solution exists and is as follows:

$\{t_3, t_4\}$	Test session length required $= 10$
$\{t_2, t_5\}$	Test session length required $= 10$
$\{t_1, t_6\}$	Test session length required $= 100$
	Total test length required $= 120$.

The algorithm fails because one of the solutions, namely $\{t_3, t_4\}$, lies outside the solution space. Therefore, for the unequal length test case, the solution space must be enlarged. This can be done by expanding the maximum PCS's obtained in the equal length test case. However, if all possible combinations of tests are considered, the size of the solution space will become unnecessarily large. In fact a smaller solution space exists and can be obtained by considering only a subset of test combinations. Let us define the following terms.

Definition 3: A *power compatible list (PCL)* H is a PCS such that the elements in H are arranged according to the descending order of lengths of the tests in the PCS.

For example, the PCL for a PCS $= \{t_4, t_3, t_1\}$ is $H = (t_1, t_3, t_4)$, since $\ell(t_1) \geq \ell(t_3) \geq \ell(t_4)$.

Definition 4: A *derived PCL (DPCL)* is generated recursively from a PCL or another derived PCL as follows. If $H = (t_{i1}, t_{i2}, \cdots, t_{ij}, t_{ik}, \cdots, t_{im})$ is a PCL or a DPCL such that $\ell(t_{i1}) = \ell(t_{i2}) = \cdots = \ell(t_{ij}) > \ell(t_{ik}) \geq \cdots \geq \ell(t_{im})$, then the derived PCL of H, denoted by H', is $(t_{ik}, \cdots, t_{im})$ provided that $H' \neq \emptyset$. If $H' = \emptyset$, then it is not a DPCL.

In other words, H' is an ordered subset of the PCL H such that test length of the first element in H' is strictly less than the test length of the first element in H. Hence, H' contains all tests of H whose test lengths are strictly less than the first element in H.

This process of deriving the DPCL's is repeated on each newly derived DPCL until no further derivation is possible. The collection of all the DPCL's so derived and the original PCL itself constitute the set of DPCL's, or equivalently, the test sessions, to be considered in the solution space for the weighted covering table approach. For the example of Fig. 6, the DPCL's of a PCL $H_1 = (t_1, t_3, t_4)$ are $H_1' = (t_3, t_4)$ and $H_1'' = (t_4)$. Similarly, for PCL $H_2 = (t_1, t_3, t_5)$, DPCL is $H_2' = (t_3, t_5)$. No more DPCL can be generated since $\ell(t_3) = \ell(t_5)$. Therefore, the set of DPCL's for H_1 consists of H_1, H_1', and H_1''. Similarly, the set of DPCL's for H_2 consists of H_2 and H_2'.

The solution space is primarily comprised of all sets of DPCL's derivable from all possible PCL's. Normally the PCL's are not disjoint. Therefore, if the DPCL's are generated from all these PCL's, repeated DPCL's are possible. It is preferable to remove multiple occurrences of the same DPCL from the solution space to reduce the size of the search space.

Definition 5: A *reduced DPCL (RDPCL) set* is the set of all DPCL's derivable from all possible PCL's such that each DPCL appears exactly once. In addition, if DPCL $h_1 = (t_1, t_2, \cdots, t_m)$, and DPCL $h_2 = (t_{i1}, t_{i2}, \cdots, t_{ik})$ such that $t_{ij} \in h_1, \quad j = 1, 2, \cdots, k$ and $\ell(h_1) = \ell(h_2)$, then h_2 is removed from the RDPCL set.

An example that clarifies the above definition is as follows. Let $h_1 = (t_a, t_b, t_c, t_d)$, $h_2 = (t_a, t_b, t_c, t_d)$, and $h_3 = (t_a, t_b, t_c)$. In addition, let $\ell(t_a) = \ell(t_b) = 10$, $\ell(t_c) = 5$, and $\ell(t_d) = 2$. The DPCL h_2 is removed from RDPCL set because $h_2 = h_1$. Further, h_3 is removed from RDPCL set because every element of h_3 is in h_1 and $\ell(h_3) = \ell(h_1)$.

