Hardware-Optimal Test Register Insertion

Albrecht P. Stroele, *Associate Member, IEEE,* and Hans-Joachim Wunderlich, *Associate Member, IEEE*

***Abstract*—Implementing a built-in self-test by a "test per clock" scheme offers advantages concerning fault coverage, detection of delay faults, and test application time. Such a scheme is implemented by test registers, for instance built-in logic block observers (BILBO's) and concurrent BILBO's (CBILBO's), which are inserted into the circuit structure at appropriate places. An algorithm is presented which is able to find the cost optimal placement of test registers for nearly all the ISCAS'89 sequential benchmark circuits, and a suboptimal solution with slightly higher costs is obtained for all the circuits within a few minutes of computing time. The algorithm can also be applied to the Minimum Feedback Vertex Set problem in partial scan design, and an optimal solution is found for all the benchmark circuits.**

The provably optimal solutions for the benchmark circuits mainly use CBILBO's which can simultaneously generate test patterns and compact test responses. Hence, test scheduling is not required, test control is simplified, and test application time is reduced.

***Index Terms*—BILBO, built-in self-test, CBILBO, test register insertion.**

I. Introduction

A. Self-Testable Structures

BUILT-IN self-test (BIST) is one of the most important techniques for testing large and complex systems. Test registers are added at the primary inputs and outputs of a circuit, and some additional test hardware is inserted into the circuit. In a "test per scan" scheme, test registers feed and evaluate a (partial) scan path (see Fig. 1).

It has been shown independently by several authors that breaking all cycles in the circuit structure bounds the length of the required test sequences to the sequential depth of the circuit [1]–[5]. To keep the hardware overhead low, the number of flip-flops that are integrated into the partial scan path in order to break all cycles should be as small as possible, and the NP-complete minimum feedback vertex set (MFVS) problem has to be solved [6]. Chakradhar, Balakrishnan, and Agrawal presented an algorithm for computing the MFVS exactly using a branch and bound technique [7].

In a "test per clock" scheme, some system registers are enhanced such that in special test modes they generate patterns or compact test responses. Examples of these multimode test registers are BILBO and GURT [8], [9]. A "test per clock" scheme has advantages with respect to test application time, delay testing, and defect coverage, but it often requires a higher hardware overhead than the "test per scan" scheme.

Manuscript October 10, 1995. This work was supported in part by the DFG under Grants Schm 623/5-1 and Wu 245/1-1. This paper was recommended by Associate Editor T. Cheng.

A. P. Stroele is with the Institute of Computer Design and Fault Tolerance, University of Karlsruhe, Germany.

H.-J. Wunderlich is with the Computer Architecture Laboratory, University of Stuttgart, Germany.

Publisher Item Identifier S 0278-0070(98)05021-0.

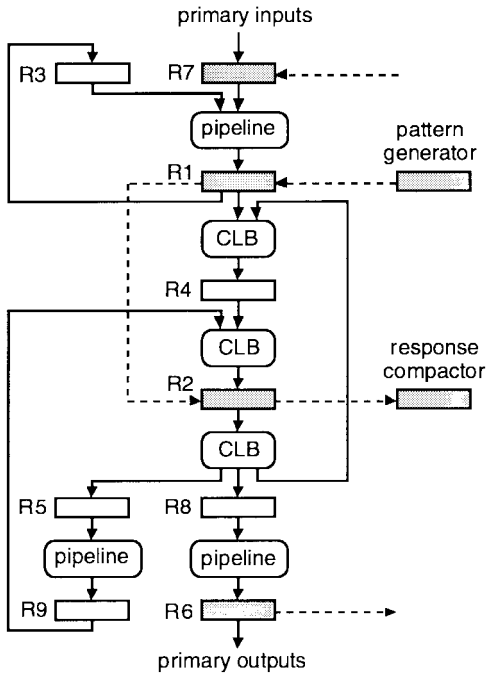

Fig. 1. "Test per scan" scheme applied to a circuit with registers R1 ··· R9, combinational logic blocks (CLB), and pipeline structures.

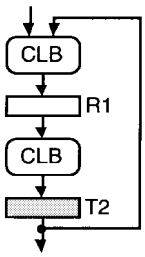

Fig. 2. Part of a data path with register R1 and test register T2.

In order to obtain self-testable circuits, test registers must be placed at appropriate positions [10]–[13]. The circuit structure obtained from breaking all cycles, however, is not *a priori* suited to a "test per clock" scheme since during self-testing some test registers may have to generate patterns and compact test responses concurrently (e.g., test register T2 in Fig. 2).

In particular with a circular self-test path, the intermediate results of response compaction are used as test patterns. This leads to a sufficiently high fault coverage if the circuit is not random pattern resistant and all the states required for testing are reachable [14], [15]. But in general, the patterns are not exhaustive, (weighted) random, or deterministic. Then, BILBO's have to be employed, and at least two of them are required in each cycle. In Fig. 2, register R1 must also be enhanced to a BILBO register.

0278–0070/98$10.00 © 1998 IEEE

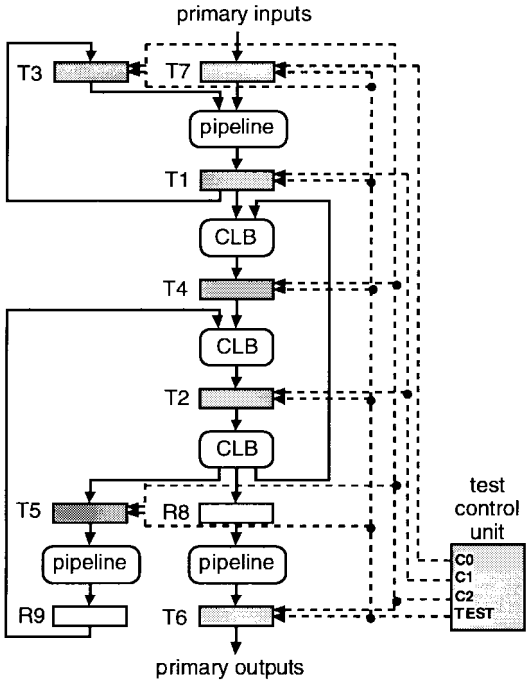

Fig. 3. "Test per clock" scheme.

