A Novel Sequence Generation Approach to Diagnose Faults in Reconfigurable Scan Networks

Riccardo Cantoro, *Member, IEEE,* Aleksa Damljanovic, *Student Member, IEEE,* Matteo Sonza Reorda, *Fellow, IEEE,* and Giovanni Squillero, *Senior Member, IEEE*

Abstract—With the complexity of nanoelectronic devices rapidly increasing, an efficient way to handle large number of embedded instruments became a necessity. The IEEE 1687 standard was introduced to provide flexibility in accessing and controlling such instrumentation through a reconfigurable scan chain. Nowadays, together with testing the system for defects that may affect the scan chains themselves, the diagnosis of such faults is also important. This article proposes a method for generating stimuli to precisely identify permanent high-level faults in a IEEE 1687 reconfigurable scan chain: the system is modeled as a finite state automaton where faults correspond to multiple incorrect transitions; then, a dynamic greedy algorithm is used to select a sequence of inputs able to distinguish between all possible faults. Experimental results on the widely-adopted ITC'02 and ITC'16 benchmark suites, as well as on synthetically generated circuits, clearly demonstrate the applicability and effectiveness of the proposed approach: generated sequences are two orders of magnitude shorter compared to previous methodologies, while the computational resources required remain acceptable even for larger benchmarks.

Index Terms—Diagnosis, Reconfigurable Scan Networks, IEEE Std 1687, Finite State Machines

1 Introduction

The complexity of the devices designed in recent years has been increasing significantly. Together with resources that back up the device functionality, a large number of auxiliary resources is embedded within the generic device to support different purposes, such as calibration, test, monitoring and debug. These additional resources are sometimes referred to as on-chip *instruments*. For example, using logic or memory BIST to perform test at the end of manufacturing, or during the device's operational life, may require some initialization values and results eventually need to be retrieved, while sensors for measuring and monitoring voltage and temperature across the device often have to be configured before acquiring the data. Hence, data transfers are required between the outside and the instruments, and different solutions have been devised to support them, e.g., by resorting to IEEE 1149.1 or IEEE 1500.

Exacerbated by the increasing number of instruments within a single scan chain, the lack of flexibility is a major issue in both IEEE 1149.1 boundary-scan (JTAG) and IEEE 1500 core test standards, along with weaknesses of the test scheduling and scalability limitations. The new IEEE 1687 standard (IJTAG) [1] was designed to be able to deal with problems caused by the long scan chains and the substantial number of instructions required to access instruments. It exploits the idea of *Reconfigurable Scan Networks* (RSNs), allowing a scan chain to be partitioned into *segments* that can be selectively included or excluded. Through dynamic configuration and variable-length scan-chain, IJTAG enables flexible and efficient access to all instruments. Thus, designers are able to take into account various configurations and choose the best trade-off between parameters such as area or access time. Although the standard does not impose an external access mechanism, the most widely accepted one is the IEEE 1149.1 Test Access Port (TAP).

To interface all on-chip instruments, special *Test Data Registers* (TDRs) are incorporated into the network. They are similar to those found in boundary-scan and include some additional control logic, depending on write/read capability. Two special reconfigurable modules, namely *Segment Insertion Bits* and *Scan Multiplexers*, are used to support dynamic configuration: users can change the configuration of the network by programming ScanMux and SIB modules, choosing which sub-set of instruments has to remain accessible; then, they can write or read the flip-flops belonging to the currently active segments. As a result, faster and more efficient access is provided. The IEEE 1687 standard introduces two languages: *Instrument Connectivity Language* and *Procedural Description Language* that allow describing the structure of the network and the protocol to access the different instruments.

To provide correct access to all instruments embedded within the device, guaranteeing that the network will behave as expected is absolutely crucial. In other words, possible defects affecting the network should be identified. This includes testing the ability to configure the network in a proper way, as well as to check if the network behaves appropriately in whichever legal configuration. In comparison with testing standard scan-chains [2]–[4], SIB and ScanMux reconfigurable modules requires special procedures. Eventually, when a fault is detected, it may be necessary

• *R. Cantoro, A.Damljanovic, M. Sonza Reorda, and G. Squillero are with the Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy.*
E-mail: riccardo.cantoro@polito.it, aleksa.damljanovic@polito.it, matteo.sonzareorda@polito.it, giovanni.squillero@polito.it

to precisely pinpoint the faulty module.

This work focuses on identifying and localizing permanent faults affecting reconfigurable modules in RSN. In more details, the paper presents a technique to produce a diagnostic sequence of stimuli able to localize the faulty element. The proposed method relies on a semi-formal approach: a Finite State Automaton is dynamically built and then used by an heuristic algorithm to generate a quite effective sequence able to diagnose all permanent faults in the target RSN. Such a diagnosis has to be complemented with the diagnosis of remaining components of the RSN, possibly implementing techniques [5]–[10].

Since it is common in the industrial practice to start the test/diagnosis generation activities early in the design process, when high-level descriptions are available only, we target a high-level fault model for the reconfigurable modules and using it enables defining more general, implementation-independent methods, since the standard does not impose rules but provides only suggestions on how certain modules should be implemented at the gate-level. Of course, such a high-level fault model is a trade-off between the complexity and the achieved defect coverage, and we also address the issue of validating the used fault-model resorting to fault-simulation. In our work we considered only RSNs containing regular SIBs and ScanMuxes, controlled both in-line and remotely, while additional aspects of RSNs, e.g., security, are not taken into account as they would render an approach to diagnose faults heavily dependent on the implementation of the chosen method.

Experimental results are reported on benchmark networks from the set introduced in [11], from [12], and on synthetically generated circuits. Results demonstrate the feasibility and effectiveness of the proposed approach: while always keeping the computational cost under control, test sequences produced on these circuits by resorting to the current approach are significantly shorter than those generated by the method described in [13].

The rest of the paper is organized as follows. In Section 2 we summarize the key characteristics of the IEEE 1687 networks, with a review on related works and some main motivations. Section 3 describes the adopted fault model and diagnostic procedure. In Section 4 we propose the technique for generating an optimized diagnostic sequence for an RSN. Section 5 reports on experimental results. Finally, Section 6 draws some conclusions.

2 Background

In this section an overview of modules used in an IEEE 1687 network is given as well as the main motivations for performing diagnosis on reconfigurable scan networks.

2.1 Overview of Reconfigurable Scan Networks

The RSN is an instrument access network residing between device interface and instrument interface. With the possibility to split the scan chain into segments connected either in series or in parallel, an RSN improves the flexibility of a single scan chain in terms of access time. Although any scan cell in a scan chain with serially-connected shift register bits is a potential point of failure, an RSN is more robust from the reliability point of view, since the rest of the circuit is likely to be still operational thanks to exploiting hierarchical structures. Moreover, RSNs offer the possibility to be dynamically configured, thus removing the need to use separate instructions to access a particular instrument or a group of instruments.

To interface each instrument, a register of a variable length is used. This corresponds to a set of scan cells, IEEE 1149.1-compatible, referred to as a TDR. TDRs can be Read-Only, Write-Only or Read-Write. Furthermore, three operations are used to control the network and read/write data from/to TDRs: capture (C), shift (S), update (U).

Apart from TDRs, the network is composed out of two types of programmable modules. These are used to partition the set of instruments; including or excluding a set of instruments obviously has an effect on the scan chain length.

A segment insertion bit (SIB) module behaves as a gateway with respect to the segment it controls: it is able either to bypass the segment or to include the segment into the active path (Figure 1(a)). In the bypass state it is referred to as *de-asserted*, while it is said to be *asserted* when is configured to expand the scan chain. SIBs can be used to obtain a hierarchical structure of the network, allowing hierarchical access to the registers interfacing the instruments. The Figure 1(b) shows a symbol used to represent a SIB.

