Layout-Aware Scan Chain Synthesis for Improved Path Delay Fault Coverage

Puneet Gupta, Andrew B. Kahng, Ion I. Măndoiu, and Puneet Sharma

***Abstract*—Path delay fault testing has become increasingly important due to higher clock rates and higher process variability caused by shrinking geometries. Achieving high-coverage path delay fault testing requires the application of scan justified test vector pairs, coupled with careful ordering of the scan flip-flops and/or insertion of dummy flip-flops in the scan chain. Previous works on scan synthesis for path delay fault testing using scan shifting have focused exclusively on maximizing fault coverage and/or minimizing the number of dummy flip-flops, but have disregarded the scan wirelength overhead. In this paper we propose a layout-aware coverage-driven scan chain ordering methodology and give exact and heuristic algorithms for computing the achievable tradeoffs between path delay fault coverage and both dummy flip-flop and wirelength costs. Experimental results show that our scan chain ordering methodology yields significant improvements in path delay coverage with a very small increase in wirelength overhead compared to previous layout-driven approaches, and similar coverage with up to 25 times improvement in wirelength compared to previous layout-oblivious coverage-driven approaches.**

***Index Terms*—Design-for-test, physical design, scan chain synthesis, VLSI.**

I. INTRODUCTION

At technology nodes 0.13 μm and below, designs are prone to several types of defects that are not caught by traditional stuck-at-fault testing. In particular, path delay faults have become more common due to higher clock rates and higher process variability caused by shrinking geometries. Therefore, at-speed path delay testing is gaining importance. Delay fault testing requires the application of two test vectors.[1] The first test vector, referred to as the *initialization vector*, initializes the logic to a known state. The second vector, referred to as the *activation vector*, activates the targeted fault, causing a transition to be propagated along the path under test. It is well-known that at-speed application of test vector pairs to the primary inputs has low path delay fault coverage [4]. However, improved fault coverage can be achieved by using scan chaining [1], which has already become the design-for-test (DFT) technique of choice for stuck-at fault testing.

A scan chain is formed of *scan flip-flops*, which are some or all of the flip-flops existing in a design. One end of the scan chain appears as a primary input (PI) and the other end appears as a primary output (PO). There are two techniques to produce the two vectors required for path delay fault testing—*launch-on-capture* and *launch-on-shift*. With launch-on-capture, the initialization vector not only sensitizes the proper paths but also produces the activation vector. On the other hand, with scan justification the activation vector is produced by a single shift of the initialization vector. Standard scan-based delay fault testing involves shifting in the initialization vector, giving one (launch-on-shift) or two (launch-on-capture) clocks to the circuit, then shifting out the resulting flip-flop values. Some pros and cons of the two techniques are as follows.

- It is known that test generation complexity using launch-on-shift is typically lower than that using launch-on-capture. To save test generation time, vectors may be generated using launch-on-shift first, and launch-on-capture may be used for faults that cannot be tested by launch-on-shift. This approach was studied in [7] and a savings of 30% of test generation time was reported.
- It has been argued that path delay faults that cannot be detected by launch-on-capture are likely to be functionally false paths, but identification of functionally untestable paths is a hard problem [17]. Several faults that cannot be detected using launch-on-capture by commercial automatic test pattern generation (ATPG) tools may be detected by scan shifting.
- The requirement that the activation vector must be obtained from the initialization vector by one-bit shifting along the scan chain [24] constrains scan chain synthesis for delay fault testing using launch-on-shift. In general, not all activation vectors can be realized in this way once we fix the order of the flip-flops in the scan chain. On the other hand, test vectors generated using launch-on-capture are compatible with any scan order. This allows scan chain synthesis to be driven by layout such that the wirelength is minimized. Nevertheless, there may be multiple equiwirelength scan chain orderings, some of which may be conducive to launch-on-shift based path delay testing.

Manuscript received January 19, 2004; revised May 10, 2004 and August 30, 2004. A preliminary version of this work has appeared in [13]. This work was supported in part by Cadence Design Systems Inc., and the MARCO Gigascale Silicon Research Center. This paper was recommended by Associate Editor N. Jha.

P. Gupta and P. Sharma are with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0114 USA (e-mail: puneet@ucsd.edu; sharma@vlsicad.ucsd.edu).

A. B. Kahng is with the Departments of Computer Science and Engineering, and of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0114 USA (e-mail: abk@ucsd.edu).

I. I. Măndoiu was with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093–0114 USA. He is now with the Computer Science and Engineering Department, University of Connecticut, Storrs, CT 06269-1155 USA (e-mail: ion@engr.uconn.edu).

Digital Object Identifier 10.1109/TCAD.2005.850900

---

[1] More general tests, such as *validatable nonrobust tests* [23], require the application of a *set* of test vector pairs. Our algorithms can handle such tests after minor modifications.

0278-0070/$20.00 © 2005 IEEE

It is therefore possible to increase fault coverage with little or no impact on layout overhead of scan.

Under the standard practice of using a single scan-enable signal and assuming scan chain edges always link the non-negated data output pin of the source flip-flop to the data input pin of the destination flip-flop, we can capture the interdependence between test vector pairs and scan chain order as follows:

Definition 1: The scan chain edge between flip-flop i and flip-flop j is *forbidden* by (or *conflicts with*) a test vector pair consisting of initialization vector u and activation vector v if either $u(i) = 0$ and $v(j) = 1$, or $u(i) = 1$ and $v(j) = 0$.

Note that this differs from the conflict definition given by Norwood and McCluskey [22], which forbids an edge between flip-flop i and flip-flop j whenever both i and j have defined values (i.e., either 0 or 1), *even if the two values are equal.* The definition in [22] leaves the freedom to arbitrarily select for each flip-flop the data output pin driving the outgoing scan chain edge, but is excessively restrictive from a coverage point of view.

Scan chain edge $i \rightarrow j$ can be made compatible with all conflicting tests by "enhancing" flip-flop j to store an additional bit or, equivalently, by inserting a separate 1-bit flip-flop between flip-flops i and j. We will refer to this operation as *inserting a dummy flip-flop* on edge $i \rightarrow j$.

An early approach to ensuring high delay fault coverage is the so-called "enhanced-scan" [10], [20], in which all scan flip-flops are enhanced. Enhanced-scan makes possible the application of any pair of initialization/activation test vectors by interleaved scanning, but has a high cost in terms of die area, test time, and circuit performance degradation. A different approach, proposed in [21], is to use standard scan flip-flops and reorder them in the scan chain so as to maximize the number of applicable test vector pairs from the given set. Combining the two approaches was first proposed in [7], which suggested to follow coverage-driven flip-flop ordering by partial dummy flip-flop insertion. More recently, [22] proposed algorithms for complete path delay coverage by simultaneous flip-flop ordering and minimal dummy flip-flop insertion, and [11] studied similar formulations with additional consideration of scan chain routing costs.

