Test-Point Insertion: Scan Paths Through Functional Logic

Chih-Chang Lin, *Member, IEEE*, Malgorzata Marek-Sadowska, *Fellow, IEEE*, Kwang-Ting Cheng, *Member, IEEE*, and Mike Tien-Chien Lee, *Member, IEEE*

***Abstract*—Conventional scan design imposes considerable area and delay overheads. To establish a scan chain in the test mode, multiplexers at the inputs of flip-flops and scan wires are added to the actual design. We propose a low-overhead scan design methodology that employs a new test-point insertion technique. Unlike the conventional test-point insertion, where test points are used directly to increase the controllability and observability of the selected signals, the test points are used here to establish scan paths through the functional logic. The proposed technique reuses the functional logic for scan operations; as a result, the design-for-testability overhead on area or timing can be minimized. We show an algorithm that uses the new test-point insertion technique to reduce the area overhead for the full-scan design. We also discuss its application to the timing-driven partial-scan design.**

***Index Terms*—Design for testability.**

I. Introduction

AUTOMATIC test pattern generation (ATPG) for sequential circuits is a difficult problem because of the lack of direct controllability of the present state lines and direct observability of the next state lines. To enhance testability, design-for-testability (DFT) techniques aimed at improving controllability and observability of the state lines have been proposed, such as full scan [5], [15], [31] and partial scan [2], [4], [6], [9], [11]–[13], [18], [24], [29]. Both scan techniques facilitate testing of a sequential circuit by interconnecting the selected flip-flops into a shift register during the test mode to control and observe the state lines directly. The complexity of ATPG is therefore reduced. However, the area and delay overheads imposed by conventional scan insertion can be significant due to the extra multiplexers in the scan flip-flops (assuming the multiplexed D flip-flops are used) and the routing area for the scan chains.

To alleviate the above DFT penalty, we propose a low-overhead scan design methodology [26] which applies the *test-point insertion* to establish the scan paths through the existing functional logic. These test points are established by appropriately inserting two-input AND gates or two-input OR gates with a common test input. Note that unlike the conventional test-point insertion, where test points are used directly to increase the controllability and observability of the selected signals, the test points are used here to establish scan paths through the functional logic. The main idea is illustrated in Fig. 1. Fig. 1(a) shows a portion of a sequential circuit, where the boxes represent flip-flops. By inserting the test point at the output of F_4 and setting the primary input x to zero during the test mode, a scan chain $F_1 \rightarrow F_2 \rightarrow F_3$ can be formed through the functional logic, as shown in the dotted line in Fig. 1(b). In this example, we established a partial-scan chain involving three flip-flops using the functional logic. The area overhead is a two-input AND gate, while the conventional scan design would require two multiplexers.

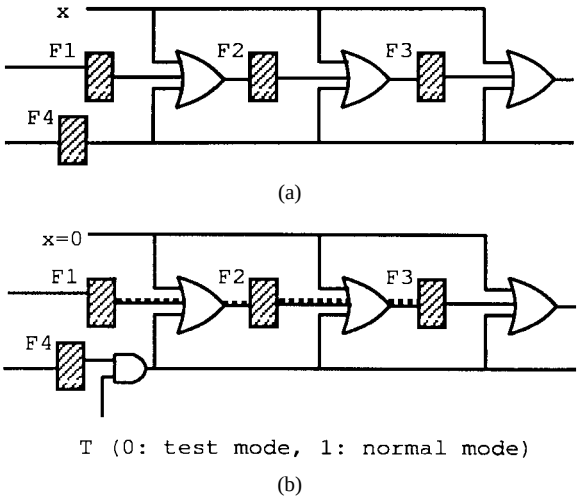


Fig. 1. Example of a test-point insertion.

In our method, the cost of inserting a test point is one AND (OR) gate and a connection from the test input T, while converting a flip-flop into a multiplexed scan flip-flop requires a multiplexer, a connection from another flip-flop, and a connection from the test input T. Inserting test points is advantageous in terms of area, if k test points can successfully establish k or more scan paths. A scan path here is defined as a physical path between two flip-flops that can be fully sensitized in the test mode. Moreover, the method of inserting test points could be applied for the timing-driven scan design. For example, we can add test points away from the critical paths while still being able to establish scan paths through the critical nets.

Manuscript received May 13, 1996; revised March 31, 1998. This work was supported in part by the National Science Foundation under Grant MIP9419119 and in part by the California MICRO program through Fujitsu/Xilinx/Actel. This paper was recommended by Associate Editor S. Reddy.

C.-C. Lin was with the University of California, Santa Barbara, CA 93106 USA. He is now with Verplex Systems, Inc., Santa Clara, CA 95054 USA.

M. Marek-Sadowska and K.-T. Cheng are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA.

M. Tien-Chien Lee is with Avant! Corp., Fremont, CA 94538 USA.

Publisher Item Identifier S 0278-0070(98)06763-3.

We discuss two applications of using the test-point insertion technique for the scan design. First, we consider the full-scan design environment, where the effects of inserting test points are examined globally. The goal is to establish as many scan paths and use as few test points as possible. The advantage of our technique in this application is the area overhead reduction. Next, we consider the partial-scan design environment. The objective is to break cycles without degrading the performance of the design. In partial-scan design, flip-flops are selected sequentially by the cycle-breaking algorithms [20], [24]. If timing constraints cannot be met after converting the currently selected flip-flop into a multiplexed scan flip-flop, the test-point insertion technique can be applied to move the test circuitry away from the critical path to avoid timing degradation.

This paper is organized as follows. Section II reviews previous works and defines terminology. Section III discusses our test-point insertion technique and its application in the full-scan design environment. Section IV shows a method to combine the conventional multiplexer insertion and the test-point insertion techniques for the timing-driven partial-scan design. Section V discusses the test methodology and ATPG. Section VI concludes and discusses possible future work.

II. Review and Terminology

To improve the ATPG fault coverage, a technique of test-point insertion was proposed in [17]. The idea was to insert a set of test cells into a circuit to improve the observability and controllability of some internal signals. The size of a test cell may be large, and the compound effect of adding such cells may result in significant area overhead (a test cell requires at least one flip-flop and two multiplexers). Our test point is simply a two-input AND gate or a two-input OR gate, and the purpose of inserting test points is to establish a scan chain, which in turn makes scanned flip-flops fully observable and controllable.

A method to create scan programmable logic array (PLA) designs with standard flip-flops and modified combinational part of the sequential circuit was described in [28]. Another method in which an easily testable state transition graph (STG) (test machine) is superimposed on the design STG (target machine) and then the composite STG is synthesized was proposed in [3]. The work in [7] and [30], presented algorithms to reduce scan overhead by attempting to merge scan multiplexers into the functional logic during logic synthesis. In [14], the concept of embedded scan was proposed, where attempts were made to embed the multiplexers for scan into the logic immediately preceding the scan flip-flops. In [21], a partition and resynthesis method is proposed to embed a test machine into the circuit under test such that all states are reachable and observable by predetermined sequences. It can also produce lower overhead testable circuits than the corresponding full-scan designs. However, the DFT design environment we are interested in is at the end of logic design flow, where major netlist changes by logic resynthesis are not allowed. In [25], a scan design methodology called *free scan* was proposed, where by setting appropriate values at primary inputs during the test mode, some combinational paths between flip-flops can be sensitized. Thus, a portion of the scan chain can be established without any DFT overhead. For highly sequential or pipelined circuits, however, it is quite possible that the primary inputs do not have sufficient influence on the internal logic to establish cost-free scan paths. In this paper, we further extend the concept of free scan and incorporate the test-point insertion technique to establish scan paths through functional logic.

We define some terminology used in the following discussion. A connection is specified by a pair of gates $[g_{\text{source}}, g_{\text{sink}}]$, where g_{source} is g_{sink}'s fan-in. A path is specified by a sequence of gates $[g_1, \cdots, g_k]$, where g_i is g_{i+1}'s fan-in. Side inputs of a path are a set of connections whose g_{sink}'s are on the path while g_{source}'s are not. Given a gate g and one of its fan-ins f, we say that the constant value v is a sensitizing value for the connection $[f, g]$ if setting f to v does not determine the value of g. On the other hand, if it does, v is called a controlling value.

For static timing analysis, we adopt the timing models used in [1], where the delay across a gate g is modeled linearly by its block delay, driving power, and load as follows:

$$delay(g) = block(g) + drive(g) \times load$$

where *load* is the total capacitive load driven by this gate. The parameters *block(g)* and *drive(g)* are given by the technology library.

The *arrival* time of a gate is defined as the latest time at which a signal switches from low to high or high to low. The *required* time of a gate is defined as the latest time at which a signal must switch from low to high or high to low in order to meet the timing constraint. By static timing analysis, the arrival time can be computed from inputs toward outputs in linear time in terms of the circuit size. Given the desired cycle time, the required time can also be computed from outputs toward inputs in linear time. We define the *slack* time of a gate as the difference between the required and the arrival times. A gate's slack time determines how much extra delay can be added to the gate's output without degrading the overall performance of the circuit. The slack times of all gates have to be nonnegative to guarantee correctness of the circuit for a given cycle time.

