Design for Test Using Partial Parallel Scan

SUNGGU LEE, STUDENT MEMBER, IEEE, AND KANG G. SHIN, SENIOR MEMBER, IEEE

***Abstract*—Traditional scan design techniques such as level-sensitive scan design, scan path, and random-access scan suffer from the drawback that the extra test application effort (which includes both time and memory) required is directly proportional to the number of latches and can become quite significant. We present a new scan design technique termed *partial parallel scan* which reduces test application effort by 1 to 2 orders of magnitude. Theoretical and practical aspects of the new design method are discussed. The practical use of the partial parallel scan technique has been demonstrated with an LSI circuit and a VLSI circuit designed using silicon compiler tools.**

***Index Terms*—Testable design, testing and maintenance, CAD for fault-tolerance, *NP*-complete, scan design, partial scan.**

I. Introduction

THERE are several variables which need to be considered in the design of a testing scheme. These include test generation complexity, test application effort (which includes both time and memory requirements and its directly dependent on test sequence length), hardware overhead (in the case of design for test (DFT) schemes), level of fault coverage, and generality of usage. All testing schemes tradeoff one or more of these variables to gain in one or more of the other variables. In this paper, we investigate the issue of reducing test application effort by reducing test sequence length.

In designing a DFT technique, it is important to keep in mind the fact that both test generation and test application times must be kept within reasonable limits. In previous papers on DFT using a scan design [1]–[6], test application effort has been essentially ignored. To justify this type of approach, Williams and Angell [1] point to an example of a 1000-gate network with the number of stuck-at and adjacent bridging faults on the order of 16 000. If the circuit has a hundred latches connected in a single chain in the scan mode, then even if a separate test is needed for each fault, the number of clock periods required would be on the order of 1.6 million. Such a set of tests could be applied in the course of a few seconds.

However, with the ever-increasing complexity of integrated circuits being designed today, this same argument might not be applicable to a circuit of, say, over 10 000 logic gates and several hundred latches. Furthermore, with the increase in test sequence length, more memory is required and more effort is required to verify the test responses. Williams and Angell [1] further state that if the test sequence length is of major importance, then the use of multiple shift registers in scan mode, where the scan-in inputs and scan-out outputs of these registers are multiplexed with the normal circuit inputs and outputs, could reduce the test sequence length by a factor of at least 10.

In this paper, we describe an alternative scan method, termed *partial parallel scan*, which provides an even greater speedup and allows the test engineer to dynamically switch between different levels of partial scan. Partial parallel scan is defined by first describing a method of scanning test vectors in parallel, termed *parallel scan*, and then making some of the latches *not* scannable in order to reduce the hardware overhead required by the method. Partial parallel scan also allows any sequence of inputs to be stored in the latches, which can at times be an important capability [3]. Partial parallel scan trades off hardware overhead for test application effort.

The rest of the paper is organized as follows. Section II provides a brief overview of currently available DFT approaches using scan design and describes the parallel scan method. Section III investigates the theoretical aspects of optimal use of the method. Section IV describes how parallel scan can be implemented efficiently using a partial scan philosophy. Section V gives results of applying the method to an LSI and a VLSI design. Finally, Section VI provides concluding remarks.

II. DFT Using Scan Design

2.1. *Existing Scan Design Techniques*

Most structured DFT techniques are based upon the concept of being able to control and observe all latches in a circuit so that it can be viewed as a combinational circuit for test generation purposes. The way in which this is done is to insert a multiplexer in front of the latch, either as a separate element [1] or embedded into the design of the latch [2], [7], and to introduce extra routing connections so that when the circuit is placed in scan mode, all of the latches form one long serial-in, serial-out shift register (referred to as a scan chain). If the application of each test to a circuit is preceded and followed by a scan operation, in which the contents of the shift register are scanned out and new values shifted in, then the generation of tests for the circuit is considerably simplified.

There have been several departures from this strictly serial scan design type of approach. In random access design [6], each latch is treated as a cell in a memory array. An addressing scheme, similar to that of a random-access memory, is used to allow each latch to be uniquely se-

Manuscript received July 18, 1988; revised April 4, 1989. This work was supported in part by NASA under Grants NAG-1-492 and NAG-1-296. This paper was recommended by Associate Editor V. K. Agarwal.

The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 8931297.

lected so that it can be controlled or observed. This approach has a high I/O pin overhead and fairly high hardware and routing overhead. In addition, it does not solve the problem of excessive test application effort because each latch has to be accessed separately.

Multiple scan-path design [1], [8] is an approach in which the single serial scan chain is replaced by multiple scan chains. The serial inputs and outputs to these scan chains can be obtained by multiplexing them off of the primary I/O pins. While this approach can reduce test application effort by about one order of magnitude, the hardware overhead required is considered to be too high, and thus the approach was dismissed in [1]. However, this approach can still sometimes be used (with less than the maximum possible number of scan chains) to break up overly long scan chains [9].

Partial scan design [10], [11] is a design approach in which only a subset of the latches are included in the scan chain in order to reduce the often unacceptably high overhead penalty of scan design. The selection of which latches to include in the scan chain is made using testability analysis programs such as SCOAP as in [10] or heuristic approaches as in [11]. Analytical and numerical analyses show that it is sometimes beneficial to use partial scan.

2.2. Parallel Scan

Parallel scan is a design approach in which the latches are scanned in parallel, with the number of I/O pins limiting the degree of parallelism. In parallel scan, there is no scan chain. When the number of latches is less than the minimum of the number of input and output pins available, then the structure shown in Fig. 1 is used. All latches can be controlled and observed in parallel. The *test_in* inputs in Fig. 1 are connected to the primary input connections. The outputs of the latches are driven through tristate buffers onto lines which are connected to the primary output pins. Note that since parallel scan does not require a shift register capability, the latches can be simple transparent *D*-latches instead of master-slave or edge-triggered latches.

