Graceful Degradation of Reconfigurable Scan Networks

Erik Larsson, Zehang Xiang, and Prathamesh Murali

***Abstract*— Modern integrated circuits (ICs) include thousands of on-chip instruments to ensure that specifications are met and maintained. Scalable and flexible access to these instruments is offered by reconfigurable scan networks (RSNs), e.g., IEEE Std. 1687. As RSNs themselves can become faulty, there is a need to exclude and bypass faulty parts so that remaining instruments can be used. To avoid keeping track and updating description languages for each individual IC, we propose an on-chip hardware block that makes adjustments according to the fault status of a particular IC. We show how this block enables test for faulty scan chains, localization of faulty scan chains, and repair by excluding faulty scan chains. We made implementations and experiments to evaluate the overhead in terms of transported data and area.**

***Index Terms*— Diagnosis, IEEE Std. 1687, IEEE Std. P1687.1, localization, repair, test.**

I. Introduction

The semiconductor development toward smaller, faster, and more transistors gives advantages, such as more functionality, better performance, and lower power consumption. However, it is increasingly challenging to avoid malfunctioning. Smaller and faster transistors lead to tighter margins, which in combination with more transistors increase the risk of malfunctioning. To avoid malfunctioning, modern integrated circuits (ICs) are increasingly equipped with embedded (on-chip) instruments for testing, tuning, trimming, configuration, and so on [1]. These instruments, which can be in the range of thousands, are accessed throughout the ICs' life cycle: from prototype, debug, test, and validation to in-field monitoring and test [2].

Access to instruments requires an on-chip infrastructure connecting the instruments and an interface (port) to the IC's boundary (pins). Reconfigurable scan networks (RSNs), such as IEEE Std. 1687 networks, offer flexible and scalable access to instruments. The main interface for IEEE Std. 1687 is the IEEE Std. 1149.1 test access port (TAP). Fig. 1 shows a system with three instruments connected using IEEE Std. 1687.

IEEE Std. 1687 includes two description languages: instrument connectivity language (ICL) and procedural description language (PDL) [3]. ICL describes how instruments are interconnected. Fig. 1 shows the schematic equivalent of the network's ICL. PDL describes how to operate on instruments. Fig. 1 shows PDL to concurrently write data to instrument *i1* and read data from instrument *i3*.[1] Access (test) patterns are created by an electronic design automation (EDA) tool or an embedded controller with PDL and ICL as inputs. For the PDL in Fig. 1, smart access patterns include instruments *i1* and *i3*, while instrument *i2* is excluded from the active scan path as the PDL specifies operations on instruments *i1* and *i3*, but not on instrument *i2*. Dynamic reconfiguration of the active scan path to include or exclude instruments can be achieved by the use of segment insertion bits (SIBs).

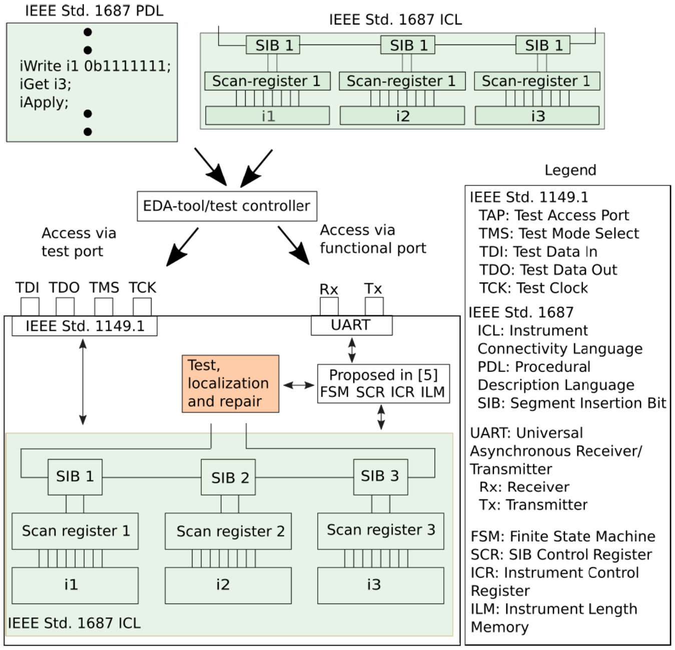

Fig. 1. Today's and future solution to access an IEEE Std. 1687 network.