Together with achieving high delay fault coverage, a significant concern during scan chain synthesis is the wirelength overhead, since excessive scan wirelength can compromise the routability of the design and degrade its performance. While the wirelength overhead has received considerable attention in the context of stuck-at fault testing [3], [5], [6], [8], [12], [16], [18], [19], previous works on scan synthesis for delay fault testing have completely disregarded this aspect, focusing exclusively on maximizing delay fault coverage [21], or achieving a certain coverage factor with minimum number of dummy flip-flops [7], [11], [22].

In this paper we consider both dummy flip-flop and wirelength costs and address post-layout scan chain synthesis formulations that capture the achievable tradeoffs between these costs and path delay fault coverage. Layout information is beneficial to scan chain synthesis in two important ways: (1) it enables higher ATPG selectivity in the choice of paths to be tested due to the availability of accurate path criticalities, and (2) it makes possible accurate estimation of scan routing cost and impact on circuit performance, thus enabling better informed coverage-cost tradeoff decisions.

Our contributions include the following.

- An efficient heuristic for maximizing delay fault coverage by simultaneous layout-aware scan chain synthesis and insertion of a bounded number of dummy flip-flops.
- A compact integer linear programming (ILP) formulation for the problem of optimally inserting a number of dummy flip-flops in a given scan chain. The integer program can be solved in practical runtime for designs with up to tens of thousands of scan flip-flops using mathematical optimization packages such as CPLEX [25].
- A comprehensive empirical evaluation of the proposed algorithms on industry testcases, including a detailed analysis of the tradeoffs between delay fault coverage on one hand and the number of dummy flip-flops and scan chain wirelength on the other hand.

All our work is equally applicable to gate delay fault testing. However, we believe this work to be most relevant for path delay fault testing, since the latter problem is more challenging. In general, the path delay fault model is more useful at debugging timing problems, since it models distributed defects as opposed to the gate delay fault model which models isolated defects. Furthermore, ATPG tools can construct gate delay test patterns by propagating the fault to one of several available sequential or primary outputs. Hence, usually there are several possible test patterns that detect a certain gate delay fault, and as a result typical gate delay fault coverages are already very high. On the other hand, path delay faults are tougher to detect and coverages achieved by current ATPG tools are fairly low. Thus, the scan chain ordering methodology can add more value without being overly constrained in a path delay context.

The rest of the paper is organized as follows. In Section II, we give an efficient heuristic for the problem of maximizing path delay coverage by scan chain synthesis and simultaneous insertion of a bounded number of dummy flip-flops. In Section III we prove the NP-hardness of the problem of computing achievable tradeoffs between delay fault coverage and the number of dummy flip-flops inserted in an already routed scan chain and give a compact ILP formulation for this problem. Finally, we present experimental results in Section IV and conclude in Section V with directions of further research.

II. FORMULATIONS FOR POST-LAYOUT COVERAGE DRIVEN SCAN CHAIN SYNTHESIS

In [11], the post-layout scan chain synthesis problem is formulated as follows.

Scan Synthesis for Complete Delay Fault Coverage (CompleteDFC-Scan)

Given:

- Set of n placed flip-flops F, scan-in/scan-out pins SI and SO
- Set $\mathcal{T}$ of m test vector pairs

Inputs:

- Set of n placed flip-flops F
- Scan-in/scan-out pins SI and SO
- Set of m test vector pairs T
- Maximum number D of dummy flip-flops

Outputs:

- Scan chain ordering π of $F \cup \{SI, SO\}$ starting with SI and ending with SO and set of covered tests $\mathcal{C} \subseteq T$

Phase 1: Run the multi-fragment greedy algorithm (Figure 2) to get scan chain fragments $P_0, \ldots, P_{D+1}$ and set of covered test vector pairs$\mathcal{C}$.

Phase 2: Construct a complete graph G' with vertex set $\{SI, SO\} \cup \{P_0, \ldots, P_{D+1}\}$ and edge-costs given by pin-to-pin wirelength, then compute a minimum cost directed path $\hat{P}$ from SI to SO by running an Asymmetric Traveling Salesman Problem (ATSP) heuristic such as LKH [14] or ScanOpt [5] on G'. Construct a scan chain π_1 by stitching the fragments $P_0, \ldots, P_{D+1}$ in the order given by $\hat{P}$, and insert dummy flip-flops on the edges used for stitching, i.e., on edges connecting flip-flops from different P_i's.

Phase 3: Construct an auxiliary graph G'' with vertex set $F \cup \{SI, SO\}$ containing all edges compatible with the faults in $\mathcal{C}$ (note that all edges of π_1 must be in G''). Assign edge-costs given by pin-to-pin wirelength, then compute a minimum cost directed path π_2 from SI to SO computed by running an ATSP heuristic such as LKH [14] or ScanOpt [5] on G''. Return the chain order π_2.

Fig. 1. Three-phase MaxDFC-scan heuristic.

Find:

- Scan chain ordering π of $F \cup \{SI, SO\}$ starting with SI and ending with SO

Such that:

- The number of dummy flip-flops needed to achieve complete fault coverage (i.e., the number of edges in π that conflict with at least one test vector pair in $\mathcal{T}$) is minimized

The above formulation is appropriate when complete fault coverage is a design requirement. However, for most designs full coverage is not required. Rather, designers decide on a design-by-design basis the best tradeoff between delay fault coverage and scan chain cost (wirelength, dummy flip-flops, impact on performance, etc.). To accurately capture this tradeoff we introduce the following problem formulation:

Scan Synthesis for Max Delay Fault Coverage (MaxDFC-Scan)

Given:

- Set of n placed flip-flops F, scan-in/scan-out pins SI and SO
- Set $\mathcal{T}$ of m test vector pairs and positive weights[2] w_t, $t \in \mathcal{T}$
- Upperbound D on the number of dummy flip-flops

Find:

- Scan chain ordering π of $F \cup \{SI, SO\}$ starting with SI and ending with SO
- Set $\mathcal{C} \subseteq \mathcal{T}$ of covered test vector pairs

Such that:

- At most D scan chain edges $\pi_i \rightarrow \pi_{i+1}$ conflict with test vector pairs in $\mathcal{C}$
- Subject to this constraint, the total weight of tests in $\mathcal{C}$ is maximized and the total length of the scan chain is minimized

MaxDFC-Scan generalizes various problem formulations in [7], [11], [21], [22], and is therefore NP-hard. Thus, we cannot expect to find polynomial-time algorithms that solve MaxDFC-Scan optimally in polynomial time [9]. In the remainder of this section, we present a MaxDFC-Scan heuristic that can efficiently handle the instances with tens of thousands of scan flip-flops and thousands of test vector pairs arising in today's high-end designs.

A. Three-Phase MaxDFC-Scan Heuristic

The overall flow of our three-phase heuristic for MaxDFC-Scan is shown in Fig. 1. The heuristic works in three phases.

In the *first phase* of the heuristic we construct a set of $D+1$ scan chain fragments using a multifragment greedy algorithm (Fig. 2) similar to that used for the traveling salesman problem (TSP) [15]. Since we are not inserting dummy flip-flops on the edges contained in the $D+1$ fragments, we want these edges to be compatible with as many faults as possible. Therefore, the multifragment greedy heuristic attempts to use edges in increasing order of the number of conflicting test vector pairs. Note that the number of conflicting test vector pairs changes during the algorithm: once a test vector pair is discarded because it conflicts with one of the edges added to the chain fragments, it should no longer be counted as conflicting with the remaining candidate edges. Simultaneously with keeping low the number of incompatible faults, the multifragment greedy algorithm attempts to keep the wirelength of the fragments as low as possible. It does so by growing the fragments as much as possible using short edges before starting to use longer edges.