When there are too many latches for the method of Fig. 1 to be used, the latches must be split up into groups of smaller sizes as shown in Fig. 2. Each "SCAN GROUP" in Fig. 2 has the structure of the circuit in Fig. 1. The *not_test* control inputs required can be obtained by decoding $\log_2 N$ test mode lines (N is the number of scan groups) which can themselves be multiplexed off of primary input lines. The *clk* control inputs can be obtained as qualified clocks using a negligible amount of combinational logic.

For ease of explanation, let us use the following notation. We will assume a sequential circuit with m latches, NI input pins, and NO output pins. Let NID be the number of input pins other than asynchronous control inputs. Asynchronous control inputs must always be controlled and thus are not available to provide test inputs. Let $\lceil x \rceil$ ($\lfloor x \rfloor$) denote the least (greatest) integer greater (less)

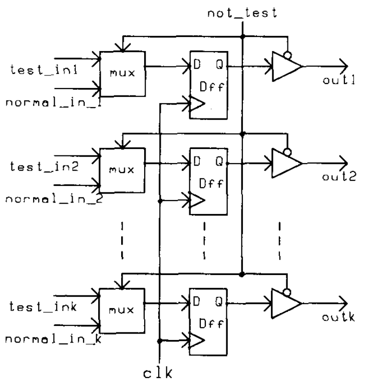


Fig. 1. Parallel scan with a small number of latches (SCAN GROUP circuit).

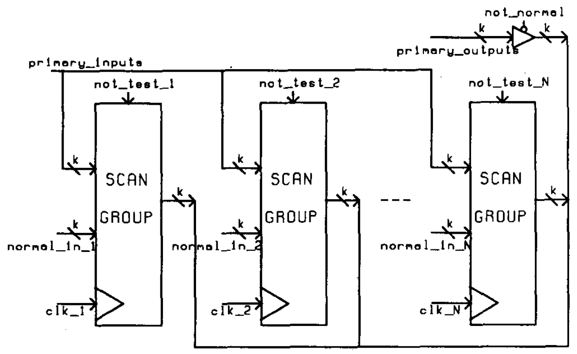


Fig. 2. Parallel scan with a large number of latches.

than or equal to x, SD be the set of m latches, k be the maximum number of latches allowed in a scan group, and N be the total number of scan groups.

Let $MINIO$ = minimum $\{NID, NO\}$ and $AVEIO = \lfloor NID + NO/2 \rfloor$. If $m \leq MINIO$, then the method of Fig. 1 can be used. If $m \leq AVEIO$, then the method of Fig. 1 can still be used with the additional requirement of changing a few input or output pins to bidirectional pins. If $m > AVEIO$, then the method of Fig. 2 must be used and the latches must be split up into groups. There will be $N = \lceil m/AVEIO \rceil$ groups with a maximum of $k \equiv AVEIO$ latches in each group. Within a single group, all of the latches can be scanned in parallel.

The selection of the group to be scanned is done with decoding logic placed adjacent to the groups that they control. The tristate buffers used in the design of the SCAN GROUP circuit of Fig. 1 combined with the tristate buffers for the normal primary outputs in Fig. 2 constitute a "distributed" multiplexer since the *not_test* and *not_normal* control lines permit only one set of outputs to be placed onto the output pins. The number of extra inputs required to address the latches is $\lceil \log_2 \lceil m/AVEIO \rceil \rceil$. As is done in [6] and elsewhere, extra hardware in the form of multiplexers or latches can be used to reduce the number of extra I/O pins required. In parallel scan, the extra inputs for addressing the latches can either be brought in directly as extra I/O pins or simply multiplexed off of the normal primary input pins; one extra primary input pin is still required to select test mode.

2.2.1. *Test Application*

In a circuit designed using the above parallel scan method, test application is done in the following manner. Tests are generated for the combinational part of the circuit assuming that all latches are controllable and observable. Next, an ordering is imposed on the set of tests. In the test generation phase, information about which latches need to be controlled and observed for each test needs to be saved along with the tests. Let c_i and r_i denote the set of latches that need to be controlled and observed, respectively, for test t_i. To apply test t_i after the application of test t_{i-1}, we need to scan out r_{i-1} and scan in c_i. Thus the latches in $r_{i-1} \cup c_i$ need to be scanned using $\leq N$ steps (the scan in and scan out operations can be combined). This can be done by selection of the appropriate scan groups. In a large circuit, it should frequently be the case that not all N scan groups need to be scanned.

The reader will note that the efficiency of this method is dependent on the order of the tests selected and on the partition of *SD* used. Optimality questions are addressed in Section III. It is not possible to efficiently select an optimal order of tests and an optimal partition of *SD* because those problems are *NP*-hard, as we shall prove in Section III. However, even if an arbitrary ordering of tests and a partition of *SD* based simply on adjacency properties is used, there should be a significant improvement in scan time over serial scan methods and even multiple scan-path design. Depending on the efficiency of the partition and on the order of the tests applied, scan time, and hence, test application time should be reduced by one to two orders of magnitude over serial scan methods assuming an IC chip with 28–64 I/O pins.

With respect to test application, parallel scan has advantages as well as disadvantages when compared to multiple scan-path design. The main advantage of parallel scan is that given the same degree of parallelism k, parallel scan will require fewer scan steps since not all N scan groups need to be scanned for each test pattern. The main disadvantage of parallel scan is that testing using the proposed structure is more complicated since we need to specify the sets of control and observation latches for each test pattern. This, however, is more of a test generation problem. Test generation is discussed in Section IV.

2.2.2. *Routing and Hardware Overhead*

The routing and hardware overhead required by the parallel scan method is similar to the overhead required using multiple scan-path design. Also, the fan-out required from primary input lines is not exorbitant.

In parallel scan, each primary input will be connected to N latches in addition to its normal connections (N is the number of scan groups). Thus given a 64-pin circuit and assuming 30 pins are used for loading test inputs, $N = \lceil m/30 \rceil$. Thus with $m = 150$, N is only 5. This is well within acceptable fan-out limitations.