As some ICs do not have an IEEE Std. 1149.1 TAP, the IEEE Std. P1687.1 [4] explores how to use functional ports, such as serial peripheral interface (SPI), interintegrated circuit (I2C), and universal serial bus (USB) to access IEEE Std. 1687 networks. Different from IEEE Std. 1149.1, where the TAP is described in detail, the working group of IEEE Std. P1687.1 is working toward a standard without detailing fixed hardware. A main question becomes: what include in the hardware placed between an IEEE Std. 1687 network and a functional port?

We have previously explored the impact on transporting data when including key information from PDL and ICL in a hardware component placed between a functional port, e.g., universal asynchronous receiver-transmitter (UART), and IEEE Std. 1687 [5] (see Fig. 1). The basic assumption was that there are no faults in the IEEE Std. 1687 networks, which means that description languages (PDL and ICL) correspond to the physical implementation (the IEEE Std. 1687 network). In this work, we explore cases when description languages do not correspond to the physical implementation due to faults in the IEEE Std. 1687 network. The motivation of the work is as follows. PDL and ICL can be stored in a central database shared among several ICs or stored embedded (compressed) locally near each individual IC. In both cases, PDL and ICL need to be updated according to the unique status of individual ICs. For example, assume a central database with PDL and ICL serving many ICs. As long as all ICs are free from faults, the same PDL and ICL can be used for all ICs. However, as soon as one IC has faults, for example, a faulty scan-register, description languages for this IC must be modified. For example, assume that scan register *3* (Fig. 1) is faulty, and then, the iApply group, for this particular IC, must be updated such that iGet *i3* is removed, which makes instrument *i3* to be excluded from

Manuscript received November 30, 2020; revised March 12, 2021; accepted March 27, 2021. Date of publication May 27, 2021; date of current version June 29, 2021. *(Corresponding author: Erik Larsson.)*

The authors are with the Department of Electrical and Information Technology, Lund University, 22100 Lund, Sweden (e-mail: erik.larsson@eit.lth.se).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3076593.

Digital Object Identifier 10.1109/TVLSI.2021.3076593

[1] iGetReadData (iGet) reads information from an instrument.

the active scan path. In the worst case, there is a need to keep individual versions of ICL and PDL for each individual IC, which is infeasible in practice.

The objective of this brief is to enable graceful degradation of IEEE Std. 1687 networks where faulty parts are excluded without the need of updating description languages (PDL and ICL). This means that original PDL and ICL assuming no faults can be used even in the case when their physical implementation, the IEEE Std. 1687 network, does not match any longer due to faults. We believe that this important aspect has not been addressed prior to this work. The objective is met by developing an on-chip hardware block that makes automatic adjustments according to the fault status of a particular IC. This hardware block makes it possible to test whether scan chains are faulty, localize (pinpointing) faulty scan chains, and repair networks by excluding faulty scan chains. We implemented IEEE Std. 1687 networks with 50, 100, and 150 instruments and proposed hardware block to evaluate the overhead in terms of data to be transported and area. We compare a theoretical computation of overhead for direct operation on the IEEE Std. 1687 network against a software-based scheme and proposed hardware-based scheme.

This brief is organized as follows. The related work is in Section II and an introduction to the hardware component and protocol to use a functional port to interface IEEE Std. 1687 is in Section III. The schemes for test, localization, and repair are in Sections IV–VI, respectively. The experimental results with implementation on a field-programmable gate array (FPGA) and evaluation of area and the amount of data transported are in Section VII. This brief is concluded in Section VIII.

