Test Time Minimization in Reconfigurable Scan Networks

R. Cantoro, M. Palena, P. Pasini, M. Sonza Reorda
Politecnico di Torino - Torino, Italy

***Abstract*—Modern devices often include several embedded instruments, such as BISTs, sensors, and other analog components. New standards, such as IEEE Std. 1687, provide vehicles to access these instruments. In approaches based on reconfigurable scan networks, instruments are coupled with scan registers, connected into chains and interleaved with reconfigurable multiplexers, permitting a selective access to different parts of the chain. A similar scenario is also supported by IEEE Std. 1149.1-2013, where a test data register can be constructed as a chain of multiple segments, some of which can be excluded or mutually selected. The test of permanent faults affecting a reconfigurable scan network requires to shift test patterns throughout a certain number of network configurations. This paper presents a method to select the list of configurations needed to apply the complete test set in the minimum amount of clock cycles. The method is based on a graph representation of the problem. Experimental results on some benchmark networks are provided, together with a comparison with other approaches based on heuristics. The provided results can be effectively used to evaluate the test time of sub-optimal approaches.**

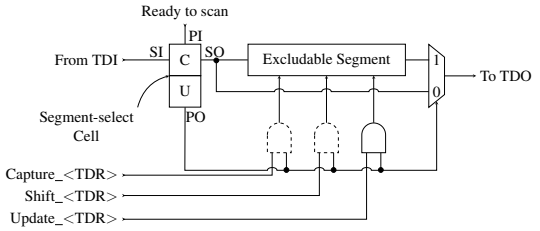

Fig. 1. Excludable TDR segment described in IEEE Std 1149.1-2013.

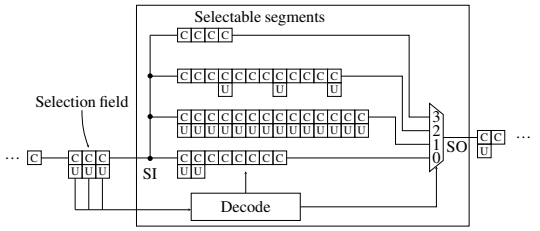

Fig. 2. Selectable TDR segments described in IEEE Std 1149.1-2013

I. Introduction

Due to the complexity of new electronic devices, several features are embedded in such systems beside the core functional logic. Examples of such features are Built-In Self-Test (BIST) modules included for test and diagnostic sake, interfaces to core testing (e.g., based on the IEEE Std 1500), analog components (e.g., temperature sensors) accessed during the chip calibration, or debug modules (e.g., trace buffers). The access to these features often happen through registers, hereinafter called *instruments*. Creating a mechanism to access instruments has led to many different legacy solutions in the past, facing the complex task of integrating all of them in the system, especially when they come from different designers. In order to mitigate these issues, new standards have been created to mitigate these issues.

The IEEE Std 1149.1 has introduced the concept of *test data register* (TDR), as the interface register between the instrument and the access mechanism. According to the standard, a set of TDRs can be defined as part of the boundary-scan architecture, each one with a register code specified at design time, and selected by acting on the test access port (TAP) controller. In the original version of the standard, each TDR must have a fixed length, chosen by the designer. In the latest revision of the standard (i.e., IEEE Std 1149-2013 [1]), a TDR can have a variable length, and can be constructed as a chain of multiple segments, some of which are always scanned while others, called *excludable segments* and *selectable segments*, are scanned only in particular situations. The excludable segment of a TDR is controlled by a configuration cell, which eventually disables the update/capture capabilities of the segment, and is followed by a switching element controlled by the configuration cell (see Fig. 1). When a logical 0 is scanned into *SegSel*, the segment will be excluded from the *active path* of the network, i.e., the list of cells connected between the test data input (TDI) and test data output (TDO) pins of the TAP. The selectable segments of a TDR are segments, even of different lengths, which are connected to a selection circuit (e.g., a multiplexer). According to the value of a selection field, only one segment at a time is selected as the scanout for the set (see Fig. 2).