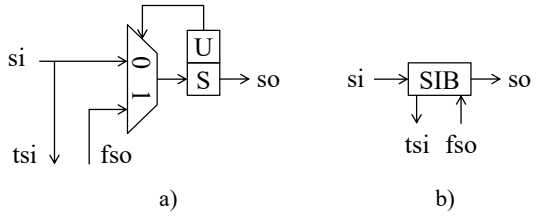

Fig. 1. Segment Insertion Bit (SIB) module

Other than using a SIB to include or bypass a segment, a different module is used to support exchange of one scan chain segment for another. A scan multiplexer with shift-update cells (ScanMux, SM) can be seen as a configurable multiplexer, which is used to alter the scan chain by selecting which of its input branch segments are to be included into the active path. Thus, the existence of mutually exclusive scan chains is supported, reinforcing the trade-off of access time with access length. The example given in Fig. 2(a) shows a scan multiplexer with a two-bit shift-update control register which is used to choose one among four segments. The symbol shown in Fig. 2(b) will be used to represent a shift-update cell.

Control registers of the modules consist out of two stage cells. A cell is referred to as a flip-flop with additional control logic. Shift (S) cell is a part of scan chain and it shifts values, while Update (U) cell stores the S cell value, when update operation is performed. The configuration of the module is defined by the value in the U cell. For example, a SIB module is configured by shifting in the desired value into the S scan cell, followed by an update, thus storing the value from the S cell to the U scan cell. Indicatively, as illustrated by Fig.1(a), in this work a SIB is considered to be asserted if the latched bit is 1 and if so, it includes the path between `tsi` and `fso` terminals.

Conversely, it is regarded as being in a de-asserted state if the latched bit is 0, bypassing the segment between `tsi` and `fso` terminals, directly connecting `si` and `so` through the S cell. The provided shortcut has the length of one bit.

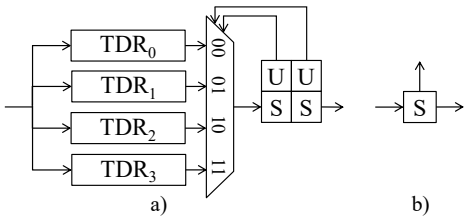

Fig. 2. ScanMux (SM) module

Depending on the position of the configuration bit(s) with respect to the programmable module itself the module can be either *inline* or *remote*. A module is considered to be inline if its associated configuration cell is located in the same segment of the related module. Otherwise, it is said to be remotely controlled.

Some basic architectural constructs, varying on the organization of TDRs, are provided in Figure 3. A simplest one is a flat structure where TDR is always accessible (Fig. 3a). MUXed TDRs enable mutually exclusive access (Fig. 3b), while excludable TDR is either a part of the active path or is bypassed (Fig. 3c). Partial configurations involve combining previous structures, thus in Fig. 3d (partially selectable TDRs) always two TDRs belong to the active path (one fixed, another selectable), while in Fig. 3e (partially excludable TDRs) one or two registers belong to the active path (path always includes one, while the other one can be inserted). The Fig. 3f shows the partially excludable and selectable configuration (one TDR is always accessible, while one of the remaining two can be included into the active path). These can be combined to design more complex networks.

Although all the examples presented in this paper resort only to SIB and 2-1 ScanMux modules with a pair of TDRs on their branches, the approach supports a wide range of scenarios including inline and remote modules. It can also be applied on networks containing ScanMux modules with higher number of input branches as well as cascaded ScanMux modules.

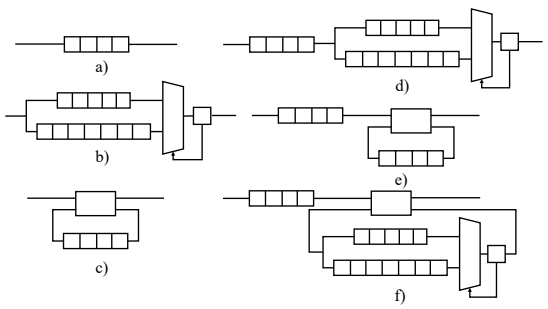

Fig. 3. Supported architectural constructs - organization of TDRs

The standard itself does not impose which type of external interface has to be used to access the RSN. However, the most common one is IEEE 1149.1 with the TAP controller. In this paper we will thus consider that as the external interface.

2.2 Related works

In recent years the IEEE 1687 standard and RSNs have been subject of many research works, addressing test, verification, security and design. However, to our knowledge, apart from [13], this is the only work addressing the issue of permanent RSN fault diagnosis.

As already discussed, although reconfigurable scan networks introduced flexibility, minimizing access time has arisen as a potential issue. The authors in [14] analyzed various structures with different access scheduling to estimate overall access time. Additionally, the same authors developed the CAD tool to support design automation of optimized 1687 SIB networks [15]. Based on the access schedule and the set of instruments, the tool is able to design a network with optimized access time and low hardware overhead.

In [16], a general approach based on heuristics algorithms and independent on network implementation is proposed to test 1687 RSN modules. Techniques used to test single elements are described and merged to form a single test. An alternative approach based on an evolutionary algorithm was developed in [17] to generate test sequence for a generic RSN with minimum duration. The usage of formal techniques to generate the minimum duration sequence able to test all reconfigurable modules in the network is explored in [18], providing a lower bound for small networks and thus assessing the effectiveness of the other approaches. Furthermore, the same issue is addressed in [19], where the state of a reconfigurable scan network is modelled with a finite state automaton. A greedy algorithm is then used to generate a functional test sequence able to detect all fault mapped to multiple state-transition faults.

The authors of [13] analyzed the effect of permanent fault on RSN elements to determine diagnostic properties. The test sequence generated by the procedure described in [16] is further extended to enable localization of faults, i.e., to satisfy the defined properties.

A number of works also analyze the use of IEEE 1687 infrastructure to support on-line health monitoring and fault management [20], [21].

Modelling, verification and optimal pattern generation is tackled in [22]. RSN formal model is presented considering structural and functional dependencies. The problem is transformed into Boolean Satisfability Problem. The formal method is also used for pattern retargeting, i.e., to generate scan-in data for reconfiguration and execution of commands in instrument access procedures. Moreover, the paper describes pattern generation method for efficient concurrent access. For the retargeting modelled as a sequential problem unrolled over number of time frames (CSU operations), the authors in [23] proposed a method for calculating an upper-bound on the number of required CSU operations. Knowing the upper-bound is used to deal with the model complexity for large designs and reduce the search space while preserving the optimum solution.

Defining and verifying security properties is addressed in [24]. In this work it is described how specified permissions and restrictions are transformed into predicated for a formal model unbounded checking.

The authors in [25] considered introducing some DfT modifications to enable observability of shadow registers

and update logic. Moreover, different test methods are proposed for stuck-at, flip-flop transparency and bridge-faults in the RSN.

Security in RSN is another important aspect considered is literature. In [26], [27], obfuscation strategies are proposed to modify the structure of the network by introducing additional logic for controlling the state of reconfigurable modules, as well as creating false paths to confuse the attacker. On the other hand, other works consider validating security properties and preventing unauthorized access by the means of external filter module both online and fixed-precomputed [28]–[30].

2.3 Motivations

Although the IEEE 1687 standard alleviated many problems and resolved many issues, it has also introduced some additional ones. Testing traditional scan-chains for permanent faults is relatively simple, since shifting so called, flush sequence (a sequence of alternated 1s and 0s) through the scan chain is sufficient to detect and even understand the type of defect affecting it, if any. On the other hand, to test an RSN, apart from testing the capability of FFs (comprising TDRs) to shift, modules such as SIBs and ScanMuxes also have to be tested. Without considering the test of reconfigurable elements, no guarantee can be given that applying any valid configuration will result in network changing its state or being in a state corresponding to the wanted one. The problem becomes even more complex when discriminating between the faults affecting SIBs and ScanMuxes is required. Even though a number of works is focused on identifying faults affecting scan-chain [5]–[10], little attention has been given to resolving the issue of fault diagnosis within RSNs. The main motivation of this work is to determine which RSN modules (TDRs, SIBs, ScanMuxes) are potentially affected by a permanent fault. Identifying the faulty reconfigurable module is a challenging task given the complexity of the current RSNs and reducing the duration of the diagnosis procedure is crucial. Once the faulty reconfigurable module is identified, the diagnostic procedure may be completed resorting to techniques already available.