Using BILBO's, it is possible to segment the circuit into a set of subcircuits that are completely bounded by test registers (see Fig. 3). For testing a portion of the circuit, at least one test register must collect test responses. Thus, the smallest region that can be tested independently (test unit) consists of one test register that can be configured as a multiple input signature register, the block of logic connected to the inputs of this register, and a set of test registers to generate test patterns for the inputs of the block (cf., [11], [16], and [17]). In this way, every test unit $u(T_i)$ is uniquely determined by the test register T_i at its outputs. In Fig. 3, the test unit $u(T_1)$ includes test register T_1 (response compaction), the pipeline structure connected to the inputs of T_1, and the test registers T_3 and T_7 (pattern generation). Since a BILBO register cannot generate pseudorandom patterns and compact test responses simultaneously, some subcircuits cannot be tested concurrently, e.g., $u(T_1)$ and $u(T_4)$. Test application must be scheduled such that these conflicts of resources are avoided.

As a test register can generate patterns, analyze signatures, and operate in system mode, it requires at least two control signals. A test control unit is needed that supplies these control signals for each test register according to the test schedule [18].

B. Multimode Test Registers

The general "test per clock" scheme requires at least two multimode test registers to be placed in each cycle of the circuit structure. In particular, problems arise when a register feeds itself through combinational logic (self-adjacent register [19]), e.g., register R3 in Fig. 4. Here register R3 must be enhanced to a BILBO register T3, and an additional test register T4 of the BILBO type that is transparent in normal mode must be inserted into the feedback path. In [20], transparent test registers as shown in Fig. 5 are called "fences."

To reduce the additional overhead of transparent test registers, some authors propose using three latches for each bit of the test register such that it can generate patterns and compact test responses independently. These test registers are called L3-BILBO's [21], [22] or CBILBO's [23]. In Fig. 4, it is sufficient to enhance R3 to a CBILBO, and an additional test register in the feedback path is not required.

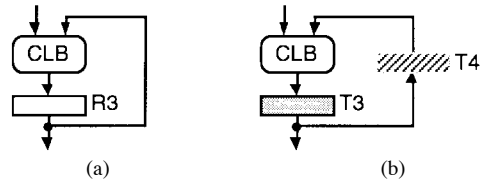

Fig. 4. (a) Part of a data path with self-adjacent register R3. (b) Self-testable structure with BILBO register T3 and transparent BILBO register T4.

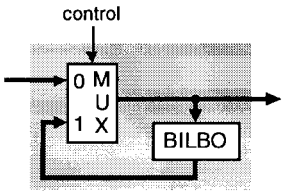

Fig. 5. Structure of a transparent test register.

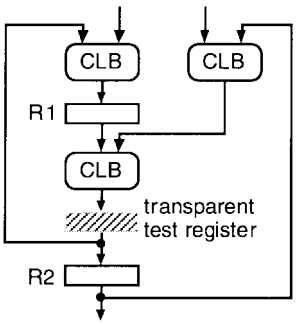

Fig. 6. Circuit with two connected self-loops.

At register transfer level, registers can be enhanced to BILBO's or CBILBO's, and additional BILBO's and CBILBO's that are transparent in normal mode can be inserted into arbitrary lines. After test registers have been inserted, each cycle of the circuit structure must contain at least one CBILBO or two BILBO's. Regarding hardware costs, CBILBO's are more expensive than BILBO's, and inserting a whole transparent test register is more expensive than enhancing an existing register. Nevertheless, in some situations transparent test registers are useful. In order to make the circuit of Fig. 6 self-testable, both registers R1 and R2 can be enhanced to CBILBO's. But the same goal can also be achieved by inserting just one transparent CBILBO.

In this paper, we present an exact algorithm that determines an appropriate placement of test registers and selects their types such that the total hardware cost of all the built-in test registers is minimum. As a special case, it also solves the MFVS problem for partial scan and the "test per scan" scheme.

C. Optimal Test Register Placement

During "top-down" design or synthesis, test registers are usually inserted at register transfer level [12], [24]–[28]. This way, hierarchy can be exploited, and the underlying complex optimization problem can be solved efficiently. But at RT-level not all of the structural knowledge is yet available, and much

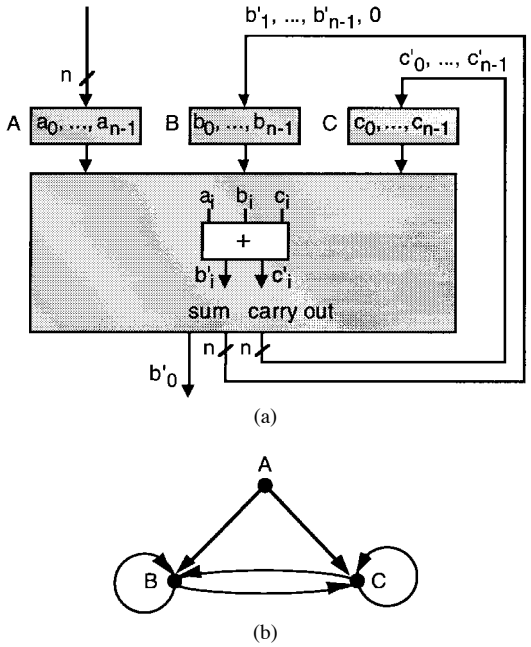

Fig. 7. CSA: (a) data path and (b) its register graph.

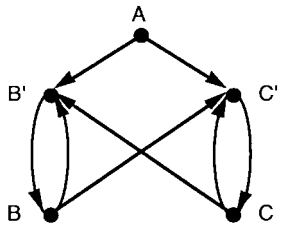

Fig. 8. Test register graph including transparent test registers B' and C'.

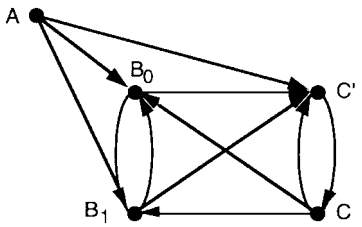

Fig. 9. Test register graph after inserting test register C' and splitting register B.

better solutions are possible if gate level information is used. As the gate level description of a system has much higher complexity than the RT-level description, highly efficient algorithms for the insertion of BIST cells (1-bit elements of test registers) are required. In this paper, a hardware-optimal algorithm is presented which finds a BIST solution with minimum transistor overhead for nearly all the ISCAS'89 benchmark circuits [29].

