A Functional Approach to Test and Debug of IEEE 1687 Reconfigurable Networks

Michele Portolan[1 2], Riccardo Cantoro[2], Ernesto Sanchez[2]

[1]Univ. Grenoble Alpes, CNRS, Grenoble INP*, TIMA, F-38000 Grenoble, France,
{first.last_name}@grenoble-inp.fr

[2] Politecnico di Torino, Italy
Dipartimento di Automatica ed Informatica
{first.last_name}@polito.it

***Abstract*— The IEEE 1687 standard introduces several novelties, most notably Reconfigurable Scan Networks (RSNs), i.e., scan chains whose length can change dynamically. These architectures offer important advantages but can result in extremely complex integrity test following traditional structural approaches. In this paper, we will present an innovative approach to RSN test and debug based on the functional features of the standard, which is able to greatly speed up test generation time while guaranteeing a precise fault coverage.**

I. INTRODUCTION

The complexity of modern integrated circuits such as System-on-Chips (SoCs) puts a huge stress on testing: while circuits grow larger, accessibility and controllability of internal components becomes critical, making advanced Design-for-Test (DfT) approaches a necessity. In this context, the IEEE 1687-2014 standard [1] offers significant advantages by building on well-established scan-based test approaches and offering two main innovations:

- Dynamic topologies,, described with the Instrument Connectivity Language (ICL).
- Functional instruments, whose operation can be described thanks to the Procedural Description Language (PDL).

The first point is particularly problematic: variable-length chains make Scan Chain Integrity an extremely computationally intensive operation, because there is the need to generate patterns for each possible configuration. The alternative would be to exploit the functional features of 1687, but these approaches traditionally lack precision when trying to evaluate coverage. In this paper, we will present an innovative approach to dynamically test the device reconfigurable scan network exploiting the functional features of IEEE 1687-2014 while providing clear coverage metrics.

II. STATE OF THE ART

In ICL, a ScanMux is simply declared by specifying its data path (Data Sources and TDO) and the signals controlling its selection, as in this code snippet taken from [1]:

```
ScanMux SIB_out SelectedBy SIBREG {
       1'b0 : base[0];
       1'b1 : aux[0];
       }
```

This code must be read as: *The signal* **`SIB_out`** *takes the value of* **`base[0]`** *when* **`SIBREG`** *is 0 and the value of* **`aux[0]`** *when* **`SIBREG`** *is 1*. This is a useful definition of the Data Path, but fails at conveying the effect of the ScanMux on the control path. To avoid corrupting the value stored into the unselected signals, the ScanMux must emit some "select" signals that will gate all or part of the JTAG control signals. This could be as simple as inhibiting Update to completely shut down register operations by clock gating. So, errors inside the ScanMux will not only affect the Data Path, but also the Control Path, resulting in Selection Errors. Structural approaches like [2]-[6] read the effect of these errors on the overall scan chain length, and devise patterns able to identify length mismatches.

III. PROPOSED APPROACH

State-of-the-art test approaches are based on direct observation of the scan chain: patterns are computed to reach and verify all possible RSN configurations. Even though efficient algorithms do exist, the sheer combinatorial complexity of the problem risks to quickly hit a "computational wall", especially for debug. This complexity is caused by the fact that the new hierarchy-enabling element like the ScanMuxes are not directly characterized: the only thing that is observed is the effect that a fault generate on the overall chain. In this section, we will first provide a full characterization of the fault behavior of ScanMuxes, and then we will exploit it to build a new testing method, depicted in Figure 1.

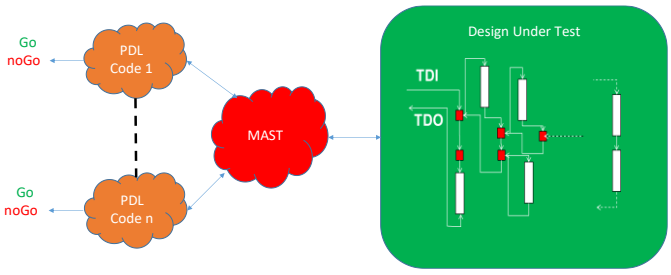

Figure 1 ScanMux testing setup

For each segment in the Design Under Test, we define a PDL-1 algorithm (i.e., PDL input/output operations wrapped in C/C++ code using the MAST tool **Erreur ! Source du renvoi introuvable.**) that writes and reads back random data over several cycles to verify the integrity of its segment. Their concurrent execution cause the opening and closing of the ScanMuxes in the path: faults will cause read/write mismatches, allowing detection.

