FP7-ICT-2013-11-619871

BASTION

Board and SoC Test Instrumentation for Ageing and No Failure Found

Instrument: Collaborative Project

Thematic Priority: Information and Communication Technologies

Report on structural analysis, verification and optimization methodology for ICL networks (Deliverable D2.3)

Due date of intermediate deliverable: December 31, 2015
Ready for submission date: February 11, 2016
Due date of final deliverable: December 31, 2015

Start date of project: January 1, 2014 Duration: Three years

Organisation name of lead contractor for this deliverable: Hochschule Hamm-Lippstadt University of Applied Sciences

Revision 2.0.4

Project co-funded by the European Commission within the Seventh Framework Programme (2014-2016)		
Dissemination Level		
PU	Public	☒
PP	Restricted to other programme participants (including the Commission Services)	☐
RE	Restricted to a group specified by the consortium (including the Commission Services)	☐
CO	Confidential, only for members of the consortium (including the Commission Services)	☐

Notices

For more information, please contact Prof. Dr. Krenz-Baath, e-mail: rene.krenz-baath@hshl.de

This document is intended to fulfil the contractual obligations of the BASTION project concerning deliverable D2.3 described in contract 619871.

Table of Revisions

Version	Date	Description and reason	Author	Affected sections
0.1	June 18, 2014	Initial document created	R. Krenz-Baath	Contents
0.2	Sept 18, 2014	Full non-polished document	R. Krenz-Baath	All
0.3	Sept 22, 2014	Update to Section 3	A. Jutman	Section 3
0.4	Sept 22, 2014	Updates all Sections added Executive Summary	All	All
1.0	Sept 30, 2014	Reviewing and updating all sections	R. Krenz-Baath, F. Zadegan, S. Devadze, A. Jutman, C. Laudert	all
2.0.1	Dec 8, 2015	Initial version for final document	R. Krenz-Baath	all
2.0.2	Dec 10, 2015	Added and updated contributions by Lund, TUT, IFAG, HSHL	R. Krenz-Baath	Sections 3.3.4, 4.3.2, 4.3.5
2.0.3	Dec 15, 2015	Merged contributions	R. Krenz-Baath	all
2.0.4	Feb 09, 2016	Final version	R. Krenz-Baath	all

Author, Beneficiary

René Krenz-Baath, HSHL
Erik Larsson, ULUND
Farrokh Zadegan, ULUND
Jaan Raik, TUT
Artur Jutman, TL
Sergei Devadze, TL
Konstantin Shibin, TL
Piet Engelke, IFAG
Carsten Laudert, IFAG

Executive Summary

This document presents BASTION contributions in the area of modelling, optimization and verification of IJTAG 1687 networks. First the state of the art is discussed and the described contributions are set into context. Next initial research regarding 1687 network topologies in connection to their ability to keep the time overhead low (robustness) when accessed throughout the lifecycle of a product is presented. Furthermore, methods to perform fault management in 1687 networks are proposed. In addition, an approach to model 1687 networks in order to perform verification and validation is described. Afterwards several verification challenges are discussed which are relevant to ensure the compliance of ICL descriptions as well as to enable efficient re-targeting of PDL instructions and debugging of ICL descriptions. Finally, conclusions are provided.

List of Abbreviations

BIST	Build Inside Test
CPU	Central Processing Unit
CSU	Capture-Shift-Update Cycle
DIN	Diagnostic Instrumentation Network
DRC	Design Rule Check
ECC	Error-Correction Code
EDA	Electronic Design Automation
FP7	European Union's 7th Framework Programme
FSM	Finite State Machine
ICL	Instrument Connectivity Language
IEEE	Institute of Electrical and Electronics Engineers
IJTAG	Internal JTAG, a short name for IEEE 1687 standard and infrastructure collectively
ILP	Integer Linear Programming
IM	Instrument Manager
IST	Information Society Technologies
ITRS	International Technology Roadmap for Semiconductors
JTAG	Joint Test Action Group
LBIST	Logic BIST
LSIB	Locking Segment Insertion Bits
MBIST	Memory BIST
MUX	Multiplexer
PN	Perfect Network
NFF	No Fault Found, also No Failure Found
OAT	Overall Access Time
OS	Operating System
PCB	Printed Circuit Board
PDL	Procedural Description Language
RM	Resource Manager
RTL	Register-Transfer-Level
SAT	Boolean Satisfiability
SCB	ScanMux Control Bit
SI	primary Scan Input
SIB	Segment Insertion Bit
SO	primary Scan Output
SoC	System-on-Chip
TAP	Test Access Point
TDR	Test Data Register
URL	Uniform Resource Locator