The covering table technique can be used to obtain the optimum solution if DPCL's are treated like prime implicants in logic minimization. Further, we must consider the test length along with each DPCL. Hence, algorithms that obtain a minimum cost cover of a weighted covering table must be used [2], [7] [16]. The cost in this case is the test length required for each DPCL. The optimum scheduling algorithm (PCULTSA) for the unequal test length case is shown in Fig. 7. It can be proved that PCULTSA will indeed find an optimum test schedule. The proof is given in Appendix.

For the example of Fig. 6, the steps of the scheduling algorithm of Fig. 7 are as follows.

1) The TCG of Fig. 6 is obtained.
2) All possible cliques from the TCG are found as

$$G1 = \{t_1, t_3, t_5\}$$
$$G2 = \{t_1, t_3, t_4\}$$
$$G3 = \{t_1, t_6\}$$
$$G4 = \{t_2, t_5\}$$
$$G5 = \{t_2, t_6\}.$$

3) All possible PCL's ($P_{\max} = 4$) are

$(t_1, t_3), (t_1, t_5), (t_3, t_5)$	obtained from G1
(t_1, t_3, t_4)	obtained from G2
(t_1, t_6)	obtained from G3
(t_2, t_5)	obtained from G4
(t_6, t_2)	obtained from G5.

4) Reduced DPCL's (RDPCL's) are

$$(t_1, t_5), (t_5), (t_3, t_5)$$
$$(t_1, t_3, t_4), (t_3, t_4), (t_4)$$
$$(t_1, t_6)$$
$$(t_2, t_5)$$
$$(t_6, t_2).$$

5) The weighted covering table is shown in Table II. The cost is the test length associated with each DPCL in the reduced set. After applying the weighted covering table minimization procedure, the optimum schedule is obtained as

(t_3, t_4)	Test session length required $= 10$
(t_2, t_5)	Test session length required $= 10$
(t_1, t_6)	Test session length required $= 100$
	Total test length required $= 120$.

```
Procedure PCULTSA ()
{ Obtain the test compatibility graph (TCG) from the resource graph;
  Obtain all cliques of the TCG;
  For each clique
      Derive the power compatible set (PCS) of tests;
  For each PCS {
      Obtain the power compatible list (PCL) by arranging the tests
      according to the descending order of the test lengths;
      Generate recursively all the derived PCLs (DPCLs);
  }       Obtain the reduced DPCL set (RDPCL);
  Apply weighted covering table procedures to the RDPCL set;
  Return (the minimum cost cover);    // The optimum test schedule.
}
```

Fig. 7. Power-constrained unequal length test scheduling algorithm.

It can be verified that this is indeed an optimum schedule for the given example. In addition, it is also noted that the equal test length case is just a special case of the unequal test length case. In the example of Fig. 6, if all tests have the same test length and the unequal test length scheduling algorithm is applied to it, then the DPCL's will not be generated since they all have the same test length as the PCL's. Consequently, the rows (t_3, t_4), (t_2), (t_3), (t_4), and (t_5) will be deleted from Table II. The resulting covering table will be the same as in Table I, and hence, the optimum schedule obtained will be the same as in the equal test length case. However, the equal test length scheduling algorithm shown in Fig. 5 is still of interest

TABLE II
COVERING TABLE FOR UNEQUAL TEST LENGTH CASE

RDPCL	t_1	t_2	t_3	t_4	t_5	t_6	Cost
(t_1, t_3, t_4)	x		x	x			100
(t_1, t_5)	x				x		100
(t_1, t_6)	x					x	100
(t_6, t_2)		x				x	100
(t_3, t_5)			x		x		10
(t_2, t_5)		x			x		10
(t_3, t_4)			x	x			10
(t_5)					x		10
(t_4)				x			5

since it is simpler, and the solution space is smaller. Therefore, if the problem can be modeled as in the equal test length case, simpler scheduling algorithm presented in Section IV should be used to obtain the solution instead of the more general and complex solution presented in this section.