---

[2] The weights w_t represent the number—or possibly average criticality—of faults tested by test vector pair t; multiple faults per test vector pair are common due to the use of test vector compaction in automatic test pattern generation tools.

Inputs:

- Set of n placed flip-flops F
- Set of m test vector pairs T
- Weights w_t, $t \in T$, reflecting the relative criticality of the test vector pairs
- Number D of dummy flip-flops to be inserted
- Algorithm parameters α, T, and f

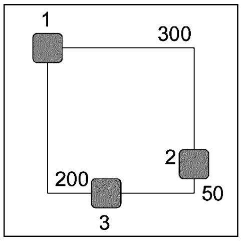
Outputs:

- $D+1$ scan chain fragments (to be connected in a single scan chain using D dummy flip-flops)
- Set of covered test vector pairs $C \subseteq T$

```
/* Step 1: Initialize */
    $E \leftarrow \{i \to j \ : \ i,j \in F, i \neq j\}$                 /* $E$: set of all edges */
    $\forall t \in T, E_t \leftarrow \{e \in E \ : \ e$ conflicts with $t\}$   /* $E_t$: set of edges that conflict with vector pair $t$ */
    $\forall e \in E, T_e \leftarrow \{t \in T \ : \ e$ conflicts with $t\}$   /* $T_e$: set of test vector pairs conflicting with edge $e$ */
    $\forall i \to j \in E, \ell(i \to j) \leftarrow$ routing distance between $i$ and $j$   /* $\ell(i \to j)$: length of edge $i \to j$ */
    $Avg \leftarrow \frac{\sum_{e \in E} \ell(e)}{|E|}$                     /*$Avg$: average edge length */
    $E' \leftarrow \emptyset$                                /* $E'$: Set of edges in the constructed tour */
    $C \leftarrow T$                                 /* $C$: Set of test vector pairs compatible with edges in $E'$ */

/* Step 2: Fill buckets */
    for each edge $i \to j \in E$ do
        if $\alpha * \frac{\ell(i \to j)}{Avg} + |T_{i \to j}| < T$ then
            put $i \to j$ in $b$-th bucket, where $b = \sum_{t \in T_{i \to j}} w_t$

/* Step 3: Construct tour */
    while $|E'| < n - D - 1$ do
        if all buckets are empty then
            $T \leftarrow f * T$ and goto Step 2
        else
            select a shortest edge $i \to j$ from the first non-empty bucket and delete it from $E$
            if $i \to j$ does not create a cycle or a vertex of degree greater than 2 with the other edges of $E'$ then
                $E' \leftarrow E' \cup \{i \to j\}$
                $C \leftarrow C \setminus T_{i \to j}$
                for each $t \in T_{i \to j}$ and each $e \in E_t$ do
                    remove $t$ from $T_e$
                    decrease the bucket number of $e$ by $w_t$

/* Step 4: Ouput results */
    return $C$ and the (up to $D+1$) paths formed by the edges of $E'$
```

Fig. 2. Multifragment greedy algorithm.

In the *second phase* of the MaxDFC-Scan heuristic, we combine the $D+1$ fragments produced by the multifragment greedy algorithm into a single scan chain with the help of D dummy flip-flops. Since the objective of this phase is to increase the wirelength of the scan chain by the least amount possible, we perform this “fragment stitching” by using a wirelength driven asymmetric traveling salesman problem (ATSP) solver. Even high-quality solvers such as LKH [14] can be used in practice since the number of fragments is small and hence runtime is not prohibitive.



$\ell(1 \to 2) = 300$
$\ell(1 \to 3) = 200$
$\ell(2 \to 1) = 300$
$\ell(2 \to 3) = 50$
$\ell(3 \to 1) = 200$
$\ell(3 \to 2) = 50$
$Avg = 183.33$

	A		B		C		D		E	
	Init.	Activ.	Init.	Activ.	Init.	Activ.	Init.	Activ.	Init.	Activ.
FF1	1	0	0	1	1	1	1	1	0	0
FF2	0	0	0	0	0	1	X	X	0	1
FF3	0	0	1	1	X	X	0	1	0	0

Fig. 3. Placement information and five test vector pairs for a MaxDFC-Scan instance with $n = 3$ flip-flops. Weights $w_t, t \in \{A, B, C, D, E\}$, are assumed to be equal to 1. The sets of test vector pairs conflicting with each edge are $\mathcal{T}_{1\to 2} = \{A, E\}, \mathcal{T}_{1\to 3} = \{A, B\}, \mathcal{T}_{2\to 1} = \{B, C\}, \mathcal{T}_{2\to 3} = \{B\}, \mathcal{T}_{3\to 1} = \{D\}$, and $\mathcal{T}_{3\to 2} = \{B, E\}$. The sets of edges conflicting with each test vector pair are $E_A = \{1 \to 2, 1 \to 3\}, E_B = \{1 \to 3, 2 \to 1, 2 \to 3, 3 \to 2\}, E_C = \{2 \to 1\}$, $E_D = \{3 \to 1\}$, and $E_E = \{1 \to 2, 3 \to 2\}$.

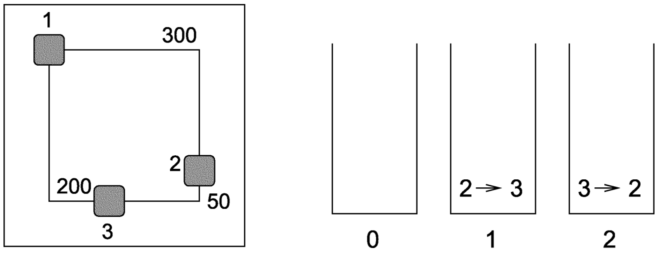


Fig. 4. Bucket contents after first execution of Step 2 of the multifragment greedy algorithm on the instance described in Fig. 3. We assume that $\alpha = 2$, $T = 3$, $f = 2$, and $D = 0$. Only edges $2 \to 3$ and $3 \to 2$ meet the constraint $\alpha * \ell(i \to j)/Avg + |\mathcal{T}_{i \to j}| \le T$. At the end of this step $\mathcal{C} = \{A, B, C, D, E\}$ and $E' = \emptyset$.