The new IEEE Std 1687 [2] tackles the same problem of accessing instruments by means of reconfigurable scan networks. Actually, the standard offers more general access mechanisms with respect to the IEEE 1149.1, and the reconfigurable scan networks are only a subset of the possibilities allowed. One of the most important structures is called *Segment Insertion Bit* (SIB), which is similar to the concept of the excludable segment, while selectable segments are implemented by means of ScanMux elements. When a SIB is said to be *asserted*, the segment it controls is included in the active path; otherwise, it is said to be *de-asserted*. Each segment controlled by a SIB or a ScanMux can be a complex network itself.

In order to move to a certain network configuration of a reconfigurable scan network, patterns are shifted through the Test Data Input (TDI) pin of the TAP. Then, an update operation moves the pattern from the shift flip-flop to the update latch of the configuration cells (i.e., the cells that control the selection logic). This will change the active path of the network. The operation of making an instrument, which is deep into the network, part of the active path may require multiple configuration phases.

This work has been supported by the European Commission through the FP7 STREP Project No. 619871 (BASTION).

This paper deals with the test of permanent faults affecting a reconfigurable scan network. Testing these faults is more complex than testing standard scan chains, which is done by means of well-known techniques (e.g., [3], [4], [5]). In fact, a proper test must check whether the network can be properly configured and whether it works as expected after the configuration (i.e., when a certain set of segments is made part of the active path). Moreover, each special module related to the network configuration must also be tested. The interface with the instruments (i.e., the capture and update logic of TDRs) is not considered in this work, since it requires knowledge of the instrument functions. However, we assume that faults affecting these parts can be detected by applying known stimuli to the instruments at the end of the proposed test flow.

The method proposed in this paper uses the fault model described in [6] and is based on a graph representation of the problem. The result of the application of the method is the minimum (in terms of required test time) sequence of configurations and test patterns needed to cover all faults. Once the sequence of operations required to test the network is known, the time required by the test can be computed using the methods described in [7].

Experimental results are reported on a set of benchmark cases, which practically demonstrate the feasibility of the above test approach, and provide an evaluation of the duration of the test. Moreover, a comparison with the sub-optimal method proposed in [6] is shown.

The rest of the paper is organized as follows. Section II presents the background of the research work on reconfigurable scan networks, with special emphasis on IEEE 1687 scan networks. The basic notions of network testability and the functional fault model are presented in Section III. The proposed approach is detailed in Section IV, followed by some experimental results in section Section V. Finally, Section VI draws some conclusions.

II. Background

Reconfigurable scan networks have been a hot research topic for years. Due to the recent adoption of the IEEE Std 1687 by commercial tools, several issues have arisen regarding design, validation, and test of such structures, as well as their usage in the field.

Given a generic circuit with several instruments, different access networks can be created, e.g., by using IEEE 1687 SIB elements. The authors of [8] have analyzed different possible scenarios and evaluated the overall access time (i.e., the time required to access each instrument in the circuit) by means of a purposely developed tool.

The design automation of IEEE 1687 reconfigurable scan networks has been targeted by different works. In [9], a method has been proposed, which constructs networks composed of multiple SIBs, and optimize it according to different targets, such as area overhead, total access time, or average access time.

Additional scenarios have been analyzed in [10], such as different positions of the configuration cells with respect to the related ScanMuxes; moreover, a formal analysis of the modeled networks has been presented in the same paper for verification sake.

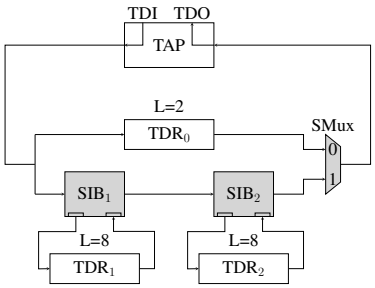

Fig. 3. Example of IEEE 1687 reconfigurable scan network.

A formal analysis of generic reconfigurable scan networks has been also presented by the authors of [11], [12]; in such papers, reconfigurable scan networks have been modeled as satisfiability (SAT) problems, upon which several structural and functional properties have been verified; that works offer a deep analysis about the related modeling, verification, and pattern generation problems.

Pattern retargeting is a well-known problem in the area of reconfigurable scan networks. IEEE 1687 provides a way to define instrument-specific operations. Each operation targets a specific segment of the network, and the specific issue arises of configuring the network such as that the time to access that segment is minimized. This problem has been tackled by different works, e.g., [13], [14], [15].