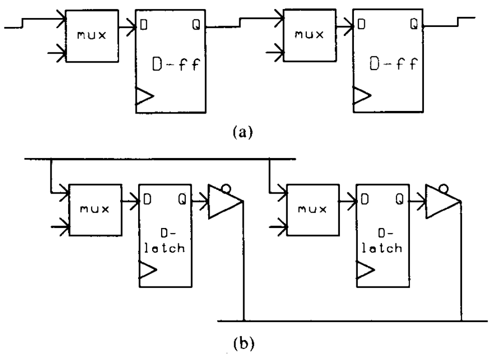


Fig. 3. Comparison of (a) multiple scan-path design and (b) parallel scan.

With multiple scan-path design, the beginning of each scan chain must be connected to a primary input pin. If extra test pins are allocated for this purpose, then there must be k extra test pins just for this purpose (recall that k is the maximum degree of parallelism permitted). If extra test pins are not allocated for this purpose, then they can simply be connected to normal primary input connections as in parallel scan.

In multiple scan-path design, the connection between two adjacent latches in a single scan chain is made from the output of one latch to the input of the multiplexer which connects to the next latch. In parallel scan, these same two latches are connected such that the outputs of the two latches are connected to primary outputs, and the wire which attaches to the input of the multiplexer for the first latch connects from that point to the input of the multiplexer of the next latch. This comparison is shown in Figs. 3(a) and 4(b). For the sake of clarity, only the major connections are shown. Thus compared to multiple scan-path design, there is only one extra routing connection for each latch in the scan design.

One difference in hardware overhead between the two methods is that each latch in the parallel scan method requires an extra tristate buffer. However, note that in parallel scan, the latches can be simple transparent *D*-latches, which are simpler than the latches required for multiple scan-path design (since the latter latches must be capable of shifting). Thus considering these two offsetting factors, the hardware overhead required by the two methods should be similar.

Even though the routing and hardware overhead required by parallel scan is similar to multiple scan-path design, the overhead required by a multiple scan-path design with *AVEIO* scan chains would probably be too high in a VLSI design. Thus parallel scan would not be feasible unless a *partial* scan design philosophy is followed. Two heuristics for selecting the latches to include in the scan design are described in Section IV. Using these heuristics, only about 10–60 percent of the latches need to be included in the scan design. *Partial parallel scan* refers to the combination of the use of parallel scan as defined in this section and the partial scan methodology described in Section IV.

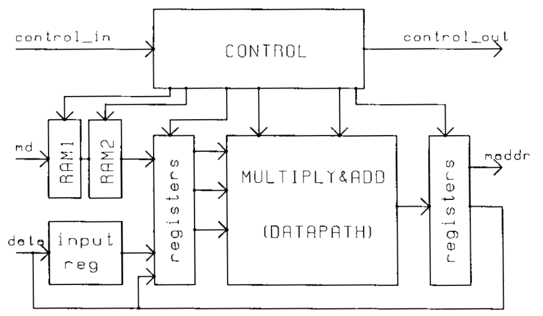


Fig. 4. Block diagram of 3-D linear interpolator.

III. Optimal Use of Parallel Scan

To speedup test application as much as possible when using a parallel scan design method, it is necessary to minimize the number of scan steps required for the application of each test in the test set. The order in which the tests are applied is important because the latches which need to be observed for test t_{i-1} can be scanned out at the same time that the latches which need to be controlled for test t_i are scanned in. Given a fixed partition of *SD* into scan groups, the number of scan groups which need to be scanned to apply test t_i and observe the result of test t_{i-1} is dependent on $r_{i-1} \cup c_i$. Thus there is a test order for which the total number of scan steps is minimized. The number of steps required to scan the latches in $r_{i-1} \cup c_i$ is also dependent on the way in which *SD* is partitioned into scan groups. Problem 1 is the problem of finding an optimal partition of *SD* given a fixed test order. Problem 2 is the problem of finding an optimal test order given a fixed partition of *SD*. Problem 3 is the problem of finding a partition of *SD* and a test order which minimizes the total number of scan steps required. Solutions to these three problems would result in more efficient implementations of the parallel scan design method.

The above problems are formalized below and shown to be *NP-hard* (which effectively means that no polynomial-time algorithms exist for their solution). We require the following notation to describe the application of tests using the parallel scan method. Let $T = \{t_1, t_2, \cdots, t_{n-1}\}$ denote the set of $n - 1$ tests required to test the circuit. Recall that each test t_i requires the control of c_i and the observation of r_i, where c_i and r_i are sets of latches. Thus $CR \equiv \{(c_1, r_1), (c_2, r_2), \cdots, (c_{n-1}, r_{n-1})\}$ is an alternative characterization of the set of tests. Let an ordering of T be denoted by a function $o: Z_{n-1} \rightarrow Z_{n-1}$, where $Z_p = \{1, 2, \cdots, p\}$, and $o(i)$, $i \in Z_p$, is the ith element of o. Thus the tests in T would be applied in the order $t_{o(1)}$, $t_{o(2)}$, etc. Let $S = \{s_1, s_2, \cdots, s_n\}$ denote the set of latches which need to be scanned for the application of the tests in T given the test order o, where $s_i = r_{o(i-1)} \cup c_{o(i)}$. A partition of *SD* is represented by $P = \{\pi_1, \pi_2, \cdots, \pi_N\}$, where $|\pi_i| \leq k$ for all i $(1 \leq i \leq N)$ and $N \equiv \lceil m/k \rceil$. Let $Q_i \subseteq P$ be the set of scan groups which must be scanned in order to account for all of the latches in s_i, i.e., $\pi_j \in Q_i \leftrightarrow \pi_j \cap s_i \neq \emptyset$. This means that $|Q_i|$ scan steps are required for the application of test $t_{o(i)}$ given the test order o and the partition P. Thus the total testing length LENGTH $= \Sigma_{i=1}^{n} |Q_i|$.

