Concurrent Test of Reconfigurable Scan Networks for Self-Aware Systems

Wang, Chih-Hao; Lylina, Natalia; Atteya, Ahmed; Hsieh, Tong-Yu; Wunderlich, Hans-Joachim

Proceedings of the IEEE International Symposium on On-Line Testing And Robust System Design (IOLTS'21), Virtual, 28 - 30 June 2021, pp. 1-7

doi: https://doi.org/10.1109/IOLTS52814.2021.9486710

Abstract: Self-aware and safety-critical hardware/software systems rely on a variety of embedded instruments, sensors, monitors and design-for-test circuitry to check the system integrity. The access to these internal instruments is supported by standards commonly called iJTAG and employs so called reconfigurable scan networks (RSNs), which are more and more used at runtime, too. They collect periodically and also concurrently the information on the circuit's health state and deliver it to some dependability management unit. The integrity of RSNs is essential for the dependability of self-aware systems and can be ensured by a combination of periodic and concurrent test methods of the RSN itself. The paper at hand presents the first concurrent online test method for RSNs by adding a brief integrity test to each access operation. The presented scheme includes a hardware extension of negligible size, supports offline test, diagnosis and post-silicon validation as well, and is further referred as ROSTI: RSN Online/Offline Self-Test Infrastructure. It exploits the original RSN control signals and does not require any modification of the underlying RSN. The hardware costs are independent of the size of the RSN, and ROSTI is flexible for generating different test sequences for different types of faults. The experimental results validate these characteristics and show that ROSTI is highly scalable.

Preprint

Concurrent Test of Reconfigurable Scan Networks for Self-Aware Systems

Chih-Hao Wang[1], Natalia Lylina[2], Ahmed Atteya[2], Tong-Yu Hsieh[1], Hans-Joachim Wunderlich[2]

[1]Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan
chwang.peter@gmail.com, tyhsieh@mail.ee.nsysu.edu.tw
[2]Institute of Computer Architecture and Computer Engineering (ITI), University of Stuttgart, Stuttgart, Germany
{lylina, atteyaad, wu}@informatik.uni-stuttgart.de

***Abstract*—Self-aware and safety-critical hardware/software systems rely on a variety of embedded instruments, sensors, monitors and design-for-test circuitry to check the system integrity. The access to these internal instruments is supported by standards commonly called iJTAG and employs so called reconfigurable scan networks (RSNs), which are more and more used at runtime, too. They collect periodically and also concurrently the information on the circuit's health state and deliver it to some dependability management unit.**

The integrity of RSNs is essential for the dependability of self-aware systems and can be ensured by a combination of periodic and concurrent test methods of the RSN itself. The paper at hand presents the first concurrent online test method for RSNs by adding a brief integrity test to each access operation. The presented scheme includes a hardware extension of negligible size, supports offline test, diagnosis and post-silicon validation as well, and is further referred as ROSTI : RSN Online/Offline Self-Test Infrastructure. It exploits the original RSN control signals and does not require any modification of the underlying RSN. The hardware costs are independent of the size of the RSN, and ROSTI is flexible for generating different test sequences for different types of faults. The experimental results validate these characteristics and show that ROSTI is highly scalable.

***Keywords*– Self-Aware Systems, Reconfigurable Scan Networks, Systems-on-a-Chip, Test, Online Test, Concurrent Test**

I. Introduction

Self-aware and safety-critical hardware/software systems rely on a variety of embedded instruments, sensors, monitors and design-for-test circuitry to check the system integrity and eventually to initiate fault-tolerance actions [1]. The access to these internal instruments is supported by the standards commonly called iJTAG (internal Joint Test Access Group) and employs so called reconfigurable scan networks (RSNs).

Each instrument is connected to a scan segment, and control segments determine via multiplexers and Segment Insertion Bits (SIBs) an activated path through the network which can transport information from one instrument to another one, or which can collect information for central evaluation and dependability management. Standards for such RSNs are found under IEEE Std. 1687-2014 (iJTAG) [2] and IEEE Std. 1149.1-2013 (JTAG) [3].

Originally, iJTAG techniques were developed for test, debug and diagnosis in post-silicon validation and production test, but they can be also used at runtime to collect information on the circuit's health state and can pass it to a dependability management unit at system level [4], or to support online test and diagnosis at runtime [5]. In any case, the dependability and fault-tolerance of the entire system depend on the integrity of the access mechanism, and first papers on fault-tolerant RSNs are already found in literature [6]. An essential part of most fault-tolerance schemes is online fault detection as a prerequisite for fault recovery, fault repair or reconfiguration.

The paper at hand presents the first scheme of a concurrent online test of RSNs for fault detection. It includes a negligible amount of additional hardware, and can also be used for an offline test of the RSN in post-silicon validation and production test. This scheme comprises an RSN Online/Offline Self-Test Infrastructure (ROSTI), which augments each workload shifted through the RSN by a brief pre-sequence which tests the integrity of the activated path through the RSN.