Table of Contents

Table of Revisions iii
Author, Beneficiary iii
Executive Summary iv
List of Abbreviations v
Table of Contents iv
1 Introduction 2
1.1 The Structure of the report 3
2 Background 3
3 Analysis, Design and Optimization of IEEE 1687 Networks 6
3.1 Introduction 6
3.2 Related Work 7
3.3 BASTION Contributions 7
3.3.1 Robustness of 1687-Compatible Scan Networks 7
3.3.2 Race on Update Signal 10
3.3.3 IEEE 1687 Extensions for Fault Management 11
3.3.4 Safety and Security Aspects of IEEE 1687 Networks 18
3.4 Summary 23
4 Verification and Validation of IEEE 1687 Networks 24
4.1 Introduction 24
4.2 Related Work 25
4.3 BASTION Contributions 25
4.3.1 Modeling of IEEE 1687 Networks for Verification and Validation 25
4.3.2 ICL Compliance, Verification and Silicon Validation 33
4.3.3 Smarter Retargeting 37
4.3.4 Co-verification of ICL and PDL 38
4.3.5 Access Time Minimization in IEEE 1687 Networks 39
4.4 Summary 51
5 Conclusions 52
6 Bibliography 53

1 Introduction

In this deliverable, results are reported about the research performed by BASTION partners in the area of structural analysis, verification and optimization methodology for ICL networks. This document is a second and final edition of the deliverable. It concludes work performed in T2.4 “Analysis, verification and optimization of IJTAG network topologies”.

The document starts with an introduction to 1687 networks and an overview of the state of the art. After that, the document contains a detailed description of basic building blocks of 1687 networks. Later, the first BASTION contributions are presented and initial conclusions are formulated.

The first contribution in the area of analysis, design and optimization of IEEE 1687 networks investigates the robustness of 1687 networks with respect to the overall access time for different use cases or scenarios. The fundamental problem is that at design time it is very difficult to predict how an IEEE 1687 network will be used through the life-time. Embedded instruments will be applied in a broad range of applications with individual requirements. The contribution evaluates the overall access time on different network topologies. Another contribution discussed in this area is a new approach to extend ICL networks to support fault management. Fault management is increasingly important in context of aging and in-field validation. One of the core contributions is the proposal of flag-based error reporting infrastructure, which allows an effective mechanism to transmit error detection signals through large and complex ICL networks requiring only small implementation effort. Also the aspects of asynchronous fault detection and possible extensions of standard SIBs are discussed. Furthermore, approaches for fault localization and fault diagnosis in 1687 networks are presented. The final presented contribution in the area of analysis, design and optimization of IEEE 1687 networks focuses on safety and security aspects in particular an access protection concept is proposed which is based on separation of the group of instruments applicable by the future device user and the group of security relevant instruments into two separate IEEE 1687 networks.

The next set of contributions is related to the area of the verification and the validation of 1687 networks. After the state of the art is discussed, an initial approach to model a given ICL network using a graph representation is presented. Providing an efficient and complete model of a complex 1687 network is an inevitable requirement in order to perform scalable verification and validation of very large networks. The different steps of the modelling process are explained in detail and demonstrated using an explanatory example of an ICL network. Next a validation flow for 1687 networks is proposed which complies with the verification challenges discussed in the same section. Several checks of ICL networks on different levels are proposed, for example the accessibility of individual instruments, instrument correctness checks and detection of hidden or undocumented structures. Furthermore, future challenges with respect to efficient re-targeting techniques as well as the co-verification of PDL instructions in connection to a specific ICL network are discussed in detail.