Problem 1 is to find the optimal partition of the set of latches given a test order. More formally, we are given *SD*, a set of m elements, k, an integer between 1 and m, and S, a set of n subsets of *SD*. The problem is to find a partition P^* of *SD* into N disjoint sets of size not more than k each such that LENGTH is minimized.

Problem 2 is to find the optimal ordering of the test set given a partition of the set of latches. More formally, we are given *SD*, a set of m elements, k, an integer between 1 and m, P, a partition of *SD* into N disjoints sets of size not more than k each, and *CR*, a set of $n - 1$ ordered pairs (c_i, s_i), where $c_i, s_i \subseteq SD$. The problem is to find an ordering o^* of the members of *CR* such that LENGTH is minimized.

Problem 3 is the combination of Problems 1 and 2. The definitions used are the same as those for Problems 1 and 2. We are given *SD*, a set of m elements, k, an integer between 1 and m, and *CR* a set of $n - 1$ ordered pairs (c_i, s_i), where $c_i, s_i \subseteq SD$. The problem is to find an ordering o^* of *CR* and a partition P^* of *SD* into N disjoint sets of not more than k each such that LENGTH is minimized.

The above three problems and the concepts described can best be illustrated with an example.

Example 1: $SD = \{1, 2, 3, 4, 5, 6, 7\}$. $T = \{t_1, t_2, t_3\}$. The sets of control and observation latches c_i and r_i are shown in Table I. Choosing $k = 2$, $m = |SD| = 7$, and $N \equiv \lceil m/k \rceil = 4$, let us choose the ordering $o(i) = i$ for all $i: 1 \leq i \leq 3$. Let us also choose the partition $P = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7\}\}$. Given this ordering and partition, the sets s_i and Q_i are shown in Table I. Given ordering o and partition P, LENGTH $= 3 + 3 + 3 + 1 = 9$. Note that using multiple scan-path design with the same degrees of parallelism $k = 2$, we would require a total of $4 \times 4 = 12$ scan steps. It is clear that with a different ordering, the sets s_i and Q_i will change, thus possibly resulting in a different total length. Also, given a different partition of *SD*, the sets Q_i required will change, again possibly resulting in a different total length. The Problems 1–3 address these issues.

All of the three problems described above are *NP*-hard. A search problem can be shown to be *NP*-hard by showing that the decision problem based on the search problem is *NP*-complete or *NP*-hard. The decision problems OPT-PARTITION and FIND-ORDER, defined below, are the decision problems based on Problems 1 and 2, respectively.

PROBLEM: OPT-PARTITION

INSTANCE: A finite set *SD* of m elements, an integer k between 1 and m, a set S of n subsets of *SD*, and a bound $B \in Z^+$ (set of positive integers).

QUESTION: Can a partition of *SD*, $P = \{\pi_1, \pi_2, \cdots, \pi_{\lceil m/k \rceil}\}$, where $\forall i: 1 \leq i \leq \lceil m/k \rceil$, $|\pi_i| \leq k$, be found such that LENGTH $\leq B$, where LENGTH is as defined above.

TABLE I
PARAMETERS c_i, r_i, s_i, AND Q_i FOR EXAMPLE I

i	c_i	r_i	s_i	Q_i
1	{1,4,6}	{2,3}	{1,4,6}	{{1,2},{3,4},{5,6}}
2	{2,7}	{4,6,7}	{2,3,7}	{{1,2},{3,4},{7}}
3	{3,4,5}	{2}	{3,4,5,6,7}	{{3,4},{5,6},{7}}
4			{2}	{{1,2}}

PROBLEM: FIND-ORDER

INSTANCE: A finite set *SD* of m elements, an integer k between 1 and m, a partition $P = \{\pi_1, \pi_2, \cdots, \pi_N\}$ of S where $|\pi_i| \leq k \ (1 \leq i \leq N)$, *CR*, a set of $n - 1$ "tests" (actually ordered pairs of sets as defined above), and a bound $B \in Z^+$.

QUESTION: Is there an ordering o of the $n - 1$ tests such that LENGTH $\leq B$, where o and LENGTH are as defined above.

Since the decision problem derived from a search problem is no harder than the search problem, if it can be shown that OPT-PARTION and FIND-ORDER are *NP*-complete or *NP*-hard, then it will have been shown that Problems 1 and 2 are *NP*-hard. Also, since Problems 1 and 2 are restrictions of Problem 3, the *NP*-hardness of Problem 3 follows from the *NP*-hardness of Problems 1 or 2. Let D_π and Y_π be the domain and yes-set of a decision problem Π. Then the *complement* of Π, Π^c, is defined as the decision problem having domain set D_π and yes-set $D_\pi - Y_\pi$.

Theorem 1: Problem OPT-PARTITION is *NP*-hard.

Proof: The proof follows if it can be shown that OP^c, the complement of OPT-PARTITION, is *NP*-complete.

$OP^c \in NP$ since a nondeterminisitic algorithm need only guess a partitioning P of S satisfying the conditions stated for OPT-PARTITION and check in polynominal time whether LENGTH $\leq B$.

Next, it shall be shown that OP^c can be restricted to SET-SPLITTING. SET-SPLITTING is defined as follows [12]:

INSTANCE: Collection C of subsets of a finite set *SC*.

QUESTION: Is there a partition of *SC* into two subsets SC_1 and SC_2 such that no subset in C is entirely contained in either SC_1 or SC_2.

Note that the definition of OP^c is the same as OPT-PARTITION except that it is required that LENGTH $> B$. Let α be a new element that is not a member of *SC*. Let $SD = SC \cup \{\alpha\}$, $k = |SC|$, $C = S$, and $B = 2|C| - 1$. The *new* element α is required to make $N = \lceil |SD|/k \rceil = 2$. Under the above restrictions, OP^c is the same as SET-SPLITTING, a known *NP*-complete problem. Therefore, it follows that OP^c is *NP*-complete and OPT-PARTITION is *NP*-hard. Q.E.D.

Theorem 2: Problem FIND-ORDER is *NP*-complete.