Although stimulating instruments and collecting responses could eventually resolve the issue of ambiguity between the modules, the functional access to the instruments is not considered in this work. Therefore, identifying classes of undistinguishable faults is necessary. A fault affecting any of the elements within the same class of equivalence has the same effect on the output, no matter which input stimuli is applied to the network.

Since the effect of a fault observed at the output differs depending on the module it affects, the main goal can be divided into two categories:

- discriminating between fault-affected TDRs and detecting the pairs of TDRs defined as *undistinguishable*;
- discovering a fault-affected reconfigurable module (SIB, ScanMux).

In given examples, SIB module is denoted with ”A” when asserted, while it is denoted with ”D” when de-asserted. ScanMux configuration when upper segment is chosen is denoted with ”0”, while the configuration with lower segment chosen corresponds to ”1”. Remark that, by the sake of generality, we consider SIB modules and ScanMux modules with two input branches containing TDRs. After the system reset, network is in a reset state, in which all SIBs are de-asserted and all ScanMuxes select upper input segment.

3 Fault model and Diagnostic Mechanism

Introducing reconfigurability has made diagnosis more challenging. Apart from localizing the faults affecting the ability of flip-flops forming TDRs to correctly shift values, distinguishing between faults affecting reconfigurable elements is also required. The high-level fault model introduced in [16] and then adopted in several works (e.g., [17], [31] and [18]) is used as an abstract representation of defects on network modules. Although it was originally designed after analyzing the effects of possible stuck-at faults, it has certain limitations regarding certain faults (e.g., faults affecting reset and enable logic).

Faults affecting TDRs are considered at the level of a single FF composing the TDR. In this case, a pair of stuck-at faults (stuck-at-0 and stuck-at-1) may affect the output of the FF. Consequently, when a faulty TDR is accessed, repeated subsequent values of 0s or 1s are observed at the output of the scan chain. Faults affecting two different FFs within the same TDR can not be distinguished between themselves, at least not using only structural information. However, performing access at the instrument level to control and collect responses is not being considered in this work. All stuck-at-0 and stuck-at-1 TDR’s FF faults are grouped into two TDR faults, respectively. While the consideration that the scan cells may be affected exclusively by stuck-at faults is an important simplification, additional fault models may be easily taken into account thanks to the high-level nature of the approach, e.g., timing faults including *slow faults* (slow-to-rise, slow-to-fall and slow) and *fast faults* (fast-to-rise, fast-to-fall and fast), resulting from setup/hold-time violations.

Although flush patterns are both sufficient and efficient to detect defects and determine their type, they cannot identify the faulty scan cell(s). As different flush patterns exist, the effect of permanent faults of different type on the inserted pattern (00110011) is shown in Table 1. Additional type to be considered is intermittent fault. Increasing diagnostic resolution which is currently at the level of the scan segment may benefit from the approaches presented in survey [6]. Different flush sequences may be used and failures from multiple failing scan patterns may be analysed to trace back failures to the origin.

Faults affecting SIB and ScanMux reconfigurable modules are modelled as high-level stuck-at faults. Accordingly, a SIB can be *stuck-at asserted* or *stuck-at de-asserted*. Regardless of the configuration, a fault-affected SIB may be permanently bypassing or including the associated segment. The effect of such a fault after configuring the network, is that the path between TDI and TDO differs from the expected one (*faulty path*). By shifting in a sequence of alternated 0s and 1s the same sequence will appear at the output after a number of clock cycles different than the expected one. Similarly, a ScanMux may be stuck-at one

TABLE 1
Set of possible scan-chain fault models and their effect on the inserted pattern 00110011.

Fault models	Output effect (permanent faults)
Fault-free	00110011
Slow-to-rise	0010001X
Slow-to-fall	0111011X
Slow	0110011X
Fast-to-rise	X0111011
Fast-to-fall	X0010001
Fast	X0011001
Stuck-at-0	00000000
Stuck-at-1	11111111

of its configurations (e.g., for a 2-to-1 ScanMux, faults are stuck-at-0 and stuck-at-1, while for a 4-to-1 ScanMux, faults are stuck-at-00, stuck-at-01, stuck-at-10, and stuck-at-11). A corresponding input branch is always selected no matter the configuration. The same effect on the scan path length can be observed in this case. By shifting in a sequence of alternated 0s and 1s the same sequence is going to appear at the output after a different number of clock cycles with respect to the expected one.

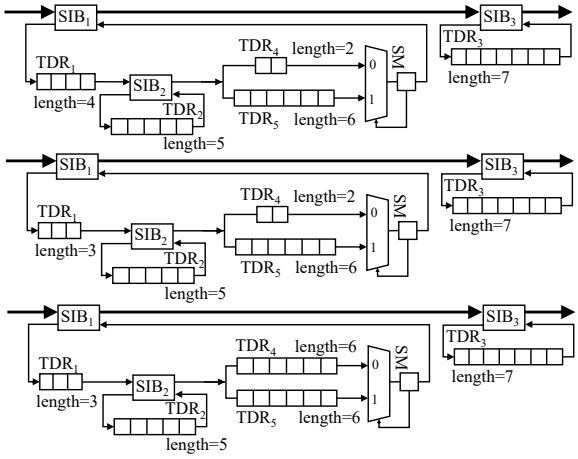

Fig. 4. Examples of IEEE 1687 RSNs

According to this high-level fault model, one can perform diagnosis on an RSN by configuring the RSN so that the target fault is excited, shift in the sequence of alternated 0s and 1s, and observe when it will appear at the output. By comparing the length of the activated path against the lengths of all other path lengths including faulty ones and the expected one the existing fault can be identified. Especially when all applicable configurations, starting from the initial one, result in having different path length, the above approach is easily applicable. As an example, the high-level fault affecting the SM in the network shown in Fig. 4 (top), which always selects the segment connected to the input 1, is considered. The faulty module can be excited by a configuration that selects its input 0; an additional requirement is for the module itself to be included into the active path, otherwise the fault is masked. Configurations C_8, C_{10}, C_{12} and C_{14} (given in Table 2) fulfill these conditions. Once one of them is activated, one can measure the length of the active path by shifting a given sequence (called diagnostic vector) through TDI and checking when it will appear on TDO. It has to be noted that the total number of clock cycles required to apply the generated diagnostic sequence, namely, cost, is not influenced in the same manner by the choice of different configurations from the aforementioned set.

However, it is not always trivial to localize the fault, since different faults may result in having the same active path length. For example, in the network from Fig. 4 (middle), stuck-at-asserted faults on SIB_1 and SIB_2 result in having the same path length ($9 = SIB_1(1) + TDR_3(7) + SIB_3(1) = TDR_1(3) + SIB_2(1) + TDR_4(2) + SM(1) + SIB_1(1) + SIB_3(1)$). This issue can be resolved by continuing to stimulate the network, until a unique sequence of path lengths is observed taking into account the previous active path lengths. The principle is described in more details in the following section. Moreover, some of the faults may remain undistinguishable. Pairs of faults affecting a ScanMux module with the same length of TDRs on its input branches belong to this group. By using this approach, it remains impossible to differentiate between stuck-at-0 or stuck-at-1 faults on the SM module ($6 = TDR_4(6) = TDR_5(6)$) from Fig. 4 (bottom).

The same procedure is used to the single permanent fault diagnosis on the RSN elements (TDRs, SIBs and ScanMuxes). The complete generated procedure is organized as a set of *sessions*, each composed of a diagnostic step and a configuration step. Configuration step corresponds to shifting configuration bits in the scan chain and performing update. Each diagnostic step consists of the following phases:

1) shifting in the first sequence consisting of same values (all 0s or all 1s), while the length of the sequence is equal to the length of the longest path in the network; the goal of this phase is the initialization of the scan cells;
2) shifting in the second sequence of alternated 0s and 1s (i.e., 0101...01), with the predetermined length of the sequence (equal to the maximum length of the expected path and all faulty paths). As a sequence terminator two identical bits (either 00 or 11) are added;
3) the last, diagnostic sequence shifts values from the currently active path.