III. Test-Point Insertion

Suppose that a pair of flip-flops is connected by a combinational path. To include this path in a scan chain, all of the side inputs along this path have to be disabled. In other words, we have to set the values of side inputs on the path to sensitizing values. If a value of *zero* is desired at a connection c to disable it during the test mode, we insert a two-input AND gate at c with the test input T as one of its inputs. The value of T is assumed to be one in the normal mode and zero in the test mode. On the other hand, if a value of *one* is desired, we insert a two-input OR gate with T' as one of its inputs, where T' is the negation of T. To establish a scan path between two flip-flops may require more than one test point. The number of side inputs along a selected combinational path is an upper

bound on the requirement of the number of test points for establishing a scan path through the selected path.

In general, assigning a constant value at a connection (by inserting a test point) may potentially disable more than one side input because its value may imply values at other connections. As a result, to utilize this methodology efficiently, we analyze the circuit's topology and determine the global effect of inserting a particular test point. The objective is to decide at which connections test points should be inserted and what constant values they should have, so that we can establish as many scan paths as possible with as few test points as possible.

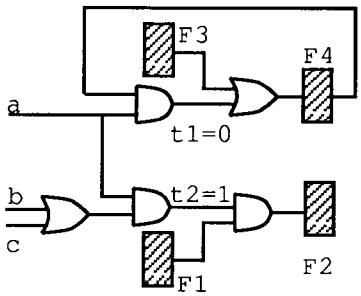


Fig. 2. Example of setting values for test points by assigning values at primary inputs.

A. Test-Point Insertion for Full-Scan Design

For the full-scan design, all the flip-flops have to be scanned. The goal here is to use the test-point-insertion technique to establish as many scan paths (through functional logic) with as few test points as possible, and then use the conventional scan conversion (multiplexer insertion) for the remaining unestablished scan paths in order to have a connected scan chain. We developed an algorithm called TPGREED for this purpose. TPGREED examines the combinational paths between flip-flops in the circuit and then, in a greedy way, sequentially inserts the test points with appropriate values. During the insertion, all the possible candidate locations are sorted according to their potential contributions in establishing scan paths. The details of the algorithm are as follows.

Given a sequential circuit, first we build a sparse matrix A, where the entry A_{ij} represents a set of combinational paths from flip-flop F_i to F_j. There might exist a large number of paths in the circuit, and in general, it is more costly to establish a scan path using a combinational path with a large number of side inputs. Our heuristic limits the number of considered paths and records only those paths with a number of side inputs smaller than a user-specified upper bound K_{bound} to save the computation time.

Given a combinational path p_k in A_{ij}, let $|p_k|$ denote the number of side inputs along this path. During the iteration of test-point insertion and the forward implication of the assigned constants, side inputs of p_k might have sensitizing, controlling, or unknown values. If there exists a side input having a controlling value, it will be impossible to build a scan path through it. We call such a path a *nullified* path and remove it from A_{ij}. On the other hand, if there is no side input with a controlling value, we use w_k to denote the number of side inputs with an unknown value. To make a path become a scan path, all the side inputs must have a sensitizing value. The gain of setting one of the side inputs to a sensitizing value is $1/w_k$. Notice that for each path p_k, the number $|p_k|$ does not change, while w_k is decreasing during the process. When w_k is reduced to zero, the path p_k successfully becomes a scan path.

Given a connection c, if we insert a test point with the value v at c, forward implication of v at c may imply new values v_i at some connections c_i in c's fan-out cone. We denote them as $\{(c_1, v_1), \cdots, (c_h, v_h)\}$. Some of the v_i's are controlling while others are sensitizing values. With these new implied constant values, the paths passing through $c, c_1, \cdots, c_{h-1}$ or c_h will be nullified because the data cannot go through these paths without being altered. Also, paths with c or c_i's as side inputs and v or v_i's as controlling values will be nullified. On the other hand, paths with c or c_i's as side inputs and v or v_i's as sensitizing values will have their w_k's reduced. We denote the set of such paths as S_c. Thus, the gain of inserting a test point with a value v on c is as follows:

$$\sum_{j=1}^{n}\left(\text{MAX}_{i=1}^{n}\left(\text{MAX}_{p_k\in A_{ij},\, p_k\in S_c}\frac{1}{w_k}\right)\right) \tag{1}$$

where n is the number of flip-flops. We sum the contribution of making flip-flops F_j as a part of the scan chain. Among all the paths in A_{ij}'s ending at a flip-flop F_j, we choose the maximal contribution instead of their summation because our objective is to establish exactly one path from some flip-flop to flip-flop F_j in the scan chain.

Based upon the cost function in (1), we can iteratively choose a connection and a value (c, v) with the highest gain as a test point and update the entries A_{ij} in the matrix A by removing the nullified paths. During the iteration of the greedy insertion process, if the scan path $F_i \rightarrow F_j$ is established, we record this path as part of the final scan chain. Since the scan chain has to be acyclic, we also remove some entries A_{pq} if adding the path $F_p \rightarrow F_q$ to the scan chain would result in a cycle. For example, consider a sequential circuit with four flip-flops F_1, F_2, F_3, and F_4. Suppose that we have already established a path $F_1 \rightarrow F_2$. Assume that adding a new test point will establish a scan path $F_2 \rightarrow F_3$. Besides recording this new scan path in the scan chain, we remove A_{31} because the path $F_3 \rightarrow F_1$ would result in a cycle $F_1 \rightarrow F_2 \rightarrow F_3 \rightarrow F_1$. Also, we have to remove all A_{i3}'s and A_{2j}'s since each flip-flop in the scan chain should have only one incoming and one outgoing edge.

B. Input Assignment

After performing the greedy insertion procedure described above, we know exactly at which connections (i.e., $c_1, \cdots, c_m$) test points should be inserted and also what values (i.e., $v_1, \cdots, v_m$) they should have. Before physically inserting AND (for value zero) or OR (for value one) gates, we make an attempt to induce as many of the proper values v_i as possible at the connections c_i by assigning appropriate values at the primary inputs to avoid inserting unnecessary test points. We call this input combination the *enabling vector*. For example, Fig. 2 shows a portion of a sequential circuit,

```
TPGREED(C, K_bound, gain_bound)
{
    A ← build_sparse_matrix(C,K_bound);
    Chain ← ∅;                                  /* initialize the scan chain */
    T ← ∅;                                      /* initialize the test points */
    loop {
        CAND ← candidate_connections(C,A);
        M    ← ∅;
        foreach candidate connection c_i in CAND
            gain_0 = analyze_gain(C,c_i,0);     /* try value 0 */
            add (c_i, 0, gain_0) in M;
            gain_1 = analyze_gain(C,c_i,1);     /* try value 1 */
            add (c_i, 1, gain_1) in M;
        end
        (c, v, gain) ← highest_gain(M);
        if ( gain ≥ gain_bound)
            add the test point (c, v) to T;
            update_sparse_matrix(C,A);          /* nullify some paths in A */
            if ( new scan paths established)
                update_scan_chain(Chain);
                update_sparse_matrix(C,A);
            end
        end
    } while (gain ≥ gain_bound)
    post_input_assignment(C,T);
    redundant_test_point_removal(C,T);
}
```

Fig. 3. Pseudocode of TPGREED (test insertion for full-scan design).

where a, b, and c are primary inputs. Assume that the greedy procedure decides to insert test points at connections $t1$ and $t2$ with values zero and one, respectively, to establish two scan paths $F_1 \rightarrow F_2$ and $F_3 \rightarrow F_4$. The desired values at $t1$ and $t2$ cannot be produced solely by applying an input vector (no input vector can produce $t1 = 0$ and $t2 = 1$). However, we can use appropriate values at primary inputs to produce one of the desired constants (e.g., $a = 0$, or $a = 1$ and $b = 1$, or $a = 1$ and $c = 1$) and use a test point to achieve another desired constant. In general, an optimization algorithm is required to decide the optimal input assignment in order to maximize the number of test points which can be set up freely. We adopt the algorithm described in [25] for this purpose.

C. Overall Algorithm

The overall algorithm is shown in Fig. 3. Besides the circuit, the user provides two extra parameters K_{bound} and $\text{gain}_{\text{bound}}$. The parameter K_{bound} is used to limit the number of side inputs for paths considered in establishing scan paths. During the iterative process, some scan paths will be established. Besides adding them as a portion of the scan chain, we also have to make sure that the subsequent insertion will not destroy the established scan paths. The procedure `candidate_connections` is used for this purpose.

The other parameter $\text{gain}_{\text{bound}}$ is used to terminate the algorithm when the highest gain computed by (1) for all candidate connections is smaller than $\text{gain}_{\text{bound}}$. This is used to avoid the case that the number of subsequently inserted test points is larger than the number of established scan paths.

In our current implementation, after a test point is inserted into a circuit, we recompute the gain of inserting a test point at each connection (`analyze_gain`) before inserting the next one. The procedure `post_input_assignment` determines the values at the primary inputs to reduce the number of required test points (as discussed in Section III-B). Moreover, since the insertion is iterative, the exact effect of inserting a test point cannot be computed until the end of the entire insertion process. As a result, there might exist redundant test points. At the last stage, we remove redundant test points.