II. Related Work

While there are a number of works on analysis [6], design [7], and fault management [8], [9] of IEEE Std. 1687, all these works assume that the IEEE Std. 1687 network is without any faults. Several works have addressed testing and localization (diagnostic) for regular scan chains [10]–[12] and for IEEE Std. 1687 networks [13]. Kundu [10] presented an early work on testing and diagnosing faults in scan chains. The basic principle is to shift a test sequence through the scan chain, like "001100…11," without performing capture. If there is a mismatch between the shift-out sequence and the shift-in sequence, there are one or more faults in the scan chain. For localization, the results from the automatic test pattern generation (ATPG) test vectors are used to pinpoint the faulty scan flip-flops. Cantoro *et al.* [13] developed a technique to test and diagnose RSNs. To the best of our knowledge, there is no work addressing the repair of RSNs.

III. Background

We previously explored the impact of including different amount of information in a hardware component placed between a functional port and an IEEE Std. 1687 network [5]. The most efficient solution, shown in Fig. 2, is based on a finite-state machine (FSM) complemented with three parts: SIB control register (SCR), instrument control register (ICR), and instrument length memory (ILM). The SCR keeps desired values of SIBs, the SCR keeps desired operation of an instrument, and ILM keeps the length of each instrument. The hardware component is operated using two types of commands: control and data. Control commands are used to set SCR and ICR and data commands are used to transport data for instruments. Hence, each iApply group is translated into one or more control commands and one or more data commands.

To illustrate, the iApply group in Fig. 1 is retargeted into two control commands and one data command, in total 7 bytes of information. The first control command, bytes 1 and 2 in Fig. 2, makes SIB 1 active and sets instrument *i1* in write mode. The details

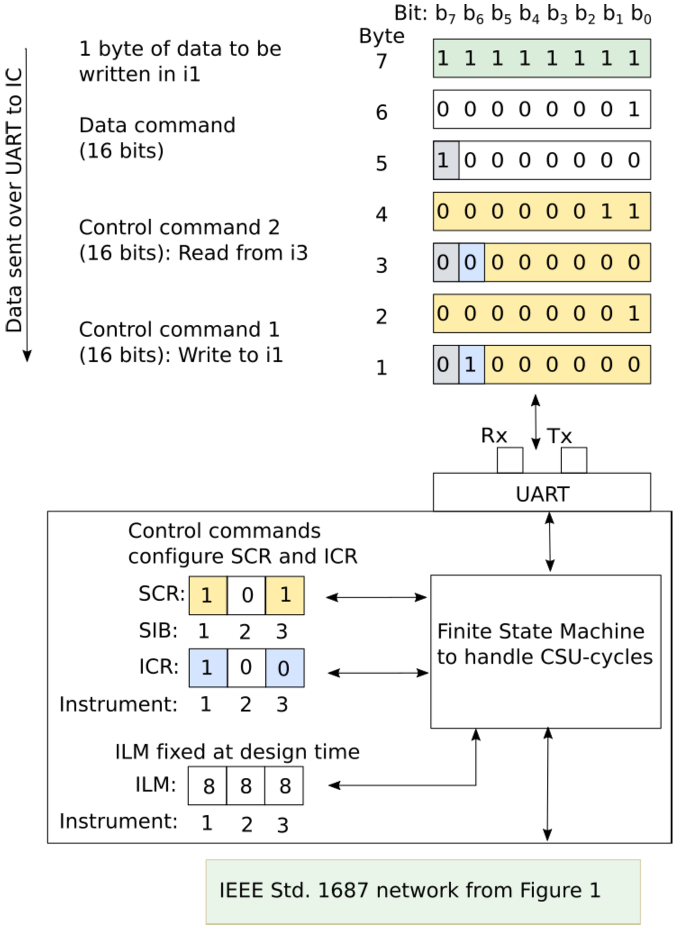
Bit: b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
Byte
Data sent over UART to IC
1 byte of data to be written in i1
7 1 1 1 1 1 1 1 1
Data command (16 bits)
6 0 0 0 0 0 0 0 1
5 1 0 0 0 0 0 0 0
Control command 2 (16 bits): Read from i3
4 0 0 0 0 0 0 1 1
3 0 0 0 0 0 0 0 0
Control command 1 (16 bits): Write to i1
2 0 0 0 0 0 0 0 1
1 0 1 0 0 0 0 0 0
Rx
Tx
UART
Control commands configure SCR and ICR
SCR: 1 0 1
SIB: 1 2 3
ICR: 1 0 0
Instrument: 1 2 3
Finite State Machine to handle CSU-cycles
ILM fixed at design time
ILM: 8 8 8
Instrument: 1 2 3
IEEE Std. 1687 network from Figure 1