III. Networks Testability

Testing a standard (non-reconfigurable) scan chain for permanent faults can be performed by shifting a suitable sequence of 0s and 1s through the scan chain. Reconfigurable scan networks (e.g., supported by IEEE 1687) are however far more complicated to test. In addition to flip-flops composing the TDRs, which have to be tested to check whether they can correctly shift values when included in the active path, the reconfigurable modules (e.g., the IEEE 1687 SIBs and ScanMuxes) have to be also tested to check whether they are able to move the network to all legal configurations.

As an example, a simple IEEE 1687 scan network is considered, which is composed of two selectable segments: the first one with a single TDR, and the other one with two TDRs, each one controlled by a SIB. The network is shown in Fig. 3, which also reports the length of each TDR. The SIB elements and the multiplexer (SMux) include an additional scan cell and are highlighted in grey. Depending on the configuration (i.e., the value of the selection cells of SIBs and SMux), the active path can include five possible subsets of TDRs, as listed in Table I. In this table, A means the SIB is in the asserted position, D means de-asserted, 0 and 1 correspond to the two possible positions of the SMux, and - appears when a module belongs to an inaccessible segment (i.e., dont care value). During the system reset, a known configuration is selected.

A functional fault model is used in this work, which has been introduced in [6]. The fault model is briefly presented and discussed below.

A single scan cell affected by a stuck-at fault produces a sequence of 0s, in case a stuck-at 0 affects it, or 1s, when a stuck-at 1 exists. Similarly a same fault model can be applied to an entire TDR. In fact, a fault in a scan cell of a TDR corrupts the

TABLE I
LIST OF POSSIBLE CONFIGURATIONS FOR IEEE 1687 NETWORK IN FIG. 3.

SMux	SIB1	SIB2	Scan length	Active path
0	-	-	3	TDI→TDR_0→TDO
1	D	D	3	TDI→TDO
1	D	A	11	TDI→TDR_2→TDO
1	A	D	11	TDI→TDR_1→TDO
1	A	A	18	TDI→TDR_1→ TDR_2→TDO

TABLE II
EFFECT OF THE FUNCTIONAL FAULT ON SMUX OF FIG. 3, WHICH ALWAYS SELECTS THE INPUT 1.

SMux	SIB1	SIB2	Faulty length	Faulty path
0	D	D	3	TDI→TDO
0	D	A	3	TDI→TDR_2→TDO
0	A	D	11	TDI→TDR_1→TDO
0	A	D	18	TDI→TDR_1→TDR_2→TDO

scan output values, when such a TDR is part of the active path. The update and capture logic is not taken into consideration in this work, since the controllability depends on the instruments attached to TDRs. For the same reason, the faults on the reset logic are not considered.

The faults affecting reconfigurable modules, such as scan multiplexers, are modeled such that a different configuration may be selected rather than the expected one. Such a fault leads to a different active path (called *faulty path*) than the expected one, and the two are likely to have a different length. For example, in Fig. 3 the multiplexer (SMux) may be affected by a permanent fault whose effect is that the segment connected to the input 0 is always selected, no matter the value in the selection cell. The same may arise for the SIBs (whose faults are named stuck-at asserted/de-asserted in [6]). The stuck-at faults in the scan bits of the selection cells are considered as detected by implication by testing such functional faults, which cover also the faults affecting the update logic of the reconfigurable modules. Moreover, such faults cover some faults affecting the reset logic, whose effect is that the module is stuck at the reset value. The other reset faults (i.e., those that make the reset ineffective) are not considered, but can be easily targeted if we grant the possibility to act on the asynchronous reset signal (this scenario is not considered in this paper for sake of simplicity).

A proper test for functional faults of a reconfigurable module compares, for a configuration able to excite a given fault, the expected path length against the length of the faulty path. As an example, the functional fault on the SMux element of Fig. 3, which always selects the segment connected to the input 1, can be excited by a configuration which selects the input 0; all such possible configurations are listed in Table II, which also reports the length of the selected faulty paths. In the table, the first configuration is not able to detect the fault, due to the fact that the faulty length is equal to the active path length (see first configuration in Table I). Thus, one of the remaining three configurations can be selected for the test.