D. Experimental Results

We tested the proposed test-point insertion method on a number of ISCAS89 and MCNC91 sequential benchmarks. All circuits are optimized by SIS `script.algebraic` script and mapped using technology libraries `nand − nor.genlib` and `mcnc_latch.genlib` for minimal area. In the current implementation, we can only handle primitive gates, including INV, AND, OR, NAND, and NOR gates. The extension to complex gates is not difficult, but it requires more programming efforts.

The results of test-point insertion are shown in Table I, where we report the number of flip-flops in the circuit (A), the number of test points inserted (B), the number of test points whose values can be set up by a proper enabling vector at the primary inputs (C), and the number of scan paths established (D). The CPU time is measured on a SUN SPARC 5 with 128 MB of memory. In our experiments, the parameters K_{bound} and $\text{gain}_{\text{bound}}$ are set to ten and 0.5, respectively. For example, we inserted 137 test points in circuit s15850 to establish 244 scan paths. Among the 137 test points, an enabling vector

TABLE I
EXPERIMENTAL RESULTS FOR ISCAS89 AND MCNC91 BENCHMARK CIRCUITS

circuit	# FF A	#insertion B	#free C	#scan paths D	area overhead reduction	CPU (sec)
s27	3	1	0	1	16.7%	0.7
s208	8	0	0	0	0.0%	1.0
s298	14	5	1	2	0.0%	2.6
s344	15	6	1	5	16.7%	3.8
s349	15	5	0	3	3.3%	6.0
s382	21	5	1	4	9.5%	4.5
s386	6	4	2	1	0.0%	2.2
s400	21	6	1	4	7.1%	6.5
s420	16	8	0	4	0.0%	5.6
s444	21	11	1	6	4.8%	9.2
s510	6	0	0	0	0.0%	4.7
s526	21	10	1	5	2.4%	6.8
s526n	21	12	1	6	2.4%	7.8
s641	17	8	6	7	35.3%	8.0
s713	17	8	5	7	32.4%	8.7
s820	5	0	0	0	0.0%	2.5
s832	5	4	2	1	0.0%	4.9
s838	32	0	0	0	0.0%	6.0
s1238	18	3	3	1	5.6%	6.5
s1488	6	0	0	0	0.0%	10.8
s1494	6	0	0	0	0.0%	8.4
s1196	18	5	4	2	8.3%	12.1
s1423	74	29	8	18	10.1%	93.3
s5378	152	28	3	62	32.6%	171.2
s9234	135	35	1	57	29.6%	296.7
s13207	453	120	2	196	30.2%	1151.7
s15850	540	137	2	244	32.7%	3907.4
s35932	1728	3	3	1440	83.3%	3019.6
s38417	1636	169	8	448	22.5%	6852.4
s38584	1294	164	1	1133	81.3%	15324.3
bigkey	224	115	3	112	25.0%	576.9
clma	33	0	0	0	0.0%	369.2
dsip	224	4	3	168	74.8%	52.6
mult32a	32	31	1	31	50.0%	24.1
mult32b	61	31	1	31	26.2%	26.0
sbc	27	12	7	4	5.6%	30.8

at the primary inputs can set up two of them. Therefore, the actual number of required test points is 135. The gate size of a multiplexer is typically twice as large as a test point (an AND or an OR gate). Also, the insertion of a multiplexer requires adding two connections (one from the scan enable signal and the other from the previous scan flip-flop) to the circuit, while insertion of a test point requires only adding one connection (one from the scan enable signal). Based on this observation, we assume that the area costs of inserting a multiplexer and a test point are two and one, respectively. The area overhead in conventional multiplexed scan design is 1080 (540×2), while the area overhead using the test-point insertion method is 135 (i.e., the cost for inserting 135 test points) plus 592 (i.e., the cost for converting the remaining 296 flip-flops into multiplexed scan flip-flops). In general, the reduction of area overhead is computed using the following formula:

$$1 - \frac{2(A-D)+(B-C)}{2A}.$$

If we use multiplexed flip-flops, the area overhead can be approximated as $2A$. In our method, the term $(B-C)$ represents the number of test points inserted, and the term $(A-D)$ represents the number of remaining flip-flops that require a multiplexer for each of them.

The amount of saving depends on the circuit's structure, the logic-synthesis algorithm, and the test-point-insertion algorithm. In the case of s35932, as much as 83% in saving of the area overhead can be achieved. The computation time for s38584 is quite high. This is because the number of paths considered by our algorithm in this case is 270 463. Possible ways to reduce the computation time are to decrease K_{bound} or to have an incremental algorithm for recomputing the gains as discussed in Section III-C. The test points are inserted iteratively. We show some intermediate results (the number of established scan paths versus the number of inserted test points) for circuits s15850 and s38417 in Fig. 4. As we can see, due to the greedy strategy, the gains of test-point insertion are high in the beginning and then gradually decrease.

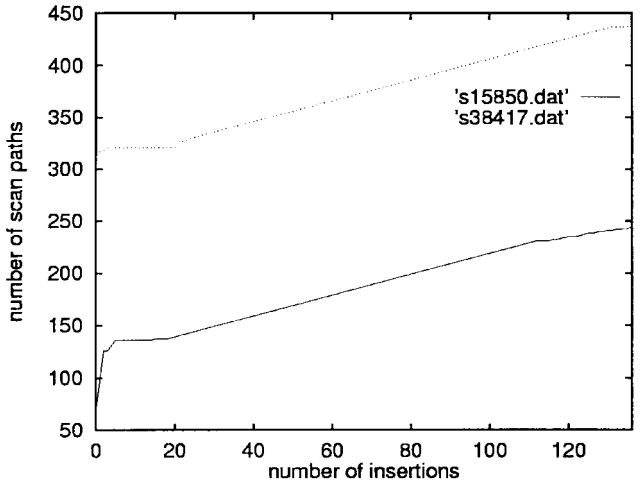


Fig. 4. Intermediate results on the circuits s15850 and s38417.

IV. Timing-Driven Scan Path Design by Test-Point Insertion

Due to its low overhead, the partial-scan design methodology has become popular as a major DFT technique for sequential circuits. A cycle-breaking strategy [11] and several associated algorithms [8], [24] for partial-scan designs were proposed in which the selection of flip-flops was aimed at breaking the cyclic structure of the circuit. Using this approach, a sequential ATPG program can achieve much better performance (in terms of both fault coverage and run times) for the resulting partial-scan circuits while still maintaining a relatively low area overhead. The cycle-breaking problem (i.e., the feedback vertex set problem) is NP-complete [16], [22]. Thus, various heuristics are used to select flip-flops [24].

Although partial scan has a lower overhead in terms of area, that may not be the case when we consider timing issues. In [20], a timing-driven partial-scan flip-flop selection algorithm was proposed. There, flip-flops with a slack time less than the propagation delay of a multiplexer are not allowed for selection, even if they have high gains for breaking cycles. As a result, the number of selected flip-flops for breaking cycles is usually larger than that selected by algorithms without considering timing issues. Moreover, there are circuits that have no cycle-breaking solutions without degrading the performance. We enhance the timing-driven partial-scan design methodology [20] by combining the cycle-breaking algorithm and the test-point-insertion method. As we show below, our objective is to establish scan paths by utilizing the functional logic through the nontiming-critical regions of the circuits.

If we scan a flip-flop by converting it to a multiplexed scan flip-flop, where the slack time of the flip-flop is less than the propagation delay of a multiplexer, such a conversion will result in timing degradation. However, by incorporating the test-point-insertion technique, we may still scan the flip-flop without any timing penalty. For example, Fig. 5(a) shows a portion of a sequential circuit, where the bold lines denote the critical path. If we scan the flip-flop F_2 by inserting a multiplexer directly behind F_2, we will increase the critical delay, which results in timing degradation, as shown in Fig. 5(b). However, there exists a combinational path from F_1 to F_2.

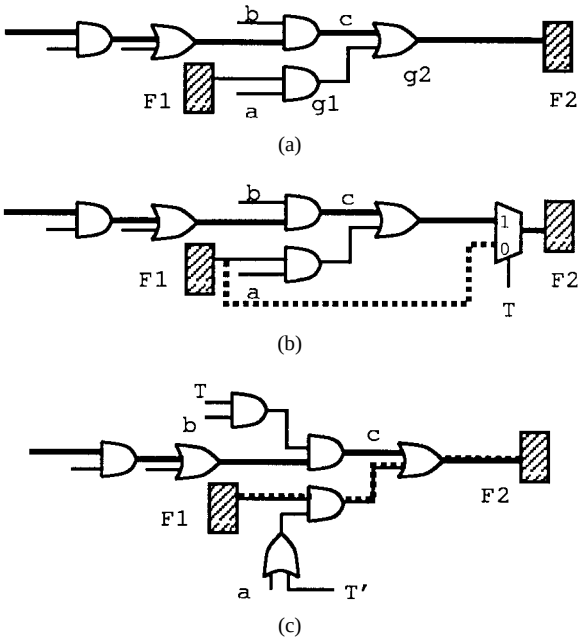


Fig. 5. Example of test-point insertion for timing-driven scan path design. The bold line represents the critical path, while the dotted line represents the scan path.