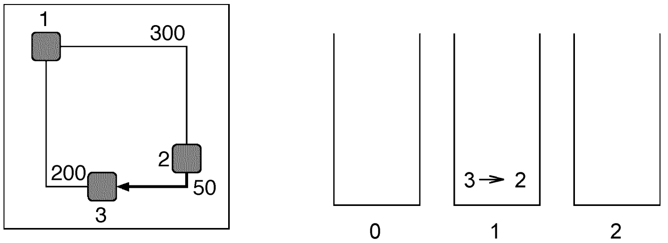


Fig. 5. Bucket contents after first execution of Step 3. Edge $2 \to 3$ (shown with a thick line) is added to E' in this step. Vector pair B, which conflicts with $2 \to 3$, is removed from $\mathcal{C}$. Edge $3 \to 2$ is moved to bucket 1 since it conflicts with a single test vector pair of the updated $\mathcal{C}$. In next execution of Step 3 the multifragment greedy algorithm selects edge $3 \to 2$, but it does not add it to E' since it creates a cycle with the already included edge $2 \to 3$. Since all buckets are now empty, the algorithm doubles T (we assume that $f = 2$), then continues from Step 2.

In the *third phase* of the MaxDFC-Scan heuristic, all edges compatible with surviving faults are added to an auxiliary graph, and an ATSP solver is called again to further decrease the length of the tour and possibly remove some of the dummy flip-flops.

Since the multifragment greedy algorithm is the most important part of the three-phase MaxDFC-Scan heuristic, we give next a line-by-line description of its operation. The algorithm begins by doing several initializations in Lines 2–8: the set E is initialized to contain all directed edges between flip-flop pairs; for each test vector pair t, the set E_t is initialized to contain all edges in E that conflict with t; for each edge $e \in E$, $\mathcal{T}_e$ is initialized to contain all test vector pairs conflicting with e; and, for each edge $i \to j \in E$, $\ell(i \to j)$ is set to the wirelength required to connect flip-flop i to flip-flop j in the scan chain. Finally, Avg is set to the average edge wirelength E' which represents the set of selected scan edges, is set to the empty set, and $\mathcal{C}$, which represents the set of test vector pairs compatible with the edges in E', is initialized to $\mathcal{T}$. In second step (Lines 11–13), the algorithm distributes edges into buckets based on the total weight of test vector pairs that are *incompatible* with these edges. To reduce the runtime, only edges that meet the criteria in Line 12 are considered. The parameter α is used to achieve different tradeoffs between wirelength and fault coverage of the final tour; in our experiments we used $\alpha = 2.0$. The parameter T is the initial threshold used in Line 12, which is increased by a factor of f every time the buckets get empty (edge removal from buckets is explained later). Our experiments indicate that solution quality is not very sensitive to the exact values used for the parameters T and f; in the results reported in Section IV we used $T = 1.4$ and $f = 1.2$.

Edge selection is done in the loop spanning Lines 16–26. Loop execution stops when the number of scan chain fragments becomes $D + 1$ (or, equivalently, the number of selected edges equals $n - D - 1$, see Line 16). The test in Line 17 checks if there are any edges available in the buckets. If all buckets are empty, Line 18 increases the threshold T and jumps to Step 2 to refill the buckets. On the other hand, if there is at least one nonempty bucket, we select (in Line 20) the shortest edge, $i \to j$, from the lowest-indexed nonempty bucket, and remove it from that bucket. In Line 21 we check if $i \to j$ can be added to the tour, i.e., if adding $i \to j$ to the tour does not form a cycle or causes the incoming or outgoing directed degree of a vertex to become greater than 2. If it can, $i \to j$ is added to the set E' (Line 22), and all test vector pairs that are incompatible with it are removed from $\mathcal{C}$ (Line 23). Furthermore, we consider each test vector pair t removed in Line 23, and for each edge e that is incompatible with t, we remove t from the set of vector pairs $\mathcal{T}_e$ that are incompatible with e (Line 25) and reduce the bucket number of edge e by the weight w_t of the removed test vector pair (Line 26).

Finally, in Line 29, the set of covered vector pairs, $\mathcal{C}$, and the constructed scan chain fragments are returned. The step-by-step execution of the multifragment greedy algorithm on a small MaxDFC-Scan instance described in Fig. 3 is illustrated in

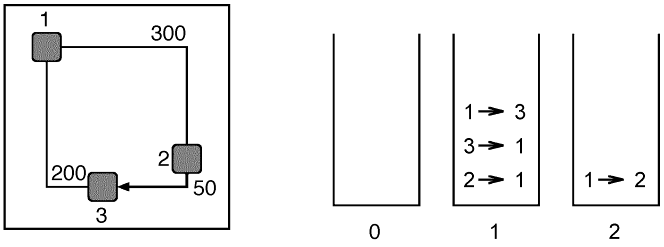


Fig. 6. Bucket contents after second execution of Step 2. All remaining edges of E meet the constraint $\alpha * \ell(i \rightarrow j)/Avg + |\mathcal{T}_{i\rightarrow j}| \leq T$ with $T = 6$. Edge $1 \rightarrow 3$ is selected when executing Step 3, however, it is not added to E' since it creates a vertex with incoming degree of 2. In next execution of Step 3 the algorithm selects edge $3 \rightarrow 1$ and adds it to E'.

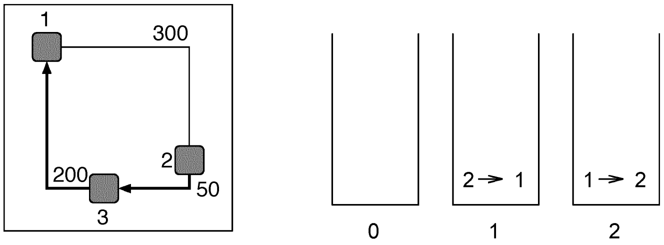


Fig. 7. Bucket contents after adding edge $3 \rightarrow 1$ to E'. Vector pair D conflicts with $3 \rightarrow 1$ and is therefore removed from $\mathcal{C}$. No bucket changes are required, since vector pair D does not conflict with other edges. Since $|E'| = n - D - 1$ (we assume that $D = 0$), the while loop in Step 3 completes. In Step 4, the algorithm returns $\mathcal{C} = \{A, C, E\}$ and the path $2 \rightarrow 3 \rightarrow 1$.

Figs. 4–7; for this illustration we assume that $\alpha = 2, T = 3, f = 2$, and $D = 0$.[3]

B. Extension to Multiple Scan Chains

Using multiple scan chains is a well-known technique to significantly reduce testing time. Our heuristics can be easily modified to take care of multiple scan chains. Assuming that all flip-flops are labeled by the scan chain they belong to (this may be done using a tool such as *Synopsys DFT Compiler*) the modified heuristic considers only those edges which connect flip-flops of the same scan chain. If up to D dummy flip-flops can be added, and we want to synthesize k scan chains, then the multifragment greedy algorithm must be stoped after it constructs $D + k$ fragments. Such a modification will in fact speedup the multifragment greedy algorithm, since the number of candidate edges decreases. Phase 2 of the MaxDFC-Scan heuristic must also be modified to label the inserted dummy flip-flops with the scan chain they belong to. Finally, Phase 3 must be performed on each scan chain independently.