1.1 The Structure of the report

This report is structured as follows. First the basic idea behind the IEEE 1687 standard including a short description of basic building blocks as well as descriptions of the Instrument Connectivity Language (ICL) and the Procedural Description Language (PDL) is discussed. Section 3 contains an introduction and set of BASTION contributions in the area of design and optimization of 1687 networks. BASTION contributions in verification and validation of 1687 networks as well as the discussion of future verification challenges in the field are discussed in Section 4. This report is concluded in Section 5.

The major differences of the 2nd edition of the deliverable compared to the initial version are the following:

- New sections 3.3.4 and 4.3.5 have been added;
- Sections 4.3.1 and 4.3.2 have been modified by updating/improving the algorithms.

Updates in other sections were only introduced where necessary and they are negligible.

2 Background

IEEE 1687 specifies JTAG as the main off-chip to on-chip interface to the instrument access infrastructure (the 1687 *network*) and is informally called Internal JTAG (IJTAG). IEEE 1687 includes (1) specifications on the hardware that interfaces the on-chip dedicated monitors, checkers, sensors, etc. collectively called instruments to the chip boundary, (2) a hardware description language called Instrument Connectivity Language (ICL) which describes the instrument's port functions and logical connection to other instruments and to the (chip-level) test access interface (e.g., JTAG test access port (TAP)), and (3) Procedural Description Language (PDL) which describes how an instrument should be operated at its terminals. The idea in introducing ICL and PDL is to provide an adequate and standardized description of the 1687 network, instruments, and instrument access procedures, and to enable ICL and PDL interpreter tools to automate the retargeting of access procedures. Retargeting is the process of translating the PDL commands at the instrument terminals to the higher design levels up to the chip pins, through the provided ICL.

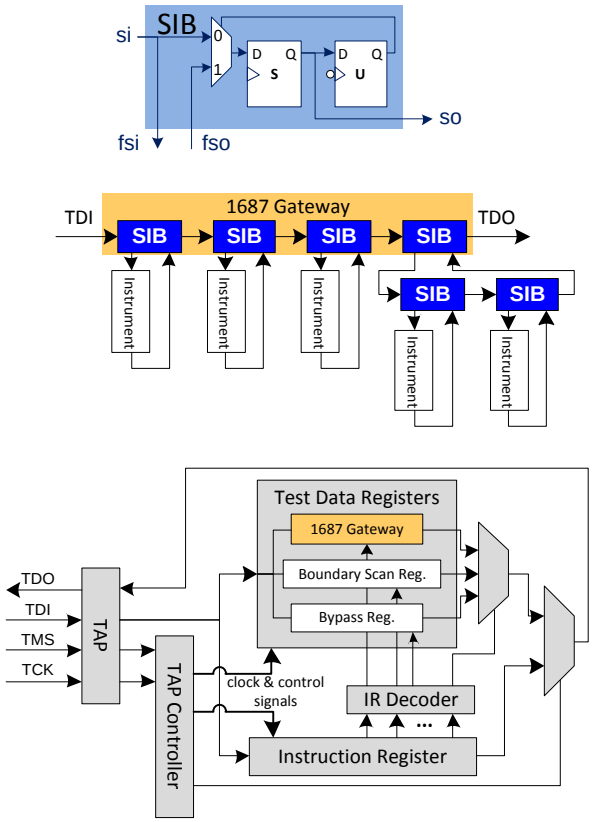

Figure 1 Example SIB, example 1687 network, and how the network is interfaced to the JTAG circuitry