VI. A Case Study

VLSI devices running in BIST mode consume more power than when running in normal mode. Zorian [23] has studied the test scheduling problem for BIST devices. We will use Zorian's example of ASIC Z as a case study to illustrate the result of the scheduling algorithm presented in this paper. Fig. 8 is the block diagram of ASIC Z, which is partitioned into four RAM's, two ROM's, one Register File (RF), and three random logic blocks (RL1, RL2, and RL3). The shaded area is the network which controls BIST activities. The block marked *B-S TAP* is the boundary scan test access port. Since the power dissipations for both the BIST control network and the B-S TAP blocks are not given, they will not be considered in this example. Table III depicts the size, the active power dissipation in mW, and the test length for each block. Test lengths for blocks were not given in [23] but for this illustration, as shown in Table III, are assumed to be proportional to the sizes of blocks. They were obtained by assuming one unit test length per 100 gates. Notice that the information for RL3 is missing in the table since it was not available in the paper. The maximum power dissipation limit $P_{\max}$ is 900 mW, which is the same as used in [23].

\hskip-7pcA few additional assumptions are made. First, all blocks in Table III can be executed concurrently. Therefore, the clique in this case contains all blocks. Second, due to the fact that the idle blocks contribute very little to the total power dissipation, they are excluded to simplify the calculation without affecting the validity of the result. However, the power of the idle blocks can be incorporated into the calculation in our algorithm if they were significant for some application. After generating the PCS's and the RDSPCL's, the weighted covering table minimization procedure is applied to obtain the result shown in (6.1). The test schedule given in [23] is shown in (6.2). Both (6.1) and (6.2) are shown at the bottom of the next page.

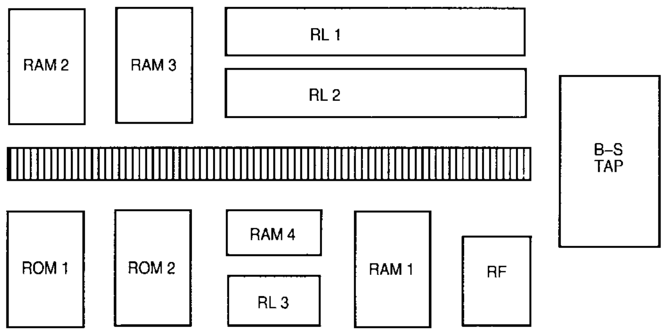

Fig. 8. ASIC Z partitioning and BIST control.

TABLE III
ASIC Z POWER DISSIPATION AND ASSUMED TEST LENGTH

Block	Size	$P_{B/active}$ (mW)	Test Length
RL1	13400 gates	295	134
RL2	16000 gates	352	160
RF	64×17 bits	95	10
RAM1	768×9 bits	282	69
RAM2	768×8 bits	241	61
RAM3	768×5 bits	213	38
RAM4	768×3 bits	96	23
ROM1	1024×10 bits	279	102
ROM2	1024×10 bits	279	102

The schedule obtained by our algorithm is indeed better, and it can be shown that this schedule is an optimum schedule. In Zorian's paper, however, the test scheduling also considered the minimization of hardware overhead in testing. In that approach an *ad hoc* scheduling was adopted to simultaneously test as many nearby blocks as possible. The schedule obtained from our algorithm also provides this minimum hardware overhead result since all blocks scheduled in the same test session are also close to each other, and the types of blocks in each session of our solution have similar features as those produced by the solution procedures given in [23]. Note, however, that our current approach does not consider the block adjacency constraint as was done in [23]. Therefore, in the current approach, alternative schedules exist which might schedule blocks that are physically far apart to execute concurrently. However, with an additional dimension of block adjacency constraint integrated in the cost function used for the weighted covering table minimization, optimum solutions satisfying both time and power constraints can be obtained.