III. OPTIMAL DUMMY FLIP-FLOP INSERTION IN A GIVEN SCAN CHAIN

In this section we consider minimum dummy flip-flop insertion in a scan chain constructed in a previous design phase (possibly using the three-phase heuristic in Section II). We assume that a set of spare sites available for dummy flip-flop insertion have been identified, and this results in the identification of a subset of the scan chain edges that are eligible for dummy flip-flop insertion. Clearly, if there is no bound on the number of dummy flip-flops that can be inserted then complete test coverage is achieved by inserting one dummy flip-flop in each scan chain edge that conflicts to at least one test (the required set of dummy flip-flops can be computed in $O(nm)$ time, where n is the number of flip-flops, and m is the number of test vector pairs). In practice it is useful to impose an upperbound on the number of inserted dummy flip-flops while maximizing path delay fault coverage.[4] This motivates the following problem formulation:

TABLE I
TEST VECTOR PAIRS USED IN THE PROOF OF THEOREM I

Vector	f_u	f_{u+1}	f_v	f_{v+1}	Case
Initialization	0	1	1	0	if $f_v = f_{u+1}$
Activation	1	1	1	0	
Initialization	1	0	0	1	if $f_u = f_{v+1}$
Activation	1	0	1	1	
Initialization	0	1	1	0	otherwise
Activation	1	1	1	0	

Maximum Coverage Dummy Insertion (MCDI) Problem

Given:
- Valid scan ordering $\pi = (\pi_0, \pi_1, \ldots, \pi_{n+1})$ of $F \cup \{SI, SO\}$ with $\pi_0 = SI$ and $\pi_{n+1} = SO$ and set $\mathcal{E}$ of scan chain edges eligible for dummy flip-flop insertion
- Set $\mathcal{T}$ of m test vector pairs and nonnegative weights w_t, $t \in \mathcal{T}$
- Upperbound D on the number of inserted dummy flip-flops

Find:
- D scan chain edges $(\pi_i, \pi_{i+1}) \in \mathcal{E}$ in which dummy flip-flops will be inserted
- Set $\mathcal{C} \subseteq \mathcal{T}$ of covered test vector pairs

Such that:
- None of the scan chain edges conflicts with the test vector pairs in $\mathcal{C}$ after dummy insertion
- The total weight of the test vector pairs in $\mathcal{C}$ is maximum possible subject to the above constraint

NP-Hardness. Cheng *et al.* [7] claim NP-hardness of MCDI based on equivalence to the set covering problem. However, MCDI is *not* equivalent to set covering, since the set of faults made testable by inserting a dummy flip-flop cannot be determined independently of the other inserted flip-flops. A correct NP-hardness proof is given below:

Theorem 1: The MCDI problem is NP-hard.

Proof: We will show that the NP-hard CLIQUE problem reduces in polynomial time to MCDI. Given a graph $G = (V, E)$ and a positive integer k, the CLIQUE problem asks if G has a complete subgraph of size k. Without loss of generality, assume that $V = \{1, \ldots, |V|\}$. We construct a MCDI instance with $n = |V| + 1$, $F = \{f_1, \ldots, f_n\}$, and $\pi_i = f_i$ for every $i = 1, \ldots, n$. For each edge $(u, v) \in E$ construct a test vector pair $t_{(u,v)}$ which conflicts with edges

---

[3] Note that in this example bucket 0 is empty throughout the execution of the algorithm; in general this may not necessarily be the case.

[4] An alternate formulation seeks a minimum number of dummy flip-flops that guarantee a certain fault coverage [7].

TABLE II
TEST CASE PARAMETERS

Test Case	Source	# Cells	# Scan FFs	#Critical Paths	Coverage by launch-on-capture	#Remaining Paths
s38417	ISCAS'89	6291	1564	506	4.74%	482
s13207	ISCAS'89	1648	627	172	20.34%	137
s9234	ISCAS'89	529	145	1206	46.43%	646
AES	opencores.org	10465	554	3050	58.03%	1280
DES3	opencores.org	3912	128	988	71.76%	279

$f_u \rightarrow f_{u+1}$ and $f_v \rightarrow f_{v+1}$ but no other edges of π. The test pair $t_{(u,v)}$ can be constructed by assigning don't care values to all flip-flops except f_u, f_{u+1}, f_v, and f_{v+1}, for which the values are set as in Table I. It is easy to see that G has a clique of size k if and only if $D = k$ dummy flip-flops can be inserted on the edges of π such that $k(k-1)/2$ tests become testable, i.e., deciding CLIQUE reduces to optimizing MCDI. ■

ILP Formulation. In the following we present an integer linear program formulation for the MCDI problem. This formulation can be optimally solved in practical running time even for scan chains with tens of thousands of flip-flops using mathematical optimization packages such as the branch and bound based CPLEX MIP solver [25]. Let E_t be the set of scan chain edges conflicting with test t. MCDI can be formulated as an integer linear program by using two sets of 0/1 variables:

- $x_i, i = 1, \ldots, n$, where x_i is set to 1 if edge $\pi_i \rightarrow \pi_{i+1} \in \mathcal{E}$ and a dummy flip-flop is inserted between π_i and π_{i+1}, and to 0 otherwise, and
- $y_t, t \in \mathcal{T}, 0 < |E_t| \leq D$, where y_t is set to 1 if test t does not forbid any of the scan chain edges after inserting the D dummy flip-flops, and to 0 otherwise.[5]

The ILP formulation is the following:

$$\text{Max} \quad \sum_{t \in \mathcal{T},\ 0<|E_t| \leq D} w_t y_t \qquad (1)$$

$$\text{s.t.}$$

$$\sum_{i=1}^{n-1} x_i \leq D \qquad (2)$$

$$|E_t| y_t \leq \sum_{i \in E_t} x_i \quad \forall t \in \mathcal{T},\ 0 < |E_t| \leq D$$

$$x_i = 0, \quad \text{if } \pi_i \rightarrow \pi_{i+1} \notin \mathcal{E}$$

$$x_i \in \{0,1\}, \quad \text{if } \pi_i \rightarrow \pi_{i+1} \in \mathcal{E}$$

$$y_t \in \{0,1\} \quad \forall t \in \mathcal{T},\ 0 < |E_t| \leq D. \qquad (3)$$

It is easy to see that ILP (1) is equivalent to MCDI: constraint (2) ensures that no more than D dummy flip-flops are inserted, while constraints (3) make sure that a test t is counted by the objective function as covered only if dummy flip-flops have been inserted on all scan edges conflicting with it. The ILP (1) has compact size (at most $n + |\mathcal{T}| - 1$ binary variables and at most $|\mathcal{T}| + 1$ constraints), and, as shown by the results in Section IV, can be solved to optimality in practical running time by the commercial solver CPLEX. Further speedups can be achieved in practice by instructing CPLEX to stop as soon as it finds feasible solutions within a small factor of the optimum.