One important feature in the IEEE 1687 specification is the concept of a variable length scan-path which can be implemented through a component called Segment Insertion Bit (SIB). A SIB is a 1-bit shift and update register on the scan-path which can be programmed to insert another segment of IEEE 1687 conform scan-path into the current (active) scan-path—hence the name Segment Insertion Bit. It is possible to build a multitude of different hierarchical 1687 networks for the same set of instruments by using SIBs. Figure 1 shows a possible SIB implementation, a small 1687 network, and an example of how a 1687 network is interfaced to the test access interface (i.e., test access interface in this case). From Figure 1 , it can be seen that the on-chip 1687 network is interfaced to the test access interface by using a dedicated Test Data Register (TDR) called Gateway. The Gateway itself is composed of one or more SIBs. The shown network also illustrates the concept of a variable length scan-path in which the first three instruments on the path are on the first hierarchical level and the remaining two instruments are on the second hierarchical level.

Another programmable component introduced by IEEE 1687 is a *ScanMux* control bit which is used to control the multiplexers on the scan-path. A ScanMux control bit (referred to as SCB in this report) is a one-bit shift-and-update register. Figure 2(a) shows a simple schematic for an SCB, and Figure 2(b) shows the symbol used for SCBs in the rest of this report.

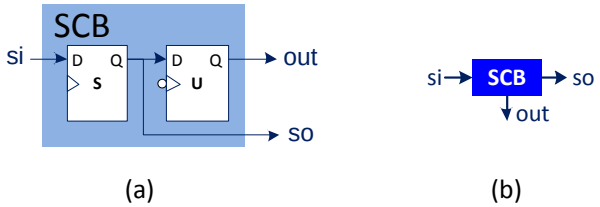

Figure 2: (a) a simplified schematic for an SCB, (b) the symbol for SCB

ICL: ICL is not meant to be an exact description of the actual hardware, but a logical description of the instrument access network. For this purpose, ICL introduces a number of building blocks such as *ScanRegister* for describing shift registers, *AccessLink* for describing the interface between the on-chip instrument access network and the test access interface, and so. In the rest of this document, a few more of these building blocks are introduced.

PDL: The PDL commands are categorized either as *setup* commands or as *action* commands. Setup commands configure the environment for the action commands, and action commands perform actual operations that make the setup commands take effect. For example, commands such as iTarget, iWrite, iRunLoop, and iRead are setup commands, whereas iPDLLevel and iApply are action commands which are immediately performed. That is, all the reads and writes that are grouped under one iApply are performed together, and the consecutive iApply commands are performed sequentially. This implies that if updates to the network are not to be applied at the same time, they should not be merged under the same iApply group. Moreover, this shows that the PDL code also determines the access schedule.

3 Analysis, Design and Optimization of IEEE 1687 Networks

This section discusses aspects of the analysis, the design and optimization of IEEE 1687 networks. After a detailed introduction and an overview of related work the contributions delivered by BASTION participants in this area are described. The first contribution investigates the optimization with respect to the overall access time as well as robustness and hardware overhead of 1687 networks by applying different network topologies. Furthermore, future design challenges and approaches for how to extend 1687 networks for fault management are described. Finally, security and safety aspects in 1687 networks are discussed and represent a new contribution of the 2nd edition of this deliverable.

3.1 Introduction

The IEEE Std. 1687 allows creation of dynamically reconfigurable on-chip instrument access networks. The flexibility in such networks brings two types of freedom: (1) the freedom to construct the network in a multitude of different ways, and (2) the freedom to schedule the access to the instruments in a variety of ways according to the given constraints (e.g., resource conflicts and power budget). As an example, consider three instruments with the interface shift-register length (L) of four, five and three flip-flops, which are to be accessed (A) ten, four and five times, respectively. Figure 3 shows three different ways to connect these three instruments as a 1687 network. It can be seen that the network in Figure 3(b) uses less hardware components (i.e., SIBs in this example) compared to the other two networks. Although the instruments are the same, the overall access time (OAT) for the network shown in Figure 3(a) is less than OAT for the networks in Figure 3(b) and Figure 3(c). This can be explained by noting that part of OAT is the time spent on programming the SIB components for every read/write access to the instruments [1]. Therefore, since in the network in Figure 3(a), it is possible to close SIB4 and keep SIB2 and SIB3 off the scan-path when accessing instrument with 10 accesses (which is the largest number of accesses among the three instruments), the time that should otherwise be spent on programming SIB2 and SIB3 is saved.