This new online test scheme for RSNs has mainly four goals:

1) **Support of system level recovery**: A violation signal has to be generated which may trigger an interruption and forbid an upload operation of the RSN. Fault recovery techniques like re-execution or reconfiguration can be initiated.
2) **Flexibility with respect to different fault types**: The above mentioned pre-sequence should be programmable and adopted to technology dependent defect models.
3) **Independence of the RSN structure**: The hardware implementation of ROSTI should only depend on the size of the flush test sequence, and be independent of the RSN size. The control of ROSTI has only to depend on global signals, and an internal generation of control signals is not required.
4) **Cost-effectiveness and easiness to implement**: ROSTI has to be compliant with the upcoming standard P1687.1 [7] and can be integrated into a TAP controller with negligible hardware costs.

To validate the approach, a set of commonly accessible benchmarks of RSNs is used. According to the simulation experiments, the online use of ROSTI reaches a comparable fault coverage with external testing. Less than 0.69% of area overhead with respect to the original RSN size is incurred for all the benchmark circuits.

The rest of the paper is organized as follows. Section II describes the required background of the RSNs. Section III explains the runtime use and integrity tests of RSNs. Section IV presents the basic strategy and algorithm of ROSTI, and Section V explains the underlying hardware implementation. Experimental results are discussed in Section VI. Finally, Section VII concludes the paper.

II. Terminology of RSNs

Reconfigurable Scan Networks (RSNs) provide a flexible and efficient mechanism to access on-chip instruments, such as aging monitors, sensors, and Built-In Self-Test registers. Typically, an RSN contains *scan segments*, *wrapped instruments*, and *scan multiplexers*, as shown in Fig. 1.

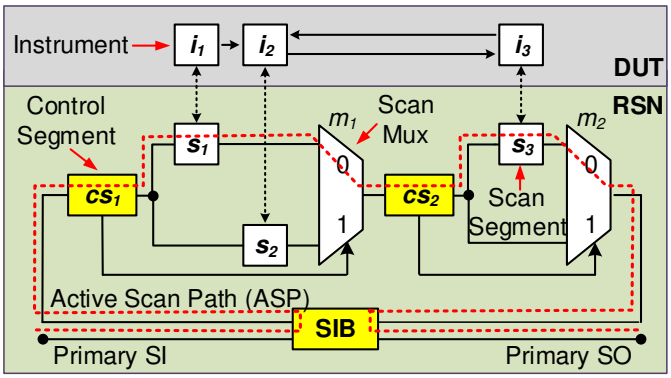

Fig. 1. RSN terminology

Scan segments are acting as a normal scan chain to transfer data from a scan-in (SI) towards a scan-out (SO) or with an instrument via a data-in (DI) and a data-out (DO). A scan segment can be connected to an instrument to form a *wrapped instrument*. In order to reconfigure the path of data transferring, *scan multiplexers* are added to switch between different scan segments. An RSN may also contain optional control primitives called *Segment Insertion Bits* (*SIBs*), which include and exclude the parts of an RSN from an activated path. A subnetwork is selected, only if the SIB is *asserted*. Otherwise, the subnetwork will be bypassed.

A non-circular path from a primary SI to a primary SO through a sequence of selected scan primitives is called an *Active Scan Path* (*ASP*). The ASP in the provided example is shown as a red dashed line in Fig. 1, and is going from SI to the SIB to SO. Since the SIB is open, the control segments cs_1, cs_2, as well as the scan segments s_1, s_3, are included into the ASP. The control signals are used to drive the select-ports of the scan primitives and thereby to build an ASP. These control signals can be generated internally from the shadow registers of the control segments, or externally from the outside of the RSN. A *Scan Configuration* c is defined by the state of the configuration registers in the RSN and by the assignments to the external control signals. In a valid scan configuration, only a single ASP is configured.

An access to an RSN includes three phases, namely the *capture*, *shift*, and *update*. During a capture phase, the data from the instruments is transferred to the scan segments. Then, during a shift phase, the data is being shifted through the ASP, while the new data is being shifted in. Finally, during an update phase, the newly shifted-in data is latched in the shadow registers of the scan segments. This data can be further used to drive the internal control signals and thereby to select a particular ASP. Alternatively, the data in the shadow registers can be written into the instruments, e.g. to control their operation.

III. Runtime Use and Integrity Tests of RSNs

A. Runtime Use of RSNs

A complete fault management solution for self-aware systems is proposed in [4], which uses RSNs to perform online fault detection and to adapt the system operation, such that only the healthy resources of the system are used.

In [8] an on-chip controller is presented to efficiently operate the instruments, which are used to enhance the reliability and the functional safety of the system during the whole life-cycle. Cross-layer evaluation procedures are presented to process the instruments' measurement data and to control the system operation.