Finally, in any scan network with at least one reconfigurable module (i.e., a reconfigurable scan network by definition), for each TDR fault (according to the presented fault model) F_i, a reconfigurable module fault F_j exists, such that F_i is dominant with respect to F_j (i.e., any test for F_j is also a test for F_i). Thus, the collapsed fault list for a reconfigurable scan chain only contains faults affecting the reconfigurable modules.

Fig. 4. List of the steps in the proposed approach.

IV. PROPOSED APPROACH

The full test of a reconfigurable network must pass through a certain number of configurations, each one able to include in the active path a subset of the reconfigurable modules. Once each target configuration is reached, a test pattern is shifted into the network, and the response is observed by monitoring the scan output values. Similarly to what proposed in [6], a sequence of alternated 0s and 1s can be used, having a length equal to the active path length (which is configuration dependent) and followed by a sequence terminator (e.g., two consecutive 1s after the alternated sequence). By counting the number of shift operations required to reach the sequence terminator, the fault effects are observed and some faults are detected.

The test is organized in *sessions*. After the system reset, the network is set to its initial configuration (which is known). In each session, a target configuration is reached, passing through some intermediate configurations (by means of a certain number of configuration patterns); then, the test pattern is applied. The initial configuration of the following test session is the target configuration of the previous session.

Since the amount of possible configurations of a network grows exponentially with the number of reconfigurable modules, the problem of identifying a sequence of sessions which guarantees the full network test coverage while minimizing the overall test time is not trivial. This paper achieves the intended goal with a tractability limitation of the approach on large circuits, due to the size of the search space. Despite this, we believe that having an indication of the minimum test time for reasonable networks is valuable for evaluating the effectiveness of further works on the same topic, e.g., based on heuristics or optimization techniques.

The approach proposed in this paper formulates the problem as a graph search at the minimum cost. By selecting useful configurations of the network and connecting them as an oriented graph, a pathfinding algorithm can be applied. The resulting path represents the list of configurations and test patterns to be scanned into the network in a given order, which test the network in the minimum number of clock cycles.

The approach is composed of three steps (see Fig. 4). In the first step, the reconfigurable scan network is analyzed and the list of possible configurations is extracted; each configuration is characterized by the scan path length and the list of testable faults. The second step is the creation of the *configuration graph*, where each node is the one related to a specific configuration, and each edge reports the costs to pass from a configuration to another. In the final step, the test sequence is extracted by applying a minimum cost search algorithm on the graph. In the following subsections, each step is analyzed in details.

A. Configuration list selection

The number of network configurations is one of the crucial parameters for applying the proposed methodology with reasonable time or memory efforts. Just to give an idea, a network with

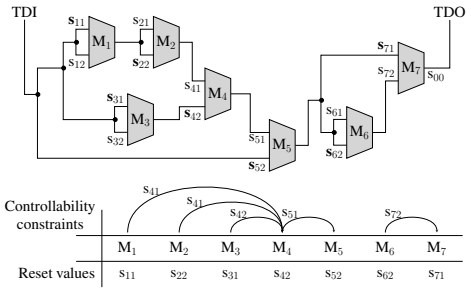

Fig. 5. Example of configuration list selection: example network topology (top) and module dependencies and reset values (bottom).

10 reconfigurable modules, each one having a single selection bit, can be configured in 1024 possible ways (in general, with n selection bits, there are two to the power of n configurations). Later on in this section, a solution to get rid of a subset of configurations is described, thus reducing the overall search space.

For each configuration, a record is created, which contains an identifier and the following fields:

- the values of the selection field of each reconfigurable module (e.g., asserted/de-asserted for IEEE 1687 SIBs, an input identifier for scan multiplexers);
- the length of the active path;
- the list of testable faults (according to the fault model presented in the previous section).

A reduced configuration list can be generated by analyzing the network topology. The idea is to remove the configurations of the reconfigurable modules that cannot be reconfigured with a single scan pattern (because their configuration cells are not part of the active path) and force such modules to their reset state. Then, additional configurations are added in order to cover the testable faults that would result untested with only the selected configurations.

First of all, controllability constraints on the reconfigurable modules can be extracted from the network topology. Each module has the following dependencies:

- the segment which includes the module has to be selected by the controlling module;
- the segment (or segments) which includes the selection cells has to be selected by the controlling module.