VII. Discussion and Conclusion

This paper defines and formulates the power-constrained test scheduling problem. The objective is to minimize the total test length or, equivalently, the total test time, subject to the power constraints which may be imposed by a particular device technology. Four possible cases combining equal/unequal power consumption and equal/unequal test lengths are identified. These four cases are generalized into two categories: a) equal length tests under power constraints and b) unequal length tests under power constraints. The algorithmic solution is based on reducing the search space, and hence, the complexity of finding the minimum test length is reduced.

In this work, each test session is assumed to have a fixed power dissipation which is the worst case dissipation of that session. In reality, power dissipation varies as test vectors are applied within a test session. Therefore, the maximum constant power dissipation assumption is pessimistic. A better test schedule could be found if power consumption was considered a function of time. This requires a more detailed analysis of the system either by simulation or by measurement of the power dissipation as tests are applied. If two tests are compatible, and have their peak powers occurring at different times, then they can be scheduled in the same test session even when the sum of their peak powers is greater than the maximum power limit. Consequently, better test schedules could be possible. In addition, a test session could allow multiple short compatible tests to be scheduled in sequence to have their power consumptions strategically skewed. Thus, two tests running in series can be scheduled in parallel with other longer tests to shorten the overall test time. Such cases are not considered in this paper.

This work considers the solution to the test scheduling problem based on a theoretical analysis. From implementation point of view, several parts of the test scheduling algorithm, namely, the identification of all cliques in a graph and the covering table minimization technique, belong to the class of NP-complete problems. Therefore, heuristic-driven algorithms must be employed to obtain practical and near-optimal solutions [10], [15], [19], [20]. Methods such as integer linear programming (ILP) and heuristic approaches to the clique identification and the covering table minimization technique are well studied and can be found in the literature. We believe that a suitable practical algorithm for the test scheduling problem can be implemented by considering the published heuristic-driven algorithms with variations.

The concept presented in this paper not only applies to IC level, but can also be extended hierarchically to board and system levels. The constraints in these cases would be different, of course. However, the general concept remains the same. As long as these components are to be tested under certain set of constraints imposed by the technology, the approach presented in this paper may be applied to solve the problem.

Appendix

Theorem: The test schedule obtained by PCULTSA is an optimum test schedule.

Proof: It is known that the weighted covering table technique can provide an optimum cover solution [16]. For example, the last step of Quine–McCluskey logic minimization algorithm uses the weighted covering table technique to find the optimum set of prime implicants of a logic function if the set of prime implicants (solution space) to be searched is complete. Therefore, it is only necessary to prove that PCULTSA can provide an adequate solution space for the covering table procedure. In other words, the optimum solution must be contained in the solution space provided by PCULTSA.

Let t_i, t_j, t_k be three separate tests, $i \neq j \neq k$. Let s_m and s_n be PCS's corresponding to the PCL's or DPCL's, $m \neq n$. And let $s_m = \{t_i, t_j\}$, $s_n = \{t_i, t_k\}$. There are two cases to consider.

Case 1: $\ell(t_i) = \ell(t_j)$. In this case, $\ell(s_m) = \ell(t_i) = \ell(t_j)$, and $\ell(s_n) = \max\{\ell(t_i), \ell(t_k)\}$. If s_m is split into different test sessions such that t_i is executed in s_n, the total test length L required to execute the tests t_i, t_j, and t_k is

$$\begin{aligned} L &= \ell(s_n) + \ell(t_j) \\ &= \max\{\ell(t_i), \ell(t_k)\} + \ell(t_j) \\ &\geq \ell(t_i) + \ell(t_j) \\ &= \ell(t_i) + \ell(s_m) \\ &= \ell(s_m) + \ell(t_j) \qquad (\text{since } \ell(t_i) = \ell(t_j) = \ell(s_m)). \end{aligned}$$