TABLE III
SCAN/TOTAL FAULT COVERAGE AND WIRELENGTH OF THE COMPARED FLOWS IN SINGLE CHAIN MODE WITHOUT DUMMY FLIP-FLOP INSERTION

Testcase	Measure	Flow I	Flow II	Flow III
s38417	Scan Coverage (%)	0.00	100.00	29.46
	Total Coverage (%)	4.74	100.00	32.80
	Wirelength (mm)	13.03	353.07	14.03
s13207	Scan Coverage (%)	5.11	100.00	100.00
	Total Coverage (%)	24.41	100.00	100.00
	Wirelength (mm)	4.16	68.11	11.86
s9234	Scan Coverage (%)	6.34	100.00	96.90
	Total Coverage (%)	49.83	100.00	98.34
	Wirelength (mm)	1.02	7.43	1.95
AES	Scan Coverage (%)	20.85	100.00	100.00
	Total Coverage (%)	66.78	100.00	100.00
	Wirelength (mm)	5.84	118.26	7.50
DES3	Scan Coverage (%)	92.11	100.00	100.00
	Total Coverage (%)	97.77	100.00	100.00
	Wirelength (mm)	1.39	12.24	1.73

IV. EXPERIMENTAL STUDY

In this section we describe our experimental setup and results. The testcases used in our experiment are described in Table II. Reported ILP runtimes are obtained using mixed integer programming solver from the CPLEX 7.0 package on a 300 MHz Sun Ultra-10 with 1 GB RAM. The three-phase MaxDFC-Scan heuristic and *ScanOpt* were run on a 2.4 GHz Intel Xeon server with 2 GB RAM.

Since vectors using launch-on-capture can be used to test faults for any scan order, we separated the paths that are testable using this method. We used a commercial automatic test pattern generation tool, *Synopsys TetraMAX*, to generate robust vectors using launch-on-capture for the testcases. We obtain a scan order using each of three different flows, and compare the final coverages and scan chain wirelengths.

---

[5] Tests which conflict with no scan chain edge ($|E_t| = 0$) are always going to be covered, while tests that conflict with more than D edges ($|E_t| > D$) cannot be made testable by inserting D or fewer dummy flip-flops. Consequently these tests are not considered in ILP (1).

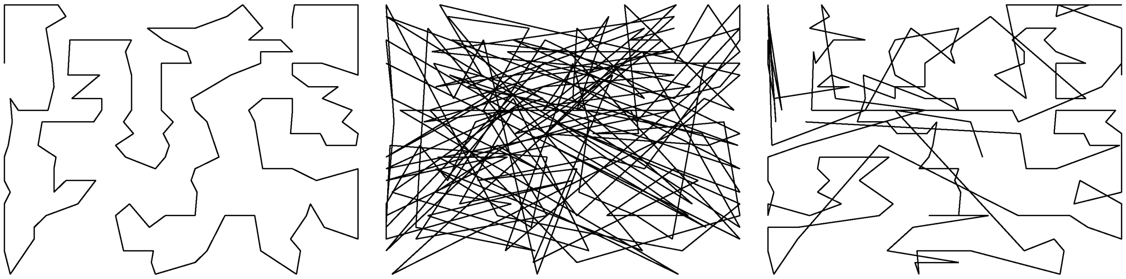


Fig. 8. Scan chains generated by Flows I, II, and III on testcase *s9234*.

For each of the testcases we conducted the experiments in the following way.

1) The Verilog RTL design is synthesized using *Synopsys Design Compiler* in an Artisan TSMC 0.13 μm library.[6]
2) The most critical paths and their sensitizing test patterns were found using *Synopsys PrimeTime*. We select the top 5000 critical paths or the paths that have a slack less than 30% of the clock period, whichever is less.[7] Then the true paths (as detected by PrimeTime) are selected and used for testing.
3) Robust vectors using launch-on-capture are generated for the synthesized netlist using *Synopsys TetraMAX*.[8] We consider only robust tests in our experiments since this type of test is guaranteed to detect excessive delay on the given path irrespective of timing on other paths. Robust tests can also be useful for characterizing the timing of a particular path, or for better diagnostic resolution of a failing path delay test. Note however that requiring only robust path delay fault tests will result in lower overall coverage.
4) Path sensitization vectors from *Synopsys PrimeTime* are used to construct test vector pairs to be applied using launch-on-shift. The paths covered using launch-on-capture in the previous step are excluded. Only this set of test vector pairs is passed on to the following scan chain ordering flows.
5) The synthesized design is placed with *Cadence PKS* to generate a placed DEF netlist.
6) We do the scan chain ordering using each of the following:
   - *Flow I*: Layout-driven scan chain ordering.

     Cell-to-cell distances from the placed netlist are used to drive the *ScanOpt* TSP solver [5]. If there are uncovered critical paths (among those not robustly testable via launch-on-capture), we perform optimal dummy flip-flop insertion by solving ILP (1) for the *ScanOpt* order.
   - *Flow II*: Layout-oblivious coverage-driven scan chain ordering.

     We use *ScanOpt* as the TSP solver to solve the 0/1 TSP generated as in [11] based on 100% coverage of the critical paths that are not robustly testable via launch-on-capture.
   - *Flow III*: Layout-aware coverage-driven scan chain ordering.

     We run our MaxDFC-Scan heuristic from Section II using test vector pairs returned by PrimeTime for critical paths that are not robustly testable via launch-on-capture. Placement information is used to generate the required number of ordered fragments, then the fragments are stitched into a single tour by inserting dummy flip-flops.
7) We calculate the fault coverage by finding the number of test vector pairs compatible with the generated scan order and report it. The scan chain wirelength is estimated in all flows by summing up cell-to-cell Manhattan distances between FF locations.

Table III gives the fault coverage and wirelength of the compared flows for zero dummy flip-flops inserted. The scan coverage rows show how many of the critical paths received as input by the three flows (i.e., of the critical paths that are not robustly testable using launch-on-capture) can be robustly tested by using launch-on-shift for the scan order produced by the corresponding flow. The total coverage rows show how many of the critical paths for which TetraMAX generates robust tests are testable using either launch-on-shift or launch-on-capture. On the reported testcases, all runtimes for *ScanOpt* range from 200 to 600 s (a substantial portion of which is spent reading the input data) while the MaxDFC-Scan heuristic runtimes range from 0.5 to 440 s.

As expected, Flow I has shortest wirelength, but poorest fault coverage. Flow II has 100% total fault coverage in all testcases, but uses as much as 25 times more wirelength than Flow I. Flow III achieves an excellent tradeoff between coverage and wirelength: it achieves 100% total fault coverage in five of the six testcases, with a wirelength comparable to that of Flow I (see also Fig. 8).

We put special emphasis on the zero dummy flop-flops case since we are able to achieve reasonably high coverage, and also since the insertion of a large number of dummy flip-flops im-

---

[6] The results reported in this section differ from the preliminary results reported in [13] since the designs were re-synthesized using the `map effort high` option. Although the absolute numbers are different, qualitatively the results remain similar to those in [13].

[7] Other methods for choosing the paths to be tested can be used as well. For instance, random paths may be added from different parts of the physical layout to better cover the entire defect space.

[8] *TetraMAX* options `set delay-diagnostic_propagation` and `add pi constraints 0 test_se` were used to get robust vectors using launch-on-capture.