The interest is twofold: first, the PDL algorithms allow for a better observability, as we can easily trace the ScanMux originating the faults. We can aim at using this approach to cover as many faults as possible and use computationally-

intensive structural approaches only for the escapes. To carefully evaluate the coverage of the approach we need to use the vectors generated by MAST as input to a Fault grading Tool: we chose Synopsys TetraMAX, an industrial reference in the field, to get the failing responses for MAST. A direct interface cannot be used, because the Tool needs to run a simulation for each fault type, so it cannot be synchronized with an external module such MAST. We therefore use a 3-step procedure: first we run once the PDL algorithms to obtain a set of vectors to be sent to the System Under Test. Second, we use this set as input for a Gate-Level Tetramax Fault Injection campaign which can do the fault grading and compute a set of output vectors from the SUT, one for each injected fault. At the end of this phase we therefore have a set of "from SUT" vectors, each one representing the output of a fault configuration. Third, we use them as input to a second execution of MAST. We consider two possible outputs:

- For a fault-free "from SUT" set (the golden reference), MAST execution must terminate normally, as the PDL algorithms will receive the expected data;
- For faulty sets, MAST execution should result in at least a PDL-level error, allowing for detection.

IV. RESULTS

We ran our characterization on a set of ScanMux configurations covering the main dynamic architectural configurations of the standard. All configurations are available in two flavors: MuxPost: (the ScanMux is placed after its controlling register), and MuxPre (the opposite), and compose a representative set of the possible usages of ScanMuxes in an IEEE 1687 compliant system. To exercise each segment, we parametrized the PDL-1 algorithm to perform three iApply, each time writing a random value: to obtain reproducible and deterministic results, we hard-coded the seed for random pattern generation. In Tetramax, we injected both Stuck-at-1 (s-a-1) and Stuck-at-0 (s-a-0) faults on each ScanMux for each design, obtaining a 100% test coverage by direct observation of the pattern output. In total, we obtained 147 output patterns. We then run MAST in Simulation mode using the TetraMAX-computed patterns as "from SUT" vectors. The results are summarized in the top half of Table 1.

V. CONCLUSIONS AND PERSPECTIVES

In our experiments, MAST functional execution reliably detected all faults identified by TetraMAX with a low count of both Capture-Shift-Update cycles and total DUT Clock Cycles. The gain in terms of ATPG time can be significant: in all cases, MAST took less than a second to generate patterns. The faults escapes in Table 1 are linked to the topology choice of putting the ScanMux before its controlling bit: in case of a faulty Mux, it is therefore difficult to access the controlling bit in a reliable manner. Further work will be done to investigate the exact causes and adapt the algorithms appropriately. Anyway, it is a good design practice to avoid such setups, so the appearance of these cases in real circuits should be limited.

This established the capability of the approach to reliably detect faults in the ScanMuxes with an extremely fast Generation Time and with good performances in terms of both coverage and pattern count.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] R. Cantoro et al. 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.

[6] R. Cantoro, et al., "A new technique to generate test sequences for reconfigurable scan networks," in 2018 IEEE International Test Conference (ITC). IEEE, Oct 2018.

[7] Portolan M., "A Novel Test Generation and Application Flow for Functional Access to IEEE 1687 instruments", IEEE European Test Symposium (ETS'2016)

Table 1 Experimental Fault Coverage comparison

Testbench	Mux Flavor	Total Registers	Total ScanMuxes	Total Bits	Total Faults	Tetramax Coverage	MAST Coverage	MAST #CSU Operations	MAST Total DUT Clock cycles
OneSib	Post	1	1	32	8	100	100	4	120
OneSib	Pre	1	1	32	8	87.5	87.5	4	120
TwoNestedSib	Post	1	2	32	16	100	100	6	137
TwoNestedSib	Pre	1	2	32	16	93.75	93.75	6	137
Two_same	Post	2	1	64	8	100	100	5	190
Two _same	Pre	2	1	64	8	100	100	5	190
Two _diff	Post	2	1	48	8	100	100	5	142
Two _diff	Pre	2	1	48	8	100	100	5	142
Four _same	Post	4	1	128	14	100	100	9	351
Four _same	Pre	4	1	128	14	100	100	9	351
Four _diff	Post	4	1	80	14	100	100	9	239
Four _diff	Pre	4	1	80	14	100	100	9	239