If the selection cells of the reconfigurable module are always located in the same segment of the module itself (as for the IEEE 1149.1 elements in Fig. 1 and 2, or in the IEEE 1687 SIB mux-pre and mux-post [2]), the previous constraints are coincident.

As an example, Fig. 5 reports a network topology, in which the TDRs have been removed, and each reconfigurable module M_1- M_7 has a configuration cell located in the same segment. Boolean variables s_{ij} are used to represent the state of each controllable segment. For each module its reset value is bolded. Given a reconfigurable module M_i, its controllability constraint $C(M_i)$ corresponds to the Boolean variable of its controlling segment. For the elements at top level, the controllability constraint is assumed to be always true ($\top$). A module M_i is controlled iff $C(M_i)$ is satisfied. For each reconfigurable module M_i, its reset values $R(M_i)$ is known. Controllability constraints are reported at the bottom of Fig. 5 alongside reset values. From such representation we can derive a Boolean formula that has to be satisfied by all admitted configurations, as follows: $\bigwedge_{M_i} \big(R(M_i) \vee C(M_i)\big)$. As an example, for the network reported in Fig. 5, the resulting Boolean formula is:

$$\begin{aligned}(s_{11} \vee s_{41}) \wedge (s_{22} \vee s_{41}) \wedge (s_{31} \vee s_{42}) \\ \wedge (s_{42} \vee s_{51}) \wedge (s_{62} \vee s_{72})\end{aligned} \quad (1)$$

For each configuration, the list of testable faults can be identified. Hence, the test escapes of the selected configuration list can be derived. Each escaping fault may be either untestable (by any possible configuration) or testable by some configuration not satisfying the Boolean formula. For each of such faults, an iterative process starts from the reconfigurable module affected by the fault, and at each iteration relaxes the constraints on the modules in the controlled segment (the faults on a reconfigurable module can be associated each one to a controlled segment). The process stops when the fault becomes testable or is proved to be untestable. If the fault is untestable, the original constraints are resumed; moreover, the fault dominance on this fault cannot be applied, and the faults affecting the TDRs located on the untestable fault-related segment are included in the fault list. In the previous example of the network in Fig. 5, let us suppose the following branch lengths $\{l_{52} = 28, l_{51} = 11, l_{42} = 11, l_{31=6}, l_{32=8}\}$. At network reset, the segment s_{51} has the same length of the segment s_{52}. In this scenario, the fault that forces the module M_5 to always select the segment s_{51} cannot be detected by the configurations that satisfy the previous formula. In order to make such a fault detectable, the constraint that forces M_3 to its reset values, when s_{52} is selected, is then relaxed. In this case, there is an admitted configuration in which s_{51} and s_{52} have different lengths and thus the fault becomes testable. The formula becomes the following:

$$\begin{aligned}(s_{11} \vee s_{41}) \wedge (s_{22} \vee s_{41}) \\ \wedge (s_{42} \vee s_{51}) \wedge (s_{62} \vee s_{72})\end{aligned} \quad (2)$$

B. Configuration graph creation

The list of selected configurations is used to build a directed graph $G = (V, E)$. The graph is composed of two classes of nodes: configuration nodes (C), and test nodes (T). Each node V_i corresponds to a network configuration si. By moving from one node to another, the network configuration changes. Moreover, when a T node is visited, after changing the network configuration by means of a configuration pattern, a test pattern is applied. After a test phase, the fault list is updated, and all faults testable with the target configuration are dropped. On the contrary, when a C node is visited, a test pattern is not applied and the fault list is not updated. An edge (V_x, V_y) contains the cost in terms of clock cycles to move from the generic node V_x to V_y. The cost associated to an edge (V_x, C_y) is equal to the scan path length of σ_x. The cost for an edge (V_x, T_y) is equal to the scan path length of σ_x plus the test cost when the network is in the configuration σ_y (e.g., equal to the σ_y scan path length plus two clock cycles). The procedure that builds the graph is described below:

1) for each selected configuration si, a node C_i and a node T_i are created, and connected by a single edge (C_i, T_i), which takes into account only the test cost (the configuration does not change);

2) for each s_i, the near configurations are identified, by getting the list of modules M_i which can be reconfigured with a single scan pattern (i.e., their selection bits are part of the active path), and generating all permutations on the selection bits of M_i;
3) each near configuration s_j has to satisfy the Boolean formula used to generate the list of selected configurations, otherwise it is discarded;
4) for each permutation s_j of s_i, each of the two nodes C_i and T_j are connected to both C_j and T_j, and vice-versa (eight new edges are created).