Fig. 2. Hardware and protocol to form shift-in sequence for PDL in Fig. 1.

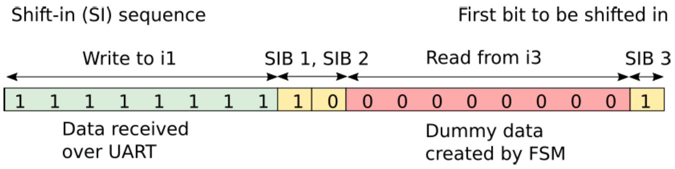

Fig. 3. Shift-in sequence from Fig. 2.

are as follows. Bit $b_7 = 0$ in the first byte indicates that the current byte and the following byte form a control command. Bit $b_6 = 1$ in the first byte indicates that a *write* operation should be performed. The following 14 bits, which hold the value 1, indicate that SIB 1 should be active so that instrument *i1* is included in the active scan path. The next two bytes, bytes 3 and 4 in Fig. 2, are also forming a control command, indicated by bit $b_7 = 0$ in byte 3. This control command has $b_6 = 0$, which informs that a *read* operation should be performed. The following 14 bits, which hold the value 3 (0b11), inform that SIB 3 should be active so that instrument *i3* is included in the active scan path. The following 3 bytes, bytes 5, 6, and 7 in Fig. 2, form a data command as $b_7 = 1$ in byte 5. The remaining 15 bits in bytes 5 and 6 are used to specify the number of bytes with data that follow. In this example, the 15 bits specify the value 1, meaning that one byte of data follows. The data in byte 7 are the data that should be written to instrument *i1*.

Fig. 3 shows the generation of shift-in data. When a control command arrives, the hardware component automatically resets SCR and ICR, and then, these registers are set according to the control commands, see above. When data commands arrive, the hardware translator begins operating the IEEE Std. 1687 network. First, the active scan path is set by traversing SCR and shifting the content to the IEEE Std. 1687 network. The bits shifted out are ignored (discarded) by the hardware component as the bits do not contain any useful information. Second, the shift sequence for the active scan path is created. We describe the shift-in sequence. The FSM begins checking the SCR at the highest value, in this example, 3 [SCR(3)], and includes that bit in the shift-in sequence. As SCR(3) $= 1$ instrument *i3* is included and ICR(3) is checked to learn that a *read* operation should be performed, this means that data need to be shifted in such that the content of instrument *i3* is shifted out. These additional (dummy) shift-in data are created by the FSM. The number of bits to shift is given by ILM(3). Then, the FSM proceeds with SCR(2). As SCR(2) $= 0$, indicating that instrument *i2* is not in the active scan path, the FSM adds a 0 to the shift-in sequence and focuses on next bit in SCR, which is SCR(1). SCR(1) $= 1$, which means that instrument *i1* should be included in the active scan path, and as ICR(1) $= 1$, a *write* operation should be performed. The FSM gets the length of instrument *i1* from ILM(1) and takes data from the UART buffer and adds it to the shift-in sequence. Fig. 2 shows the created shift-in sequence and how its information will set the SIBs and the instruments. The hardware component can with the support of SCR, ICR, and ILM, create dummy bits when needed, and discard not needed data such that only useful information is transported out from the IC. Applying the PDL in Fig. 1 results in that only the information in instrument *i3* is returned, as this is the only requested information.