In a standard TAP controller, reaching the ShiftDR state from the UpdateDR state requires visiting CaptureDR state. Therefore, Step 1) is required when capture values are either not defined or not considered. Each configuration and diagnostic step can be translated into a JTAG vector-*Scan Data Register* (SDR). SDR is a state command to perform an IEEE 1149.1 Data Register scan and is defined within Serial Vector Format (SVF). It is used with 4 arguments: TDI, TDO, MASK and SMASK, i.e., the value to be scanned into the target, the values to be compared against the actual values scanned out of the target, the mask to be used when comparing TDO values against the actual values scanned out of the target, and mask for specifying TDI data that is "dont care", respectively. A configuration step corresponds to a JTAG vector (SDR) of a predetermined length taking into account active path, while an SDR vector corresponding to an observation step followed by a configuration step

contains increased number of shift operations. The latter situation is known as "overscanning" when scanning longer than the length of the longest path.

Determining fault-caused modifications of values in the scan chain and the length of the active scan path is performed by verifying inserted diagnostic vector; in parallel with observing the values appearing at the output, new configuration vector is shifted in. The path length is deduced from the position of sequence termination symbol. Finally, applying the configuration vector demands an update operation. The duration of the complete diagnostic procedure, referred to as a total cost, depends on the duration of each step and is composed of configuration step cost and diagnostic step cost, both expressed in terms of number of clock cycles. The configuration step cost is the time needed to apply configuration vectors. The time overhead of the JTAG protocol is also included, since moving the TAP controller from shift to update state and vice versa also requires a few clock cycles. The diagnostic phase cost is the time required to shift in the diagnostic sequence. Furthermore, the duration of a session is determined by the length of the TDRs included in the path, as well as by the previous configuration.

Consideration that the TAP controller is used to access and control the network imposes certain restrictions. The TAP Finite State Machine with its defined transitions is able to traverse 3 main states (capture, shift and update) in the following order: either capture, shift and then update or capture and then update, avoiding the shift state. Therefore, without acquiring and applying certain design techniques to improve the observability of such registers [25], it is not possible to check if one shift operation destroys/overwrites previously shifted-in data. If such defect is present, after shifting data into some other scan element in parallel or rather than into the desired one, some update operation must be performed, followed by a capture. Consequently, any previously stored data is overwritten, thus removing any trace of unwanted/undesired access caused by the fault.

TABLE 2
Set of possible configurations of the RSN in Fig. 4 (top).

Config.	SIB_1	SIB_2	SM	SIB_3	Active path	Len.
C_0	D	D	0	D	-	2
C_1			1			
C_4		A	0			
C_5			1			
C_2	D	D	0	A	TDR_3	9
C_3			1			
C_6		A	0			
C_7			1			
C_8	A	D	0	D	TDR_1, TDR_4	10
C_9	A	D	1	D	TDR_1, TDR_5	14
C_{10}	A	D	0	A	TDR_1, TDR_3, TDR_4	17
C_{11}	A	D	1	A	TDR_1, TDR_3, TDR_5	21
C_{12}	A	A	0	D	TDR_1, TDR_2, TDR_4	15
C_{13}	A	A	1	D	TDR_1, TDR_2, TDR_5	19
C_{14}	A	A	0	A	TDR_1, TDR_2, TDR_3, TDR_4	22
C_{15}	A	A	1	A	TDR_1, TDR_2, TDR_3, TDR_5	27

4 METHODOLOGY

In this paper, the RSN of IEEE 1687 is modeled as a finite state automaton (FSA) as in [31]. Each state of SIBs and ScanMuxes in the network, referred to as *configuration*, represents an automaton state. The input alphabet corresponds to the possible network's reconfiguration operations. Apart from being in relation to the fault model, the length of the active path is an easy obtainable property [16]. Therefore, output symbols are mapped to the lengths of the active paths. Although the high-level model is exact in modelling the circuit, it is deliberately incomplete, since the FSA's states encode only a subset of the possible configurations. Due to the particular structural properties of the RSN, not all transitions are possible in all states. When an input does not correspond to a transition, the FSA is brought to a special *sink state* (Ω), that is a state with no output transitions and a *null* output symbol. For instance, in some networks it is possible to use a Scan Multiplexer whose configuration is based on the values of multiple configuration bits (n). However, such multiplexers do not necessarily have defined inputs for all possible configurations (2^n) in the ICL description of the network. Even though at the implementation level, either at the gate- or RTL-level, these pins might be tied to some other input or to logical $0/1$, to prevent any ambiguities, transitions to such unspecified configurations lead to a special state.

Given the stuck-at faults affecting SIBs and ScanMuxes, the same configuration operations may result in different network statuses on faulty circuits. Therefore, such faults are mapped to multiple transition faults on the high-level automaton. Taking into account the faulty automata and the good one, the goal of the diagnostic procedure is to produce a sequence of inputs able to make a unique discrimination between each one of them.

A greedy search strategy represents the basis of the proposed algorithm. Not all possible states nor all possible input symbols are considered, and, consequently, not all possible transitions. Nevertheless, the simulation of the automaton is exact, while any missing state or transition will cause the automaton to reach the *sink state*, that by construction cannot be further distinguished from any other state.

4.1 Finite State Automaton

Initially, the FSA is composed of only a state with no output transition and a *null* output symbol. Such *sink state* is used to denote a pathological condition, where the algorithm is not able to provide reliable results due to the approximation of the model. This state is characterized by its ambiguity with respect to any other state. Once entered, the FSA permanently remains in this state.

Next, the state when all configuration bits are set to the initial value, denoted as *reset state*, is added to the automaton. Then, for each SIB_i, two states are created: one with the SIB asserted and one with the SIB de-asserted. Similarly, for each SM, one state is created for each possible configuration.

Such a straightforward approach, however, is not always sufficient. Scan segments may be nested, and a resource accessible only when its parent SIB is asserted. The procedure for building the FSA detects such situations, and creates the necessary states to handle them. The transitions from the *reset state* to all these states are eventually added.

For instance, SIB_2 in Fig. 6 is only accessible when SIB_1 is asserted. Therefore, the FSA would first include the reset state ($\overline{SIB}_1, \overline{SIB}_2, \overline{SIB}_3$), and then the three states with only one SIB asserted { (SIB_1, $\overline{SIB}_2$, $\overline{SIB}_3$), ($\overline{SIB}_1$, SIB_2, $\overline{SIB}_3$), ($\overline{SIB}_1$, $\overline{SIB}_2$, SIB_3) }, finally, the state (SIB_1, SIB_2, $\overline{SIB}_3$).

Since we build the FSA in an incremental mode the following modifications are performed:

- for each transition in the good automaton, the possible faulty transitions are added;
- if the faulty transition would bring the automaton in a configuration not already encoded as a state, that specific state is added to the FSA;
- all nonexistent transitions between existing states are added to the automaton;
- eventually, all possible faulty transitions from all existing states are also added, but if one would bring the automaton in a configuration not encoded as a state, its destination is set to the *sink state*, meaning that the FSA is unable to model such situation.

As almost only the states with a hamming distance of 1 from the reset state are added to the FSA, the size of the automaton is linear in the number of configuration bits. It is possible to define an automaton with more states: for instance, at some point of the creation, all complementary states may be added as well. It is important to remember that the size of the automaton influences both the quality of the results and the performance of the algorithm.

While considering the possible transitions, some additional states corresponding to the configurations which may reduce the cost are also examined. Of course, it is important to remember that the highest priority is to continuously increase the number of diagnosed faults, while reducing the cost is a secondary goal. For example, states representing configurations in which accessible SIBs that provide access to the deepest hierarchical level are not asserted may increase the number of required sessions and therefore the cost. Additionally, configurations in which a SIB is still asserted while already all faults associated with it and its sub-hierarchical modules are diagnosed may increase the cost. If all faults affecting ScanMux and its sub-hierarchical modules are diagnosed, the configurations in which a ScanMux branch with not minimal length is included into the path might also increase the cost. It is worth noting that, although designers may explicitly define and include some additional states to this state or provide additional heuristic, experimental validations suggest that such extensions are unlikely to be beneficial.