The paper is organized as follows. In Section II, we discuss the tradeoffs between test register insertion at RT-level and at gate level. Practical examples show significant savings if the more complex gate level description is used. In Section III, test cell insertion at gate level is formulated as an optimization problem in graph theory. In Section IV, a branch and bound algorithm is presented which exploits an efficient divide and conquer approach. Experimental results are presented in Section V. Surprisingly, they show that BIST structures with minimum hardware overhead often do not use BILBO-type registers but mainly CBILBO's. This also simplifies test scheduling and test control and has strong impact on the appropriate target structures of self-testable systems aimed at by synthesis algorithms. These consequences are discussed in Section VI.

II. Test Register Insertion: Gate Level Versus RT Level

Often, for a gate level circuit there is a variety of corresponding register graphs. The register graph is determined by the way the flip-flops are partitioned and assembled to registers. The register configuration of the system mode is not always optimal for testing. As an example, Fig. 7 shows a carry save adder (CSA) and its register graph. Such a circuit is often used for implementing sequential multiplication [30].

The register graph contains two self-loops, and two additional transparent test registers B' and C' of length n are required for making it self-testable. The resulting graph is shown in Fig. 8.

But looking at this circuit in more detail it is found that the transparent test register B' is superfluous. At gate level, the topology of the storage elements can be represented by a so-called S-graph whose vertices correspond to flip-flops and whose edges indicate combinational paths between flip-flops. Flip-flop b_i just feeds flip-flops b_{i-1} and c_i, and flip-flop c_i feeds b_{i-1} and c_i; so, the S-graph contains self-loops for the flip-flops c_i, but not for the flip-flops b_i. Hence, it is more appropriate to split register B into two registers of length $n/2$ during testing, namely $B_0 := (b_0, b_2, \cdots, b_{n-2})$ and $B_1 := (b_1, b_3, \cdots, b_{n-1})$. Then, the register graph contains only one self-loop, and only one additional test register C' is required (Fig. 9).

This example shows significant hardware savings if the test registers are determined at gate level, but it also shows the increase of the problem complexity. BIST insertion into the register graph of Fig. 7 is trivial, but the corresponding gate level description of the circuit has at least $5n$ nodes, is not very regular, and requires a very sophisticated algorithm. If test cells are inserted at gate level, they have to be assembled to test registers with appropriate feedback connections. Objectives of this clustering may be a minimal routing and area overhead, minimal test lengths, or an optimal test schedule [3], [31], [32].

III. Problem Statement and Simplification

At gate level, a circuit is usually modeled by a directed graph $G = (V, E)$ with nodes V and edges $E \subset V \times V$. The nodes $V = I \cup O \cup V_C \cup V_S$ represent primary inputs (I), primary outputs (O), gates (V_C), and flip-flops (V_S). The edges describe the lines that connect these elements. At the primary inputs and outputs, test registers must be inserted in any case and are not considered explicitly. The placement of test cells is described by labels l that have the following meaning:

for $v \in V_S$:

$$l(v) := \begin{cases} 0 & \text{if flip-flop } v \text{ is not modified;} \\ 1 & \text{if flip-flop } v \text{ is enhanced to a BILBO cell;} \\ 2 & \text{if flip-flop } v \text{ is enhanced to a CBILBO cell} \end{cases}$$

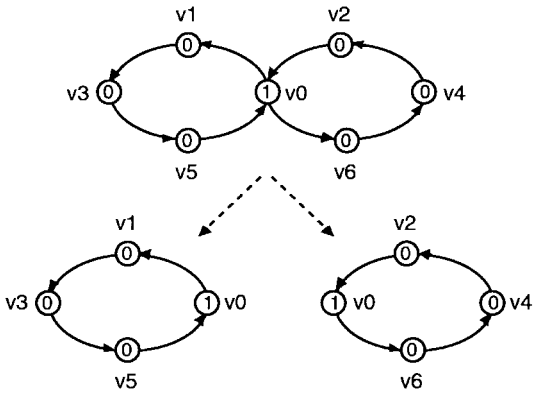

Fig. 10. Labeled graph (top) and its TCC's (bottom).

for $v \in V_C$:

$$l(v) := \begin{cases} 0 & \text{if gate } v \text{ is not modified;} \\ 1 & \text{if a transparent BILBO cell is inserted at the output of } v; \\ 2 & \text{if a transparent CBILBO cell is inserted at the output of } v. \end{cases}$$

In order to find an optimal placement, the following problem has to be solved:

Minimum cost placement (MCP)

Given: Circuit graph $G = (V, E)$, costs of enhancing a flip-flop to a BILBO cell or a CBILBO cell, costs of inserting a transparent BILBO cell or a transparent CBILBO cell at the output of a gate.

Find: Labeling l such that $\Sigma_{v \in Z}\, l(v) \geq 2$ is true for each cycle Z of G, and the total cost associated with the labeling l is minimum

Problem MCP includes the problem of selecting flip-flops for partial scan as a special case (costs of CBILBO cells smaller than costs of BILBO cells), and it is NP-complete. The authors of [7] developed an exact algorithm for selecting partial scan flip-flops. But their graph transformations and partitioning using strongly connected components cannot be applied here.

The presented algorithm uses a preprocessing step similar to [5] where iteratively all the nodes without predecessors or without successors are removed since they cannot be part of any cycle. Moreover, as the transistor cost of a transparent test cell is the same for all the gate inputs and outputs, it is sufficient to consider the outputs of combinational fanout-free regions for possible insertion of transparent test cells.

Every cycle of a directed graph G is completely included in a strongly connected component (SCC) of G [5]. Hence, all SCC's of G can be considered separately. A minimum cost placement for G consists of minimum cost placements for all the SCC's of G. The SCC's of G are extracted by deleting edges (v_i, v_j) where v_i and v_j belong to different SCC's.

IV. Branch and Bound Search

At the beginning, the labels of all the nodes of G are reset, $l(v) = 0$ for all $v \in V$, corresponding to the circuit without any test cells. Then in each step a node v_i with $l(v_i) = 0$ is selected and its label is set to 1 or 2. If the assigned label makes $\sum_{v \in V} l(v) \geq 2$ true for a cycle Z of G, this cycle need not be considered any more. If there are still cycles and two of them share a node v_i with $l(v_i) = 0$, then both cycles have to be considered together since a test cell placed at v_i would have an impact on both cycles. On the other hand, if it is possible to partition the cycles into subsets such that cycles from different subsets do not share any nodes with label 0, then these subsets can be considered separately. In this way we get subproblems that can be solved independently, and the optimal solution can be found much more efficiently. These subsets of cycles are called T-connected components (TCC's, connected during test application).