IV. Test

The objective of the test procedure is to determine whether there are any faults in any of the scan chains. The section is organized into three parts: IEEE Std. 1687-, software-, and hardware-based tests. For each part, we describe the effort needed to perform the test. The basic principle of the three parts is built on a traditional scan-chain test where a test sequence is shifted through the scan chain, but no capture and update is used. For test evaluation, the shifted output sequence is compared against the applied test sequence. Different from traditional scan chains, RSN offers the possibility to configure the active scan path. For RSNs designed as in Fig. 1, our test principle is to first set the active scan path such that all instruments are included. For the example in Fig. 1, this means that the active scan chain includes instruments *i1, i2,* and *i3*.

A. IEEE Std. 1687-Based

The scheme is straightforward. First, the active scan path is set to include all instruments, which means that three bits are shifted in and, concurrently, three bits are shifted out, in total six bits of data. In general, N bits are shifted in and N bits are shifted out for a flat RSN with N SIBs. Second, a test sequence, $001100\ldots11$, equal to the active scan path, which for the example in Fig. 1 is 27 $(8+8+8+3)$ bits, is shifted in. During the shift-in of this pattern of length 27 bits, 27 bits are shifted out. To "push through" the test sequence such that the test response is observable, another 27 bits are shifted in, and consequently, 27 bits of actual test response are shifted out. The total number of bits becomes 27×4. In general, for an RSN with N SIBs, one instrument per SIB, and the length of instrument i that is given by $l(i)$, the total number of bits is given by

$$6 \times N + 4 \times \sum_{i=1}^{N} l\left(i\right). \tag{1}$$

B. Software-Based

The software-based test scheme assumes a hardware component and protocol [5], which we extended with a mechanism to not perform capture and update when applying an iApply group if desired. The idea of the test function is to include all instruments in the active scan path, apply a test sequence, $001100\ldots11$, to all instruments, and receive the output from the IEEE Std. 1687 network. For the system in Fig. 1, the sequence would be as follows:

```
iWrite i1 0b00110011;
iWrite i2 0b00110011;
iWrite i3 0b00110011;
iApply (no capture and no update);
iGet i1
iGet i2
iGet i3;
iApply (no capture and no update);
```

C. Hardware-Based

In the hardware-based test scheme, the hardware component includes the proposed block and a command to perform the test of scan chains. The test command consists of 2 bytes, in a similar way as the data and control commands (see Section III). When the hardware receives a test command, the block automatically sets the active scan path to include all instruments, generates and shifts in a test sequence, and compares the output sequence with the expected test sequence. The output (return value) is a single bit indicating whether there were any faults or not (which becomes a byte, the smallest unit to transport in UART).

V. Localization

The objective of localization is to pinpoint faulty scan chains. The principle is built on traditional scan-chain test and diagnosis (localization). The IEEE Std. 1687 network is configured so that only one scan chain is active at a time. For each individual segment of the scan chain, a test sequence is shifted through the scan chain and the output is compared against the input sequence. The section is organized into three parts: IEEE Std. 1687-based, software-based, and hardware-based, and for each, we describe the effort needed to perform localization.