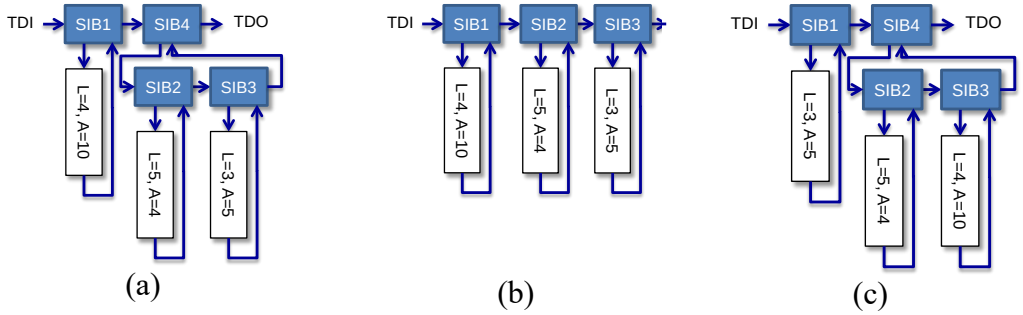

Figure 3. Three different networks for the same instruments

Moreover, for each of the shown networks in Figure 3, the access to the instruments can be scheduled in a variety of different ways, each way potentially resulting in a different OAT number [2]. The above example shows that there is a need to methods

for design of optimal 1687 networks (in terms of instrument access time, hardware overhead, etc.). Development of such methods, in turn, requires an exact analysis of different trade-offs (e.g., time overhead vs. hardware overhead). In the following sections, we will review the previous work in this regard, and will elaborate on how the BASTION project will enhance the state of the art.

3.2 Related Work

Access time analysis for SIB-based 1687 networks, under sequential and concurrent access schedules, is presented in [1]. Optimized SIB-based network construction is presented in [3], for concurrent and sequential schedules. The work in [2] presents an access time calculation method for general schedules, as well as optimized power- and resource-constrained test scheduling, for SIB-based 1687 networks. In none of these works, it is considered that instruments might be accessed differently under diverse circumstances.

The extension in form of design rules as proposed in [4] for IEEE 1687 standard is aiming at asynchronous interrupt-based in-situ error detection. The scheme targets fault management applications and attempts to minimize the error detection latency as well as the faulty resource localization time. In the BASTION project, enhancements to the work in [4] will be carried out in order to reduce the latency in error detection and localization.

3.3 BASTION Contributions

In this section, the contributions of the BASTION project regarding analysis, design, and optimization of 1687 networks are presented: Section 3.3.1 briefly describes a study performed to compare a number of 1687-compatible networks in terms of overall instrument access time (OAT), hardware overhead, and robustness [5]. Section 3.3.2 describes an example of potential design challenges that should be addressed in design automation for IEEE 1687 network design and PDL retargeting. Section 3.3.3 details how the use of 1687 networks can be extended to fault management. Section 3.3.4 focuses on security and safety aspects in 1687 networks.

3.3.1 Robustness of 1687-Compatible Scan Networks

Embedded instruments might be used in different situations during the life time of a chip, such as for post-silicon validation, debugging, wafer sort, package test, burn-in, printed circuit board (PCB) bring-up, PCB assembly manufacturing test, power-on self-test, and operator-driven in-field test. For each of these situations (referred to as usage scenarios hereafter) it is of interest to access some but not all of the instruments [6]. As an example, a memory built-in-self-test (MBIST) instrument might be accessed (1) during yield learning for a new process to choose the most suitable algorithms, (2) during wafer sort and package test to detect defective devices and perform repair, (3) in the burn-in process to cause activity in the chip and to detect infant mortality [7], [8], (4) during PCB bring-up [9], (5) during PCB assembly manufacturing test [9], and (6) during power-on self-test and operator-driven in-field tests.