Inline with the above mentioned solutions, an overall architecture, which utilizes an RSN online, is shown in Fig 2. A Test Access Port (TAP) Controller accesses an RSN through a primary SI, and the data from an RSN comes to a primary SO port. The external Capture-, Shift- and Update signals, driven by a TAP controller, are used to control the operation of the RSN. The standard extension proposal P1687.1 [7] allows to access RSNs through alternate non-JTAG interfaces such as e.g. I^2C (Inter-Integrated Circuit), as long as the iJTAG-compliant control signals are generated by the access mechanism to drive the RSN. The RSN captures the evaluation data from the dependability instruments of the hardware/-software system, which is referred as a Device-under-Test (DUT in Fig. 2), through the scan segments and transfers this information to the dependability management unit through a P1687.1 compliant access interface. This data is processed in a dependability management unit concurrently to the functional operation of the DUT. Depending on the health status of the DUT and its instruments, the dependability management unit decides, which of the system resources are healthy, and controls the operation of the dependability instruments through the RSN online.

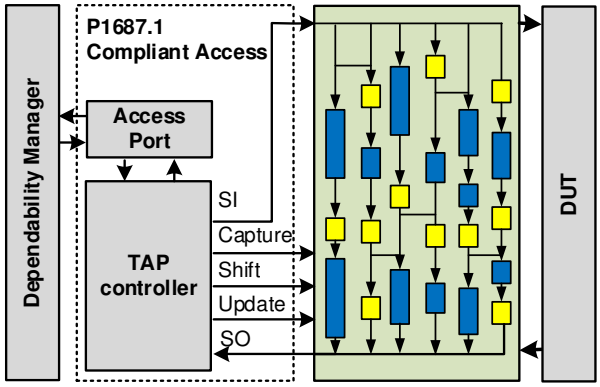

Fig. 2. Runtime use of RSNs

However, even a single fault in an RSN itself can lead to inaccessibility of some instruments through the RSN. An online fault detection mechanism for RSNs is essential to ensure the dependable RSN operation and thereby to support the system-level dependability. To the best of our knowledge, concurrent online test of the RSN has not been addressed yet.

B. Integrity Test of RSNs

The dependability and test infrastructure may occupy a significant amount of the chip area [9, 10], and has to be tested as well.

Faults in conventional scan chains can be detected by using special sequences, which are shifted through a tested scan chain. These sequences are usually referred as *flush test sequences* [11] and can be applied to ensure the integrity of the scan cells and their interconnection. Already conventional scan-chain testing for realistic fault models is challenging [12–14]. The test of RSNs is even more complicated due to their high sequential depth, distributed control structure, as well as the complex sequential and combinational dependencies [15]. Moreover, specific fault effects may become observable only for certain RSN configurations, and sequential test sequence generation (TSG) for such faults may be unfeasible for existing TPG tools. As a highly sequential structure, an RSN is also prone to soft-errors and transient faults which require concurrent test methods.

Faults in scan segments of an RSN are mostly modeled as stuck-at faults. In [16], a particular scan segment is tested by configuring an ASP to select it and applying a flush sequence. For "stuck-at-faults", flush sequences such as "01100" or "10011" are used for the integrity test of scan cells by generating all possible transitions, including "00", "01", "10", and "11". The flush sequences are modifiable for more complex fault models, such as delay faults. In [17, 18], a test method for gate-level stuck-at-faults is presented and the quality of different test strategies is investigated. Additionally, a design-for-test approach is presented to cover also the faults in the capture- and update-circuitry in scan segments with shadow registers. Due to the increased observability and the controllability of shadow flip-flops, more realistic faults models, which include flip-flop transparency faults for both shift and shadow flip-flops, and also bridge faults, can be handled.

It should be noted that a fault in an RSN may arise not only inside the scan segments but also inside the scan multiplexers, control logic, or interconnects. Since the scan segments occupies the largest area of the RSN, the integrity test is mandatory. Offline test methods to the other parts of the RSN can be found in the literature.

IV. ROSTI

ROSTI generates integrity test sequences and attaches them to the workload sequences, which include the configuration sequences and the instrument access sequences. The integrity of the scan path is tested as a part of the access operation.

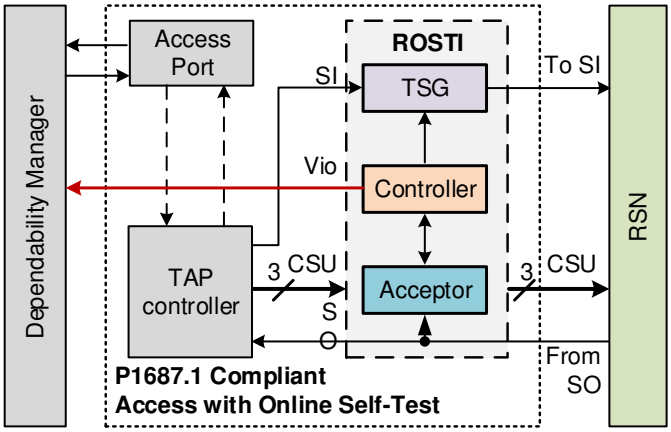

Fig. 3. The ROSTI scheme for online RSN test

A. General Idea of ROSTI

ROSTI exploits the original RSN control signals, including *capture*, *shift*, and *update*, to conduct an integrity test:

- After the capture signal and with the shift signal, a brief pre-sequence is put in front of the workload sequence. This pre-sequence implements a flush test and is programmable and flexible.
- The pre-sequence and the workload sequence are shifted through the ASP.
- If the ASP is not corrupted, the bits of the pre-sequences have been shifted-out unchanged, and the workload sequence is at the target instrument.