TABLE IV
SCAN FAULT COVERAGE, CPLEX MIP NUMBER OF ITERATIONS, NUMBER OF BRANCH&BOUND NODES, AND RUNTIME FOR TESTCASE *s38417* FOR VARYING NUMBER OF DUMMY FLIP-FLOPS

#Dummy	Flow I				Flow III			
	Scan Coverage(%)	CPLEX			Scan Coverage(%)	CPLEX		
		#Iterations	#Nodes	Time (sec.)		#Iterations	#Nodes	Time (sec.)
0	0.00	0	0	0.00	29.46	0	0	0.00
5	1.87	39	1	0.02	36.31	70	0	0.03
10	7.47	1493	53	3.31	38.17	16744	303	25.04
15	13.07	19569	266	44.37	40.87	212413	11317	213.56
20	14.94	204563	7499	283.21	45.85	752862	50734	712.94
25	20.54	717894	15809	883.02	51.87	933741	76751	879.83
30	22.82	2792800	85469	3427.32	59.34	641835	21313	629.64
35	27.18	11306677	535700	14133.07	67.84	302460	6954	311.93
40	32.78	30872009	1734447	37637.56	76.56	24370	396	44.66
45	39.42	51103249	1707025	62099.02	85.68	2347	16	13.74
50	48.76	1650431	49248	2144.60	90.87	820	2	3.36
55	57.26	38629	656	101.44	95.85	1070	4	3.81
60	64.11	6078	94	82.81	99.17	219	3	0.19
65	69.50	3137	16	12.11	99.59	64	2	0.09
68	72.82	1330	2	15.55	100.00	0	0	0.01

plies significant overheads.[9] However, as shown by the results of our heuristic for the testcase *s38417* in Table IV, dummy flip-flop insertion results in drastic improvement in coverage.

The quality of scan orderings produced by Flow III is further reflected by the fact that 100% coverage for testcase *s38417* is achieved after inserting a smaller number of dummy flip-flops than the purely wirelength-driven Flow I (see Fig. 9 and Table IV). Flow I needs 152 dummy flip-flops to achieve 100% total coverage for *s38417*, while Flow III needs only 68 (the classic enhanced-scan methodology would indiscriminately enhance all 1564 flip-flops).

Table IV also gives more detailed information on the scalability of the CPLEX MIP solver for the MCDI problem. The MIP solver uses a branch and bound algorithm, and hence the runtime is greatly affected by the number of branch&bound iterations and the number of branch&bound nodes that are generated during the search. The table shows that, when CPLEX is run on testcase *s38417*, the number of iterations and branch&bound nodes depends on the given upperbound on number of dummy flip-flops, as well as the quality of the underlying scan ordering. For very small or very large dummy flip-flop upperbounds, the ILP is easy to solve and CPLEX needs few branch and bound iterations. For intermediate upperbound values the branch&bound tree grows larger and more iterations are needed to prove optimality. Nevertheless, the runtime required to insert the optimum set of dummy flip-flops for the scan chain order produced by Flow III remains acceptable for the whole range of upperbound values.

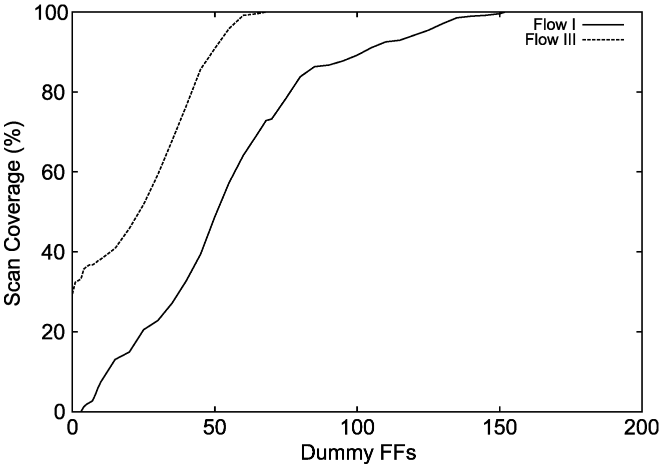


Fig. 9. Scan fault coverage (in percents) on testcase *s38417* as function of added dummy flip-flops.

Table V gives the results obtained by running the compared flows on testcase *s38417* without dummy flip-flop insertion but with varying number of scan chains. For multiple scan chains the three flows achieve wirelength/coverage tradeoffs similar to those achieved in the case of a single scan chain. When the number of scan chain increases, the wirelength of Flow II improves slightly, but still remains much longer than that of Flows I and III. In our experiments, the fault coverage of Flow III improved when increasing the number of scan chains, but wirelength also increased slightly.

---

[9] When used, dummy flip-flops are typically inserted at spare sites available in the design. Spare sites are selected for each dummy flip-flop by solving a classic minimum cost assignment problem [2], in which the cost of assigning a spare site to a scan chain edge selected for dummy flip-flop insertion is equal to the detour wirelength.

TABLE V
EFFECT OF THE NUMBER OF SCAN CHAINS ON SCAN/TOTAL FAULT COVERAGE AND WIRELENGTH FOR TESTCASE *s38417* (FLOWS RAN WITHOUT DUMMY FLIP-FLOP INSERTION)

Flow	Flow I			Flow II			Flow III		
#Scan chains	1	2	3	1	2	3	1	2	3
Scan Coverage (%)	0.00	0.00	0.00	100.00	100.00	100.00	29.46	33.40	37.75
Total Coverage (%)	4.74	4.74	4.74	100.00	100.00	100.00	32.80	36.56	40.71
Wirelength (mm)	13.03	14.09	14.72	353.07	334.89	281.96	14.03	16.06	16.33

V. CONCLUSION

In this work we have proposed algorithms for computing the achievable tradeoffs in scan chain synthesis between number of dummy flip-flops, scan chain wirelength, and path delay fault coverage. Our layout-aware coverage-driven scan chain ordering methodology yields significant improvements in path delay coverage with a very small increase in wirelength overhead compared to previous layout-driven scan chain ordering approaches. Also, our method yields similar coverage and up to 25 times improvement in wirelength compared to previous layout-oblivious coverage-driven scan chain ordering approaches.

Our ongoing work seeks to extend the applicability of the proposed algorithms to redundant test vector pairs. We also explore the possibility to exploit additional degrees of freedom such as the ability to select the flip-flop data output pins used to connect scan chain edges, and to improve routability of resulting scan chains by using the available congestion information. Finally, we are integrating our methods with dummy flip-flop placement and detailed routing to confirm that estimated wirelength savings reported in Section IV correspond to actual (post-detailed routing) wirelength savings.

ACKNOWLEDGMENT

The authors would like to thank J. Abraham, R. Kapur, W. B. Maloney, R. Parekhj, and T. Williams for useful discussions and S. Mantik for help in obtaining the testcases. They also thank the anonymous reviewers for many useful comments that have helped us improving the presentation.

REFERENCES

[1] V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A tutorial on built-in self-test, part 2: applications," *IEEE Des. Test Comput.*, vol. 10, no. 2, pp. 69–77, 1993.

[2] R. K. Ahuja, T. L. Magnanti, and J. L. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[3] S. Barbagello, M. L. Bodoni, D. Medina, F. Corno, P. Prinetto, and M. S. Reorda, "Scan-insertion criteria for low design impact," in *Proc. VLSI Test Symp.*, 1996, pp. 26–31.