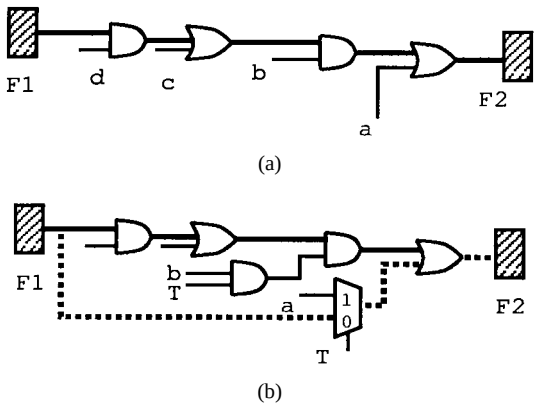


Fig. 6. Example of a multiplexer and a test-point insertion for timing-driven scan path design.

To make this combinational path $F_1 \rightarrow g1 \rightarrow g2 \rightarrow F_2$ a scan path, all of the side inputs a and c must have sensitizing values in the test mode. To assure this, we insert a test point (OR gate) on a. However, we cannot insert a test point at c without degrading the performance, since c is on a critical path. Instead, we can insert a test point (AND gate) at b, which in turn will induce a sensitizing value zero at c. The insertion of test points at a and b causes no timing violation and establishes a scan path from F_1 to F_2. The result is shown in Fig. 5(c).

The above transformation has two disadvantages. First, since the scan path is from F_1 to F_2, we have to scan F_1 too in order to have a connected scan chain. In the partial-scan environment, scanning F_1 might not help break cycles. Also, there is no guarantee that we can scan F_1 without timing degradation. Second, the number of side inputs requiring test points may be large, and the area overhead may be significant. For example, in Fig. 6(a), we have to insert four test points at a, b, c, and d in order to have a scan path from F_1 to F_2.

To overcome these disadvantages, we consider the insertion of multiplexers as well. However, the multiplexers need not be placed immediately behind the flip-flops. We insert them at connections with enough slack times. If necessary, we also insert test points at the corresponding side inputs to sensitize the scan path. For example, in Fig. 6(a), we insert a multiplexer at a and a test point at b to establish a scan path F_1 to F_2. The result is shown in Fig. 6(b). Notice that, using the above transformation, the predecessor of F_2 in the scan chain need not be F_1 and can be any other flip-flop.

A. Topological Feasibility Analysis

Given a flip-flop (selected by the cycle-breaking algorithm) for scan, we derive the following formula to check if we can scan it without timing degradation. For simplicity, we assume that a gate has one of the following five types: AND, OR, INV, FLIP-FLOP, or INPUT. The propagation delays of a multiplexer, a two-input AND, and a two-input OR are t_{mux}, t_{and}, and t_{or}, respectively. The slack time of a connection c_i is denoted as slack(c_i), while the gate type gate_type(c_i) is the gate type of c_i's source gate. The fan-in(c_i) denotes the set of fan-ins of c_i's source gate.

To scan a flip-flop, some connection in its fan-in cone has to carry the signal from the scan chain. Such a signal is denoted as scan_{in}. Also, to avoid timing degradation, some connections have to be set to one or zero. For example, to convert the circuit in Fig. 6(a) to (b), we assign a constant value zero at b and assign a as the scan_{in}. Thus, we can define cost(c_i, value) in the following as the area cost of assigning a connection c_i as value, where value is scan_{in}, one, or zero.

In (2), shown at the bottom of the page, if the slack time of c_i is greater than the propagation delay of a multiplexer, we simply insert a multiplexer, and the cost is the area of a multiplexer. Otherwise, we recursively check if we can use c_j (one of c_i's fan-ins) to be part of the scan path (i.e., assigning it as scan_{in}) and make other fan-ins c_k ($k \neq j$) have sensitizing values (i.e., assigning them to one or zero). Since there may exist multiple solutions, we choose the one with a minimal area overhead. Notice that if the gate type of c_i is FLIP-FLOP, the cost will be ∞ since it has no fan-ins to allow further recursion. Equations (3) and (4), shown at the bottom of the page, are defined similarly. For a flip-flop with fan-in connection c_i, if the cost function cost(c_i, scan_{in}) is less than ∞, we can scan this flip-flop without timing degradation.

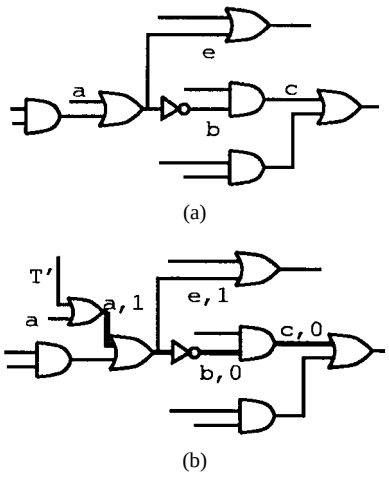


Fig. 7. Example for classification of constants.

The selection of scan flip-flops and the insertion of test points are done sequentially. It is important to keep track of the created scan paths and make sure that the subsequent insertions will not destroy the previous efforts. That is, there are some connections that have constant values associated with them due to the previous insertions. We classify them into two categories: *desired constants* and *side-effect constants*. For example, in Fig. 7(a), to make the connection c be zero in the

$$\text{cost}(c_i, \text{scan}_{\text{in}}) = \begin{cases} \text{area(MUX)} & \text{if slack}(c_i) > t_{\text{mux}} \\ \text{MIN}_{c_j \in \text{fan-in}(c_i)}(\text{cost}(c_j, \text{scan}_{\text{in}}) + \sum_{c_k \in \text{fan-in}(c_i), c_k \neq c_j}(\text{cost}(c_k, 1))) & \text{if gate\_type}(c_i) = \text{AND} \\ \text{MIN}_{c_j \in \text{fan-in}(c_i)}(\text{cost}(c_j, \text{scan}_{\text{in}}) + \sum_{c_k \in \text{fan-in}(c_i), c_k \neq c_j}(\text{cost}(c_k, 0))) & \text{if gate\_type}(c_i) = \text{OR} \\ \text{cost(fan-in}(c_i), \text{scan}_{\text{in}}) & \text{if gate\_type}(c_i) = \text{INV} \\ \infty & \text{if gate\_type}(c_i) = \text{FLIP-FLOP} \end{cases} \tag{2}$$

$$\text{cost}(c_i, 0) = \begin{cases} \text{area(AND)} & \text{if slack}(c_i) > t_{\text{and}} \\ \text{MIN}_{c_j \in \text{fan-in}(c_i)}(\text{cost}(c_j, 0)) & \text{if gate\_type}(c_i) = \text{AND} \\ \sum_{c_j \in \text{fan-in}(c_i)}(\text{cost}(c_j, 0)) & \text{if gate\_type}(c_i) = \text{OR} \\ \text{cost(fan-in}(c_i), 1) & \text{if gate\_type}(c_i) = \text{INV} \\ \infty & \text{if gate\_type}(c_i) = \text{FLIP-FLOP} \end{cases} \tag{3}$$

$$\text{cost}(c_i, 1) = \begin{cases} \text{area(OR)} & \text{if slack}(c_i) > t_{or} \\ \text{MIN}_{c_j \in \text{fan-in}(c_i)}(\text{cost}(c_j, 1)) & \text{if gate\_type}(c_i) = \text{OR} \\ \sum_{c_j \in \text{fan-in}(c_i)}(\text{cost}(c_j, 1)) & \text{if gate\_type}(c_i) = \text{AND} \\ \text{cost(fan-in}(c_i), 0) & \text{if gate\_type}(c_i) = \text{INV} \\ \infty & \text{if gate\_type}(c_i) = \text{FLIP-FLOP} \end{cases} \tag{4}$$

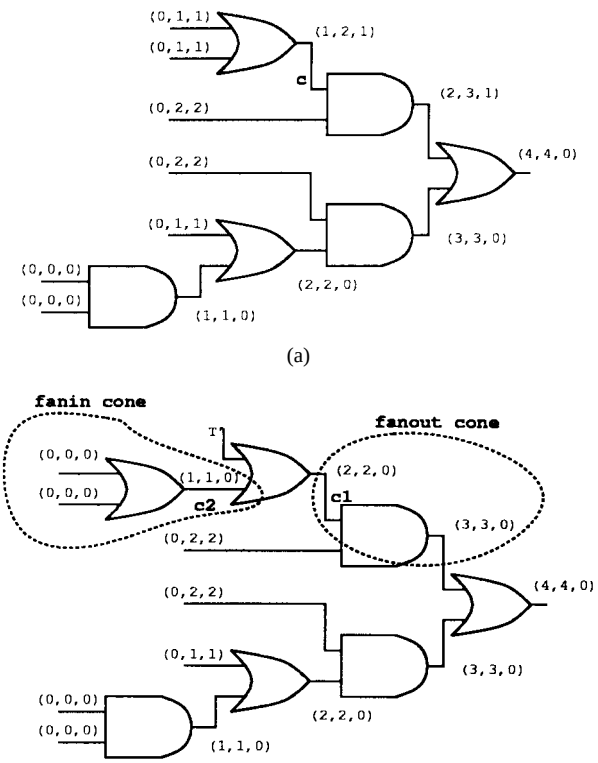


Fig. 8. Example for slack-time changes due to the test-point insertion. The triple represents the arrival time, required time, and slack time.

test mode, we can insert an AND gate at c (i.e., set c to zero), insert an AND gate at b (i.e., set b to zero), or insert an OR gate at a (i.e., set a to one). Assume that the slack times of b and c do not satisfy the requirement, while the slack time of a does. A test point can be inserted at a [Fig. 7(b)]. As a result, we have $a = 1$, $c = 1$, $b = 0$, and $c = 0$. Among them, $a = 1$, $b = 0$, and $c = 0$ are *desired constants* [as shown in the bold line of Fig. 7(b)] while $c = 1$ is a *side-effect constant*. To preserve the efforts of this insertion, the desired constants should not be changed by subsequent test-point insertions. On the other hand, we are free to change the constant values of side-effect constants.

