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

Puneet Gupta Andrew B. Kahng Ion Măndoiu† Puneet Sharma

ECE Department, University of California at San Diego, La Jolla, CA 92093
†CSE Department, University of Connecticut, Storrs, CT 06269
E-mail: {puneet,abk}@ucsd.edu, ion@engr.uconn.edu, sharma@vlsicad.ucsd.edu

Abstract

Path delay fault testing becomes 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 consider both dummy flip-flop and wirelength costs, and focus on post-layout formulations that capture the achievable tradeoffs between these costs and delay fault coverage in scan chain synthesis.

1 Introduction

Scan-based path delay fault testing requires the application of two test vectors: the first test vector, or *initialization vector*, initializes the logic to a known state while the second vector, or *activation vector*, activates the targeted fault, causing a transition to be propagated along the path under test.[1] It is well-known that at-speed application of test vector pairs to the primary inputs has low path delay fault coverage [4]. Improved coverage can be achieved by using scan chaining, which has become the design-for-test (DFT) technique of choice for stuck-at fault testing [1].

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). Standard scan-based delay fault testing involves justifying a *test vector* by giving clocks to the circuit placed in *test mode*, giving one (scan shifting) or two (functional justification) clock(s) to the circuit in *normal mode*, then shifting out the resulting flip-flop values by giving clocks in test mode.

There are two techniques to produce the vector pairs for path delay fault testing - *functional justification* and *scan shifting*. With functional justification, 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. Some pros and cons of the two techniques are as follows.

- It is known that test generation complexity using scan shifting is typically lower than that using functional justification. To save test generation time, vectors may be generated using scan shifting first, and functional justification may be used for faults that cannot be tested by scan shifting. 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 which cannot be detected by functional justification are likely to be functionally false paths, but identification of functionally untestable paths is a hard problem [16]. Several faults that cannot be detected using functional justification (by commercial ATPG tools) may be detected by scan shifting.
- Vectors generated using functional justification have the advantage of being scan order independent. This allows scan order to be driven by layout such that the wirelength is minimized. However, there may be multiple equi-wirelength scan chain orderings, some of which may be conducive to scan justification based path delay testing. It is therefore possible to increase fault coverage with little or no impact on layout overhead of scan.

The requirement that the activation vector must be obtained from the initialization vector by one-bit shifting along the scan chain [23] constrains scan chain synthesis for delay fault testing using scan shifting. 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. Under the standard practice of using a single scan-enable signal, with scan chain edges always linking 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 with 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 [21], 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 [21] 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 $i \to j$ can be made compatible with all conflicting tests either by "enhancing" flip-flop j to store an additional bit or by inserting a separate 1-bit flip-flop between i and j. We will refer to this operation as *inserting a dummy flip-flop* in the edge $i \to j$.

*Work partially supported by Cadence Design Systems, Inc. and the MARCO Gigascale Silicon Research Center. The work of I.M. was performed while he was with the ECE Department at Univeristy of California, San Diego.

[1] More general tests, such as *validatable non-robust tests* [22], require the application of a *set* of test vector pairs. Our algorithms can handle such tests after minor modifications.

An early approach to ensuring high-coverage delay fault testing is the so-called "enhanced-scan" [19, 10], in which all scan flip-flops are augmented by dummy flip-flops. 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 [20], is to use standard scan flip-flops and order 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, [21] 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 high fault coverage, a significant concern in scan synthesis is the wirelength overhead, since excessive scan wirelength can compromise the routability of the design and degrade its performance. While this overhead has received considerable attention in the context of stuck-at fault testing [3, 5, 6, 8, 12, 15, 17, 18], previous works on scan synthesis for delay fault testing have focused on maximizing delay fault coverage without regard to any scan overheads [20], achieving a certain coverage factor with minimum number of dummy flip-flops but without regard of wirelength cost [7, 21], or achieving full coverage regardless of a potentially high wirelength cost [11].

In this paper we consider both dummy flip-flop and wirelength costs and focus on *post-layout* formulations that capture the achievable tradeoffs between these costs and delay fault coverage in scan chain synthesis. 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:

- 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 ILP formulation for the problem of optimally inserting a number of dummy flip-flops in a given scan chain. This ILP is solved in practical runtime using the CPLEX commercial optimizer for designs with up to tens of thousands of scan flip-flops.
- 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 number of dummies and scan chain wirelength on the other hand.