[4] Z. Barzalai and B. K. Rosen, "Comparison of AC self-testing procedures," in *Proc. Int. Test Conf.*, 1983, pp. 89–94.

[5] K. D. Boese, A. B. Kahng, and R. S. Tsay, "Scan chain optimization: Heuristic and optimal solutions," UCLA CS Dept. Internal Rep., Oct. 1994.

[6] C. S. Chen and T. T. Hwang, "Layout driven selection and chaining of partial scan flip-flops," *J. Electron. Test.: Theory Appl.*, vol. 13, pp. 19–27, 1998.

[7] K.-T. Cheng, S. Devadas, and K. Keutzer, "Delay-fault test generation and synthesis for testability under a standard scan design methodology," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 12, pp. 1217–1231, 1993.

[8] M. Feuer and C. C. Koo, "Method for rechaining shift register latches which contain more than one physical book," *IBM Tech. Disclosure Bulletin*, vol. 25, no. 9, pp. 4818–4820, 1983.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[10] C. T. Glover and M. R. Mercer, "A method of delay fault test generation," in *Proc. ACM/IEEE Design Automation Conf.*, 1988, pp. 90–95.

[11] P. Gupta, J. Abraham, and R. A. Parekhji, "Improving path delay coverage in embedded cores—methodology and experiments," presented at the *Texas Instruments Symp. Test*, 2001.

[12] P. Gupta, A. B. Kahng, and S. Mantik, "Routing-aware scan chain ordering," in *Proc. Asia and South-Pacific Design Automation Conf.*, 2003, pp. 857–862.

[13] P. Gupta, A. B. Kahng, I. I. Màndoiu, and P. Sharma, "Layout-aware scan chain synthesis for improved path delay fault coverage," in *Proc. IEEE-ACM International Conf. Computer-Aided Design*, 2003, pp. 754–759.

[14] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *Eur. J. Operations Research*, vol. 12, pp. 106–130, 2000.

[15] D. S. Johnson and L. A. McGeoch, "Experimental analysis of heuristics for the stsp," in *The Traveling Salesman Problem and Its Variations*, G. Gutin and A. Punnen, Eds. Dordrecht, Germany: Kluwer, 2002, pp. 369–443.

[16] S. Kobayashi, M. Edahiro, and M. Kubo, "A vlsi scan-chain optimization algorithm for multiple scan-paths," *IEICE Trans. Fundamentals*, vol. E82-A, no. 11, pp. 2499–2504, 1999.

[17] A. Krstic, S. T. Chakradhar, and K.-T. Cheng, "Testable path delay fault cover for sequential circuits," in *Proc. EURO-DAC*, 1996, pp. 220–226.

[18] K.-H. Lin, C.-S. Chen, and T. T. Hwang, "Layout driven chaining of scan flip-flops," *IEE Proc. Computers Digital Techn.*, vol. 143, no. 6, pp. 421–425, 1996.

[19] S. Makar, "A layout based approach for ordering scan chain flip-flops," in *Proc. Intl. Test Conf.*, 1998, pp. 341–347.

[20] Y. K. Malaiya and R. Narayanaswamy, "Testing for timing faults in synchronous sequential integrated circuits," in *Proc. Intl. Test Conf.*, 1983, pp. 560–571.

[21] W. Mao and M. D. Ciletti, "Arrangement of latches in scan-path design to improve delay fault coverage," in *Proc. Int. Test Conf.*, 1990, pp. 387–393.

[22] R. B. Norwood and E. J. McCluskey, "delay testing for sequential circuits with scan," Center for Reliable Computing, Stanford University, Stanford, CA, Tech. Rep. CRC TR 97-5, 1976.

[23] S. M. Reddy, C. J. Lin, and S. Patil, "An automatic test pattern generation for the detection of path delay faults," in *Proc. IEEE-ACM Int. Conf. Computer-Aided Design*, 1987, pp. 284–287.

[24] J. Savir, "Skewed-load transition test: part I, calculus," in *Proc. Int. Test Conf.*, 1992, pp. 705–713.

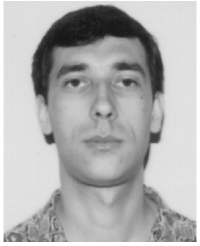
[25] CPLEX Optimization Suite. [Online]. Available: http://www.ilog.com/products/cplex

Puneet Gupta received the B.Tech. degree in electrical engineering from Indian Institute of Technology, New Delhi, India, in 2000. He is currently pursuing the Ph.D. degree in the Electrical and Computer Engineering Department at University of California at San Diego, La Jolla.

He worked at Mindtree Technologies Pvt. Ltd., Bangalore, India, as a VLSI Design Engineer from 2000 to 2001. He is recipient of IBM Ph.D. Fellowship and has given an embedded tutorial on Manufacturing-Aware Physical Design at ICCAD 2003. He has published 18 papers in the past three years in the fields of design-manufacturing interface, layout aware scan chain synthesis and power/thermal considerations in layout. His primary research interests lie in the impact of manufacturing variations on performance and yield and design-aware mask flow.

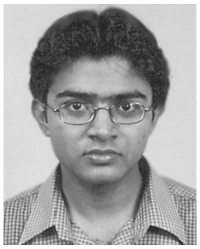
Andrew B. Kahng received the A.B. degree in applied mathematics is from Harvard University, Harvard University, Cambridge, MA, and the M.S. and Ph.D. degrees in computer science from University of California (UC) at San Diego, La Jolla.

He is currently is Professor of computer science engineering and electrical computer engineering at UC San Diego. He has published over 200 papers in the VLSI CAD literature, receiving three Best Paper awards and an NSF Young Investigator award. From 1989 to 2000, he was a member of the UC Los Angeles Computer Science Faculty. His research is mainly in physical design and performance analysis of VLSI, as well as the VLSI design–manufacturing interface. Other research interests include combinatorial and graph algorithms, and large-scale heuristic global optimization. Since 1997, he has defined the physical design roadmap for the International Technology Roadmap for Semiconductors (ITRS), and since 2001 has chaired the U.S. and international working groups for design technology for the ITRS. He has been active in the MARCO Gigascale Silicon Research Center since its inception. He was also the founding General Chair of the ACM/IEEE International Symposium on Physical Design, and co-founded the ACM Workshop on System-Level Interconnect Planning.



Ion I. Măndoiu received the M.S. degree from Bucharest University, Romania, in 1992 and the Ph.D. degree from Georgia Institute of Technology, Atlanta, in 2000, both in computer science.

He is currently an Assistant Professor with the Computer Science and Engineering Department, University of Connecticut, Storrs. His research focuses on the design and analysis of exact and approximation algorithms for NP-hard optimization problems, particularly in the areas of VLSI computer aided design, bioinformatics, and ad hoc wireless networks. He authored over 40 refereed scientific publications in these areas, including a Best Paper at the joint Asia-South Pacific Design Automation/VLSI Design Conferences.



Puneet Sharma received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi, India, in 2002, and is currently working toward the Ph.D. degree in the Electrical and Computer Engineering Department, University of California at San Diego, La Jolla.

His research interests include delay fault testing, power reduction, and design for manufacturing.