ROSTI includes a test sequence generator, an acceptor and a controller, and it is placed between the RSN and the TAP controller as illustrated in Fig. 3. Together with a TAP controller and an access port, ROSTI represents an access interface, which enables online RSN self-test and is compliant with the P1687.1 standard proposal.

The inputs to ROSTI are the standard iJTAG interface signals, which include the external Capture-, Shift- and Update signals (referred as CSU in Fig. 3). The outputs are the Vio violation signal, which indicates the defectiveness of the RSN, and a To_SI signal, which is used to forward data to the RSN. All the signals, which are required by ROSTI, are taken directly from the TAP controller and additional control signals are not needed. The controller triggers the select signals of the scan multiplexers for generating the proper ASP test bit.

The flush test sequences are generated and evaluated by ROSTI to test each ASP. These sequences can be modified and extended according to the needs of the underlying technology for fault detection. They test not only the correct scan operation but also multiplexers and selected branches to a wide extent.

In general, a flush test sequence is symmetric with respect to inversion, i.e., if $T =< t_{n-1}, \dots t_0 >$ is a flush test for the ASP, $\overline{T} =< \overline{t_{n-1}}, \dots, \overline{t_0} >$ is one as well. If $W =< w_{m-1}, \dots, w_0 >$ is the workload bit sequence, the tails of either T or $\overline{T}$ overlap with the head of W by at least one bit, and there is no need to repeat these bits in T or $\overline{T}$ (see Fig. 4).

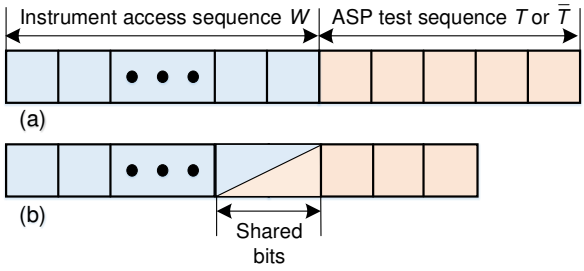

Fig. 4. Test-bit sharing mechanism

Since an overlap of a flexible number of bits between workload and test sequence would require complex control, we consider just a constant overlap of a single bit for the rest of the paper. E.g., a widely used flush test is "01100" ("10011"), ROSTI generates four remaining bits "1100" ("0011") and is thereby reducing the test time overhead by 20%.

Whenever the self-aware system mechanism accesses instruments via the RSN, the ASP test sequence is automatically generated and added to the instrument access sequence or the configuration sequence. The RSN violation signal prevents the fault manager from performing an invalid online test when the RSN contains defects or soft errors. The violation signal from ROSTI may also be a system primary output to trigger a warning signal to the user showing that the system is unreliable.

V. Hardware Implementation of ROSTI

ROSTI generates a test sequence as described above and shifts it into the RSN. Fig. 5 illustrates the architecture of ROSTI. It only affects the scan-in (SI) of the RSN, and thereby does not require any modifications of the DUT and the RSN.

In the following, the hardware implementation of ROSTI will be explained in a block-by-block manner, including three major parts: 1) Test Sequence Generator (TSG), 2) Acceptor, and 3) Controller. The major definitions are summarized in Table I.