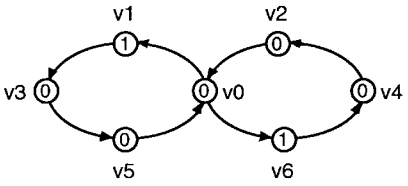

Fig. 11. Labeled graph that is composed of a single TCC.

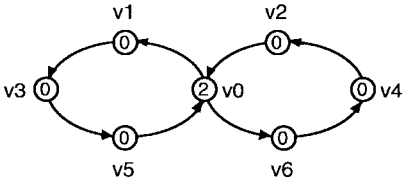

Fig. 12. Labeled graph with an empty set of TCC's.

Definition: A T-connected component of a graph G is a minimal subgraph of G with the following characteristics:

- A TCC includes at least one cycle Z of G with $\sum_{v \in V} l(v) < 2$ (i.e. a cycle that contains no nodes with CBILBO cells and at most one node with a BILBO cell).
- If G includes two cycles Z and Z' with $\sum_{v \in V} l(v) < 2$ and $\sum_{v' \in V'} l(v') < 2$, and if these cycles share at least one node with label 0, then all the nodes and edges of both cycles belong to the same TCC.

The labeled graph of Fig. 10 has two TCC's. The graph of Fig. 11, however, cannot be divided since it is composed of a single TCC. Fig. 12 shows a graph that does not contain any TCC's.

The TCC's of a graph are unique. They describe a partition of the set of nodes with label 0 that are included in at least one cycle Z with $\sum_{v \in V} l(v) < 2$. Exactly these nodes are the candidates for the insertion of further test cells. The TCC's do not contain any node with label 2. Nodes with label 1 may be included in more than one TCC (for this purpose the node is copied, see Fig. 10). If a graph does not contain any nonzero labels, its TCC's and its SCC's agree.

The problem "minimum cost placement" is solved by a depth first search algorithm building a tree whose nodes represent TCC's. The root node contains an SCC where all the nodes are labeled with 0 and for each node v the set of

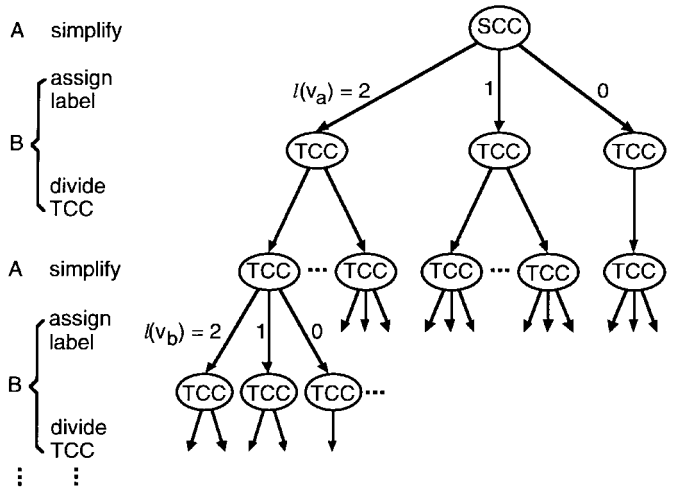

Fig. 13. Search tree for a single SCC.

admissible labels $L(v)$ contains all the possible labels 0, 1, and 2. In the first level of the search tree, a node v_a is labeled using all the admissible labels from $L(v_a)$. The assignment of a label can divide the SCC into smaller TCC's (second level). Next, a second node v_b is labeled (third level) and so on. Fig. 13 shows an example.

The search is implemented by two alternating procedures. Procedure A restricts the admissible labels for the nodes and tries to simplify the graph by local transformations. Procedure B selects a node v that has not yet been considered and tries all the admissible values $l(v) \in L(v)$. When a specific label is assigned to node v, the set of admissible labels $L(v)$ is restricted to this single label. After the assignment, procedure B tries to divide the TCC into smaller ones. Then procedure A is called for each of the resulting (smaller) TCC's.

A. Procedure A

In procedure A, the following rules are applied until no more changes are feasible. An optimal solution for the modified graph is still an optimal one for the original graph:

Rule (i): *If* v is a combinational node, $v \notin V_s$, and the number of incoming or outgoing edges is 1, $\text{indeg}(v) = 1$ or $\text{outdeg}(v) = 1$, and v is not part of a cycle with only one or two nodes, *then* ignore node v, i.e. remove v and replace every pair of edges $(w', v), (v, w'')$ by an edge (w', w'').

Rule (ii): *If* v is a sequential node, $v \in V_s$, with $l(v) = 0$, and the number of incoming and outgoing edges is 1, $\text{indeg}(v) = \text{outdeg}(v) = 1$, and its direct predecessor and successor are two sequential nodes, and v is not part of a cycle with only one or two nodes, *then* ignore node v.

By local inspection some labels of nodes can be excluded as they cannot lead to an optimal solution:

Rule (iii): *If* there is a self-loop (v, v), *then* the only admissible label for node v is 2, $L(v) := L(v) \setminus \{0, 1\}$.

Rule (iv): *if* there is a cycle with two nodes, (v, v', v), and $l(v') = 1$, *then* label 0 is not admissible for node v, $L(v) := L(v) \setminus \{0\}$.

As far as cycle breaking is concerned, increasing the label of a node v from 0 to 1 is equivalent to adding 1 to all the direct predecessors of v that are labeled with 0 or 1. It is also equivalent to adding 1 to all the direct successors of v that are labeled with 0 or 1. Consequently, for a minimum cost labeling the node v must not get the label 1 if one of the equivalent options is less expensive.

We define $cost1(v)$ and $cost2(v)$ as the costs for labels $l(v) = 1$ and $l(v) = 2$, respectively, and $in1(v)$ as the cost for adding 1 to all the direct predecessors of v that are labeled with 0 or 1. $out1(v)$ is the cost for adding 1 to all the direct successors of v that are labeled with 0 or 1. An admissible labeling is restricted by the following rule:

Rule (v): *If* $l(v) = 0$ and $cost1(v) \geq \min\{in1(v), out1(v)\}$, *then* label 1 is not admissible for node v, $L(v) := L(v) \setminus \{1\}$.