The rest of our paper is organized as follows. In Section 2, 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 3 we prove the NP-hardness of, and give a compact ILP formulation for, the problem of computing achievable tradeoffs between delay fault coverage and the number of dummy flip-flops inserted in an already routed scan chain. Finally, we present experimental results in Section 4 and conclude in Section 5.

2 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 of m delay fault tests $\mathcal{T}$

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 coverage (i.e., the number of edges in π that conflict with at least one test of $\mathcal{T}$) 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.). A formulation that captures this tradeoff is the following:

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 of m delay fault tests $\mathcal{T}$ and positive weights w_t, $t \in \mathcal{T}$ (The weights w_t represent the number – or possibly average criticality – of faults tested by test t; multiple faults per test vector pair are common due to the use of test vector compaction in ATPG.)
- 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 of covered tests $\mathcal{C} \subseteq \mathcal{T}$

Such that:

- At most D scan chain edges $\pi_i \rightarrow \pi_{i+1}$ conflict with tests 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, 20, 21], and therefore is NP-hard. Thus, we cannot expect to find polynomial-time algorithms that solve MaxDFC-Scan optimally in polynomial time [9]. We now present a MaxDFC-Scan heuristic which can efficiently handle instances with tens of thousands of scan flip-flops and thousands of test vector pairs arising in today's high-end designs.

The Three-Phase MaxDFC-Scan Heuristic. Our heuristic for MaxDFC-Scan (Figure 1) runs in three phases. In the first phase we construct a set of $D+1$ scan chain fragments using a multi-fragment greedy heuristic (Figure 2) similar to that used in the TSP literature [14]. Since the edges within each of the $k+1$ fragments will not be augmented by dummy flip-flops, we want them to be compatible with as many faults as possible. Therefore, the multi-fragment greedy heuristic attempts to use the edges in the order of decreasing number of conflicting tests. Note that the number of conflicting tests changes during the algorithm, since once a fault

Input: Set of n flip-flops F, scan-in/scan-out pins SI and SO, set of m delay fault tests $\mathcal{T}$, maximum number D of dummy flip-flops
Output: Scan chain ordering π of $F \cup \{SI, SO\}$ starting with SI and ending with SO and set of covered tests $\mathcal{C} \subseteq \mathcal{T}$

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 tests $\mathcal{C}$.

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 return the ATSP path from SI to SO computed by running on G', e.g., LKH [13] or ScanOpt [5]. Insert dummy flip-flops on the edges of the path, then use these edges to stitch the fragments $P_0, \ldots, P_{D+1}$ into a scan chain π.

Phase 3: Construct the auxiliary graph G'' by adding to π the edges compatible with all faults in $\mathcal{C}$. Assign edge-costs given by pin-to-pin wirelength, then return the ATSP path from SI to SO computed by running on G'', e.g., LKH [13] or ScanOpt [5].

Figure 1: The three-phase MaxDFC-Scan heuristic.

is made untestable by the inclusion of an edge into the chain fragments, it should no longer be counted as conflicting with the remaining edges.

Simultaneously, the multi-fragment greedy heuristic also 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. To consider both wirelength and coverage simultaneously, we rank the edges according to a weighted combination of length normalized by the average length, and number of incompatible faults (see Step 2 in Figure 1). First, the algorithm considers edges with a weighted combination value below a threshold value T (we use an initial threshold of 1.4 in our experiments). The parameter w determines the relative weight of normalized length vs. lost coverage, and can be modified to achieve different tradeoffs between the wirelength and the coverage of the final tour. In our experiments we use $w = 2.0$. The fragments are then extended iteratively using these edges (edges with least number of incompatible faults first, and breaking ties based on wirelength). When no edges are left, we increase the threshold T by a multiplicative factor (we use $f = 1.2$), and attempt to grow the fragments in the same manner using the edges that now become eligible, i.e., have a weighted combination of length and number of incompatible faults below T. The experimental conclusions that we report below are not very sensitive to the choice of the T and f parameter values.

In the second phase, we combine the $D+1$ fragments 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 ATSP solver (even high-quality solvers such as LKH [13] can be used in practice since the number of fragments is small).

In the third phase, all edges compatible with surviving faults are added to the tour, and an ATSP solver is called to further decrease the length of the tour and possibly remove some of the dummy buffers.