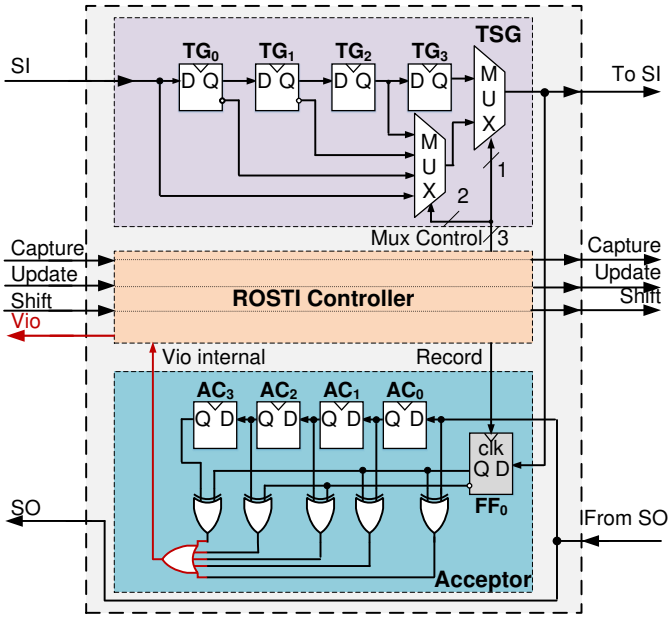

Fig. 5. Detailed architecture of ROSTI

TABLE I
Definition of terms

Terms	Descriptions
n	The length of the current ASP.
m	The length of the test sequence. $m = 5$ in this case.
$p_i \in P, 0 \leq i < n$	The sequence sent (capture) to (from) the instrument, called instrument access sequence.
$t_j \in T, 0 \leq j < m$	The m-bit flush test sequence for testing the ASP (ASP test sequence).
$TG_k, AC_k, 0 \leq k < m-1$	The flip-flops in the TSG and in the acceptor, one bit is shared with an instrument test sequence.
FF_0	The flip-flop to store the first bit of instrument access sequences (p_0)

A. Test Sequence Generator (TSG)

The ASP test sequence is added as a pre-sequence to the instrument access sequence and is shifted-in together without adding any hold cycle.

The operating flow of TSG is shown in Algorithm 1. Further, we explain this flow step-by-step:

Algorithm 1: Operating flow of the TSG

Data: The length of the current ASP n.
Instrument access sequence:
$\{p_0, p_1, p_2, p_3, p_4 \cdots p_{n-1}\}$
ASP sequence:
$\{t_0, t_1, t_2, t_3, t_4\} = \{p_0, \overline{p_0}, \overline{p_0}, p_0, p_0\}$
Result: Test sequence generator bits:
$\{TG_0, TG_1, TG_2, TG_3\}$

1 **Shift operations:** {
2 1^{st} *clock:*
3 $p_0 \rightarrow TG_0$; $t_0 \rightarrow RSN$;
4 2^{nd} *clock:*
5 $p_1 \rightarrow TG_0$; $p_0 \rightarrow TG_1$; $t_1 \rightarrow RSN$;
6 3^{rd} *clock:*
7 $p_2 \rightarrow TG_0$; $p_1 \rightarrow TG_1$; $p_0 \rightarrow TG_2$; $t_2 \rightarrow RSN$;
8 4^{th} *clock:*
9 $p_3 \rightarrow TG_0$; $p_2 \rightarrow TG_1$; $p_1 \rightarrow TG_2$; $p_0 \rightarrow TG_3$;
$t_3 \rightarrow RSN$;
10 5^{th} *clock:*
11 $p_4 \rightarrow TG_0$; $p_3 \rightarrow TG_1$; $p_2 \rightarrow TG_2$; $p_1 \rightarrow TG_3$;
$t_4 \rightarrow RSN$;
12 }

Step 1 (Line 2-3): As soon as the first bit of the instrument access sequence (p_0) is ready at the global scan-in port (SI), it is directly used as the first bit of the ASP test sequence t_0 and shifted into the RSN.

Step 2 (Line 4-9): In order to generate the rest of the ASP test sequence (i.e. $\overline{p_0}\overline{p_0}p_0p_0$), a four-bit shift register (TG_{0-3}) is used as illustrated in Fig. 5. When the first shift operation is done, the first bit of the instrument access sequence is at TG_1. The inversed logic value ($\overline{Q}$) of TG_0 is selected via the multiplexers to shift $t_1(=\overline{p_0})$ into the RSN. This procedure iterates for $m-1$ clock cycles in total, where m is the length of the flush test sequence.

Step 3 (Line 10-11): At the m^{th} clock cycle (in this case, the 5^{th}), the first bit of the instrument access sequence is shifted into the RSN and acts as the m^{th} bit of ASP test sequence $t_4(=p_0)$. sequence

B. Acceptor

As described above, the ASP test sequence is generated according to the assignment of p_0. This value is recorded in a flip-flop FF_0 of the acceptor (marked in gray in Fig. 5) for the further test response comparison. A finite state machine (FSM) works as an acceptor of the test sequence. If the last five bits shifted out of the RSN are identical with the pre-sequence $p_0\overline{p_0}\overline{p_0}p_0p_0$, the violation signal is not triggered and it outputs $Vio := 0$, which means that the ASP test passed. If the compared bits are different, the violation signal ($Vio := 1$) is triggered.