If later the labels of some predecessors or successors of v are increased from 0 to 1, this restriction will still hold.

Similarly, increasing the label of a node v from 0 to 2 can be considered. Let $in2(v)$ be the total cost for increasing the labels of all the direct predecessors of v to 2, and let $out2(v)$ be the total cost for increasing the labels of all the direct successors of v to 2. Then analogously to rule (v) the following rule is established:

Rule (vi): *If* $l(v) = 0$, and there is not a self-loop (v, v), and $cost2(v) \geq \min\{in2(v), out2(v)\}$, *then* label 2 is not admissible for node v, $L(v) := L(v) \setminus \{2\}$.

After rules (v) and (vi) have been applied, it is possible that for some nodes the only admissible label is 0. If these nodes are involved in self-loops or have a label different from 0, a solution with this (partial) labeling does not exist, and the search tree can be pruned (see Subsection IV-C). Otherwise these nodes can be eliminated from further consideration by the following rule:

Rule (vii): *If* $L(v) = \{0\}$, and there is not a self-loop (v, v), *then* ignore node v.

After all possible restrictions and simplifications have been made, procedure B follows.

B. Procedure B

Procedure B selects a node v that has not yet been considered for labeling (hence with label 0) and tries all the

admissible assignments $l(v) \in L(v)$. The node v is selected by the following criteria:

- The number of admissible labels for v, $|L(v)|$, should be minimum.
- Setting $l(v) = 2$ should make it possible to divide the TCC into smaller TCC's, and the largest of these smaller TCC's should contain a minimal number of nodes.

The first of these two points is most important. Of course, selecting a node v with $|L(v)| = 1$ is best since no decisions are required that may have to be reversed later.

If the newly assigned label is 1 or 2, it is tried to divide the TCC into smaller TCC's, and procedure A is called for each of these. If the new label is $l(v) = 0$, the TCC cannot be divided. In this case we set $L(v) := \{0\}$, and when procedure A is called, the application of rule (vii) reduces the graph by ignoring node v.

C. Pruning the Search Tree

At each node of the search tree the cost of the current labeling is computed. If this sum (*current_cost*) equals or exceeds the value obtained by the best solution found so far (*best_cost*), sons of this node in the search tree need not be investigated and the search tree can be pruned. Also if there is a node v with $l(v) \neq 0$ and $l(v) \notin L(v)$ or a node v with no admissible label, $L(v) = \emptyset$, the current (partial) labeling cannot be extended to a minimum cost labeling, and the search tree can be pruned at this point. Using these criteria for pruning, most of the search space may be skipped while it is still guaranteed that an optimal solution will be found.

After pruning, backtracking may be necessary. Starting from the current node, the search tree is traversed backward until a node is encountered where procedure B made a choice among several possible assignments of labels. A different assignment is made, and the search algorithm continues by again calling procedure A and procedure B alternatingly.

As the underlying problem is NP-hard, some circuits might be intractable by the exact algorithm. In this case good suboptimal solutions are obtained by a heuristic approach. A parameter *quality* ≤ 1 is introduced. This factor is used during two steps of the algorithm. In procedure A the rules (v) and (vi), which restrict the admissible labels, are modified to

Rule (v′): *If* $l(v) = 0$ and $cost1(v) \geq \min\{in1(v), out1(v)\} * quality$, *then* label 1 is not admissible for node v, $L(v) := L(v) \setminus \{1\}$.

Rule (vi′): *If* $l(v) = 0$ and there is not a self-loop (v, v), and $cost2(v) \geq \min\{in2(v), out2(v)\} * quality$, *then* label 2 is not admissible for node v, $L(v) := L(v) \setminus \{2\}$.

The second step where this parameter is applied occurs during pruning the search tree. Here the tree is pruned if *current_cost* $>$ *best_cost* $*$ *quality*. *quality* $= 1$ yields an optimal placement. Usually the costs of the suboptimal solutions are distinctly below the limit $\frac{optimal\_cost}{quality^2}$ as shown in the next section.

V. Experimental Results

The presented algorithm has been applied to the ISCAS'89 benchmark circuits [29]. For a given circuit the minimum cost placement strongly depends on the hardware overheads associated with the built-in test cells:

c_B: cost of replacing a D-flip-flop by a BILBO cell
c_{Bt}: cost of inserting a transparent BILBO cell
c_C: cost of replacing a D-flip-flop by a CBILBO cell
c_{Ct}: cost of inserting a transparent CBILCO cell.

For simplicity, it is assumed that the costs c_B, c_C, c_{Bt} and c_{Ct} do not depend on the locations in the netlist, and increased signal propagation times are not considered. Technology, design style, and the cell library have an impact on the cost distribution.

For $c_B + c_{Bt} < c_C, 2c_{Bt} < c_{Ct}$, a minimum cost solution does not have any CBILBO cells since a pair of BILBO cells is always more favorable than a single CBILBO cell. For $c_C < c_B, c_{Ct} < c_{Bt}$, an optimal solution does not have any BILBO cells. In this case the problem reduces to determining a minimum feedback vertex set.

The most interesting situations occur when $c_B < c_C < c_B + c_{Bt}$ and $c_{Bt} < c_{Ct} < 2c_{Bt}$ hold. In order to get realistic values, the hardware overheads were estimated by counting the additional transistors in a way similar to [22] for two different design styles. The resulting cost distributions are

parameter set I: $c_B = 11$, $c_C = 21$, $c_{Bt} = 23$, $c_{Ct} = 34$
parameter set II: $c_B = 10$, $c_C = 35$, $c_{Bt} = 30$, $c_{Ct} = 55$.

For the validation of the algorithm, we are interested in provably optimal solutions, computing times, and the impact of the factor *quality* on the costs of the found solutions. The solutions and computing times on a SUN Sparc 10 workstation are listed in Table I for parameter set I, and in Table II for parameter set II. The first column denotes the circuit, and #B, #Bt, #C, and #Ct are the number of BILBO cells, transparent BILBO cells, CBILBO cells, and transparent CBILBO cells, respectively. Test cells at the primary inputs and outputs are not counted. "Cost" is the sum of the overheads for these cells, and q is the value of the parameter quality where this solution is found. The last column gives the computing time.

The correctness of the solutions can easily be verified. For each node v, it is checked if the set of nodes that can be reached on paths with label sum less than 2 includes the node v.