There is a problem in using the recursive operations defined above. That is, when a test point is inserted at a connection c, the slack times of gates in c's fan-in or fan-out cone may be affected. Consequently, the function slack(c_i) is not a constant value but depends on the decisions made in the previous recursions. For example, in Fig. 8, if we insert a test point at c, it is split into two connections $c1$ and $c2$. The arrival time of some gates in the fan-out cone of $c1$ will be delayed, while the required time of some gates in the fan-in cone of $c2$ will be decreased. In other words, the slack time of these gates will decrease due to the insertion. As a result, taking such an update into account will result in a very complicated process. To simplify this problem, we restrict the application of recursion only to the *nonreconvergent fan-in regions* as defined below. With this restriction, we do not have to update the slack times during the recursion, and the result is guaranteed to be correct. Notice that since we restrict our solution space to the nonreconvergent fan-in regions, the obtained solution might be suboptimal.

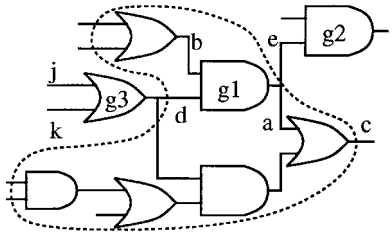


Fig. 9. Example of nonreconvergent fan-in region.

Definition 1: Given a connection c, we define its nonreconvergent fan-in region to be a set of connections in its fan-in cone, such that each connection has exactly one path to c.

See the circuit in Fig. 9 for illustration. The dotted region is the nonreconvergent fan-in region of the connection c. Notice that although the gate $g1$ has two fan-outs a and c, there is only one path from $g1$ to c passing through a. As a result, the connections a, b, and d are in the nonreconvergent fan-in region of c. On the other hand, since the gate $g3$ has two paths to c, the connections j and k are not in the nonreconvergent fan-in region of c.

Lemma 1: The nonreconvergent fan-in region of a connection c forms a tree rooted at c.

Proof: Let NRFR(c) denote the nonreconvergent fan-in region of c. It is obvious that $c \in$ NRFR(c), and we can prove by contradiction that NRFR(c) is connected. Since it is connected and each connection has exactly one path to c, we know that NRFR(c) must be a tree rooted at c. ■

Theorem 1: Given a connection c, when applying (2)–(4) for computing cost$(c, \text{scan}_{\text{in}})$, if we restrict the recursion only on the connections that are inside c's nonreconvergent fan-in region, multiple test-point insertions will not affect the slack time of a gate at the same time. As a result, we do not have to update the slack time during the recursions.

Proof: Assume that a multiplexer or test points are inserted at connections $c_1, \cdots, c_p$ using (2)–(4) recursively. First, we show that c_i will not be in c_j's fan-in cone, for $i \neq j$. Since the nonreconvergent fan-in region is a tree, if c_i is in c_j's fan-in cone, then c_i has no effect on c. This is because c_i has only one path to c, and this path is blocked by c_j. Consequently, it is redundant. But since the test points inserted during the recursion have to be irredundant, we conclude that c_i will not be in c_j's fan-in cone. Since none of the test points will be in the others' fan-in or fan-out cones, we do not have to update the slack time during the recursion. ■

The nonreconvergent fan-in region of a connection can be constructed in linear time in terms of its size by using breadth-first traversal from the connection toward the inputs.

B. Timing-Driven Partial-Scan Algorithm

The overall algorithm that integrates a cycle-breaking algorithm and our test-point insertion is shown in Fig. 10. The cycle-breaking algorithm we used is originally from [24] and then was modified in [20]. It consists of two major steps: graph reduction and heuristic selection. In the graph-reduction step, there are five operations. The first three (source operation, sink operation, self-loop operation) are exactly the same as those

```
timing_driven_partial_scan(C)
{
    G = build_dependence_graph(C);
    while ( cycle_exist(G) ) {
        ff = cycle_breaking(G);
        if( ff == NULL)
            break;
        points = test_point_insertion(ff,C);
        if( points ≠ NULL)
            update_network(C,points);   /* incremental static timing analysis */
            update_graph(G,ff);         /* graph reduction */
        else
            mark ff;                    /* cycle_breaking() won't select it */
        end
    }
    while ( cycle_exist(G) ) {
        [ff, points] = least_timing_violation(C,G);
        update_network(C,points);       /* incremental static timing analysis */
        update_graph(G,ff);             /* graph reduction */
    }
}
```

Fig. 10. Pseudocode for our timing-driven partial-scan design.

given in [24], while the last two reduction operations (unit-in and unit-out operation) are modified to take into account the slack times of the flip-flops. In the heuristic selection step, the algorithm chooses the one with maximal total number of fan-ins and fan-outs. For more details, refer to [20 and [24].

In our algorithm, the procedure `build_dependence_graph` examines the topological structure of the given circuit and builds the flip-flop connectivity graph excluding self-loops. Given a flip-flop *ff* selected by the procedure `cycle_breaking`, `test_point_insertion` performs test-point insertion analysis as described in (2)–(4) and tries to find a solution without degrading performance to scan *ff* in the nonreconvergent fan-in regions of *ff*. If such a solution exists, it always finds it and returns the set of test points. The algorithm then inserts appropriate MUX, AND, or OR gates into the circuit and performs an incremental static timing analysis for the next run. If no solutions without degrading performance exist, it returns "NULL," and the algorithm marks this flip-flop and instructs the procedure `cycle_breaking` to choose another one. This first *whole* loop continues until no cycles are left in the resulting graph or all flip-flops left have been marked. If cycles in G still exist, we know there are no solutions without degrading performance for this circuit. The algorithm then enters the second *while* loop and uses the procedure `least_timing_violation` iteratively to select a flip-flop with minimal timing degradation using the equations similar to those described in (2)–(4).

C. Experimental Results

We have implemented a system, named TPTIME, based on the SIS-1.2 package. The experimental results for a number of ISCAS89 and MCNC91 sequential benchmarks and the experimental setup are described as follows.

All of the circuits are first optimized by SIS `script.delay` script and then mapped for minimal delay. Since we target the timing-driven partial-scan design, it is more reasonable to optimize the original circuits for minimal delay. The longest delay of the optimized circuit is used as the circuit timing constraint. The technology library is used for mapping and it is based on the `nand−nor.genlib` and `mcnc_latch.genlib` from the SIS-1.2 package. We choose `nand − nor.genlib` because the current implementation can only handle primitive gates. To facilitate test-point insertion (adding AND, OR, and MUX gates into the circuit), we appended three entries in the technology library in order to perform static timing analysis in SIS-1.2. Each library cell's *drive(g)* is set to 0.2, and the input capacitive load is set to one. In other words, adding one fan-out will result in an extra 0.2-ns delay. For example, inserting a multiplexer at a connection will decrease its slack time by 2.2, since its block delay is 2.0 and the extra 0.2 is due to the fan-out of the multiplexer. For more details of the technology library specification, please refer to [1].

The statistics of the SIS-1.2 optimized circuits are shown in Table II. Notice that the test input T might have many fan-outs (connected to test points and multiplexers), and its capacitive load will be large to cause timing problems if we use the static timing analysis. However, in the mission (normal) mode, since the value of T is fixed to one, the paths from T to test points or multiplexers are false paths. So we should disable the paths originating from T during the static timing analysis.