A simple implementation of the acceptor can be represented as a combination of a shift register and a comparator. Algorithm 2 shows the operating flow of the acceptor:

Algorithm 2: Operating flow of the acceptor

Data: The length of the current ASP n.
Instrument access sequence:
$\{p_0, p_1, p_2, p_3, p_4 \dots p_{n-1}\}$
ASP sequence:
$\{t_0, t_1, t_2, t_3, t_4\} = \{p_0, \overline{p_0}, \overline{p_0}, p_0, p_0\}$
Result: Test sequence generator bits:
$\{TG_0, TG_1, TG_2, TG_3\}$
Acceptor bits:
$\{AC_0, AC_1, AC_2, AC_3\}$
Violation Vio

1 **Shift operations:** {
2 n^{th} *clock:*
3 $t_0 \rightarrow SO$;
4 $(n+1)^{th}$ *clock:*
5 $t_1 \rightarrow SO$; $t_0 \rightarrow AC_0$;
6 $(n+2)^{th}$ *clock:*
7 $t_2 \rightarrow SO$; $t_1 \rightarrow AC_0$; $t_0 \rightarrow AC_1$;
8 $(n+3)^{th}$ *clock:*
9 $t_3 \rightarrow SO$; $t_2 \rightarrow AC_0$; $t_1 \rightarrow AC_1$; $t_0 \rightarrow AC_2$;
10 $(n+4)^{th}$ *clock:*
11 $t_4 \rightarrow SO$; $t_3 \rightarrow AC_0$; $t_2 \rightarrow AC_1$; $t_1 \rightarrow AC_2$; $t_0 \rightarrow AC_3$;
12 }
13 **Update operation:** {
14 $Vio \rightarrow$ Primary output or TAP controller
15 }

Step 1 (Line 2-3): If the current ASP has the length n, then, at the n[th] cycle of the shift operation, the first bit of the ASP test sequence is shifted-out and this bit is simultaneously written into the acceptor.

Step 2 (Line 4-11): During the test response comparison, the acceptor is working aside the RSN and the shift operation is performed normally, and thereby does not require any additional control.

Step 3 (Line 12-13): As soon as the shift operation is done, the violation signal Vio is fetched at the update operation of the RSN. This signal serves as a primary output of ROSTI, and can be forwarded to the TAP controller, in order to interrupt the update, or to the dependability management unit directly.

The presented acceptor is independent of the length of the ASP and its hardware costs depend only on the length of the test sequence since the acceptor is controlled by the available global shift and update signals.

C. ROSTI Controller

The most important task of the controller is to generate the violation signal (Vio) with a correct timing. When a violation occurs, i.e. if an RSN test fails, ROSTI raises the violation signal. This signal is triggered by the rising transition of the clock signal after the removal of the shift signal (*shift*) and it holds for one cycle as shown in Fig. 6. The controller is also responsible for writing the first-bit of the instrument access sequence into the flip-flop FF_0 of the acceptor.

VI. Experimental Results

The presented approach is evaluated with the help of the ITC'02 SoC (System-on-a-Chip) benchmark set [19]. The RSNs are constructed for these benchmarks by using the method from [20] and [15], and have a hierarchical bypass (SIB-based) structure. Table II summarizes the characteristics of the considered RSNs. The RSNs contain up to 197,208 flip-flops (Column 2), which can form 327 to 9,174 possible ASPs (Column 3).

All the generated RSNs and ROSTI structures are synthesized for the TSMC 90nm technology library by commercial tools.

TABLE II
Benchmark Characteristics

Circuits (1)	# FFs (2)	# ASPs (3)
u226	2,930	615
d281	7,742	774
d695	16,792	2,385
h953	11,280	702
g1023	10,770	1,005
f2126	31,658	540
q12710	52,366	327
p22810	60,220	3,972
p34392	46,482	1,815
p93791	197,208	9,174
t512505	154,010	2,001
a586710	83,348	534

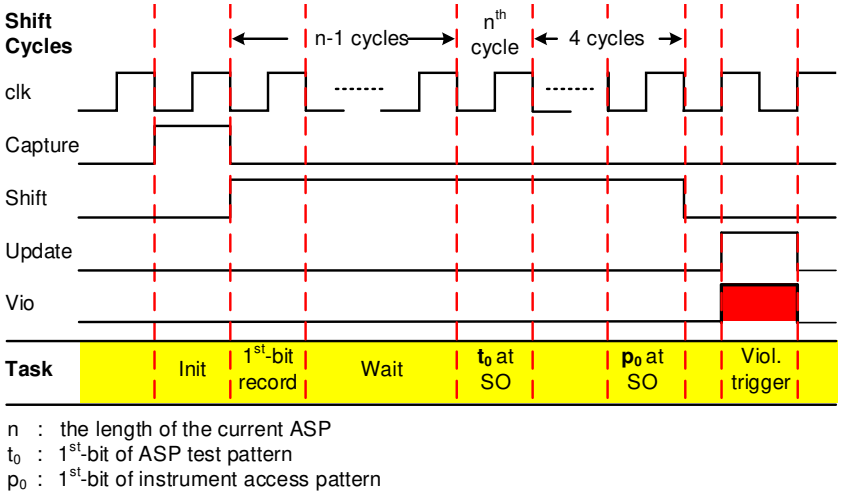

Fig. 6. Waveform of the ROSTI scheme

The design-for-testability scheme, which has been presented in [18], is integrated into the considered RSN designs. Feedback routes are injected to enhance the observability of the shadow registers in the RSNs and thereby to achieve higher fault coverage. The test generation process is implemented in C++ as described in [18] with the RSNs described in Instrument Connectivity Language (ICL) [2]. Stuck-at faults are considered to evaluate the fault coverage. The commercial fault simulation is very compute intensive. So for the largest circuits, fault sampling [21] is used for determining the fault coverage with a confidence interval of 99% .