For the network given in Fig. 7, FSA with its states and transitions is given in Fig. 5. The states are represented by circular shapes with the label and output symbol, while the lines with arrows denote the transitions between the states. Initially, only states filled with white color (*sDDDD*, *sDDDA*, *sDDAD*, *sDADD*, *sADDD*) are created. States in light grey are created consequently, dynamically, after applying the chosen input symbols and performing transitions on fault-free FSA. Remaining states, in dark grey are created as a result of transition on faulty FSAs. As for the transitions, some of them are created for examining the end states (dash lines). The transitions drawn in bold lines correspond to transitions of the fault-free FSA (t_1–{*sDDDD→sDDDA*}, t_2–{*sDDDA→sDDAD*}, t_3–{*sDDAD→DADD*}, t_4–{*sDADD→sADDD*}). The remaining transitions are executed on the faulty FSAs while applying the sequence of chosen input symbols (in solid style).

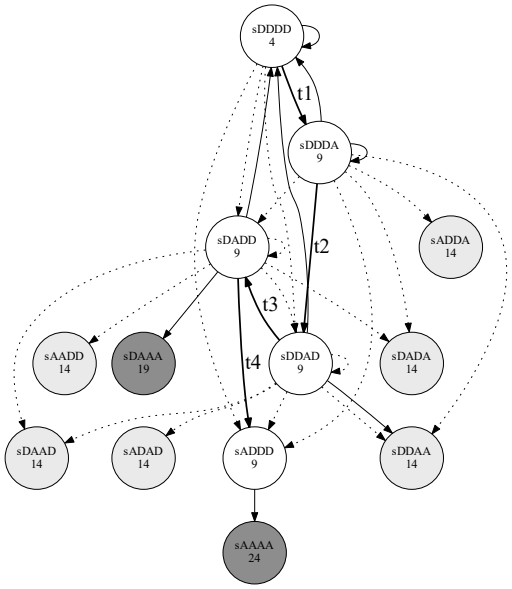

Fig. 5. Example of generating FSA for the network from Fig. 7

.

4.2 Search Algorithm

A sequence of *transition* and *observation* steps is constructed by the search algorithm. A *transition* corresponds to a change in the configuration of the RSN and it is performed by shifting an array of bits into the scan chain. On the other hand, an *observation* step does not change the configuration of the RSN and does not affect the FSA, since it is comprised only out of shift operations.

The goal of the diagnosis sequence generation procedure is to make the good circuit and the faulty ones pass through different states to be able to distinguish between them by comparing sequences of output symbols of the traversed states. For every circuit a sequence of scan chain lengths

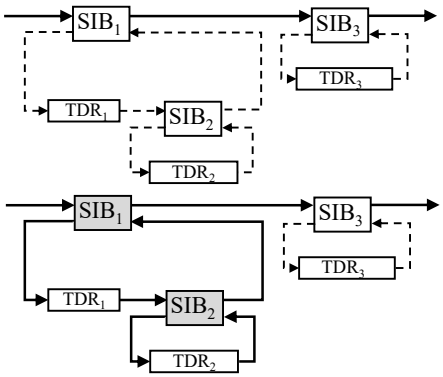

Fig. 6. Example of generating input symbols for SIB RSN.

is observed; if it is unique then a fault is considered to be diagnosed.

Let x be an input symbol for the FSA. The reset operation is denoted with **reset**, and it requires a single clock cycle to be performed; measuring the length of the scan chain is characterized with **observe** symbol. Moreover, it requires a certain number of clock cycles and does not affect the state of the FSA. Both concatenation of the two sequences and appending a symbol to an input sequence are expressed as additions, as no ambiguities are possible. The symbol $\varnothing$ denotes an empty symbol and has no effect on an input sequence, e.g., $\mathbf{t} = \mathbf{t} + \varnothing$. The state of the FSA is unambiguously defined with a sequence $\mathbf{t}$ of inputs starting with a **reset**, i.e., $\mathbf{t} = (\mathbf{reset}, +t_0, t_1, ..., t_k)$.

Two states that have undistinguishable output symbols are equivalent and are denoted with $s' \cong s''$. Conversely, non equivalent states have distinguishable output symbols and are denoted with $s' \ncong s''$. By definition, the sink state is equivalent to any other state $\forall s : s \cong \Omega$.

Let $\bar{\Lambda}_{\mathbf{t}}$ be the sequence of states $[\bar{s}_{\mathbf{reset}}, \bar{s}_{t_0}, \bar{s}_{t_1}, ..., \bar{s}_{t_k}]$, the FSA representing the fault-free circuit goes through after applying the input sequence $\mathbf{t}$, while $\Lambda^i_{\mathbf{t}}$ be the sequence of states $[s^i_{\mathbf{reset}}, s^i_{t_0}, s^i_{t_1}, ..., s^i_{t_k}]$ the FSA representing the circuit when fault i is present goes through when the same input sequence is applied. The two Λ sequences Λ^i_t and Λ^j_t are considered to be different if there are at least two different output symbols, corresponding to the same state position in both of the sequences (e.g., $s^i_{t_{k-1}}$ and $s^j_{t_{k-1}}$). Moreover, the sequence $\Lambda^i_{\mathbf{t}}$ is unique, if $\Lambda^i_{\mathbf{t}} \ncong \bar{\Lambda}_{\mathbf{t}}$ and $\forall j, j \neq i$, $\Lambda^i_{\mathbf{t}} \ncong \Lambda^j_{\mathbf{t}}$. Having a unique sequence Λ^i_t allows to mark the fault i as diagnosed, by appending an **observe** input symbol to $\mathbf{t}$. The fault i is said to be "diagnosable".

Algorithm 1 Greedy score step

```
function GREEDY(t, score)
    m ← ( )                                  ▷ Empty sequence of inputs
    for x ∈ {valid input symbols in s̄_t} do
        u ← t
        Append x to u
        FS_u ← (Λ̄_u, F_u, score)
        FS_m ← (Λ̄_m, F_m, score)
        if FS_u > FS_m then
            m ← u
    return m                                 ▷ Most promising sequence
```

Let $\mathrm{DF}(\bar{\Lambda}_{\mathbf{t}}, \mathbf{S_t})$ be the set of potentially diagnosable faults when the good circuit traversed the states in $\bar{\Lambda}_{\mathbf{t}}$ and the faulty ones $\mathbf{S_t} = (\Lambda^0_{\mathbf{t}}, \Lambda^1_{\mathbf{t}}, ..., \Lambda^{f-1}_{\mathbf{t}})$, i.e., the set of all faults that caused the faulty circuit to traverse the states in a unique Λ^i sequence. If an observation is performed, measuring the actual length of the RSN path, any unique difference would be observed and all such faults, diagnosed. Initially, all faults are annotated with an identical score (equal to -1). During the execution of the sequence generation procedure, these values are updated with the index number of the session in which a fault was diagnosed. If by running the DIAGNOSTIC SEQUENCE GENERATION procedure, not all faults from the $\mathbf{F}$ list are diagnosed, the same procedure is run from the beginning, this time with the updated **score** priority list.

A sequence of input symbols is generated iteratively since in each run the function GREEDY searches for the most optimistic input symbol for the sequence of inputs $\mathbf{t}$ to be extended with (Algorithm 1). In other words, the appended input symbol brings circuits in states where the highest number of faults with the lowest score can be diagnosed. A fault with a low score assigned means that it was either not diagnosed in the previous run, or it was diagnosed before some others (with a higher score). If no new fault can be diagnosed by adding a single transition, the function returns an empty input sequence.