Three different experiments were performed for each optimized circuit. First, we ran the Lee–Reddy [24] cycle-breaking (CB) algorithm, which does not take timing into account. Second, we ran the timing-driven cycle-breaking (TD-CB) algorithm shown in [20]. Third, we ran our program (TPTIME), which is based on the algorithm shown in Fig. 10. The results are shown in Table III. For each experiment, we report the number of selected flip-flops and the area and delay of the resulting circuit. Since no cycles exist for circuits s208, s420, s838, s1196, and s1238, no overhead is incurred for all three cases. We will exclude them from

TABLE II
STATISTICS ON ISCAS89 AND MCNC91 BENCHMARK CIRCUITS AFTER DELAY OPTIMIZATION. CIRCUITS MARKED "*" ARE OPTIMIZED WITHOUT USING FULL SIMPLIFY

circuit	#I	#O	#FF	area	delay (ns)
s27	4	1	3	52.0	9.4
s208	10	1	8	210.0	21.0
s298	3	6	14	354.0	15.4
s344	9	11	15	442.0	18.5
s349	9	11	15	437.0	19.4
s382	3	6	21	543.0	15.3
s386	7	7	6	348.0	16.8
s400	3	6	21	566.0	17.1
s420	18	1	16	475.0	21.1
s444	3	6	21	541.0	17.7
s510	19	7	6	532.0	16.5
s526	3	6	21	655.0	16.1
s526n	3	6	21	632.0	16.1
s641	35	23	17	528.0	21.3
s713	35	23	17	543.0	21.5
s820	18	19	5	711.0	20.6
s832	18	19	5	739.0	19.5
s838	34	1	32	985.0	31.2
s1196	14	14	18	1533.0	21.8
s1238	14	14	18	1715.0	29.9
s1423	17	5	74	2189.0	46.2
s1488	8	19	6	1329.0	20.4
s1494	8	19	6	1294.0	20.3
s5378	35	49	163	4286.0	26.9
s9234	36	39	135	3619.0	29.5
s13207	31	121	453	8511.0	35.8
s15850	14	87	540	13442.0	54.7
s35932	35	320	1728	40881.0	31.0
s38417*	28	106	1462	40611.0	42.4
s38584*	12	278	1449	36646.0	39.6
bigkey*	262	197	224	14461.0	27.8
clma*	382	82	33	46367.0	73.2
dsip*	228	197	224	8288.0	23.1
mult32a*	33	1	32	1655.0	95.8
mult32b*	32	1	61	1505.0	12.2
sbc*	40	56	27	1789.0	24.1

the following discussion. Without taking timing into account, the first method (CB) selected fewer flip-flops and had a smaller area overhead, but all of the tested circuits have timing degradation ranging from 2.2 to 16.4%. On the other hand, the TD-CB algorithm selected more flip-flops and had a larger area overhead, but the timing degradations for the tested circuits are smaller, ranging from 0.0 to 16.4%.

Our method (TPTIME) incorporates the test-point-insertion technique to scan a timing-critical flip-flop. Compared to CB, TPTIME has a larger area overhead due to the extra AND or OR gates. However, compared to TD-CB, since the number of selected flip-flops is in general smaller, the area overhead may be smaller. For example, for circuit s1423, the area overheads for designs produced by TPTIME and TD-CB are 6.1 and 11.2%, respectively. In terms of timing degradation, our method TPTIME obtains the best results among the three methods. In most cases, there is no timing degradation at all.

V. TEST STRATEGY

Testing of the test logic is addressed in [10], where the idea in [21] is extended to synthesize a fault-tolerant test machine such that when it merges with the machine under test, a fault in the test logic has predictable effect on the state transition of the composite machine. However, the target implementation is for two-level PLA circuits, and the effectiveness is shown by only a few small circuits with up to seven flip-flops.

In this section, we discuss how to test a circuit with a scan chain whose portion has been established by inserting test points. Similar to the test application procedure for conventional scan designs, the test contains two parts: one tests the scan chain and the other (i.e., ATPG phase) tests the functional logic.

Typically, to test the scan chain in a conventional scan design, applying a periodically alternating sequence (01 010 101$\cdots$) to the scan chain in the test mode will suffice. This is because a stuck-at-one (zero) fault that affects the functionality of a scan chain forces the scan-out data's having a tailing ones (zeroes) sequence [23]. By checking the existence of a tailing ones or zeroes sequence in the scan-out data, the scan chain can be tested.

However, the above sequence may not be sufficient to test the scan chain with test points inserted. Let the input and output of flip-flop F_i be z_i and y_i, respectively. Assume that a scan path exists from flip-flop F_i to flip-flop F_j. We have the following relation:

$$z_j = \begin{cases} y_i \text{ or } \bar{y}_i, & \text{if no faults occur} \\ 0 \text{ or } 1, & \text{if a fault occurs in a conventional scan design} \\ f(y_1, \cdots, y_k,), & \text{if a fault occurs in a scan design with test points inserted} \end{cases}$$

where k is the total number of flip-flops in the z_j's fan-in cone. That is, a fault in a scan design with test points inserted does not necessarily force the scan-out data's having a tailing ones or zeroes sequence. We give an example in Section V-A and discuss how to generate a more robust sequence to test the scan chain.

To test the functional logic, test vectors are generated by ATPG in the same way as in the conventional full- or partial-scan environment. When test vectors are scanned into flip-flops and test responses are scanned out from flip-flops in the test mode, we have to set a value of zero at the test input T and the enabling vector (see Section III-B) at the primary inputs in order to activate the desired values at the inserted test points so that the scan chain can be fully established.

In testing the scan chain, certain faults in the functional logic can also be tested without using conventional ATPG. We will address such cases in Section V-B. Section V-C discusses the testing of the extra test logic.

A. Alternating Zeroes and Ones Sequence

In the conventional scan design (using multiplexed D flip-flops), before testing the combinational portion of the circuit, sequential elements (including flip-flops and a scan chain) have to be tested first. To test flip-flops' functionality (i.e., to check if they can correctly latch the incoming data), a periodically alternating zeroes and ones sequence (01 010 101$\cdots$) containing both zero-to-one and one-to-zero transitions will suffice.

TABLE III
EXPERIMENTAL RESULTS ON ISCAS89 AND MCNC91 BENCHMARK CIRCUITS FOR TIMING-DRIVEN PARTIAL-SCAN DESIGN

	CB			TD-CB			TPTIME		
circuit	#FF	area	delay	#FF	area	delay	#FF	area	delay
s27	1	57.0 9.6%	10.2 8.5%	1	57.0 9.6%	10.2 8.5%	1	59.5 14.4%	9.8 4.3%
s208	0	210.0 0.0%	21.0 0.0%	0	210.0 0.0%	21.0 0.0%	0	210.0 0.0%	21.0 0.0%
s298	1	359.0 1.4%	17.6 14.3%	2	364.0 2.8%	17.2 11.7%	1	361.5 2.1%	15.4 0.0%
s344	5	467.0 7.7%	20.7 11.9%	5	467.0 7.7%	19.2 3.8%	5	472.0 6.8%	18.5 0.0%
s349	5	462.0 5.7%	21.6 11.3%	5	462.0 5.7%	20.4 5.2%	5	472.0 8.0%	19.4 0.0%
s382	9	588.0 8.3%	17.5 14.4%	9	588.0 8.3%	17.4 13.7%	9	615.5 13.4%	16.9 10.3%
s386	5	373.0 7.2%	18.4 9.5%	5	373.0 7.2%	18.4 9.5%	5	385.5 10.8%	16.8 0.0%
s400	8	606.0 7.1%	19.3 12.9%	9	611.0 8.0%	19.2 12.3%	9	628.5 11.0%	17.2 0.6%
s420	0	475.0 0.0%	21.0 0.0%	0	475.0 0.0%	21.0 0.0%	0	475.0 0.0%	21.1 0.0%
s444	9	586.0 8.3%	19.9 12.4%	9	586.0 8.3%	19.9 12.4%	9	606.0 12.0%	18.8 6.2%
s510	5	557.0 4.7%	18.3 10.9%	5	557.0 4.7%	18.3 10.9%	5	597.0 11.2%	17.1 3.6%
s526	3	670.0 2.3%	18.1 12.4%	5	680.0 3.8%	18.1 12.4%	3	675.0 3.1%	16.1 0.0%
s526n	3	647.0 2.4%	18.1 12.4%	5	657.0 4.0%	18.1 12.4%	4	659.5 4.4%	16.3 1.2%
s641	7	563.0 6.6%	23.5 10.3%	13	593.0 12.3%	23.1 8.5%	7	573.0 8.5%	21.3 0.0%
s713	7	578.0 6.4%	23.7 10.2%	11	598.0 10.1%	23.5 9.3%	7	588.0 8.3%	21.5 0.0%
s820	4	731.0 2.8%	22.8 10.7%	4	731.0 2.8%	22.8 10.7%	4	741.0 4.2%	20.6 0.0%
s832	4	759.0 2.7%	21.7 11.3%	4	759.0 2.7%	21.6 10.8%	4	769.0 4.1%	19.5 0.0%
s838	0	985.0 0.0%	31.2 0.0%	0	985.0 0.0%	31.2 0.0%	0	985.0 0.0%	31.2 0.0%
s1196	0	1533.0 0.0%	21.8 0.0%	0	1533.0 0.0%	21.8 0.0%	0	1533.0 0.0%	21.8 0.0%
s1238	0	1715.0 0.0%	29.9 0.0%	0	1715.0 0.0%	29.9 0.0%	0	1715.0 0.0%	29.9 0.0%
s1423	22	2299.0 5.0%	48.4 4.8%	49	2434.0 11.2%	48.4 4.8%	22	2321.5 6.1%	46.2 0.0%
s1488	5	1354.0 1.9%	22.6 10.8%	5	1354.0 1.9%	22.6 10.8%	5	1374.0 3.4%	21.5 5.4%
s1494	5	1319.0 1.9%	22.5 10.8%	5	1319.0 1.9%	22.5 10.8%	5	1336.5 3.3%	20.3 0.0%
s5378	29	4431.0 3.4%	29.0 7.8%	29	4431.0 3.4%	26.9 0.0%	29	4431.0 3.4%	26.9 0.0%
s9234	24	3739.0 3.3%	31.6 7.1%	25	3744.0 3.5%	29.5 0.0%	24	3754.0 3.7%	29.5 0.0%
s13207	41	8716.0 2.4%	38.0 6.1%	42	8721.0 2.5%	35.8 0.0%	42	8721.0 2.5%	35.8 0.0%
s15850	91	13897.0 3.4%	56.9 4.0%	91	13897.0 3.4%	55.9 2.2%	91	13909.5 3.5%	54.7 0.0%
s35932*	306	42411.0 3.7%	33.2 7.1%	306	42411.0 3.7%	31.0 0.0%	306	42411.0 3.7%	31.0 0.0%
s38417*	366	42441.0 4.5%	44.6 5.2%	388	42551.0 4.8%	44.6 5.2%	382	43351.0 6.7%	44.2 4.2%
s38584*	175	37521.0 2.4%	41.8 5.6%	233	37811.0 3.2%	41.4 4.5%	183	37808.5 3.2%	40.6 2.5%
bigkey*	112	15021.0 3.9%	30.0 7.9%	112	15021.0 3.9%	30.0 7.9%	112	15686.0 8.5%	28.7 3.2%
clam*	19	46462.0 0.2%	75.4 3.0%	29	46512.0 0.3%	75.3 2.9%	19	46472.0 0.2%	73.2 0.0%
dsip*	150	9038.0 9.0%	25.3 9.5%	180	9188.0 10.8%	25.3 9.5%	162	10555.5 27.4%	23.1 0.0%
mult32a*	16	1735.0 4.8%	97.9 2.2%	17	1740.0 5.1%	97.9 2.2%	16	1740.0 5.1%	95.8 0.0%
mult32b*	2	1515.0 0.6%	14.2 16.4%	22	1616.0 7.4%	14.2 16.4%	19	1647.5 9.5%	12.2 0.0%
sbc*	11	1844.0 3.1%	26.3 9.1%	12	1849.0 3.4%	24.1 0.0%	12	1849.0 3.4%	24.1 0.0%