With the cost distribution of parameter set I, only CBILBO cells are used. Parameter set II significantly increases the costs for CBILBO's, but even then the number of inserted BILBO cells is very small (see Table II).

It is seen that the efficiency of the algorithm depends on the cost distribution. For parameter set I, an optimal solution is found for all the circuits but one, whereas with parameter set II five circuits are hard to handle.

TABLE I
TEST CELL PLACEMENT FOR PARAMETER SET I

circuit	#B	#Bt	#C	#Ct	cost	q	CPU sec
s27	-	-	1	1	55	1	<1
s208	-	-	8	-	168	1	<1
s298	-	-	14	-	294	1	<1
s344	-	-	15	-	315	1	<1
s349	-	-	15	-	315	1	<1
s382	-	-	15	-	315	1	<1
s386	-	-	6	-	126	1	<1
s400	-	-	15	-	315	1	<1
s420	-	-	16	-	336	1	<1
s444	-	-	15	-	315	1	<1
s510	-	-	6	-	126	1	<1
s526	-	-	21	-	441	1	<1
s526n	-	-	21	-	441	1	<1
s641	-	-	7	4	283	1	<1
s713	-	-	7	4	283	1	<1
s820	-	-	5	-	105	1	<1
s832	-	-	5	-	105	1	<1
s838	-	-	32	-	672	1	<1
s953	-	-	6	-	126	1	<1
s1196	-	-	-	-	0	1	<1
s1238	-	-	-	-	0	1	<1
s1423	-	-	71	-	1491	1	2
s1488	-	-	6	-	126	1	<1
s1494	-	-	6	-	126	1	<1
s5378	-	-	30	-	630	1	1
s9234	-	-	152	-	3192	1	2
s13207	-	-	308	1	6502	1	4
s15850	-	-	441	-	9261	1	4
s35932	-	-	306	-	6426	1	8
s38417	-	-	1050	8	22322	1	32
s38584	-	-	1116	-	23436	1/2	1034

TABLE II
TEST CELL PLACEMENT FOR PARAMETER SET II

circuit	#B	#Bt	#C	#Ct	cost	q	CPU sec
s27	2	1	1	-	85	1	<1
s208	-	-	8	-	280	1	<1
s298	-	-	14	-	490	1	<1
s344	-	-	15	-	525	1	<1
s349	-	-	15	-	525	1	<1
s382	-	-	15	-	525	1	<1
s386	-	-	6	-	210	1	<1
s400	-	-	15	-	525	1	<1
s420	-	-	16	-	560	1	<1
s444	-	-	15	-	525	1	<1
s510	-	-	6	-	210	1	<1
s526	-	-	21	-	735	1	<1
s526n	-	-	21	-	735	1	<1
s641	8	4	7	-	445	1	<1
s713	8	4	7	-	445	1	<1
s820	-	-	5	-	175	1	<1
s832	-	-	5	-	175	1	<1
s838	-	-	32	-	1120	1	<1
s953	-	-	6	-	210	1	<1
s1196	-	-	-	-	0	1	<1
s1238	-	-	-	-	0	1	<1
s1423	-	-	71	-	2485	1	1
s1488	-	-	6	-	210	1	<1
s1494	-	-	6	-	210	1	<1
s5378	-	-	30	-	1050	1/3	<1
s9234	-	-	46	106	7440	1/3	4
s13207	-	-	310	-	10850	1/2	16
s15850	6	-	438	-	15390	1	4
s35932	36	-	288	-	10440	1	29
s38417	11	4	1049	4	37165	1/2	211
s38584	52	9	1079	-	38555	1/2	549

The impact of q on the costs is investigated in Table III where the costs are compared for different values of q. For values $q \geq \frac{1}{2}$ the heuristic solutions are very close to the provably minimum cost solution found by $q = 1$, but for $q < \frac{1}{2}$ there is a distinct loss in quality for some circuits.

In another experiment the algorithm has been applied to the S-graphs of the circuits. If the costs of CBILBO cells are set less than the costs of BILBO cells, then the MFVS-problem for implementing a partial scan path is solved. The algorithm is not tuned to handle S-graphs and the MFVS-problem and cannot use many of the graph reductions applied in [5] and [7], e.g., as it is designed for a much more general problem. But for all the benchmark circuits it found the provably optimal solution of the MFVS problem within a few seconds of computing time.

Some authors propose not to break self-loops for a partial scan design [1], [2], [5]. This problem can be solved by removing the self-loop edges from the S-graph. Also for these modified graphs the algorithm provides an optimal solution for all the benchmark circuits with the exception of s38584, where the factor *quality* must be reduced. The optimal solutions found agree with the numbers reported in [7].

Generally, if testability analysis shows that some parts of the circuit are easily testable without modifications or that some cycles need not be broken as they do not cause poor controllability or poor observability, then we can remove some parts from the circuit graph before we solve the MCP problem. Thus, the requirements for test registers can be reduced further.

It is also possible to model the effect of increased delays due to inserted test cells. The cost of a test cell can be adjusted such that it reflects the time slack of the considered gate or flip-flop. Alternatively, the insertion of test cells may be prohibited at some nodes by restricting the set of admissible labels. In the latter case, however, it is no longer guaranteed that a solution to the MCP problem always exists.

Furthermore, an estimation of the overheads regarding BIST control logic and routing can be incorporated into the costs of test cells. CBILBO's generally require a smaller control effort than BILBO's (see next section).

VI. SELF-TESTABLE TARGET STRUCTURES

The solutions for the ISCAS'89 circuits give new insight into the structure of self-testable circuits with a "test per clock" technique and minimal hardware overhead. In most of the circuits only CBILBO cells have to be placed, and in the remaining circuits only very few BILBO cells are required. We assumed a CBILBO cell up to 3.5 times more expensive than a BILBO cell, but the overall hardware overhead of a CBILBO-based approach is still smaller than a BILBO solution.

Hardware is not only saved within the data path, but test control and routing the test control lines are also simplified since CBILBO's have a uniform test mode for both pattern generation and response compaction. For example, the hardward-minimal solution for the circuit of Fig. 3 uses only two CBILBO's, T1 and T2, as shown in Fig. 14.