Algorithm 2 Diagnostic Sequence Generation

```
procedure DPG(score)
    t ← (reset)                              ▷ Initial diagnostic sequence
    H ← {t}                                  ▷ History
    F ← {all diagnosable faults}             ▷ Active faults
    tscore ← {−1}                            ▷ Initialize fault score
    while |F| ≠ 0 do
        g ← Greedy(t, score)
        if empty(g) then                     ▷ The greedy failed
            Append reset to t                ▷ Start over
            for t' ∈ H do
                g' ← Greedy(t')
                if |DF(Λ̄_g', S_g')| > |DF(Λ̄_g, S_g)| then
                    g ← g'                   ▷ Alternative sequence
        Append g to t
        Append observe to t
        H ← H ∪ {g}                          ▷ Save sequence
        Remove DF(Λ̄_t, S_t) from F
        Set tscore for DF(Λ̄_t, S_t)
    score ← tscore
```

4.3 Diagnostic analysis

According to the fault model considered in this work, it is obvious that the faults affecting different types of RSN modules have a different effect and can therefore, be distinguished one from another. The faults affecting TDRs corrupt the inserted diagnostic sequence, changing the values of particular bits. For example, a stuck-at-1 on a scan cell within a TDR will result in observing only fixed values of 1 on all vector positions. On the other hand, SIB or ScanMux module affected by a fault may result in a path length different than the expected one. Consequently, the diagnostic sequence is not corrupted, but is observed before or after the expected number of clock cycles.

Faults affecting scan cells of a single TDR are all made equivalent. Accordingly, it is said for a fault to affect a TDR and the fault is distinguished at the TDR instance level. The presented approach is able to generate input symbols, i.e., configuration vectors that modify the state of a network in such a way that in each session, some of the SIB modules may become de-asserted and consequently some TDRs may be excluded from the active path, but only one module can be reconfigured to include a new segment to the active path. As shown in Fig. 7, all inputs applied to the network not affected by the reconfigurable module faults result in only one TDR being included in the active path in each session ($0001 - \mathrm{TDR}_1, 0010 - \mathrm{TDR}_2, 0100 - \mathrm{TDR}_3, 1000 - \mathrm{TDR}_4$).

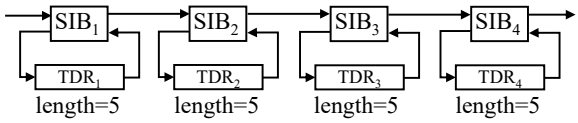

Fig. 7. 4 serially connected SIBs with TDRs of equal length of 5

Therefore, when a diagnostic sequence is applied, a session that fails will specify the faulty TDR.

If the network is designed in such a way, that two or more registers are located in the same segment, they can not be distinguished using only structural information and are referred to as *undistinguishable*. Diagnosing faults affecting this sub-set of TDRs may in some cases be possible using default capture values of the registers, since the TAP controller has to pass through the capture state in order to reach the shift state. Additionally, reading circuit's default capture values without inserting initialization sequence may be used to distinguish faults at the level of a single bit. However, this issue was not considered in this work. As stated in the standard, capture values may be fixed and predefined for some scan elements, i.e., TDRs. However, if the scan elements are defined, not as Write-Only (here, capture functionality is not necessary) but as Read-Only or Read-Write, capture source may be defined as an external signal or value from the shadow, i.e., update stage of that register. Additionally, an approach such as the one we propose may rely exclusively on structural information (given by ICL), before any detailed information on which types of instruments are to be integrated and how they are operated is available.

Faults affecting ScanMux modules may also not always be diagnosable. If a ScanMux module has at least two branches containing equal fixed length segments, then all faults forcing the ScanMux configuration to always select those branches are considered as *undistinguishable*.

One of the advantages of the proposed approach is that is able to handle types of networks constructed out of equally long TDRs placed behind serially connected SIBs. An example is given in Figure 7, with four SIB modules and four TDRs with the register length equal to 5. Since it is enough to observe a difference in length, SIB stuck-at-asserted faults in this case are easily detected. However, distinguishing between them is a more challenging task.

Table 3 shows the configurations the fault-free and the fault affected networks go through, when input sequence generated by the proposed approach is applied.

As illustrated by Fig. 8, after applying reset, depending on the location of the fault affecting a SIB module, the resulting scan chain can have not only a different length, but also a different position of particular scan cells. It can be observed that initially, the length of the scan chain in case of a fault free network and all stuck-at de-asserted faults ($\bar{s}, s^0, s^2, s^4, s^6$) is 4. In this case, the position of configuration bits within the scan chain is the same ($\mathrm{CB}_1, \mathrm{CB}_2, \mathrm{CB}_3, \mathrm{CB}_4$). On the other hand, for all stuck-at asserted faults (s^1, s^3, s^5, s^7), the scan chain is of length 9, with a different configuration bits arrangement, due to the TDRs which are now part of the active path. In case of s^1, register TDR_1 is in the active path, while e.g., in case of s^5, TDR_3 is a part of the active path. In the first case, all four SIB configuration bits (supposed that post-SIBs are used) are located after scan cells comprising TDR_1. On the other hand, in the second case two SIB configuration bits (CB_1 and CB_2) are preceding TDR_3, while two are following it (CB_3 and CB_4). As a consequence of having the same path length for pairs of different faults, at this point no fault can be diagnosed. Faults s^0, s^2, s^4, s^6 can not be distinguished between themselves and between correctly operating circuit (length 4), while faults s^1, s^3, s^5, s^7 are equivalent between themselves (length 9). Regardless the fault, a configuration sequence cv can be shifted in the scan chain. In cases where the actual length of the scan chain is lower than the one of the configuration vector, not all values in the configuration vector will be stored in the chain cells; some of them will be "cut-off", i.e., shifted out. By applying the update, new states will correspond to the ones shown in the second row of Table 3, thus allowing five faults to be diagnosed (s^1, s^3, s^5, s^6, s^7). Even though the lengths of the active path after applying the cv configuration vector in case of the fault-free network (s) and the network in which SIB_4 is affected by stuck-at asserted fault (s^7) are equal, the fault is diagnosed. When array of lengths of the active path in presence of the fault $[9, 9]$ is compared against others $\{[4, 9], [4, 9], [9, 24], [4, 9], [9, 19], [4, 9], [9, 14], [4, 4]\}$, it is determined to be unique.

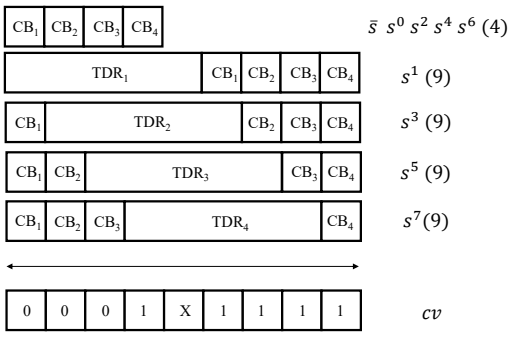

Fig. 8. Initial scan chain structure with conf. vector for network from Fig. 7

5 Experimental Results

For the purpose of validating the proposed algorithm, a subset of the ITC'16 suite of benchmark reconfigurable scan networks [11] was chosen. Not all benchmarks have been used since some of them include constructs which are currently not supported by our environment. However, some additional benchmarks, considered in [13], were included into the validation set to provide effectiveness comparison with the present approach. It is worth noting that some of the latter benchmarks are part of the ITC'16 set.

An in-house tool implementing the proposed algorithm was developed in Java. The tool is first of all able to find all reconfigurable modules, for which the faults affecting them are not distinguishable. As already discussed, this is due to their inability to produce a different path length. Moreover, the tool can generate a list of sets of TDRs; faults affecting TDRs in the same set are considered to be undistinguishable

TABLE 3
Diagnostic procedure for the network in Fig.7

Input	Fault free	SIB_1		SIB_2		SIB_3		SIB_4	
	$\bar{s}$	s@D-s^0	s@A-s^1	s@D-s^2	s@A-s^3	s@D-s^4	s@A-s^5	s@D-s^6	s@A-s^7
reset observe	DDDD (4)	DDDD (4)	ADDD (9)	DDDD (4)	DADD (9)	DDDD (4)	DDAD (9)	DDDD (4)	DDDA (9)
0001 observe	DDDA (9)	DDDA (9)	AAAA (24)	DDDA (9)	DAAA (19)	DDDA (9)	DDAA (14)	DDDD (4)	DDDA (9)
0010 observe	DDAD (9)	DDAD (9)		DDAD (9)		DDDD (4)			
0100 observe	DADD (9)	DADD (9)		DDDD (4)					
1000 observe	ADDD (9)	DDDD (4)							

between themselves, e.g., because they belong to the same segment.