A. Hardware Cost

For the ROSTI architecture (Fig. 5), where a flush test sequence consists of five bits, four flip-flops are required for the TSG, and another four bits are required for the acceptor. For the controller of ROSTI, only eight flip-flops are used. The architecture of ROSTI is independent of the RSN and the number of the required flip-flops is also fixed for any RSN under test. ROSTI is placed between the TAP controller and the RSN, but the delay overhead of ROSTI is negligible and is less than 0.07 ns, since ROSTI contains only a few gates.

The area overhead of ROSTI is reported in the second column of Table III. According to the results, ROSTI always needs less than 1% of the area needed by the RSN itself. The average cost of the investigated examples is around 0.15%.

B. Fault Coverage

The access traffic on an RSN equipped with ROSTI was analyzed with a commercial fault simulator, and the fault coverage obtained in online mode is reported in column (3) of Table III. Fault coverage reached in external offline testing are listed in column (4).

According to the results, the ROSTI scheme provides a similar fault coverage as a conventional external test scheme. The fault coverage is above 92.60% for nine benchmark circuits without fault sampling, and is 94.72% on average. It validates that the approach is able to generate and compare the proper sequences for the RSN test. The missing percentage of the fault coverage requires testing the interfaces to instruments and logic as explained above.

TABLE III
AREA OVERHEAD, FAULT COVERAGE, AND TEST TIME

Circuits (1)	AO* (2)	Fault coverage**		# of clock cycles for a complete test	
		Online (3)	Offline (4)	External w/o ROSTI (5)	External w/ ROSTI (6)
u226	0.69%	93.48%	93.65%	23,262	22,647
d281	0.27%	95.44%	95.51%	34,887	34,113
d695	0.19%	92.60%	92.17%	118,619	116,234
h953	0.18%	96.28%	96.32%	38,949	38,247
g1023	0.20%	95.59%	95.66%	51,423	50,418
f2126	0.06%	95.10%	95.11%	75,809	75,269
q12710	0.03%	95.43%	95.43%	110,231	109,904
p22810	0.08%	93.89%	93.96%	298,755	294,783
p34392	0.06%	94.67%	94.70%	156,403	154,588
p93791	0.04%	84.57±0.94%	85.98±0.90%	888,873	879,699
t512505	0.02%	85.77±0.87%	86.37±0.85%	404,930	402,929
a586710	0.02%	85.99±0.90%	86.90±0.88%	178,655	178,121

*AO: Area overhead, compared to the underlying RSN
**Fault sampling is used to p93791, t512505, and a586710 with 99% of confidence interval

ROSTI as any Design for Testability (DfT) circuitry has to be tested itself as well before using it. The test generation process is done by a commercial tool with full-sequential ATPG setting, and achieves a fault coverage of 96.84% with 16 patterns and 278 test cycles.

C. Support of off-line test

The presented ROSTI scheme can also be reused for an offline test, as shown below. The remainder of the subsection compares the following test schemes:

- **External Offline Test**: An external tester is used to shift-in the test sequences into the RSN. A complete test sequence includes an instrument test sequence W and a flush test sequence T, as shown in Fig. 4.(a). The instrument access sequence can be generated by an external tester and is used e.g. for testing the interfaces to instruments, whereas the flush test sequence is responsible for performing the integrity test of the selected ASPs. Each complete test sequence requires $|W| + |T|$ shift cycles.
- **External Offline Test + ROSTI**: The same instrument test sequences are shifted-in into the RSN externally, but the flush test sequences are generated by ROSTI. Thanks to the presented bit sharing mechanism, a complete test sequence would require only $|W| + |T| - l$ cycles, where l denotes the number of shared bits.

An external test tries to cover all primitives and instrument interfaces of the RSN by activating a minimized number of activated scan paths. Column 3 of Table II shows the number of ASPs to be activated by the test generation method according to [15]. Column 5 of Table III denotes the number of clock cycles including capture, shift and update cycles and the cycles of the flush sequences for the used ASPs. Column 6 of Table III shows that the test time is reduced, if the flush test sequence generation is supported by ROSTI. The results validate that ROSTI is able to reduce the test time and the test volume when used to support an external external test.

VII. CONCLUSION

This paper presents the scheme ROSTI to generate online testable reconfigurable scan networks to be used for monitoring, sensing and self-testing of self-aware systems. The method greatly reduces the test cost by self-generating and self-comparing test sequences and enables an online test strategy. The control of ROSTI does not cause significant overhead, since it exploits the existing RSN control signals.