Extension to Multiple Scan Chains. Multiple scan chains are known to reduce the testing time significantly. Our heuristics are easily modifiable to take care of multiple scan chains. Assuming that all flip-flops are labeled by the scan chain they belong to,[2] 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

[2]This may be done by a DFT tool like *Synopsys DFT Compiler*.

Input: Set of n flip-flops F, set of m delay fault tests $\mathcal{T}$, upperbounds D, weight w, threshold T and increment factor f
Output: $D+1$ scan chain fragments and set of covered tests $\mathcal{C} \subseteq \mathcal{T}$

---

1. Initializations:

For each $t \in \mathcal{T}$, $E_t \leftarrow \emptyset$

For each $e \in E$, $\mathcal{T}_e \leftarrow \emptyset$

For each $(i,j) \in E$, $\ell(i,j) \leftarrow$ length of edge (i,j)

For each $(i,j) \in E$ and $t \in \mathcal{T}$, if t forbids (i,j) then $E_t \leftarrow E_t + (i,j)$ and $\mathcal{T}_{(i,j)} \leftarrow \mathcal{T}_{(i,j)} + t$

$E' \leftarrow \emptyset$; $\mathcal{C} \leftarrow \mathcal{T}$

2. Distribute all edges $(i,j) \in E$ with
$w * \frac{\ell(i,j)}{\text{Average } \ell(e),\, e \in E} + |\mathcal{T}_e| < T$
into buckets based on $\sum_{t \in \mathcal{T}_e} w_t$

3. While $|E'| < |F| - D - 1$ do

If all buckets are empty then $T \leftarrow f * T$ and goto 2
Else, select a shortest edge (i,j) from the lowest non-empty bucket and delete it from E;

If (i,j) does not create a cycle or a vertex of degree greater than 2 with the other edges of E' then

For each $t \in \mathcal{T}_{(i,j)}$ and each $e \in E_t$ remove t from $\mathcal{T}_e$ and update e's bucket accordingly

$E' \leftarrow E' + (i,j)$

$\mathcal{C} \leftarrow \mathcal{C} - \mathcal{T}_{(i,j)}$

3. Return $\mathcal{C}$ and the $D+1$ paths formed by the edges of E'

Figure 2: The multi-fragment greedy algorithm - Phase 1

added, and there are k scan chains, then the algorithm stops with $D+k$ fragments. Such a modification is likely to speedup the multi-fragment greedy heuristic since fewer edges are considered. Phase 2 inserts the dummy flip-flops in the scan chains and labels them with the scan chain they belong to. Phase 3 is performed on each scan chain independently.

3 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 2). 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 optimally guaranteed by inserting one dummy flip-flop in each scan chain edge that conflicts to at least one test; the required set of dummies can be computed in $O(nm)$ time. In practice it is useful to impose an upperbound on the number of inserted dummies while maximizing path delay fault coverage.[3] This motivates the following problem formulation:

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

[3]An alternate formulation seeks a minimum number of dummies that guarantee a certain fault coverage [7].

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	

Table 1: Flip-flop values used in proof of Theorem 1

- Set of m delay fault tests $\mathcal{T}$ and non-negative 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 of covered tests $\mathcal{C} \subseteq \mathcal{T}$

Such that:

- None of the scan chain edges conflicts with the tests in $\mathcal{C}$ after dummy insertion
- The total weight of tests 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 $f_u \to f_{u+1}$ and $f_v \to f_{v+1}$ 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 1. It is easy to see that G has a clique of size k if and only if $D = k$ dummies 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 (ILP) formulation for the MCDI problem. This formulation can be optimally solved in practical running time using commercial general-purpose ILP solvers such as CPLEX even for very large testcases (scan chains with tens of thousands of flip-flops). Let E_t be the set of scan chain edges conflicting with test t. MCDI can be formulated as an integer linear program (ILP) by using two sets of 0/1 variables:

- x_i, $i = 1, \ldots, n$, where x_i is set to 1 if edge $\pi_i \to \pi_{i+1} \in \mathcal{E}$ and a dummy flip-flop is inserted between π_i and π_{i+1}, and to 0 otherwise, and
- y_t, $t \in \mathcal{T}$, $0 < |E_t| \le D$, 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.[4]

[4]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 dummies. Consequently these tests are not considered in ILP (1).