Informal Proof: The proof follows the basic approach used for proving the *NP*-completeness of TRAVELING SALESMAN [12]. The formal proof is rather lengthy, and therefore, relegated to the Appendix. Given here is an intuitive, informal proof.

An arbitrary instance of the directed Hamiltonian path with given starting and ending points (DHPSE) problem is transformed to an instance of FIND-ORDER. Since DHPSE is a known *NP*-complete problem [12], if DHPSE can be transformed to FIND-ORDER in polynomial time, then FIND-ORDER will have been shown to be *NP*-complete. Let $G = (V, E)$ be an arbitrary instance of DHPSE with starting vertex v_0 and ending vertex v_{N-1}, where V and E are the vertices and directed edges of the digraph G, respectively. Also, let $V = \{v_0, v_1, \cdots, v_{N-1}\}$. Each vertex $v_i \in V$ is considered as a test $t_i = (c_i, r_i)$. The elements of c_i and r_i and the elements of the partition P are chosen in such a manner that if an edge exists from v_i to v_j, then the elements of $r_i \cup c_j$ are included in $N - 1$ subsets of P; otherwise, $r_i \cup c_j$ is included in N subsets of P. Also, c_0 and r_{N-1} are included in one subset of P each. The bound B is then set to $(N-1)^2 + 2$.

Any directed Hamiltonian path with starting point v_0 and ending point v_{N-1} that is found in the graph G will correspond to an ordering of the tests such that LENGTH $\leq B$ because each test transition involves the use of exactly $N - 1$ subsets of P and the first and last tests require 1 subset of P each. If such a directed path does not exist, then there does not exist an ordering of the corresponding tests such that LENGTH $\leq B$ because any complete ordering of the test must involve at least one test transition which uses exactly N subsets of P. The interested reader is referred to the Appendix for a more formal proof.

Q.E.D.

Corollary 1: Problem 1 is *NP*-hard.
Corollary 2: Problem 2 is *NP*-hard.
Corollary 3: Problem 3 is *NP*-hard.

While optimal use of the parallel scan method is an *NP*-hard problem, heuristic approaches can be used to obtain "good" solutions. In most situations, routing considerations will dictate the partition of *SD* that is actually used. Thus for instance, in a w-bit wide data-path, the set of w latches in a given column of the data-path would be grouped into a single scan group. While manipulating the order of test application to reduce test application time is a more feasible approach, due to the size of test vector sets for large circuits (on the order of 1 million test vectors), optimizing the order of test application would be extremely expensive computationally even if a polynomial-time solution were available. If there are a large number of latches that need to be scanned and optimal use of the parallel scan technique is nevertheless desired, an A^* algorithm with an *admissible* heuristic could be used to obtain an optimal solution.

IV. Partial-Scan Implementation of Parallel Scan

To gain the advantages of parallel scan with reasonable hardware overhead, a partial-scan design philosophy must be used, since the hardware and routing overhead can be slightly more than even multiple scan-path design. However, as demonstrated by its use in an LSI and VLSI chip design, it can be efficiently implemented using a partial scan method where the set of latches to be included in the partial scan (to be referred to as *PSD*) are determined using heuristic methods described below.

Although their analyses are different, Trischler [10] and Agrawal *et al.* [11] show that it can be beneficial to use a partial scan design philosophy. Trischler [10] uses experimental data to conclude that maximum cost-effectiveness is achieved by including 15–25 percent of the latches in *PSD*. Trischler used testability analysis to determine the set *PSD*. Agrawal *et al.* [11] also used experimental data to show that less than 65 percent of the latches can be included in *PSD* to achieve fault coverage higher than 95 percent. They used heuristics based on the frequency of usage of the latches in tests and the distance of the latches from observation and control points.

Given that heuristics are used to determine *PSD*, it makes sense to use knowledge about the design of the specific circuit being tested to determine *PSD*. There are commercial tools such as HITEST [13]–[15] which allow the user to enter knowledge about how to control and observe the contents of latches, and to use this knowledge in the test generation task. However, the utility of this capability is limited by the fact that some latches are difficult to control and observe and some latches cannot be set to desired values without distributing the contents of other critical latches.

The heuristic proposed here is to let the IC designer determine those latches which are easy to control and observe and those which are not. The measure of ease is partly determined by the number of clock cycles required and its effect on other critical latches. Note that if certain latches are constrained to be part of the scan design, then the number of clock cycles required to control and observe other latches will change. Thus this heuristic by itself is not that simple to implement. A testability analysis tool such as SCOAP could be used to aid in the implementation of this heursitic.

If the target circuit is composed of a control section and a datapath section (as is typical in many digital designs), an additional heuristic is to include the latches in the control section of the IC in *PSD*. By enabling the test engineer to control the state of the control circuitry and the inputs to the control section, the signals for controlling the flow of data through the data-path section can be controlled. Since the flow of data can be controlled, it should be possible to control and observe the contents of all of the latches in the data-path. For example, those latches used to hold the intermediate results of an ALU computation are clearly part of the data-path and can be left out of the scan design. Whatever latches are left undetermined by this procedure should be picked up using the first heuristic described above.

For each latch in the circuit, the above two heuristics to select *PSD* and the use of the parallel scan method with the latches in *PSD* should result in a method for controlling and observing every latch in the circuit.

4.1. Test Generation with Partial Scan

Our partial scan method depends on the availability of a tool such as HITEST [13]–[15] which allows knowledge about how to control and observe the contents of latches to be included into its test generation procedure. The test generation facility in HITEST uses knowledge about the target circuit, such as would be used by an expert test engineer, to constrain the search for test sequences [13]. HITEST can use knowledge about how to control and observe any set of signals, including but not limited to, those signals at the boundaries of latches. Information has to be provided on all of the signals at the boundaries of the sequential parts of the circuit [14]. These signals are then considered to be pseudoprimary inputs and outputs when generating tests for the rest of the circuit. A combination of RAPS and PODEM [9], constrained by whatever extra knowledge is provided, is used to generate tests for the combinational parts of the circuit.