ROSTI has excellent scalability with nearly fixed, negligible hardware cost and supports also an external test in production. Furthermore, ROSTI is also modifiable and programmable for more complex test sequences.

ACKNOWLEDGMENTS

This work was supported in part by the German Research Foundation (DFG) under Contract Number WU245/17-2, by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart, and by the Ministry of Science and Technology of Taiwan under Contract Number MOST 108-2911-1-110-503, MOST 108-2628-E-110-004-MY3 and MOST 107-2221-E-110-006-MY2.

BIBLIOGRAPHY

[1] A. Jantsch, N. Dutt, and A. M. Rahmani, "Self-Awareness in Systems on Chip—A Survey," *IEEE Design & Test*, vol. 34, no. 6, pp. 8–26, 2017.

[2] "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," *IEEE Std 1687-2014*, pp. 1–283, 2014.

[3] "IEEE Standard for Test Access Port and Boundary-Scan Architecture," *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)*, pp. 1–444, 2013.

[4] K. Shibin, S. Devadze, A. Jutman, M. Grabmann, and R. Pricken, "Health Management for Self-Aware SoCs Based on IEEE 1687 Infrastructure," *IEEE Design & Test*, vol. 34, no. 6, pp. 27–35, 2017.

[5] M. A. Kochte and H.-J. Wunderlich, "Self-Test and Diagnosis for Self-Aware Systems," *IEEE Design & Test*, vol. 35, no. 5, pp. 7–18, 2018.

[6] S. Brandhofer, M. A. Kochte, and H. Wunderlich, "Synthesis of Fault-Tolerant Reconfigurable Scan Networks," in *Proc. Design, Automation Test in Europe Conf. Exhibition (DATE)*, 2020, pp. 798–803.

[7] A. L. Crouch, B. G. Van Treuren, and J. Rearick, "P1687.1: Accessing Embedded 1687 Instruments using Alternate Device Interfaces other than JTAG," in *Proc. IEEE European Test Symp. (ETS)*, May 2020, pp. 1–6.

[8] A. M. Y. Ibrahim and H. G. Kerkhoff, "An On-chip IEEE 1687 Network Controller for Reliability and Functional Safety Management of System-on-Chips," in *Proc. Int'l. Test Conf. in Asia (ITC-Asia)*, 2019, pp. 109–114.

[9] R. Guo and S. Venkataraman, "A Technique for Fault Diagnosis of Defects in Scan Chains," in *Proc. Int'l. Test Conf. (ITC)*, 2001, pp. 268–277.

[10] 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 *Proc. VLSI Test Symp. (VTS)*, 2008, pp. 79–84.

[11] K.-J. Lee and M. A. Breuer, "A Universal Test Sequence for CMOS Scan Registers," in *Proc. of the Custom Integrated Circuits Conference*, 1990, pp. 28.5/1–28.5/4.

[12] S. R. Maka and E. J. McCluskey, "ATPG for Scan Chain Latches and Flip-Flops," in *Proc. VLSI Test Symp. (VTS)*, 1997, pp. 364–369.

[13] F. Yang, S. Chakravarty, N. Devta-Prasanna, S. M. Reddy, and I. Pomeranz, "Detection of Internal Stuck-open Faults in Scan Chains," in *Proc. Int'l. Test Conf. (ITC)*, 2008, pp. 1–10.

[14] W. Cheng, G. Mrugalski, J. Rajski, M. Trawka, and J. Tyszer, "On Cyclic Scan Integrity Tests for EDT-based Compression," in *Proc. VLSI Test Symp. (VTS)*, 2019, pp. 1–6.

[15] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Reconfigurable Scan Networks: Modeling, Verification, and Optimal Pattern Generation," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 2, p. 30, 2015.

[16] R. Cantoro, M. Montazeri, M. S. Reorda, F. G. Zadegan, and E. Larsson, "On the Testability of IEEE 1687 Networks," in *Proc. Asian Test Symp. (ATS)*, 2015, pp. 211–216.

[17] M. A. Kochte, R. Baranowski, M. Schaal, and H. Wunderlich, "Test Strategies for Reconfigurable Scan Networks," in *Proc. Asian Test Symp. (ATS)*, 2016, pp. 113–118.

[18] D. Ull, M. Kochte, and H. Wunderlich, "Structure-Oriented Test of Reconfigurable Scan Networks," in *Proc. Asian Test Symp. (ATS)*, 2017, pp. 127–132.

[19] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs," in *Proc. Int'l. Test Conf. (ITC)*, 2002, pp. 519–528.

[20] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Modeling, Verification and Pattern Generation for Reconfigurable Scan Networks," in *Proc. Int'l. Test Conf. (ITC)*, 2012, pp. 1–9.

[21] V. D. Agrawal and H. Kato, "Fault sampling revisited," *IEEE Design & Test of Computers*, vol. 7, no. 4, pp. 32–35, 1990.