If $\ell(t_i) = \ell(t_j)$, running t_i and t_j in different test sessions does not provide a better schedule than running them together in the

$$\begin{array}{ll} \{\text{RAM1, RAM3, RAM4, RF}\} & \text{Test session length required} = 69 \\ \{\text{RL1, RL2}\} & \text{Test session length required} = 160 \\ \{\text{ROM1, ROM2, RAM2}\} & \text{Test session length required} = 102 \\ & \quad\text{Total test length required} = 331. \end{array} \tag{6.1}$$

$$\begin{array}{ll} \{\text{RAM1, RAM4, RF}\} & \text{Test session length required} = 69 \\ \{\text{RL1, RL2}\} & \text{Test session length required} = 160 \\ \{\text{RAM2, RAM3}\} & \text{Test session length required} = 61 \\ \{\text{ROM1, ROM2}\} & \text{Test session length required} = 102 \\ & \quad\text{Total test length required} = 392. \end{array} \tag{6.2}$$

same test session. Therefore, the solution space for search in the covering table procedure need not include these tests or test sessions for obtaining an optimum schedule.

Similar reason holds for the following three cases:

a) $s_{m1} = \{t_i, s_j\}, \quad t_i \notin s_j, \quad \ell(t_i) = \ell(s_j)$

b) $s_{m2} = \{s_i, t_j\}, \quad t_j \notin s_i, \quad \ell(s_i) = \ell(t_j)$

c) $s_{m3} = \{s_i, s_j\}, \quad s_i \cap s_j = \phi, \quad \ell(s_i) = \ell(s_j)$.

In these three cases, test t_i or test session s_i may be kept in the same session as test t_j or test session s_j.

Case 2: $\ell(t_i) > \ell(t_j)$. In this case, $\ell(t_j) < \ell(t_i, t_j) = \ell(s_m)$. If s_m is split into different test sessions such that t_i is executed in s_n, the total test length L required to execute tests t_i, t_j, and t_k is

$$\begin{aligned} L &= \ell(s_n) + \ell(t_j) \\ &= \max\{\ell(t_i), \ell(t_k)\} + \ell(t_j). \end{aligned}$$

If $\ell(t_i) \leq \ell(t_k)$, then

$$L = \ell(t_k) + \ell(t_j) < \ell(t_k) + \ell(s_m) \quad (\text{since } \ell(t_j) < \ell(s_m)).$$

If $\ell(t_i) > \ell(t_k)$, then

$$L = \ell(t_i) + \ell(t_j) < \ell(t_k) + \ell(s_m) \quad (\text{since } \ell(t_k) < \ell(t_i) \text{ and } \ell(t_j) < \ell(s_m)).$$

Hence, if $\ell(t_i) = \ell(t_j)$, splitting t_i and t_j into two different test sessions can potentially provide a better solution than executing both t_i and t_j together in the same test session s_m.

Similar reason holds for the following three cases:

a) $s_{m1} = \{t_i, s_j\}, \quad t_i \notin s_j, \quad \ell(t_i) > \ell(s_j)$

b) $s_{m2} = \{s_i, t_j\}, \quad t_j \notin s_i, \quad \ell(s_i) > \ell(t_j)$

c) $s_{m3} = \{s_i, s_j\}, \quad s_i \cap s_j = \phi, \quad \ell(s_i) > \ell(s_j)$.

In these three cases, considering test t_i or test session s_i in a different test session than test t_j or test session s_j can potentially provide a better solution. Hence, these sessions must be included in the solution space for search in the covering table procedure to provide the optimum schedule.

To provide the adequate solution space for the covering table procedure, all PCS's are considered in PCULTSA. In addition, all necessary DPCL's are generated from each PCL (ordered PCS) using the procedures conforming to the above two cases. Note that it is not essential to order PCS's into PCL's as provided in PCULTSA. Ordering PCS's just provides an easier way of identifying and generating DPCL's (test sessions) as described in the above two cases. Therefore, the procedures provided in PCULTSA will generate all necessary test sessions to be considered in the selections of optimum schedule by the covering table technique. Hence, PCULTSA provides an optimum solution. □

ACKNOWLEDGMENT

The authors would like to thank Prof. C. R. Kime of the University of Wisconsin-Madison for many helpful suggestions.