Moreover, to test the integrity of a scan chain (i.e., to check if test patterns can be shifted serially into flip-flops), the same sequence can also serve the purpose. This is because a stuck-at-zero or a stuck-at-one fault in the scan chain forces the scan-out data's having a tailing ones or zeroes sequence, and thus causes discrepancies between the scan-in and scan-out data.

In a scan design with test points inserted, however, the above periodically alternating sequence may not suffice. For example, Fig. 11 shows two sequential circuits: one using conventional multiplexed scan design [Fig. 11(a)] and the other with one test point inserted [Fig. 11(b)]. This test point is used to assert a value of zero at the connection a in the test mode, so that the path from F_2 to F_3 can be sensitized. Note that in Fig. 11(a), a stuck-at-one fault at the connection between F_2 and F_3 forces the flip-flop F_3 to have a value of one permanently during scan, so that the scan-out data must have a tailing ones sequence. On the other hand, the scan path from F_2 to F_3 in Fig. 11(b) goes through the functional logic. If the connection b is stuck at zero, the path (from F_2 to F_3) is no longer sensitized. The input function of F_3 becomes $\overline{F}_1 \vee F_2$ instead of F_2. To detect the difference, we have to

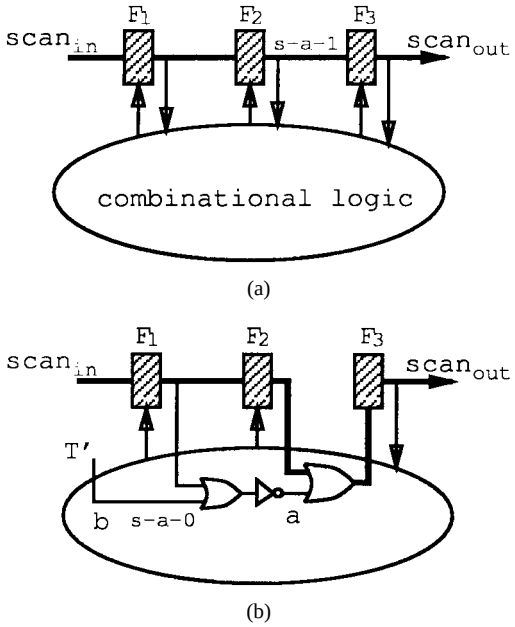


Fig. 11. Example of testing the scan chain.

introduce at flip-flops $[F_1, F_2]$ a pattern that is a min-term of the function $(\overline{F}_1 \vee F_2) \oplus F_2$ (i.e., $\overline{F}_1\overline{F}_2$). In other words, we need to have a pattern (00) at flip-flops $[F_1, F_2]$ in order to detect the fault. Since the periodically alternating sequence (01 010 101$\cdots$) does not contain the (00) pattern, in this case it cannot detect the stuck-at-zero fault at the connection b. We refer to the function $(\overline{F}_1 \vee F_2) \oplus F_2$ as the *error signature* for the stuck-at-zero fault at the connection b.

To overcome this problem, we should not apply a periodical sequence. Previously, the reason for having a periodical sequence was not only that it is sufficient to test the conventional scan chain but also that it is easier to apply and observe. Instead, in testing a scan chain with test points inserted, we apply a *pseudorandom sequence*, which can be generated efficiently. Before generating a pseudorandom sequence, a *fault list* containing all the stuck-at faults that affect the functionality of the scan chain has to be constructed. To check if a stuck-at fault affects the functionality of the scan chain during the test mode, we perform fault simulation to see if all scan paths are sensitized in the presence of this fault. If there exists a scan path that cannot be sensitized, we add this fault into the fault list. For example, in Fig. 11(b), the stuck-at-zero fault at the connection b corrupts the scan path from F_2 to F_3. It is added into the fault list. On the other hand, the stuck-at-one fault at the connection a does not affect the functionality of the scan chain. Therefore, it is not in the fault list.

Then, during the pseudorandom sequence generation, we perform fault simulation again for the faults in the fault list to check if they can be detected or not by the generated pseudorandom sequence. For example, when the sequence contains a subsequence (00), the stuck-at-zero fault at the connection b is tested and can be removed from the fault list. The total length of the generated sequence depends on the probability of being able to generate min-terms in the error signature functions of the faults in the fault list. Using the fault simulation, as soon as we cover all the faults in the fault list or the fault coverage is high enough, we terminate the sequence.

Besides using a pseudorandom sequence to test the scan chain, we also can employ a deterministic sequence generation process (similar to the conventional ATPG). The goal is to generate min-terms in the error signature functions. Integration of the deterministic sequence generation with pseudorandom sequence generation is desirable, but it will not be discussed here.

B. Testing Faults Without ATPG

In testing a scan chain, certain faults in the functional logic can be tested without using ATPG vectors. These are the faults that will affect the correctness of a scan chain. Fig. 12 shows a portion of a sequential circuit. A scan path from F_2 to F_1 is established through the functional logic by assigning zero to x_1 and one to x_2 (note that no test points are inserted; however, the scan path goes through the functional logic). If a fault in the fan-in cone of F_1 corrupts this scan path, then during the scan chain testing phase (Section V-A), the scan-in and scan-out data will be different and the fault can be detected. In Fig. 12, there are seven faults (a stuck at one and b, c, d stuck at zero or stuck at one) that will corrupt the scan path from F_2 to F_1. For example, if the connection a is stuck at one, the value of F_1 will always be one, regardless of the value of F_2. When we scan in the pseudorandom sequence, we will observe a sequence with tailing ones as the scan-out signal in the presence of this fault.

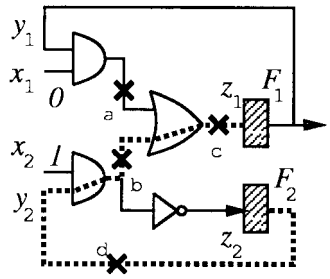


Fig. 12. Example to show that certain faults can be detected during scan chain correctness testing.

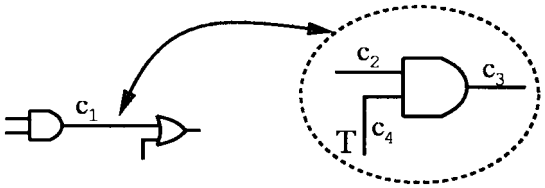


Fig. 13. Example of testing the stuck-at faults for a test point.

The faults that can be detected in this way depend on the circuit structure and the chosen scan paths. Integration of this feature more closely with the scan design is desirable but is not discussed here.

C. Testing Test Points

Next, we consider how to test the faults in the test logic (caused by the inserted test points). For simplicity, we assume that an AND gate is inserted, as shown in Fig. 13. Note that after the insertion of a test point, a connection c_1 in the original circuit corresponds now to three different connections c_2, c_3, and c_4. The purpose of inserting this test point is to have a value of zero at c_3 in the test mode ($T = 0$). Among the six stuck-at faults of these three connections, the faults c_3 stuck at one and c_4 stuck at one affect the correctness of the scan chain. Thus, they can be detected during the scan chain testing phase. For the remaining faults (i.e., c_2 stuck at zero, c_2 stuck at one, c_3 stuck at zero, and c_4 stuck at zero), since they do not affect the correctness of the scan chain, they should be targeted at the ATPG phase (assuming a value of one at T).