The proposed procedure creates a configuration graph, which is composed of 2^n nodes, where n is the number of selected configurations. The outdegree of a node (i.e., the number of edges pointing from it) depends on how many reconfigurable modules are controllable in the configuration associated to the node. For example, if a certain configuration sk exists, such that all the reconfigurable modules are controllable, the corresponding nodes C_k and T_k have edges pointing to all other nodes in the graph.

C. Test sequence extraction

After the reset, the network is set to a fixed configuration s_0. Thus, the corresponding initial node C_0 can be identified in the configuration graph. Starting from C_0, a path p on the configuration graph represents an ordered list of scan patterns, each one able to change the network configuration as expected, and eventually (in case p traverses at least one T node) perform a test. A fault F affecting the network modules might deviate the expected behavior of p, and the T nodes eventually traversed by p might fail the test. If such a situation happens, p is a test sequence for F. The set of all T nodes in the configuration graph is able to detect all testable faults affecting the network. This implies that paths that visits all T nodes starting from C_0 guarantees the full test coverage of the network. The reverse implication is not true, in fact a subset of T nodes may be sufficient to detect all possible faults. The purpose of this work is to find one possible path p on the configuration graph corresponding to a test sequence that minimizes the test cost. Due to the redundant T nodes in the graph, more than one global optimum may exist. It can be demonstrated that an optimal test sequence (in terms of test time) $p = C_0, \ldots, T_i, [C_j, \ldots, C_k], T_p, \ldots, T_n$ does not traverse the same T node more than once, but may traverse the same C node multiple times. The proposed approach makes use of a modified implementation of the A^* algorithm [16] to find the optimal test sequence. The reason of this modification is detailed after a brief description of the algorithm. The classical A^* algorithm finds the optimal path from the source node to the destination node (also referred to as goal) efficiently. It uses an estimated cost function $f(n) = g(n) + h(n)$ to determine the order in which the nodes are visited. The term $g(n)$ is the cost from the source node to the generic node n, and $h(n)$ is the heuristic estimate cost to reach the goal from n. A common implementation of A^* maintains an OPEN list and a CLOSED list of nodes. The nodes that need to be examined are selected from the OPEN list, according to $f(n)$, while the CLOSED list keeps track of nodes that have already been examined. The node with the smallest $f(n)$ is the first selected. The heuristic function $h(n)$ must be admissible, i.e., the estimated cost must be less than the actual cost, in order for the algorithm to reach the optimum. The classical version of A^* cannot be applied to the problem of finding the optimal test sequence, since the goal is not a single node in the configuration graph, but is a set of T nodes that guarantee the full test coverage of the network. Hence, the goal is unknown until is not found. In the proposed implementation, every node n in the OPEN list can be a C node or a T node. Moreover, n maintains its own OPEN_P$_n$ list and CLOSED_P$_n$ list of paths, where each path p is a possible path from C_0 to n and is associated to a $f(p)$ value. Nodes are extracted from the OPEN list according to the best $f(p)$ value in the corresponding OPEN_P list and are moved into the CLOSED list only when the corresponding OPEN_P list is empty. When n is examined, each neighbor node m is included in the OPEN list (if not already there). Given the new path q, consisting of the path p (i.e., the best path reaching n) followed by the edge (n, m), $g(q)$ is computed as $g(p)$ plus the cost of the edge (n, m). Details on how to compute the $h(q)$ value are given later on in this section. If the computed $f(q)$ is lower than the previous $f(q')$ value, a new path is included in OPEN_P$_m$. The effectiveness of the A^* algorithm highly depends on the heuristic used in the computation of the $h(q)$ value. The proposed heuristic uses the length of the segments connected to each configurable element. Given a reconfigurable module M with k selectable segments (e.g., two segments in case of a IEEE 1687 SIB), a fault that forces M to select the $i-$th segment can be detected by configuring M so that a $j-$th segment, with j different than i, is included of the active path, and then shifting a test pattern into the network. Such test pattern is at least as long as the $j-$th segment. According to this reasoning, the contribution of the fault F to $h(q)$ is comparable to the length of the shortest segment other than the $-$th one, plus the number of selection cells of the $F-$related module; if such a segment includes other reconfigurable modules, such modules and the segments they control are not counted; this permits to not take a segment into consideration multiple times in the $h(q)$ computation. As an example, the fault that forces the SMux module of Fig. 3 to the value 1 is detected by a configuration in which SMux is set to 0 and selects the segment that includes TDR2. The contribution to $h(q)$ of such a fault is the TDR2 length plus the selection bit (i.e., 11). The cost of the opposite SMux fault is estimated as the length of the other segment (which includes the two SIBs), with the inner SIBs detached; thus, the cost is zero for the segment plus the selection bit (i.e., 1). When a node T_i is visited, each fault that is tested by any test pattern in the path $q = C_0, \ldots, T_i$ is removed from the fault list. The $h(q)$ value is computed by considering each of the remaining untested faults (i.e., the *active faults*). When the active fault list is empty, the goal is reached (i.e., q is a test sequence), and $f(q)$ is the real test cost.