REFERENCES

[1] M. Abadir and M. A. Breuer, "Constructing optimal test schedules for VLSI circuits having built-in test hardware," in *Proc. Int. Symp. Fault-Tolerant Comp.,* June 1985, pp. 165–170.

[2] R. Batni, J. Russell, and C. R. Kime, "An efficient algorithm for finding an irredundant set cover," *J. ACM,* vol. 21, no. 3, pp. 351–355, July 1974.

[3] L. Benini, M. Favalli, P. Olivo, and B. Riccò, "A novel approach to cost-effective estimate of power dissipationin CMOS ICs," in *Proc. European Conf. Design Automation,* Feb. 1993, pp. 354–360.

[4] R. M. Chou, K. K. Saluja, and V. D. Agrawal, "Power constraint scheduling of tests," in *Proc. 7th Int. Conf. VLSI Design,* Jan. 1994, pp. 271–274.

[5] C. Craig, C. R. Kime, and K. K. Saluja, "Test scheduling and control for VLSI built-in self-test," *IEEE Trans. Comput.,* vol. 37, pp. 1099–1109, Sept. 1988.

[6] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in CMOS combinational circuits using Boolean function manipulation," *IEEE Trans. Computer-Aided Design,* vol. 11, pp. 373–383, Mar. 1992.

[7] D. L. Dietmeyer, *Logic Design of Digital Systems,* 2nd ed. Boston, MA: Allyn and Bacon, 1979.

[8] F. Dresig, P. Lanchès, O. Rettig, and U. Baitinger, "Simulation and reduction of cmos power dissipation at logic level," in *Proc. European Conf. Design Automation,* Feb. 1993, pp. 341–346.

[9] M. Gonzales, "Deterministic processor scheduling," *ACM Comput. Surv.,* vol. 9, no. 3, pp. 173–204, Sept. 1977.

[10] F. Harary, *Graph Theory.* Reading, MA: Addison-Wesley, 1969.

[11] N. Jarwala, C. Yau, P. Stiling, and E. Tammaru, "A framework for boundary-scan based system test and diagnosis," in *Proc. Int. Test Conf.,* Sept. 1992, pp. 993–998.

[12] W. B. Jone, C. Papachristou, and M. Pereira, "A scheme for overlaying concurrent testing of VLSI circuits," in *Proc. Design Automation Conf.,* June 1989, pp. 531–536.

[13] C. R. Kime and K. K. Saluja, "Test scheduling in testable VLSI circuits," in *Proc. Int. Symp. Fault-Tolerant Comput.,* June 1982, pp. 406–412.

[14] Z. Kohavi, *Switching and Finite Automata Theory,* 2nd ed. New York: McGraw-Hill, 1978.

[15] A. Krasniewski and A. Albicki, "Automatic design of exhaustively self-testing chips with BILBO modules," in *Proc. Int. Test Conf.,* Nov. 1985, pp. 362–371.

[16] E. J. McCluskey, *Introduction to the Theory of Switching Circuits.* New York: McGraw-Hill, 1965.

[17] J. P. Roth, "Computer solutions to minimum cover problems," *Operations Research,* vol. 17, no. 3, pp. 455–465, May/June 1969.

[18] K. K. Saluja, C. R. Kime, and C. Craig, "Design of control for scheduling tests in testable VLSI circuits," Dep. Elec. Comput. Eng., Univ. Newcastle, New South Wales, Australia, Tech. Rep. EE8540, Sept. 1985.

[19] A. Shneider, "Classification analysis of heuristic algorithms for graph coloring," *Cybernet.,* vol. 20, pp. 484–492, July/Aug. 1984.

[20] M. N. Swamy and K. Thulasiraman, *Graphs, Networks, and Algorithms.* New York: Wiley, 1981.