A. IEEE Std. 1687 Based

The localization procedure assumes that the IEEE Std. 1687 network is in a reset state, which for the example in Fig. 1 means that the active scan path includes only the three SIBs. First, the active scan path is set to include the first instrument, which means that three bits are shifted in and, concurrently, three bits are shifted out, in total six bits of data. Second, a test sequence, $001100\ldots11$, is equal to the active scan path, which includes instrument 1. For the example in Fig. 1, 11 $(8+3)$ bits are shifted in. During the shift-in, 11 bits are shifted out. To "push through" the test sequence such that it becomes observable, another 11 bits are shifted in, and consequently, 11 bits are shifted out, the actual test response. In this example, the number of bits shifted in and shifted out is 50 $(3+3+11+11+11+11)$ for one instrument. As there are three instruments in Fig. 1, the total number of bits becomes 150 (3×50). In general, the number of bits shifted in and shifted out during a localization procedure of a flat RSN with N SIBs, one instrument per SIB, and $l(i)$ the length of instrument i that is given by

$$6 \times N^2 + 4 \times \sum_{i=1}^{N} l\left(i\right). \tag{2}$$

B. Software-Based Localization

The software-based localization scheme has several similarities with the test function (see Section IV-B). We assume the hardware component and protocol [5] and have added a mechanism to not perform capture and update of iApply if desired. Different from testing (Section IV-B), localizations include one instrument at a time in the active scan path, apply the test sequence, 001100 . . . 11, and receive the output from the IEEE Std. 1687 network. For the system in Fig. 1, the commands would be as follows:

```
iWrite i1 0b00110011;
iApply (no capture and no update);
iGet i1;
iApply (no capture and no update);
iWrite i2 0b00110011;
iApply (no capture and no update);
iGet i2;
iApply (no capture and no update);
iWrite i3 0b00110011;
iApply (no capture and no update);
iGet i3;
iApply (no capture and no update);
```

C. Hardware-Based Localization

The hardware-based localization resembles the hardware-based test, with the difference that the proposed block, when initiated, automatically traverses the instruments one at a time. When the block receives a localization command, the block sets up the active scan path to include instruments one at a time, shifts in a test sequence, and compares the output sequence with the expected test sequence. We created a dedicated command to make the block initiate localization. The command is constructed in the same way as the test command, 2 bytes of data to initiate and 1 return bit to indicate whether any faults were detected (one byte as the smallest unit for UART is one byte).

VI. Repair

Repair is to make it possible to make use of a partially faulty RSN by excluding instruments with faulty scan chains. Given is knowledge about which of the scan chains in the IEEE Std. 1687 network that is faulty. We explore two alternative solutions to repair: software- and hardware-based.

A. Software-Based Repair

In software-based repair, the PDL is modified according to the faults in scan chains. For the system in Fig. 1, assume that it is known that the scan chain related to instrument *i3* is faulty. This information is considered together with ICL and PDL in the retargeting such that the PDL is changed from this

```
iWrite i1 0b1111111;
iGet i3;
iApply;
```

to this PDL where instrument *i3* is excluded

```
iWrite i1 0b1111111;
iApply;
```

With the above modification of the PDL, the partially faulty RSN can be used.

B. Hardware-Based Repair

For hardware-based repair, the original PDL is applied and the hardware block automatically excludes faulty scan registers from the active scan path. For example, if the scan chain related to instrument *i3* in Fig. 1 is faulty, the test and localization process has set the repair register to hold the value 110. This indicates that instrument *i3* will not be included in the scan path due to the 0, while the other instruments, which are not faulty, indicated by 1 (see Fig. 4). When the original PDL in Fig. 1 is applied, the SCR will contain 101 as the PDL specifies that instruments *i1* and *i3* should be active (see Fig. 4). Given the combination of the repair register and SCR, the FSM performs a bitwise AND-operation between the two registers to receive the SCR to be used $\boxed{1}\boxed{0}\boxed{0}$. We observe that the "used SCR" does not include instrument *i3*, which is faulty, and hence, the FSM in our component automatically excludes instrument *i3*, while instrument *i1* is included. The key advantage is that the original PDL can be used and there is no need of additional retargeting due to faults in the IEEE Std. 1687 network.