VI. Conclusions and Future Work

We proposed a low-overhead scan design methodology that employs the test-point-insertion technique to establish scan paths through the functional logic. Applications to both full- and partial-scan designs for reducing either area or timing overhead are addressed, and the experimental results demonstrate its usefulness.

We also show a new methodology to test the scan chain under this new method. This is important because the established scan chain goes through the functional logic. Moreover, by testing the scan chain, we show that certain faults that affect the correctness of the scan chain can be tested before the application of scan tests.

Future work includes performing further experiments to investigate the possible reduction in layout overhead using this methodology and integrating this method into the logic-synthesis process.

REFERENCES

[1] "SIS: A system for sequential circuit synthesis," University of California, Berkeley, Rep. M92/41, 1992.

[2] M. Abramovici, J. J. Kulikowski, and R. K. Roy, "The best flip-flops to scan," in *Proc. Int. Test Conf.*, 1991, pp. 166–173.

[3] V. D. Agrawal and K.-T. Cheng, "An architecture for the synthesis of testable finite state machines," in *Proc. Eur. Conf. Design Automation*, 1990, pp. 612–616.

[4] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin, "Designing circuits with partial scan," *IEEE Design Test Comput. Mag.*, pp. 8–15, Apr. 1988.

[5] V. D. Agrawal, S. K. Jain, and D. M. Singer, "Automation in design for testability," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1984, pp. 159–163.

[6] P. Ashar and S. Malik, "Implicit computation of minimum-cost feedback-vertex sets for partial scan and other applications," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 77–80.

[7] S. Bhatia and N. K. Jha, "Synthesis of sequential circuits for easy testability through performance-oriented parallel partial scan," in *Proc. IEEE Int. Conf. Computer Design*, 1993, pp. 151–154.

[8] S. Bhawmik and N. K. Jha, "PASCANT: A partial scan and test generation system," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1991, pp. 17.3.1–17.3.4.

[9] S. T. Chakradhar, A. Balakrishman, and V. D. Agrawal, "An exact algorithm for selecting partial scan flip-flops," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 81–86.

[10] S. T. Chakradhar, S. Kanjilal, and V. D. Agrawal, "Finite state machine synthesis with fault tolerant test function," *J. Electron. Testing*, pp. 57–69, Feb. 1993.

[11] K.-T. Cheng and V. D. Agrawal, "A partial scan method for sequential circuits with feedback," *IEEE Trans. Comput.*, vol. 39, pp. 544–548, Apr. 1990.

[12] V. Chickermane and J. H. Patel, "An optimization based approach to the partial scan design problem," in *Proc. Int. Test Conf.*, 1990, pp. 377–386.

[13] ———, "A fault oriented partial scan design approach," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1991, pp. 400–403.

[14] H. Cox, "On synthesizing circuits with implicit testability constraints," in *Proc. Int. Test Conf.*, 1994, pp. 989–998.

[15] 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.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness.* New York: Freeman, 1979.

[17] H. H. S. Gundlach and K. D. Muller-Glaser, "On automatic testpoint insertion in sequential circuits," in *Proc. Int. Test Conf.*, 1990, pp. 1072–1079.

[18] R. Gupta, R. Gupta, and M. A. Breuer, "The ballast methodology for structural partial scan design," *IEEE Trans. Comput.*, vol. 39, pp. 538–543, Apr. 1990.

[19] F. C. Hennie, "Fault detecting experiments for sequential circuits," in *Proc. 5th Ann. Symp. Switching Circuit Theory and Logic Design*, 1964, pp. 95–110.

[20] J.-Y. Jou and K.-T. Cheng, "Timing-driven partial scan," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1991, pp. 404–407.

[21] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "A partial and resynthesis approach to testable design of large circuits," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 1268–1276, Oct. 1995.

[22] R. M. Karp, "Reducibility between combinational problems," in *Symposium on the Complexity of Computer Computations*, R. R. Miller and J. W. Thatcher, Eds. New York: Plenum, 1972, pp. 118–125.

[23] S. Kundu, "On diagnosis of faults in a scan-chain," in *Proc. IEEE VLSI Test Symp.*, 1993, pp. 303–308.

[24] D. H. Lee and S. M. Reddy, "On determining scan flip-flops in partial-scan designs," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1990, pp. 322–325.

[25] C.-C. Lin, M. T.-C. Lee, M. Marek-Sadowska, and K.-C. Chen, "Cost-free scan: A low-overhead scan path design methodology," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1995, pp. 528–533.

[26] C.-C. Lin, M. Marek-Sadowska, K.-T. Cheng, and M. T.-C. Lee, "Test point insertion: Scan paths through combinational logic," in *Proc. ACM/IEEE Design Automation Conf.*, 1996, pp. 268–273.

[27] A. Miczo, *Digital Logic Testing and Simulation.* New York: Harper and Row, 1986.

[28] S. M. Reddy and R. Dandapani, "Scan design using standard flip-flops," *IEEE Design Test Comput. Mag.*, pp. 52–54, 1987.

[29] E. Trishler, "Incomplete scan path with an automation test generation approach," in *Proc. Int. Test Conf.*, 1980, pp. 153–162.

[30] B. Vinnakota and N. K. Jha, "Synthesis of sequential circuits for parallel scan," in *Proc. Eur. Conf. Design Automation*, 1992, pp. 366–370.

[31] 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.

Chih-Chang Lin (M'96) received the B.S. degree in computer science and information engineering from National Taiwan University, Taiwan, R.O.C., in 1987. He received the M.S. degree in computer science and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1991 and 1996, respectively.

He was a Software Engineer with Mentor Graphics Corp., San Jose, CA, developing logic synthesis and test synthesis tools. He is currently the Engineer Director of Verplex Systems, Inc., Santa Clara, CA, developing formal verification tools. His research interests include logic synthesis and circuit testing.

Malgorzata Marek-Sadowska (M'87–SM'95–F'97) received the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the Politechnika Warszawska, Warsaw, Poland, in 1971 and 1976, respectively.

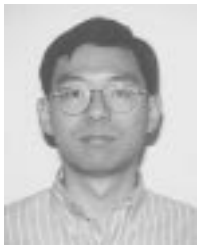
From 1976 to 1982, she was an Assistant Professor at the Institute of Electron Technology at the Politechnika Warszawska. She was a Visiting Professor in the Electrical Engineering Department of the University of California, Berkeley, from 1979 to 1980. She became a Research Engineer at the Electronics Research Laboratory in 1979 and continued there until 1990, when she joined the Department of Electrical and Computer Engineering at the University of California, Santa Barbara, as a Professor. Her research interests are in the area of computer-aided design with an emphasis on layout and logic synthesis of very-large-scale-integration circuits and systems. She has been a member of numerous technical committees, including the Technical Committee of the International Conference on Computer Aided Design, the Technical Committee of the Design Automation Conference, and the Technical Committee International Workshop on Placement and Routing. She has been Associate Editor of the *Journal of Circuits, Systems and Computers* since 1990 and is a reviewer for numerous technical journals.

Prof. Marek-Sadowska was Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 1989 to 1993 and Editor-in-Chief from 1993 to 1995.

Kwang-Ting Cheng (S'88–M'88) received the B.S. degree in electrical engineering from National Taiwan University, Taiwan, R.O.C., in 1983 and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1988.

From 1988 to 1993, he was with AT&T Bell Laboratories, Murray Hill, NJ. He joined the Faculty of the University of California, Santa Barbara, in 1993, where he is currently Professor of electrical and computer engineering. His current research interests include very-large-scale-integration testing, design synthesis, and design verification. He has published more than 120 technical papers, coauthored two books, and received six U.S. patents in these areas. He also has been working closely with U.S. industry on projects in these areas. He is a member of the editorial board of the *Journal of Electronic Testing: Theory and Application.* He has served on the technical program committees for several international conferences on computer-aided design and testing.

Prof. Cheng received the Best Paper Award at the 1994 Design Automation Conference and the 1987 AT&T Conference on Electronic Testing. He is a member of the editorial boards of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and IEEE DESIGN AND TEST OF COMPUTERS MAGAZINE. He has been General Chair and Program Chair of the IEEE International Test Synthesis Workshop.



Mike Tien-Chien Lee (M'94) received the B.S. degree in computer science from National Taiwan University, Taiwan, R.O.C., in 1987 and the M.S. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 1991 and 1993, respectively.

He was on the Research Staff at Fujitsu Laboratories of America from 1994 to 1996, conducting research on low-power design, embedded system design, high-level synthesis, and test synthesis. Currently, he is the Project Leader on deep-submicrometer timing optimization focusing on synthesis and layout integration, a switch-level design validation. He was a Consulting Researcher at the Center of Reliable Computing, Stanford University, Stanford, CA, in 1994. He is the author of *High-Level Test Synthesis of Digital VLSI Circuit*.

Dr. Lee has been on the program committees of the IEEE Pacific Northwest Test Workshop, IEEE International Test Synthesis Workshop, and IEEE International High Level Design Validation and Test Workshop. He received Best Paper Awards at ASP-DAC'95 and DAC'96.