To perform test generation in a circuit designed using partial scan, HITEST has to be provided with the knowledge about how to control and observe every latch. The sequence of input vectors to control and observe those latches which are scannable are first specified. Then, to control those latches in the datapath, the input vectors required to set the state of the control circuitry (with scan-in operations) and the input vectors to bring in values from the input ports to the desired latches have to be specified. Likewise, how to observe the contents of latches in the datapath also has to be specified. Once this information is provided to HITEST, HITEST can generate tests for the combinational parts of the circuits, regarding all latches as being controllable and observable. The test vectors generated by HITEST consist of input vectors to set the appropriate latches to their desired states, followed by input vectors to test combinational parts of the circuit, followed by input vectors to observe and control latches, followed by more input vectors to test combinational parts of the circuit, and so on. Since all latches are controllable and observable, fault coverage is not sacrificed due to the requirement of using partial scan.

V. Experimental Results

The partial parallel scan method with the heuristics described above for determining the latches to be scanned was implemented in two IC designs, a 300-transistor BCD-to-binary converter circuit and a 20 000-transistor 3-D linear interpolator circuit. In these designs, the *SD* latches were partitioned simply on the basis of adjacency. Both designs were done using Seattle Silicon's Concorde silicon compiler and Mentor Graphics' network editor and simulator tools.

The BCD-to-binary converter, intended to covert 12-bit BCD into 8-bit binary numbers, had 15 latches and required approximately 300 transistors. Of the 15 latches, 9 could easily be set to any desired value. Therefore, partial parallel scan was used with only 6 latches. In addition, 9 multiplexers were used to make the remaining 9 latches easily observable. Given these DFT extensions,

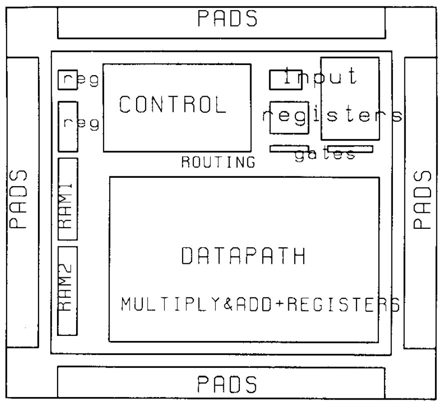


Fig. 5. Map of layout for 3-D linear interpolator.

TABLE II
BCD-TO-BINARY CONVERTER

Technology	MOSIS 3 μm p-well CMOS
Area (original)	2.970 mm X 3.051 mm
Transistor Count	approx. 300
Area (DFT)	3.415 mm X 3.155 mm
Total # FF's	15
Scan FF's	6
Extra Observ. Pts.	9
Area Overhead	19%
Fault Coverage	95.14%
Speedup	11.25 over serial scan

the entire circuit could be treated as combinational for test generation purposes and only two clock cycles were required to control and observe the contents of all latches. To obtain a fault coverage figure for this circuit, test vectors were generated using the PODEM algorithm [16]. Fault coverage of 95.14 percent (for gate-level single stuck-at faults) can be achieved using either serial or partial parallel scan. 100 percent fault coverage cannot be achieved because the use of standard components introduced some redundancy and unused outputs. The speedup figures given are based on the number of clock cycles required for the application of all test vectors. Information on the original chip and its DFT extension are given in Table II.

The 3-D linear interpolator was designed to be a prototype for a fast linear interpolator on a chip which could be used to control the spark advance in an automobile engine control system. The completed circuit had 241 latches (excluding a 4×29 bit on-chip RAM) and required approximately 20 000 transistors. In the DFT extension version, 25 flip-flops were included in the scan design and 7 extra observation points were added. With this DFT extension, each set of 16 latches in the data-path (since the data-path was 16 bits wide) could be controlled and observed independently using 2–3 clock cycles. Bounds on the speedup in test application time were obtained by assuming inefficient and theoretically optimum implementations of partial parallel scan. A block diagram of the circuit is shown in Fig. 4, a map of the chip layout is shown in Fig. 5, and a color print of the chip layout is shown in Fig. 6. Information on the original chip and its DFT extension are given in Table III.

The area overhead consumed by the use of the partial parallel scan technique on these two designs was 19 percent and 6 percent. The relative area overhead was much higher in the smaller design because of the relatively large impact that even a small addition has on a small chip. The fact that the area overhead consumed by including 25 latches in the scan design of the larger design was only 6 percent is very encouraging. In the smaller design, only one extra input pin was required to implement the partial parallel scan. In the larger design, no extra input pins were required as one of the operation modes of the chip happened to be available.

VI. Concluding Remarks

The purpose of using parallel scan with a partial scan philosophy is to speed up the test application time without negatively affecting test generation effort and with little hardware overhead. As in traditional serial scan techniques, test generation can take place viewing the target circuit as purely combinational. The benefit of using partial parallel scan is that at the expense of very little hardware overhead, test application effort can be reduced by 1 to 2 orders of magnitude. Such a decrease in test application effort will allow more thorough testing and increase throughput.

The main drawback of the (partial) parallel scan technique is that it is no longer a *structured* DFT technique. Because optimal use of the parallel scan technique is an *NP*-hard problem, heuristics must be used in determining the partitioning of the latches in the circuit and the ordering of the set of tests to be applied. Also, due to its additional area overhead requirements, a partial scan technique must be used. As was done by previous researchers [10], [11], a heuristic method was used to determine the latches to include in the scan design. However, unlike previous methods, fault coverage was not sacrificed due to the requirement of using partial scan. Instead, the test generation program is required to have a knowledge-base component such as in HITEST [14] in order to automatically generate a test program.

When combined with a partial scan approach, parallel scan is a practical method for reducing test application effort. The routing and hardware overhead required is similar to multiple scan-path design, given the same number of latches in the scan design and k scan chains, where k is the size of the scan group (Fig. 1). The use of the partial scan heuristics described in Section IV allows the parallel scan method to be implemented with low hardware and area overhead.