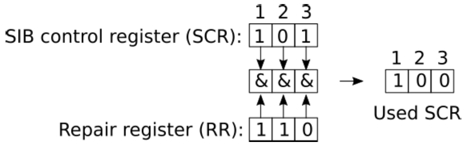

Fig. 4. Repair by excluding instrument i3.

VII. Experimental Results

The objective of the experiments is to evaluate overhead in terms of data, that is, the number of bits, transported to and from the IC and the area utilization of the proposed scheme for test and localization of RSNs. For repair, there are no separate results as the hardware solution automatically repairs the IEEE Std. 1687 network, and with the software solution, the PDL is modified according to the defects in the IEEE Std. 1687 network.

As an experimental platform, we used a Nexys 4 DDR with an Artix-7 (XC7A100T-1CSG324C) FPGA. We implemented three IEEE Std. 1687 designs with 50, 100, and 150 instruments, respectively. The instruments are connected in a flat manner with one SIB per instrument, as shown in Fig. 1. The length of each instrument is 8 bits, and for communication with the outside, the IEEE Std. 1687 network is connected using UART. The overhead for the IEEE Std. 1687 network scheme is computed with (1) and (2).

Table I shows the number of bits transported to and from the IC for the test process. The hardware-based solution only needs 16 bits to initiate the command and 1 bit is to report whether there were any faults. As UART is used for communication, the least amount of data to be produced is packaged in one byte. The total number of bits becomes 24. As expected, the number of bits for the IEEE Std. 1687 and the software-based alternatives increase with the number of instruments. Interesting to note is the high number of bits needed for the software-based alternative, higher than that of IEEE Std. 1687.

Table II shows the number of bits transported to and from the IC for the localization process. The hardware-based solution needs 16 bits to initiate the process and 1 bit to report whether there were any faults. As discussed above, UART needs at least one byte, which means that the overhead becomes 24 bits in total. In the same way as for the test process, the number of bits increases with the number of instruments for the IEEE Std. 1687 and the software-based alternatives. Note that the number of bits for the software-based localization is significantly lower than that for the IEEE Std. 1687 alternative.

The results on data overhead for test and localization show that when the IEEE Std. 1687 solution is used, it makes sense to first do a test to check whether there are faults, and if faults are present, a localization action takes place. However, in the case of a software-based solution, the difference between test and localization is quite low, which means that a localization function can be used without using a test procedure before. For the hardware-based solution,

TABLE I

NUMBER OF BITS TRANSPORTED TO PERFORM TEST

Instruments	IEEE Std. 1687	Hardware-based	Software-based
50	1900	24	2432
100	3800	24	4832
150	5700	24	7232

TABLE II

NUMBER OF BITS TRANSPORTED TO PERFORM LOCALIZATION

Instruments	IEEE Std. 1687	Hardware-based	Software-based
50	16600	24	4000
100	63200	24	8000
150	139800	24	12000

TABLE III

AREA FOR HARDWARE-BASED SOLUTION, IEEE STD. 1687 NETWORKS, AND RATIO BETWEEN THE TWO

Instruments	IEEE Std. 1687	Hardware-based	Ratio (%)
50	145	83	57
100	290	130	44
150	433	161	37

we have two separate functions: test and localization. While they only require 24 bits each, it would be possible to implement them as a single command performing test localization repair. A single command of 16-bits would initiate the process. The output could be a single bit to report whether the operation was performed correctly or not. Additional output could include a number and position of faults.

Table III shows the area for the hardware solution and the IEEE Std. 1687 network at 50, 100, and 150 instruments. The area is given as configurable logic blocks (CLBs), which constitutes the basic FPGA cell. The ratio (%) is the area of the hardware solution over the area of the IEEE Std. 1687 network times 100. Interesting to note is that the ratio decreases as the number of instruments in the IEEE Std. 1687 network increases, which indicates that the relative impact of the hardware solution decreases as the number of instruments increases.