TABLE III
COSTS FOR SOLUTIONS FOUND WITH PARAMETER SET II AND DIFFERENT VALUES OF THE QUALITY FACTOR

circuit	q = 1	q = 5/6	q = 4/6	q = 3/6	q = 2/6
s27	85	85	85	105	110
s208	280	280	280	280	400
s298	490	490	490	490	690
s344	525	525	525	525	665
s349	525	525	525	525	665
s382	525	525	525	525	545
s386	210	210	210	210	210
s400	525	525	525	525	545
s420	560	560	560	560	560
s444	525	525	525	525	545
s510	210	210	210	210	210
s526	735	735	735	735	815
s526n	735	735	735	735	815
s641	445	445	445	525	545
s713	445	445	445	525	605
s820	175	175	175	175	195
s832	175	175	175	175	175
s838	1120	1120	1120	1120	1120
s953	210	210	210	210	210
s1196	0	0	0	0	0
s1238	0	0	0	0	0
s1423	2485	2485	2485	2485	2645
s1488	210	210	210	210	210
s1494	210	210	210	210	210
s5378	-	-	-	-	1050
s9234	-	-	-	-	7440
s13207	-	-	-	10850	12755
s15850	15390	15390	15390	15390	19335
s35932	10440	10440	10440	10440	10710
s38417	-	-	-	37165	46090
s38584	-	-	-	38555	39410

The hardware-minimal circuit structures found by the algorithm have also advantages with respect to test application time. All test registers may work in parallel, and the maximal degree of concurrency is not restricted by conflicts regarding the modes of test registers, but only by the limits of power dissipation during BIST operation [33].

The results obtained so far give also hints for an appropriate data path structure to be used by high-level synthesis systems and designers in order to implement self-testable systems. Recent synthesis for testability approaches try to reduce the number of cycles in a circuit. Avra and McCluskey try to reduce the number of self-adjacent registers using a so-called "incompatibility graph" [26], [34]. This graph indicates possible self-loops as well as constraints between variables due to the selected schedule, and graph coloring provides an admissible assignment of variables to registers. In [35] and [36], binding is formulated as a network flow problem for each control step, such that area and timing is optimized and the number of self-loops is minimized. Papachristou, Chiu, and Harmanani consider "(extended) testable functional blocks" consisting of functional units and the necessary test registers as the basic blocks of a self-testable RT structure [12], [27]. Allocation and binding of resources correspond to covering a given data path with a minimal number of testable functional blocks.

But looking at the results of the hardware-optimal algorithm, it is found that not only the number of cycles has an impact on BIST costs. More important is the existence of a small number of registers such that each cycle contains at least one of these registers. The hardware-minimal BIST solution will enhance these registers to CBILBO's, and if binding maps all the self-loops of variables to just these registers, a self-loop does not cause any additional cost for BIST.

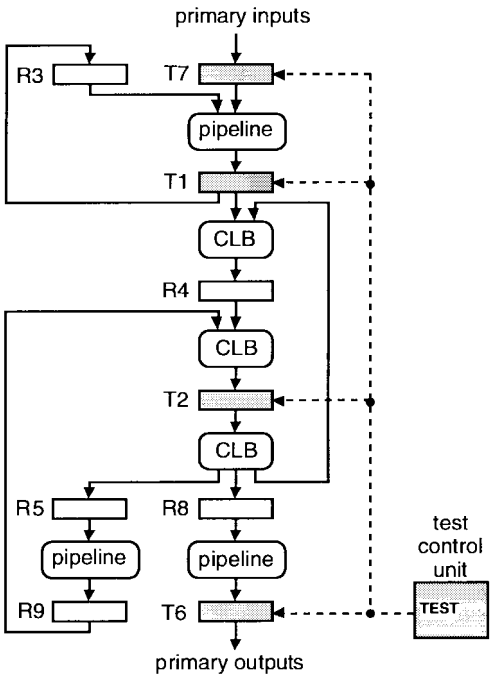

Fig. 14. CBILBO solution for the circuit of Fig. 3.

This problem is similar to designing a circuit that has an MFVS with low cardinality and is optimal for partial scan. In [37]–[39], retiming and resynthesis procedures have been presented that reduce the MFVS of a circuit described at gate level. In [40], a high-level synthesis method is proposed that synthesizes data paths such that they have a MFVS consisting of a small number of flip-flops. In contrast to partial scan designs, BIST cells for a "test per clock" scheme may also be inserted at combinational nodes. This can be exploited for further optimizations.

VII. CONCLUSION

An exact algorithm has been presented that selects flip-flops to be incorporated into multimode test registers or into a partial scan path in order to implement a "test per clock" or a "test per scan" scheme with minimum hardware overhead. The algorithm finds provably optimal solutions for nearly all the ISCAS'89 benchmark circuits, and the remaining circuits can be handled by a heuristic version very efficiently. It also considers BIST cells within the combinational logic, and takes into account that the hardware costs of BIST cells depend on their type. The MFVS problem of partial scan design is solved as a special case.

The hardware-optimal solutions mainly use CBILBO's such that test control logic, routing of test control signals, and even test application time are minimized, too.

ACKNOWLEDGMENT

The authors would like to thank G. Kiefer for assistance with the implementation and the experiments.

REFERENCES

[1] K.-T. Cheng and V. D. Agrawal, "A partial scan method for sequential circuits with feedback," *IEEE Trans. Comput.*, vol. 39, pp. 544–547, Apr. 1990.

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

[3] R. Gupta, R. Gupta, and M. A. Breuer, "The BALLAST methodology for structured partial scan design," *IEEE Trans. Comput.*, vol. 39, pp. 538–544, Apr. 1990.

[4] A. Kunzmann and H.-J. Wunderlich, "An analytical approach to the partial scan problem," *J. Electronic Testing: Theory and Applications*, vol. 1, no. 2, pp. 163–174, 1990.

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

[6] M. R. Garey and D. S. Johnson, *Computers and intractability*. New York: Freeman, 1979.

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

[8] B. Koenemann, J. Mucha, and G. Zwiehoff, "Built-in logic block observation techniques," in *Proc. Test Conf.*, Cherry Hill, NJ, 1979, pp. 37–41.

[9] H.-J. Wunderlich, "Self test using unequiprobable random patterns," in *Proc. Int. Symp. Fault-Tolerant Computing*, 1987, pp. 258–263.

[10] M. S. Abadir and M. A. Breuer, "A knowledge-based system for designing testable VLSI chips," *IEEE Design & Test Mag.*, vol. 2, no. 3, pp. 56–68, 1985.