The basic information on the benchmarks' evaluation set is reported in Table 4. Columns 2 and 3 give the number of SIBs and SMs for each network. The total number of SIB and SM configuration bits is reported in the fourth column. The depth of a module is equal to the number of nested modules controlling its segment. The hierarchical depth of the network corresponds to the highest module depth in the network and is provided in column 5. The sixth and seventh columns represent the maximum path length and the total number of scan cells in a given network, respectively. The upper part of the table contains the list of networks used to evaluate the approach proposed in [13], while the list of considered ITC'16 benchmarks is contained in the lower part of the same table.

TABLE 4
Benchmark networks list

Network	SIB	SM	Tot bits	Max depth	Max path	Scan cells
A586710	6	0	6	2	41,972	41,972
N17D3	7	8	15	4	372	462
N32D6	13	10	23	4	84,039	96,158
N49D0	16	18	34	1	949	1,114
N61D2	11	22	33	2	1,162	1,422
N73D14	29	17	46	12	190,526	218,869
N88D8	32	32	64	4	1,637	2,013
N100D2	31	37	68	3	1,833	2,293
N132D4	39	40	79	5	2,555	2,991
P22810	30	0	30	2	30,915	30,915
P34392	22	0	22	2	23,478	23,478
Mingle	10	3	13	4	171	270
TreeBalanced	43	3	48	7	5,219	5,581
TreeFlat_Ex	57	3	62	5	5,100	5,195
TreeUnbalanced	28	0	28	11	42,630	42,630
a586710	0	32	32	4	42,381	42,410
p22810	270	0	270	2	30,356	30,356
p34392	0	96	96	4	27,899	27,990
q12710	27	0	27	2	26,185	26,185
t512505	159	0	159	2	77,005	77,005
NE600P150	207	194	401	78	23,423	28,250
NE1200P430	381	430	811	127	88,471	108,148

A computer with an Intel i5-7200U processor and 8 GB of RAM was used to perform the experiments. Table 5 reports the experimental results obtained from running the proposed algorithm on the set of benchmark networks. In columns 2 and 3, the number of configuration vectors, i.e., the number of diagnosis vectors is given. In this table, the cost of performing the diagnostic procedure on a given network by applying the generated sequence is given in terms of number of clock cycles required to apply all configuration steps (cv, column 4) and all diagnostic steps (dv, column 5).

TABLE 5
IEEE 1687 algorithm experimental results

Network	cv	dv	Conf. cost [cc]	Test cost [cc]
A586710	6	7	44,168	377,435
N17D3	15	16	3,287	9,492
N32D6	23	24	989,654	3,041,145
N49D0	34	35	20,429	55,029
N61D2	33	34	25,882	67,055
N73D14	46	47	4,883,376	13,945,235
N88D8	63	64	54,488	160,405
N100D2	67	68	73,903	200,260
N132D4	77	78	115,928	317,291
P22810	30	31	62,191	1,023,787
P34392	22	23	65,008	606,897
Mingle	14	15	921	3,591
Tree Balanced	47	48	132,172	393,223
TreeFlat_Ex	62	63	184,536	515,997
TreeUnbalanced	28	29	149,021	1,386,204
a586710	61	62	75,525	2,703,562
p22810	301	302	6,472,380	15,697,407
p34392	187	188	50,712	5,296,399
q12710	25	26	34,082	716,281
t512505	159	160	190,736	12,512,221
NE600P150	399	400	4,186,745	13,563,116
NE1200P430	809	810	39,615,088	111,319, 225

A number of experiments has been run to evaluate the effectiveness of the high-level fault model used in this work. Selected benchmarks have been synthesised using NanGate 45 nm Open Cell Library. Synopsys tool TetraMAX[1] was used to perform fault simulation on the designs at the gate-level by applying test sequences generated by the method in [18]. The fault simulation results showed that in general a high or complete stuck-at fault coverage is achieved. Certain corner cases appeared as a result of faults affecting modules positioned deep in the hierarchy and with the ScanMuxes having more than 2 inputs. Finally, the stuck-at faults affecting the update logic and the flipflops are either covered or they propagate long sequences of Xs in the circuit. Since the proposed test sessions demand for a precise sequence of 0 and 1 s to be observed on scan output ports, faults that propagate sequences of Xs can be safely marked as covered.

1. TetraMAX ATPG User Guide, Version M-2016.12, Synopsys, www.synopsys.com.

By using the proposed approach and according to the fault model which was described previously, all distinguishable faults were diagnosed, thus reaching 100% diagnostic coverage. Considering the discussion in Section 2.1, faults affecting modules that are not considered as undistinguishable are considered to be distinguishable. The comparison of the proposed approach with the approach from [13] is given in the upper part of Table 6. The lower part of the same table refers to the results obtained on the sub-set of ITC'16 benchmarks. The second column of the table gives the total number of clock cycles needed to apply the input sequence generated by the approach proposed in Section 4.2. Comparison data on diagnosis duration in clock cycles are taken from [13] and are provided in column 3. The fourth column shows the comparison against the current result. The ratio between the duration of the diagnostic sequence generated by the previous and new approach is reported, thus showing how faster the latter one is. Due to the Java's non-determinism at run-time no accurate timing in terms of CPU time for generating the sequence is possible. Therefore, only the program's total execution time is reported in column 5. The number of pairs of undistinguishable faults affecting TDRs and SMs is reported in columns 6 and 7.

As it can be observed from the Table 6, in all of the cases where comparison data exists, the time required to perform the diagnosis is significantly reduced, up to 43 times. Although no data regarding the execution time of the previous algorithm is provided, it is evident from the Table 6 that the presented algorithm is efficient and fast to execute, since in most of the cases, the required time is measured in the order of seconds. Exceptionally, more than one hour was needed for three networks.

6 Conclusions

In summary, we have proposed a new sequence generation technique to diagnose permanent faults in RSNs resorting to an FSA model of the circuit and a greedy search algorithm. Experimental results demonstrate that the presented approach outperforms the previous ones in terms of number of clock cycles required to run the generated diagnostic sequence. Furthermore, this technique can be applied to a wide range of network types of different complexity since for all of the test cases and benchmark networks full diagnostic coverage has been reached while keeping the computation effort under control. The future work should focus on extending the technique to support different fault models.

Acknowledgements

The work has been partially supported by the European Commission through the Horizon 2020 RESCUE-ITN project under the agreement No. 722325.

References

[1] "IEEE standard for access and control of instrumentation embedded within a semiconductor device," *IEEE Std 1687-2014*, pp. 1–283, Dec 2014.

[2] K.-J. Lee and M. A. Breuer, "A universal test sequence for cmos scan registers," in *Proceedings of the IEEE 1990 Custom Integrated Circuits Conference*. IEEE, 1990, pp. 28–5.

[3] S. Maka and E. J. McCluskey, "ATPG for scan chain latches and flip-flops," in *1997 15th IEEE VLSI Test Symposium*. IEEE, 1997, pp. 364–369.

[4] 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," in *2008 26th IEEE VLSI Test Symposium (VTS)*. IEEE, 2008, pp. 79–84.

[5] H. Chen, Z. Qi, L. Wang, and C. Xu, "A scan chain optimization method for diagnosis," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, Oct 2015, pp. 613–620.

[6] Y. Huang, R. Guo, W. T. Cheng, and J. C. M. Li, "Survey of scan chain diagnosis," *IEEE Design Test of Computers*, vol. 25, no. 3, pp. 240–248, May 2008.

[7] Y. Huang, X. Fan, H. Tang, M. Sharma, W. T. Cheng, B. Benware, and S. M. Reddy, "Distributed dynamic partitioning based diagnosis of scan chain," in *2013 31st IEEE VLSI Test Symposium (VTS)*, April 2013, pp. 1–6.

[8] W. H. Lo, A. C. Hsieh, C. M. Lan, M. H. Lin, and T. Hwang, "Utilizing circuit structure for scan chain diagnosis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2766–2778, Dec 2014.