VIII. CONCLUSION

We have shown that by including key information in an on-chip hardware component, it is possible to get graceful degradation of IEEE Std. 1687 networks. The main advantage is with respect to maintaining description languages, PDL and ICL, through the lifetime of ICs. As soon as an IEEE Std. 1687 network becomes faulty, PDL and ICL no longer match the IEEE Std. 1687 hardware. Instead of keeping copies of PDL and ICL for each individual IC, which is impractical due to large volumes, we showed that a small hardware block can perform the automatic test, localization, and repair, such that the original PDL assuming a fault-free IEEE Std. 1687 network is applied and the proposed block automatically, on-chip, adjusts the PDL to the fault situation of each particular IC. We demonstrated that such a component gives a significant reduction in the amount of data (information) that needs to be sent to and from an IEEE Std. 1687 network via a functional port as proposed by IEEE Std. P1687.1. This is highly important as it shows that access with IEEE Std. P1687.1 can be performed without significant impact on the normal (functional) operation, which is crucial, for example, during the periodic test in the automotive industry.

Future work may include handling of general IEEE Std. 1687 networks, advance combinations of instruments using IEEE Std. 1687 in combination with IEEE Std. 1500 and IEEE Std. 1838, and addressing other faults than scan-chain faults.

ACKNOWLEDGMENT

This submission is an extension of our two-page European Test Symposium 2020 paper where we presented the basic idea but did not include implementation and experiments.

REFERENCES

[1] "Embedded instrumentation: Its importance and adoption in the test & measurement marketplace," Frost & Sullivan, White Paper, 2010, p. 20.

[2] K. Posse, "Component manufacturer perspective," in *Proc. Int. Test Conf.*, 2015, pp. 1–10.

[3] *IEEE Standard for Access and Control of Instrumentation Embedded Within a Semiconductor Device*, IEEE Standard 1687-2014, 2014.

[4] *Standard for the Application of Interfaces and Controllers to Access 1687 IJTAG Networks Embedded Within Semiconductor Devices*, Standard IEEE P1687.1, Dec. 2016.

[5] E. Larsson, P. Murali, and G. Kumisbek, "IEEE Std. P1687.1: Translator and protocol," in *Proc. IEEE Int. Test Conf. (ITC)*, Nov. 2019, pp. 1–10.

[6] F. G. Zadegan, U. Ingelsson, G. Carlsson, and E. Larsson, "Access time analysis for IEEE P1687," *IEEE Trans. Comput.*, vol. 61, no. 10, pp. 1459–1472, Oct. 2012.

[7] F. G. Zadegan, U. Ingelsson, G. Carlsson, and E. Larsson, "Design automation for IEEE P1687," in *Proc. Design, Autom. Test Eur.*, Mar. 2011, pp. 1–6.

[8] F. G. Zadegan, D. Nikolov, and E. Larsson, "A self-reconfiguring IEEE 1687 network for fault monitoring," in *Proc. 21th IEEE Eur. Test Symp. (ETS)*, May 2016, pp. 1–6.

[9] A. Jutman, S. Devadze, and J. Aleksejev, "Invited paper: System-wide fault management based on IEEE P1687 IJTAG," in *Proc. 6th Int. Workshop Reconfigurable Commun.-Centric Syst.-on-Chip*, Jun. 2011, pp. 1–4.

[10] S. Kundu, "Diagnosing scan chain faults," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 512–516, Dec. 1994.

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

[12] D. Adolfsson, J. Siew, E. J. Marinissen, and E. Larsson, "On scan chain diagnosis for intermittent faults," in *Proc. Asian Test Symp.*, 2009, pp. 47–54.

[13] R. Cantoro, F. G. Zadegan, M. Palena, P. Pasini, E. Larsson, and M. S. Reorda, "Test of reconfigurable modules in scan networks," *IEEE Trans. Comput.*, vol. 67, no. 12, pp. 1806–1817, Dec. 2018.