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

[12] C. A. Papachristou, S. Chiu, and H. Harmanani, "A data path synthesis method for self-testable designs," in *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 378–384.

[13] S.-P. Lin, C. A. Njinda, and M. A. Breuer, "A systematic approach for designing testable VLSI circuits," in *Proc. Int. Conf. CAD*, 1991, pp. 496–499.

[14] A. Krasniewski and S. Pilarski, "Circular self-test Path: A low cost BIST technique for VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 46–55, Jan. 1989.

[15] F. Corno, P. Prinetto, and M. Sonza Reorda, "Making the circular self-test path effective for real circuits," in *Proc. Int. Test Conf.*, 1994, pp. 949–957.

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

[17] A. P. Stroele, "Partitioning and hierarchical description of self-testable designs," *IFIP Transactions A: Computer Science and Technology*, vol. A-42, pp. 113–122, 1994.

[18] O. F. Haberl and H.-J. Wunderlich, "The synthesis of self-test control logic," in *Proc. COMPEURO*, 1989, pp. 5.134–5.136.

[19] C. L. Hudson and G. D. Peterson, "Parallel self-test with pseudo-random test patterns," in *Proc. Int. Test Conf.*, 1987, pp. 954–963.

[20] R. Illman and S. Clarke, "Built-in self-test of the MACROLAN chip," in *Proc. Int. Test Conf.*, 1989, pp. 735–744.

[21] S. Das Gupta, R. G. Walther, E. B. Eichelberger, and T. W. Williams, "An enhancement to LSSD and some applications of LSSD in reliability, availability and serviceability," in *Proc. Int. Symp. Fault-Tolerant Computing*, 1981, pp. 32–34.

[22] M. J. Ohletz, T. W. Williams, and J. P. Mucha, "Overhead in scan and self-test designs," in *Proc. Int. Test Conf.*, 1987, pp. 460–470.

[23] L.-T. Wang and E. J. McCluskey, "Concurrent built-in logic block observer (CBILBO)," in *Proc. Int. Symp. Circuits and Systems*, 1986, pp. 1054–1057.

[24] V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A tutorial on built-in self-test, Part 1: Principles," *IEEE Design & Test Mag.*, vol. 10, no. 1, pp. 73–82, 1993.

[25] ———, "A tutorial on built-in self-test, Part 2: Applications," *IEEE Design & Test Mag.*, vol. 10, no. 2, pp. 69–77, 1993.

[26] L. Avra, "Allocation and assignment in high-level synthesis for self-testable data paths," in *Proc. Int. Test Conf.*, 1991, pp. 463–472.

[27] H. Harmanani and C. A. Papachristou, "An improved method for RTL synthesis with testability tradeoffs," in *Proc. Int. Conf. CAD*, 1993, pp. 30–35.

[28] M. L. Flottes, D. Hammad, and B. Rouzeyre, "Automatic synthesis of BISTed data paths from high level specification," in *Proc. European Design and Test Conf.*, 1994, pp. 591–598.

[29] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. Int. Symp. Circuits and Systems*, 1989, pp. 1929–1934.

[30] D. Goldberg, "Computer arithmetic," in *Computer Architecture: A Quantitative Approach*, J. L. Hennessy and D. A. Patterson, Eds. San Mateo, CA: Morgan Kaufmann, 1990.

[31] S. Narayanan, C. Njinda, and M. Breuer, "Optimal sequencing of scan registers," in *Proc. Int. Test Conf.*, 1992, pp. 293–302.

[32] A. P. Stroele and H.-J. Wunderlich, "Configuring flip-flops to BIST registers," in *Proc. Int. Test Conf.*, 1994, pp. 939–948.

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

[34] L. Avra and E. J. McCluskey, "Behavioral synthesis of testable systems with VHDL," in *Proc. COMPCON Spring'90*, 1990, pp. 410–415.

[35] A. Mujumdar, R. Jain, and K. Saluja, "Incorporating testability considerations in high-level synthesis," *J. Electronic Testing: Theory and Applications*, vol. 5, no. 1, pp. 43–55, 1994.

[36] A. Mujumdar, R. Jain, and K. Saluja, "Behavioral synthesis of testable designs," in *Proc. Int. Symp. Fault-Tolerant Computing*, 1994, pp. 436–445.

[37] D. Kagaris and S. Tragoudas, "Partial scan with retiming," in *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 249–254.

[38] S. T. Chakradhar and S. Dey, "Resynthesis and retiming for optimum partial scan," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 87–93.

[39] P. Pan and C. L. Liu, "Partial scan with pre-selected scan signals," in *Proc. ACM/IEEE Design Automation Conf.*, 1995, pp. 189–194.

[40] S. Dey, M. Potkonjak, and R. Roy, "Synthesizing designs with low-cardinality minimum feedback vertex set for partial scan application," in *Proc. VLSI Test Symp.*, 1994, pp. 2–7.

Albrecht P. Stroele (A'90) received the diploma degree in electrical engineering from the University of Darmstadt, Germany, in 1980, and the Dr.rer.nat. (Ph.D.) degree in computer science from the University of Karlsruhe, Germany, in 1992.

From 1981 to 1987, he was with Siemens where he was involved in image processing and computer design. Currently, he is Privatdozent at the Institute of Computer Design and Fault Tolerance, University of Karlsruhe. His research interests include fault modeling, design for testability, built-in self-test techniques, and cellular automata.

Dr. Stroele is a member of the steering committee of the GI/ITG Special Interest Group on "Testmethoden und Zuverlässigkeit von Schaltungen und Systemen."

Hans-Joachim Wunderlich (A'86) received the Dr.rer.nat. (Ph.D.) degree in computer science from the University of Karlsruhe, Germany, in 1986.

From 1986 to 1991, he was the head of a research group on automation of circuit design and test at the University of Karlsruhe. From 1991 to 1996 he was a full Professor of Computer Science at the University of Siegen. Since October 1996, he has been the head of the Division for Computer Architecture at the University of Stuttgart. He is author and coauthor of three books and over 70 papers in the field of test, synthesis, and fault tolerance of digital systems.

Dr. Wunderlich has been a member of the program committee at numerous conferences and a reviewer of research proposals submitted to NSF and NATO. Within the European projects EUROCHIP and EUROPRACTICE he has been a lecturer for courses on VLSI design and test.