This paper has introduced and demonstrated the feasibility of using partial parallel scan. In the example circuits on which partial parallel design was used, the extra area overhead required was 19 percent and 6 percent (DFT extension version 2). The latter figure of 6 percent is more representative because it corresponds to a circuit of large enough size for partial parallel scan to be useful and desirable. Only one extra input pin is required by this DFT technique. The small I/O pin and area overhead required by the partial parallel scan DFT technique make it a

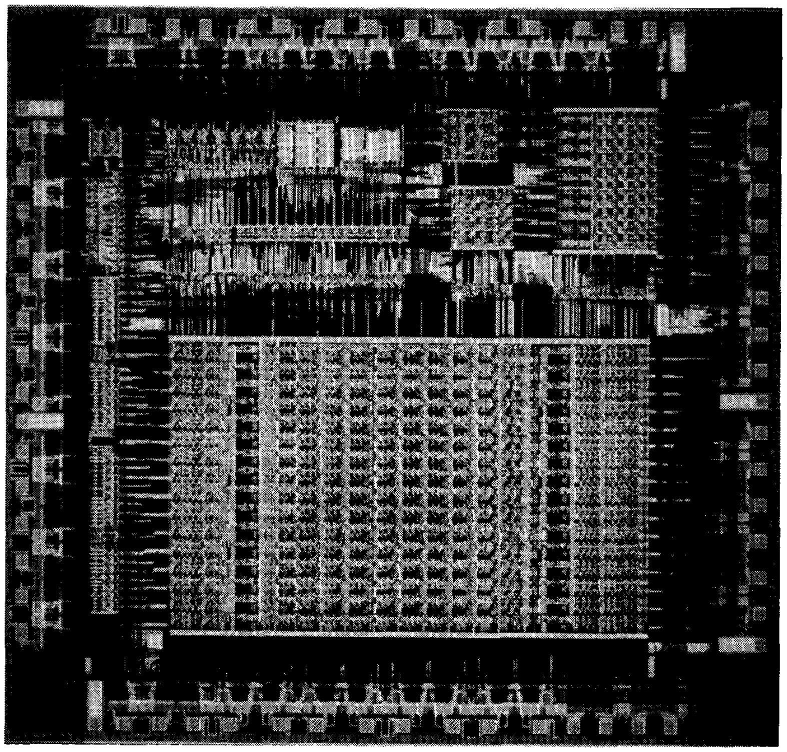


Fig. 6. Print of layout for 3-D linear interpolator.

TABLE III
3-D LINEAR INTERPOLATOR

Technology	NCR 2 μm n-well CMOS
Area (original)	6.742 mm X 6.331 mm
Transistor Count	approx. 20,000
Area (DFT)	6.994 mm X 6.465 mm
Total # FF's	241
Scan FF's	25
Extra Observ. Pts.	7
Area Overhead	6%
Speedup	8 to 164 over serial scan

promising method for significantly decreasing test application effort.

APPENDIX
FORMAL PROOF OF THEOREM 2

FIND-ORDER $\in$ *NP* since a nondeterministic algorithm need only guess an ordering for *CR* and check in polynomial time whether LENGTH $\leq B$.

Next, we shall transform an arbitrary instance of the DHPSE problem to an instance of FIND-ORDER. Since DHPSE is a known *NP*-complete problem [Garey 79], if DHPSE can be transformed to FIND-ORDER in polynomial time, then FIND-ORDER will have been shown to be *NP*-complete.

Let the graph $G = (V, E)$ be an arbitrary instance of DHPSE with starting point $v_0 \in V$ and ending point $v_{N-1} \in V$, $|V| = N$, and $|E| = M$. The corresponding instance of FIND-ORDER is generated as follows. Assign an arbitrary ordering to the elements of $V - \{v_0, v_{N-1}\}$ such that v_i is the ith element of $V - \{v_0, v_{N-1}\}$. Execute the following algorithm:

```
1. $\pi_0 := \{q_0\}$;  { $\pi_i$'s are the blocks of the partition }
2. $\pi_N := \{p_{N-1}\}$;  { $p_i$'s are the second halves of tests }
3. For $i$ from 1 to $N-1$ do
     $\pi_i := \{p_{i-1}\}$;
     LEFT$_i$ := $\{q_{i,1}, q_{i,2}, \cdots, q_{i,N-1}\}$;
                { $q_{i,j}$'s are members of the first half of a test }
     MIN$_i$ := 1
   endfor;
4. For $i$ from 0 to $N-2$ do
     For $j$ from 1 to $N-1$ do
       if $(v_i, v_j) \in E$ then
         $\pi_{i+1} := \pi_{i+1} \cup \{q_{j,\mathrm{MIN}_j}\}$;
         MIN$_j$ := MIN$_j$ + 1
       endif
     endfor
   endfor;
5. $r := N$;
6. For $i$ from 1 to $N-1$ do  { pick up $q_{i,j}$'s not included in any partition block }
     For $j$ from MIN$_i$ to $N-1$ do
       $r := r + 1$;
       $\pi_r := \{q_{i,j}\}$
     endfor
   endfor;
7. $maxsize$ := maximum$(|\pi_i|)$ $(1 \leq i \leq r)$;
8. $s := 0$ and $Pt := \{\}$;
9. For $i$ from 0 to $r$ do  { make sizes of blocks of partition equal }
```

```
       For j from $|\pi_i| + 1$ to maxsize do
         $s := s + 1$;
         $\pi_i := \pi_i \cup \{z_s\}$   { $z_s$'s are "new" variables }
       endfor;
       $Pt := Pt \cup \pi_i$
     endfor;
10. The corresponding instance of FIND-ORDER has:
       $B = (N-1)^2 + 2$,
       $k = maxsize$,
       $P = Pt$,
       $CR = \{ \{ \{q_0\}, \{p_0\} \}, \{ \{q_{1,1}, q_{1,2}, \cdots, q_{1,N-1}\}, \{p_1\} \},$
             $\{q_{2,1}, q_{2,2}, \cdots, q_{2,N-1}\}, \{p_2\} \},$
             $\cdots,$
             $\{ \{q_{N-1}, q_{N-1,2}, \cdots, q_{N-1,N-1}\},$
             $\{p_{N-1}\} \} \}$, and
       $S = flatten(CR) \cup \{z_1, z_2, \cdots, z_s\}$.
             { flatten ($CR$) = set of all lowest level elements of $CR$ }.
```

The above transformation is a polynomial-time algorithm that is $O(N^2)$.