[21] R. E. Tulloss and C. Yau, "BIST and boundary-scan for board level test: Test program pseudocode," in *Proc. European Test Conf.,* Apr. 1989, pp. 106–111.

[22] P. Vanoostende, P. Six, and H. De Man, "Priti: Estimation of maximal currents and current derivatives in complex cmos circuits using activity waveforms," in *Proc. European Conf. Design Automation,* Feb. 1993, pp. 347–353.

[23] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," in *Proc. 11th IEEE VLSI Test Symp.,* Apr. 1993, pp. 4–9.

Richard M. Chou received the B.S. degree in electrical engineering and computer science from the University of California, Berkeley, and the M.S. degree in electrical and computer engineering from the University of Wisconsin, Madison. Currently, he is working toward the Ph.D. degree in electrical and computer engineering at the University of Wisconsin-Madison.

From 1985 to 1988, he was a Member of the Technical Staff with Hewlett-Packard, Fort Collins, CO. His research interests include design and synthesis of testable VLSI circuits and systems, test generation, and computer architecture.

Kewal K. Saluja (S'70–M'73–SM'89–F'95) received the B.E. degree from University of Roorkee, India, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Iowa, Iowa City.

He is currently with the Department of Electrical and Computer Engineering at the University of Wisconsin, Madison, as a Professor, where he teaches logic design, computer architecture, microprocessor-based systems, VLSI design, and testing. Prior to this, he was at the University of Newcastle, Australia. He has held visiting and consulting positions at eight institutions such as the University of Southern California, University of Iowa, Hiroshima University, and University of Roorkee. He has also served as a Consultant to the United Nations Development Program. His research interests include test generation, design for testability, fault-tolerant computing, VLSI design, and computer architecture.

Dr. Saluja is the Associate Editor for the Letters Section of the *Journal of Electronic Testing: Theory and Applications (JETTA).* He is a member of Eta Kappa Nu and Tau Beta Pi.

Vishwani D. Agrawal (S'68–M'70–SM'80–F'86) received the B.Sc. degree from the University of Allahabad, Allahabad, India, in 1960, the B.E. degree from the University of Roorkee, Roorkee, India, in 1964, the M.E. degree from the Indian Institute of Science, Bangalore, India, in 1966, and the Ph.D. degree in electrical engineering from the University of Illinois, Urbana-Champaign, in 1971.

He is a Distinguished Member of Technical Staff in the Computing Sciences Research Center of Bell Laboratories (R&D arm of Lucent Technologies), Murray Hill, NJ, and a Visiting Professor of Electrical and Computer Engineering at Rutgers University, New Brunswick, NJ. Over the past two decades, he has developed numerous techniques and tools for test generation, fault simulation, testability analysis, design for testability, and timing analysis, now used in the design of VLSI circuits. His current interests are testing, synthesis for testability, and parallel algorithms. He has published more than 200 papers and coauthored four books. He holds ten U.S. patents on design for testability.

Dr. Agrawal is the founding Editor-in-Chief of the bimonthly *Journal of Electronic Testing: Theory and Applications*. From 1985 to 1987, he was the Editor-in-Chief for the IEEE DESIGN & TEST OF COMPUTERS magazine. He is the Consulting Editor for the *Frontiers of Electronic Testing* book series of Kluwer-Academic Publishers, Boston, MA. In 1985, he cofounded the International Conference on VLSI Design in India. Currently, he chairs the steering committee of that conference. From 1989 to 1990, he served on the Board of Governors of the IEEE Computer Society. In 1994, he chaired the Fellow Committee of the IEEE Computer Society. Dr. Agrawal is a Fellow of the IETE (India) and a Member of the ACM and the VLSI Society of India. He has received five Best Paper Awards and one Honorable Mention Award. He is the recipient of an Outstanding Contribution Award (1988), a Meritorious Service Award (1989), a Golden Core Member Award (1996) of the IEEE Computer Society, and a Distinguished Alumnus Award (1993) of the University of Illinois at Urbana-Champaign.