TABLE III
CHARACTERISTICS OF BENCHMARK NETWORKS

Network	#TDRs	#Config bits	Cumulative path	Longest path	Depth	Search space
A586710	5	6	41,972	41,972	2	75.0%
N17D3	27	15	462	372	3	7.5%
N12D4	15	12	632	481	4	2.6%
N14D4	16	14	196	102	4	0.3%
N9D5	11	9	350	302	5	12.7%
N12D9	14	12	593	593	9	1.7%

TABLE IV
EXPERIMENTAL RESULTS ON THE BENCHMARKS AND COMPARISON WITH DEPTH FIRST SEARCH.

	A^* results				Depth-first [6] results				
Network	#C nodes	#T nodes	C cost [cc]	T cost [cc]	#C nodes	#T nodes	C cost [cc]	T cost [cc]	Efficiency
A586710 reset A	3	3	41998	167.914	5	3	42009	167914	∼100%
A586710 reset B	3	3	41998	167.914	4	3	83644	167914	83.4%
N17D3 reset A	6	5	1047	2810	8	5	1337	2876	91.5%
N17D3 reset B	5	5	967	2827	6	5	1262	2876	91.7%
N12D4 reset A	17	12	1303	6997	16	12	1701	6997	100%
N12D4 reset B	12	12	1225	6997	15	12	1356	6997	98.4%
N14D4 reset A	20	14	1110	2202	24	14	1241	2202	96.2%
N14D4 reset B	14	14	774	2202	15	14	843	2202	97.7%
N9D5 reset A	6	6	509	2321	10	6	835	2325	89.6%
N9D5 reset B	6	6	509	2321	7	6	624	2325	96.0%
N12D9 reset A	14	10	773	6652	19	10	844	6652	99.1%
N12D9 reset B	10	10	722	6652	11	10	1317	6652	92.5%

V. EXPERIMENTAL RESULTS

The effectiveness of the proposed approach has been validated with an in-house tool on a set of benchmark reconfigurable scan networks. The A^* algorithm has been compared against the depth-first search proposed in [6]. The tool, written in Java, reads the network topology (described in XML format), which also includes the reset state of each reconfigurable module. A set of heterogeneous reconfigurable scan networks, representing different kinds of topologies, has been manually generated. The key characteristics of each network are detailed in Table III, which also includes the percentage of selected configurations (column 6) according to the rules discussed in Subsection IV-A. The first two networks have been inherited from [6]. Interestingly, for two of the considered networks (namely N14D4 and N12D9) the search space is highly reduced with respect to all possible network configurations. The reason for this reduction is that instruments in such networks are deeply hidden by configuration modules. Conversely, the network with depth equal to 5 (N9D5) has a long series of instruments on the same level, which makes it harder to constrain the search space. For each network, two reset configurations are considered. In the first configuration (referred to as reset A), each reconfigurable element selects the first controllable segments, while the second segment is selected in the second one (reset B). According to [6], the cost of each pattern is increased by 5 clock cycles. Additionally, each test operation requires two patterns (one composed of as many 0s as the longest path, followed by one long as the active path, alternating 0s and 1s). Experimental results on the selected benchmarks are shown in Table IV. Each line of the table contains the results of A^* (columns 2-5) and of the depth-first approach (columns 6-10). For both methods, the number of configuration (#C nodes) and test (#T nodes) nodes of the test sequence are reported, as well as the configuration (C cost) and test (T cost) costs obtained. Costs are expressed in terms of number of clock cycles (cc). Finally, the efficiency of the depth-first approach in terms of final cost (with respect to the optimal cost of A^*) is reported in column 10. The efficiency values on the experimented networks are very high, greater than 80% in the worst case, and more than 95% in at least half of the cases. Note that in some cases the depth-first approach achieves 100% efficiency, meaning that it performs as good as an exaustive search.