The ILP formulation is the following:

$$\text{Max} \quad \sum_{t \in \mathcal{T},\ 0 < |E_t| \le D} w_t y_t \tag{1}$$

s.t.

$$\sum_{i=1}^{n-1} x_i \le D \tag{2}$$

$$|E_t| y_t \le \sum_{i \in E_t} x_i \quad \forall\, t \in \mathcal{T}, 0 < |E_t| \le D \tag{3}$$

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

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

$$y_t \in \{0,1\} \quad \forall\, t \in \mathcal{T}, 0 < |E_t| \le D$$

It is easy to see that ILP (1) is equivalent to MCDI: constraint (2) ensures that no more than D dummies are inserted, while constraints (3) make sure that a test t is counted by the objective function as covered only if dummies have been inserted on all scan edges conflicting with it. 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 4, can be solved to optimality in practical running time by the commercial solver CPLEX. If needed, significant speedups can be achieved in practice by instructing CPLEX to stop as soon as it finds feasible solutions known to be within a small cost of the optimum.

4 Experimental Study

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

Test Case	Source	# Cells	# Scan FFs	# Paths	Functional Coverage
eth	Industry	19575	9841	2792	0%
s38417	ISCAS'89	6291	1564	806	48.76%
s13207	ISCAS'89	1648	627	1563	45.68%
s9234	ISCAS'89	529	145	456	35.31%
AES	opencores.org	10465	554	441	76.19%
DES3	opencores.org	3912	128	1078	73.75%

Table 2: Test Case Parameters

Since vectors using functional justification can be used to test faults irrespective of the scan order, we separate the paths that are testable by functionally justified vectors. We use a commercial ATPG tool, *Synopsys TetraMAX* to generate robust vectors using functional justification for the testcases. We obtain a scan order using each of three different flows, and compare the final coverages and scan chain wirelengths.

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.
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. Then the true paths (as detected by PrimeTime) are selected and used for testing.

3. Robust vectors using functional justification are generated for the synthesized netlist using *Synopsys TetraMAX*.[5] 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 robust vectors to be applied using scan justification. The paths tested using functional justification in the previous step are excluded. Only this set of test vectors is passed on to the 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:* Placement driven scan chain ordering flow. 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 functional justification, we perform optimal dummy flip-flop insertion by solving ILP (1) for the *ScanOpt* order.
   - *Flow II:* Test driven scan chain ordering flow. 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 functional justification.
   - *Flow III:* Layout aware test driven scan chain ordering. We run our multi-fragment greedy heuristic using the robust tests returned by PrimeTime for critical paths that are not robustly testable via functional justification. 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 as described in Section 2.

7. We calculate the coverage by finding the number of faults 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 3 gives the path coverage and wirelength of the compared flows for zero dummies inserted. The scan coverage rows show how many of the critical paths received as input by the 3 flows (i.e., of the critical paths that are not robustly testable using functional justification) can be robustly tested by using the scan order produced by each flow. The total coverage rows show how many of the critical paths for which TetraMAX generates robust tests are testable using either scan or functional justification. On the reported testcases, all runtimes for *ScanOpt* range from 200 to 600 seconds, but are in some sense not comparable directly since alternative flows either read in and solve, or simply solve, the ATSP instance; the read-in time is a substantial portion of total runtime in the former case. MFG runtimes range from 0.5 to 440 seconds.

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× more wirelength than Flow I. Flow III

Testcase	Measure	Flow I	Flow II	Flow III
eth	Scan Coverage (%)	38.50	100.00	100.00
	Total Coverage (%)	38.50	100.00	100.00
	Wirelength (mm)	74.33	852.98	77.69
s38417	Scan Coverage (%)	0.00	100.00	19.61
	Total Coverage (%)	48.76	100.00	58.81
	Wirelength (mm)	13.06	365.12	13.95
s13207	Scan Coverage (%)	68.90	100.00	100.00
	Total Coverage (%)	83.12	100.00	100.00
	Wirelength (mm)	5.05	80.13	5.27
s9234	Scan Coverage (%)	33.22	100.00	100.00
	Total Coverage (%)	56.80	100.00	100.00
	Wirelength (mm)	1.22	10.09	1.46
AES	Scan Coverage (%)	100.00	100.00	100.00
	Total Coverage (%)	100.00	100.00	100.00
	Wirelength (mm)	5.78	123.07	5.78
DES3	Scan Coverage (%)	86.57	100.00	100.00
	Total Coverage (%)	96.47	100.00	100.00
	Wirelength (mm)	1.39	11.75	1.43

Table 3: Scan coverage and wirelength of the compared flows for 0 dummies inserted.