It now remains to be shown that the transformation is correct. What the above transformation does is to cause each transition from one test to another to require exactly $N - 1$ subsets of P if the corresponding edge exists in E and exactly N subsets of P otherwise. To see this, note that steps 2 and 3 create "private" sets in P for each r_i, where c_i and r_i refer to the first and second parts of the test corresponding to v_i. In step 4, a previously unused element of c_j is included in r_i's "private" set if there is an *edge* from v_i to v_j. In step 6, all remaining elements of the first parts of all tests are given "private" sets in P. Therefore, in a transition from v_i to v_j, exactly $N - 1$ subsets of P will be needed to cover all of the elements of c_j. An additional subset of P will be needed for r_i only if there is no directed edge from the vertex v_i to v_j.

Suppose that there exists a directed Hamiltonian path in G with starting vertex v_0 and ending vertex v_{N-1}. Then according to the above construction, the corresponding ordering of the tests results in LENGTH $= (N - 1)(N - 1) + 2 = B$ since there are $N - 1$ transitions between tests, requiring $N - 1$ subsets of P each, and the first and last test require exactly one subset of P each. Conversely, suppose that there does not exist such a directed Hamiltonian path in G. Then, any complete ordering of the set of tests which have been created by the above transformation must involve at least one test transition requiring N subsets of P (corresponding to a non-edge), thereby resulting in LENGTH $> B$. Also, if the first and last tests are any tests other than those corresponding to the starting and ending vertices, then LENGTH $> B$ since the first part of the first test and the second part of the last test have their own "private sets" in P. Therefore, the transformation is correct. Q.E.D.

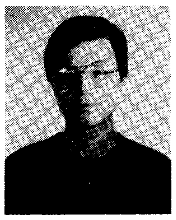
Acknowledgment

The authors would like to thank P. Mazumder of the University of Michigan and the anonymous reviewers for their constructive comments on an earlier draft of this paper.

References

[1] M. J. Y. Williams and J. B. Angell, "Enhancing testability of large-scale integrated circuits via test points and additional logic," *IEEE Trans. Comput.*, vol. C-22, pp. 46-60, Jan. 1973.
[2] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," *J. Design Automat. Fault-Tolerant Comput.*, vol. 2, pp. 165-178, May 1978.
[3] K. K. Saluja, "An enhancement of LSSD to reduce test pattern generation effort and increase fault tolerance," in *Proc. 19th Design Automation Conf.*, pp. 489-494, 1982.
[4] F. Lee, V. Coli, and W. Miller, "On-chip circuitry reveals system's logic states," *Electron. Design*, pp. 119-123, Apr. 1983.
[5] S. Funatsu, N. Wakatsuki, and T. Arima, "Test generation systems in Japan," *Proc. 12th Design Automation Symp.*, pp. 114-122, June 1975.
[6] H. Ando, "Testing VLSI with random access scan," in *Dig. Comp. Conf.*, pp. 50-52, Feb. 1980.
[7] V. G. Oklobdzija and M. D. Ercegovac, "Testability enhancement of VLSI using circuit structures," in *Proc. IEEE ICCC*, pp. 198-201, 1982.
[8] S. M. Reddy and R. Dandapani, "Scan design using standard flip-flops," *IEEE Design Test*, Feb. 1987.
[9] R. G. Bennetts, *Design of Testable Logic Circuits*. Reading, MA: Addison-Wesley, 1984.
[10] Erwin Trischler, "Testability analysis and incomplete scan path," in *Proc. 1983 Int. Conf. CAD*, pp. 38-39, Oct. 1983.
[11] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin, "A complete solution to the partial scan problem," in *Proc. 1987 Int. Test Conf.*, pp. 44-51, 1987.
[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[13] G. B. Robinson, "HITEST—Intelligent test generation," in *Proc. Int. Test Conf.*, pp. 311-323, 1983.
[14] —, *HITEST Reference Manual*, GenRad Inc., Hampshire, U.K., 1987.
[15] —, *HITEST Programming Course: Trainee's Guide*, GenRad Inc., Hampshire, U.K., 1987.
[16] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 215-222, Mar. 1981.

*



Sunggu Lee (S'87) received the B.S.E.E. degree from the University of Kansas, Lawrence, in 1985 and the M.S.E. degree from the University of Michigan, Ann Arbor, in 1987. He is currently working towards the Ph.D. degree in the Department of Electrical Engineering and Computer Science of the University of Michigan, Ann Arbor.

His research interests include fault-tolerant computing, distributed computing, and artificial intelligence.

Mr. Lee is a member of the Tau Beta Pi Engineering Honor Society and the Phi Alpha Phi Honor Society.

*

Kang G. Shin (S'75-M'78-SM'83) received the B.S. degree in electronics engineering from Seoul National University, Korea, in 1970, and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

From 1972 to 1974, he was on the research staff of the Korea Institute of Science and Technology, Seoul, Korea. From 1978 to 1982, he was an Assistant Professor at Rensselaer Polytechnic Institute, Troy, NY. In 1982, he joined the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, where he is currently a Professor. In 1985, he founded the Real-Time Computing Laboratory. He has published over 150 technical papers in the areas of fault-tolerant computing, distributed real-time computing, computer architecture, and robotics and automation.

Dr. Shin was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium and the General Chairman in 1987. In 1987, he received the Outstanding Paper Award from the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.