VI. CONCLUSIONS

The paper presented a methodology to evaluate the test cost of reconfigurable networks, according to the functional fault model introduced in [6]. The proposed methodology uses the network topology to build a graph of possible network configurations. An optimal test sequence is generated by applying the A^* search algorithm on such a graph, with a suitable heuristic. The search space was constrained without compromising the final results. Experimental results on selected benchmarks proved the validity of the approach. The major goal of evaluating the effectiveness of sub-optimal approaches was reached, demonstrating the high efficiency of the depth-first heuristic.

REFERENCES

[1] IEEE Standard for Test Access Port and Boundary-Scan Architecture, IEEE Standard 1149.1-2013, 2013.

[2] IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, IEEE Std 1687-2014.

[3] K. Lee and M. A. Breuer, A Universal Test Sequence for CMOS Scan Registers, IEEE CICC, pp. 28.5/14, 1990.

[4] S. R. Makar and E. J. McCluskey, ATPG for Scan Chain Latches and Flip-Flops, IEEE VTS, pp. 364-369, 1997.

[5] F. Yang, S. Chakravarty, N. Devta-Prasanna, S. M. Reddy and I. Pomeranz, On the Detectability of Scan Chain Internal Faults An Industrial Case Study, IEEE VTS, pp. 7984, 2008.

[6] R. Cantoro, M. Montazeri, M. Sonza Reorda, F. G. Zadegan and E. Larsson, "On the testability of IEEE 1687 networks," 2015 IEEE 24th ATS, Mumbai, 2015, pp. 211-216.

[7] F. G. Zadegan, U. Ingelsson, G. Asani, G. Carlsson and E. Larsson, Test Scheduling in an IEEE P1687 Environment with Resource and Power Constraints, IEEE ATS, pp. 525-531, 2011.

[8] F. G. Zadegan, U. Ingelsson, G. Carlsson and E. Larsson, Access Time Analysis for IEEE P1687, IEEE Trans. on Computers, Vol. 61, No. 10, pp. 1459-1472, 2012.

[9] F. G. Zadegan, U. Ingelsson, G. Carlsson and E. Larsson, Design Automation for IEEE P1687, DATE 2011.

[10] F. G. Zadegan, E. Larsson, A. Jutman, S. Devadze and R. Krenz-Baath, "Design, Verification, and Application of IEEE 1687," 2014 IEEE 23rd ATS, Hangzhou, 2014, pp. 93-100.

[11] R. Baranowski, M. A. Kochte and H. J. Wunderlich, "Modeling, verification and pattern generation for reconfigurable scan networks," 2012 IEEE ITC, Anaheim, CA, 2012, pp. 1-9.

[12] R. Baranowski, M. A. Kochte and H. J. Wunderlich, Reconfigurable Scan Networks: Modeling, Verification, and Optimal Pattern Generation, ACM Trans. Des. Autom. Electron. Syst. 20, 2, Article 30, 27 pages, 2015.

[13] F. Ghani Zadegan, U. Ingelsson, E. Larsson and G. Carlsson, Reusing and Retargeting On-Chip Instrument Access Procedures in IEEE P1687, Design and Test of Computers, IEEE, vol. 29, no. 2, pp. 7988, april 2012.

[14] R. Baranowski, M. A. Kochte and H. J. Wunderlich, Scan Pattern Retargeting and Merging with Reduced Access Time, in Proc. IEEE ETS. IEEE Computer Society, 2013, pp. 39-45.

[15] R. Krenz-Baath, F. Zadegan and E. Larsson, Access time minimization in ieee 1687 networks, in ITC 2015. IEEE International, Oct 2015, pp. 1-10.

[16] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, 1968.