#Dummy	Flow I		Flow III	
	Coverage(%)	Time(s)	Coverage(%)	Time(s)
0	0.00	0.04	19.61	2.94
5	0.97	0.05	41.40	2.97
10	3.39	0.31	66.10	2.88
15	9.69	3.10	75.30	2.91
20	13.32	70.70	84.02	2.90
25	16.22	330.64	88.86	2.86
30	22.52	623.88	94.67	2.85
35	34.38	361.89	97.58	2.84
40	45.28	135.92	98.31	2.81
45	53.75	176.80	99.76	2.84
50	67.55	25.19	100.00	0.51

Table 4: Fault coverage using scan shifting on testcase *s38417* as function of added dummies. Time reported for flow I is the time taken by the CPLEX ILP implementation for dummy insertion.

achieves an excellent tradeoff between coverage and wirelength: it achieves 100% total fault coverage in 5 of the 6 testcases, with a wirelength comparable to that of Flow I (see also Figure 3).

We put special emphasis on the zero dummies case since we are able to achieve reasonably high coverages, and also since ECO insertion of a large number of dummies implies significant overheads.[6] In some cases, dummy insertion results in drastic improvement in coverage as shown by the results of our heuristic for the testcase *s38417* in Table 4.

The quality of flip-flop orderings produced by Flows II and III is reflected by the fact that 100% coverage for testcase *s38417* is achieved after inserting only a small number of dummies. In contrast, the purely wirelength-driven Flow I needs over 100 dummy flip-flops to achieve 100% total coverage, and the classic enhanced-scan methodology would indiscriminately enhance all 1564 flip-flops. Note that, as the number of dummies increases, the wirelength of scan chains produced by Flows I and II remains constant, while the wirelength cost incurred by Flow III is slightly decreasing due to the second-phase optimization.

Table 4 also shows the runtime needed by CPLEX to solve the

[5] The options `set delay -diagnostic_propagation` and `add pi constraints 0 test_se` are used to get robust vectors using functional justification.

[6] 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 insertion is equal to the detour wirelength (recall that Phase 2 of the MaxDFC heuristic in Figure 1 relies on point-to-point distances).

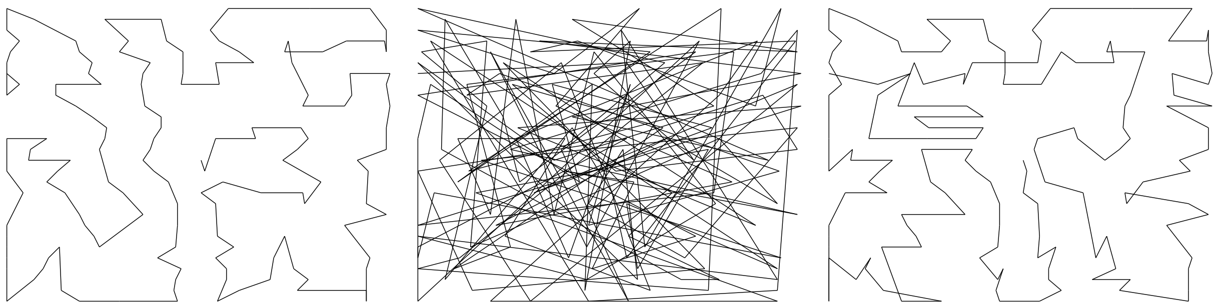

Figure 3: Scan chains generated by Flows I, II & III on testcase *s9234*.

ILP in Section 3 for testcase *s38417*. The table shows that CPLEX runtime is dependent on the bound on the number of dummies. For very small or very large bounds the ILP is easy to solve and the branch and bound algorithm needs few iterations. For intermediate bound values the branch and bound tree grows larger and more iterations are needed to prove optimality. However, even in this case the runtime remains acceptable.