[9] S. Kundu, S. Chattopadhyay, I. Sengupta, and R. Kapur, "Scan chain masking for diagnosis of multiple chain failures in a space compaction environment," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 7, pp. 1185–1195, July 2015.

[10] J. Ye, Y. Huang, Y. Hu, W. T. Cheng, R. Guo, L. Lai, T. P. Tai, X. Li, W. Changchien, D. M. Lee, J. J. Chen, S. C. Eruvathi, K. K. Kumara, C. Liu, and S. Pan, "Diagnosis and layout aware (DLA) scan chain stitching," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 3, pp. 466–479, March 2015.

[11] A. Tšertov, A. Jutman, S. Devadze, M. Sonza Reorda, E. Larsson, F. G. Zadegan, R. Cantoro, M. Montazeri, and R. Krenz-Baath, "A suite of IEEE 1687 benchmark networks," in *2016 IEEE International Test Conference (ITC)*. IEEE, 2016, pp. 1–10.

[12] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of SOCs," in *Proceedings. 2002 International Test Conference*. IEEE, 2002, pp. 519–528.

[13] R. Cantoro, M. Montazeri, M. Sonza Reorda, F. G. Zadegan, and E. Larsson, "Automatic generation of stimuli for fault diagnosis in IEEE 1687 networks," in *2016 22nd IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2016, pp. 167–172.

[14] F. G. Zadegan, U. Ingelsson, G. Carlsson, and E. Larsson, "Access time analysis for IEEE P1687," *IEEE Transactions on Computers*, vol. 61, no. 10, pp. 1459–1472, 2012.

[15] ——, "Design automation for IEEE P1687," in *2011 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Mar 2011, pp. 1–6.

[16] R. Cantoro, M. Montazeri, M. Sonza Reorda, F. G. Zadegan, and E. Larsson, "On the testability of IEEE 1687 networks," in *2015 24th IEEE Asian Test Symposium (ATS)*. IEEE, 2015, pp. 211–216.

[17] R. Cantoro, L. San Paolo, M. Sonza Reorda, and G. Squillero, "An evolutionary technique for reducing the duration of reconfigurable scan network test," in *2018 21st IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2018.

[18] R. Cantoro, F. G. Zadegan, M. Palena, P. Pasini, E. Larsson, and M. Sonza Reorda, "Test of reconfigurable modules in scan networks," *IEEE Transactions on Computers*, 2018.

[19] R. Cantoro, A. Damljanovic, M. Sonza Reorda, and G. Squillero, "A new technique to generate test sequences for reconfigurable scan networks," in *2018 IEEE International Test Conference (ITC)*. IEEE, Oct 2018.

[20] A. Jutman, S. Devadze, and K. Shibin, "Effective scalable IEEE 1687 instrumentation network for fault management," *IEEE Design Test*, vol. 30, no. 5, pp. 26–35, Oct 2013.

[21] A. Jutman, K. Shibin, and S. Devadze, "Reliable health monitoring and fault management infrastructure based on embedded instrumentation and IEEE 1687," in *2016 IEEE AUTOTESTCON*, Sept 2016, pp. 1–10.

[22] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Reconfigurable scan networks: Modeling, verification, and optimal pattern generation," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 2, p. 30, 2015.

[23] F. G. Zadegan, R. Krenz-Baath, and E. Larsson, "Upper-bound computation for optimal retargeting in IEEE 1687 networks," in *2016 IEEE International Test Conference (ITC)*. IEEE, 2016, pp. 1–10.

TABLE 6
Experimental comparison of the proposed algorithm vs. [13]

Network	Diagnostic sequence duration [clock cycles]	[13]	[13] vs. proposed	Runtime (wall clock)	Und. pairs of faults TDR	Und. pairs of faults CM
A586710	421,603	3,879,326	9.20x	0s	0	0
N17D3	12,779	48,099	3.76x	1s	4	0
N32D6	4,030,799	25,038,071	6.21x	0s	16	0
N49D0	75,458	263,866	3.50x	1s	105	0
N61D2	92,937	333,280	3.59x	1s	260	0
N73D14	18,828,611	320,437,112	17.02x	2s	78	0
N88D8	214,893	2,065,800	9.61x	4s	68	0
N100D2	274,163	2,219,397	8.10x	5s	206	0
N132D4	433,219	3,900,952	9.00x	17s	435	4
P22810	1,085,978	46,601,832	42.91x	1s	0	0
P34392	671,905	20,665,284	30.76x	0s	0	0
Mingle	4,512			19s	0	0
Tree Balanced	525,395			48s	9	3
TreeFlat_Ex	700,533			40s	14	3
TreeUnbalanced	1,535,225			23s	14	0
a586710	2,779,087			22s	5	0
p22810	22,169,787			1.3h	6	0
p34392	5,347,111			3m	14	0
q12710	750,363			17s	0	4
t512505	12,702,957			4m	0	0
NE600P150	17,749,861			5h	427	4
NE1200P430	150,934,088			15h	643	4

[24] M. A. Kochte, R. Baranowski, M. Sauer, B. Becker, and H.-J. Wunderlich, "Formal verification of secure reconfigurable scan network infrastructure," in *2016 21th IEEE European Test Symposium (ETS)*. IEEE, 2016, pp. 1–6.

[25] D. Ull, M. Kochte, and H. J. Wunderlich, "Structure-oriented test of reconfigurable scan networks," in *2017 26th IEEE Asian Test Symposium (ATS)*, Nov 2017, pp. 127–132.

[26] J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz, and M. Thornton, "Don't forget to lock your sib: hiding instruments using p1687," in *2013 IEEE International Test Conference (ITC)*, Sep. 2013, pp. 1–10.

[27] A. Zygmontowicz, J. Dworak, A. Crouch, and J. Potter, "Making it harder to unlock an lsib: Honeytraps and misdirection in a p1687 network," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[28] R. Baranowski, M. A. Kochte, and H. Wunderlich, "Securing access to reconfigurable scan networks," in *2013 22nd Asian Test Symposium*, Nov 2013, pp. 295–300.

[29] M. A. Kochte, M. Sauer, L. R. Gomez, P. Raiola, B. Becker, and H. Wunderlich, "Specification and verification of security in reconfigurable scan networks," in *2017 22nd IEEE European Test Symposium (ETS)*, May 2017, pp. 1–6.

[30] A. Atteya, M. A. Kochte, M. Sauer, P. Raiola, B. Becker, and H. Wunderlich, "Online prevention of security violations in reconfigurable scan networks," in *2018 IEEE 23rd European Test Symposium (ETS)*, May 2018.

[31] R. Cantoro, A. Damljanovic, M. Sonza Reorda, and G. Squillero, "A semi-formal technique to generate effective test sequences for reconfigurable scan networks," in *2018 IEEE International Test Conference in Asia (ITC-Asia)*. IEEE, Aug 2018.

Riccardo Cantoro (M'18) received the MS degree in computer engineering from Politecnico di Torino, Italy, in 2013, and his PhD degree in computer and control engineering in 2017 from the same university. He is a currently a researcher in the Department of Computer Engineering, Politecnico di Torino. His research interests include software-based functional test of SoCs and memories, data analysis, and reconfigurable scan networks.

Aleksa Damljanovic (StM'18) received the MS degree in electrical engineering and computing from the School of Electrical Engineering, University of Belgrade, Serbia, in 2017. He is currently a PhD student in the Department of Computer Engineering, Politecnico di Torino. His research interests include reconfigurable scan networks, and validation of secure and reliable electronic systems.

Matteo Sonza Reorda (F'16) received the MS degree in electronics and PhD degree in computer engineering both from the Politecnico di Torino, in 1986 and 1990, respectively. He currently is a full professor with the Department of Control and Computer Engineering of the same University. He is an IEEE Fellow. His research interests include test of SoCs and fault tolerant electronic system design.

Giovanni Squillero (SM'14) received his MS degree and PhD degree in computer science both from the Politecnico di Torino in 1996 and 2001, respectively. He is an associate professor with the Department of Control and Computer Engineering of the same University. His research interests include the whole spectrum of bio-inspired metaheuristics, computational intelligence, and selected topics from machine learning.