5 Conclusions

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. With our layout and test-aware scan chain ordering methodology, we see up to 200% improvement in path delay coverage with just 20% increase in wirelength overhead compared to a layout-driven scan chain ordering approach. Also, up to 25x improvement in wirelength is achieved on the testcases compared to a test-driven scan chain ordering approach.

Ongoing work seeks to extend these algorithms to redundant test-vector pairs, to exploit additional degrees of freedom such as selection of 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. As discussed above, extensions to multiple scan chains are also possible. We are also integrating our methods with dummy flip-flop placement and detailed routing to confirm that estimated wirelength savings reported in Section 4 correspond to actual (post-detailed routing) wirelength savings.

Acknowledgements

We thank Jais Abraham, Rohit Kapur, William B. Maloney, Rubin Parekhji and Tom Williams for useful discussions and Stefanus Mantik for help in obtaining the testcases. We 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 Design and Test of Computers*, 10(2):69–77, 1993.

[2] R.K. Ahuja, T.L. Magnanti, and J.L. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ,, 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 Symposium*, pages 26–31, 1996.

[4] Z. Barzalai and B.K. Rosen. Comparison of AC self-testing procedures. In *Proc. Intl. Test Conference*, pages 89–94, 1983.

[5] K.D. Boese, A.B. Kahng, and R.S. Tsay. Scan chain optimization: Heuristic and optimal solutions. Internal Report, UCLA CS Dept., October 1994. Downloadable from *http://www.gigascale.org/bookshelf/Slots/ScanOpt/*.

[6] C.S. Chen and T.T. Hwang. Layout driven selection and chaining of partial scan flip-flops. *Journal of Electronic Testing: Theory and Applications*, 13:19–27, 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 Transactions on Computer-Aided Design*, 12:1217–1231, 1993.

[8] M. Feuer and C. C. Koo. Method for rechaining shift register latches which contain more than one physical book. *IBM Technical Disclosure Bulletin*, 25(9):4818–4820, 1983.

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

[10] C.T. Glover and M.R. Mercer. A method of delay fault test generation. In *ProceedingsACM/IEEE Design Automation Conference*, pages 90–95, 1988.

[11] P. Gupta, J. Abraham, and R.A. Parekhji. Improving path delay coverage in embedded cores – methodology and experiments. In *Texas Instruments Symp. on Test*, 2001.

[12] P. Gupta, A.B. Kahng, and S. Mantik. Routing-aware scan chain ordering. In *ProceedingsAsia and South-Pacific Design Automation Conference*, pages 857–862, 2003.

[13] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operations Research*, 12:106–130. Code available at *http://www.dat.ruc.dk/∼keld/research/LKH/*., 2000.

[14] D.S. Johnson and L.A. McGeoch. Experimental analysis of heuristics for the stsp. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443, Dordrecht, 2002. Kluwer Academic Publishers.

[15] S. Kobayashi, M. Edahiro, and M. Kubo. A vlsi scan-chain optimization algorithm for multiple scan-paths. *IEICE Trans. Fundamentals*, E82-A(11):2499–2504, 1999.

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

[17] K.-H. Lin, C.-S. Chen, and T.T. Hwang. Layout driven chaining of scan flip-flops. *IEE Proc., Computers and Digital Techniques*, 143(6):421–425, 1996.

[18] S. Makar. A layout based approach for ordering scan chain flip-flops. In *Proc. Intl. Test Conference*, pages 341–347, 1998.

[19] Y.K. Malaiya and R. Narayanaswamy. Testing for timing faults in synchronous sequential integrated circuits. In *Proc. Intl. Test Conference*, pages 560–571, 1983.

[20] W. Mao and M.D. Ciletti. Arrangement of latches in scan-path design to improve delay fault coverage. In *Proc. Int. Test Conference*, pages 387–393, 1990.

[21] R.B. Norwood and E.J. McCluskey. Delay testing for sequential circuits with scan. Technical Report CRC TR 97-5, Center for Reliable Computing, Stanford University, 1976, available at *http://www-crc.stanford.edu/crc_papers/CRC-TR-97-5.pdf*.

[22] S.M. Reddy, C.J. Lin, and S. Patil. An automatic test pattern generation for the detection of path delay faults. In *Proceedings IEEE-ACM International Conference on Computer-Aided Design*, pages 284–287, 1987.

[23] J. Savir. Skewed-load transition test: Part I, Calculus. In *Proc. Intl. Test Conference*, pages 705–713, 1992.