Also, the number of accesses to a given instrument typically varies between different scenarios. For example, during yield learning, an embedded memory might be tested

several times by running multiple BIST algorithms. Another example is reading out the memory contents for diagnostic purposes [10]. In both examples many accesses might be needed. In contrast, during manufacturing tests, an embedded memory might be tested only by accessing the associated MBIST engine a few times to setup the algorithm, start the BIST, check for its completion, and read the results.

Furthermore, at design time it is (1) difficult to foresee all needed scenarios, and (2) how many times an instrument will be used at each of the scenarios. The number of needed scenarios and the number of accesses might be affected by late design changes, adding/excluding tests, or change of constraints, such as power consumption. Some changes may only be known after manufacturing.

Previous works on network design assume one known scenario where the number of accesses is fixed [3]. However, a 1687 network optimized for one scenario might not be as efficient for another one. And, a network optimized for a fixed number of accesses, might not be optimal if the number of accesses changes. Therefore, we studied [5] the robustness of seven approaches for designing 1687 networks, and we examined their efficiency, in respect to OAT and hardware overhead.

The studied network design approaches are (1) a flat network, (2) a single hierarchical network, and (3) multiple networks, as well as a daisy-chained counterpart for each of those three, namely (4) flat daisy-chain, (5) hierarchical daisy-chain, and (6) multiple daisy-chains. In addition to the six enumerated approaches, we study one more approach in which two separate JTAG test data registers (TDRs) are used for the instrument access network: one for network configuration, and one to access the instruments. This approach will be referred to as “separate control and data TDRs” hereafter.

Since there was a need to OAT calculation methods for performing the comparison among the mentioned network design approaches, we presented in [5] OAT calculation methods for those studied approaches for which such methods were not available from prior work.

Robustness: Ideally, a network should incur zero (or minimal) configuration time overhead (i.e., clock cycles spent on network configuration and the operation of test access interface FSM) regardless of the scenario in which it is used. Therefore, a network can be considered robust against changes in the scenario—namely, changes in number of accesses and changes in the access schedule (for example, change from a concurrent schedule to a sequential schedule)—if such changes do not change the overhead significantly. However, since a change in the number of accesses to instruments changes both instrument data and configuration time overhead, the overhead alone cannot be used as a measure for robustness. That is, it might happen that despite an increase in the overhead (in terms of clock cycles), it becomes negligible compared to the data transferred over the network to the instruments (i.e., instrument data) in the new scenario. Therefore, considering that OAT consists of instrument data and overhead, a network can be said to be robust if the ratio of OAT to instrument data does not change significantly between scenarios. Therefore, we calculate the ratio of OAT to instrument data for each scenario for a given approach, and we consider the standard deviation of the calculated ratios as the metric for

robustness of that approach. The smaller the metric value is, the more robust the approach will be.

Experimental results:

To compare the studied network design approaches, two sets of experiments were performed. In the first set, eight scenarios were considered and it was assumed that a weight is assigned to each scenario by the designer, indicating the relative importance of OAT reduction for that scenario in comparison with other scenarios. For each of the above-mentioned seven approaches, OAT achieved under each scenario was calculated, multiplied by the weight assigned to corresponding scenario, and summed up. This experiment was performed for three cases where (A) only one scenario (out of eight) is known at design time, (B) five out of eight scenarios are known at chip design time, (C) all scenarios are known at design time. The chart in Figure 5 presents the results for the first set of experiments. The second set of experiments studied robustness against change of concurrency in a given scenario. Figure 6 shows the results of this experiment in which for one of the given scenarios, the number of concurrently active instruments is gradually increased from one active instrument to 100 concurrently active instruments.

The experimental results for both sets of experiments showed that the network generated by the "separate control and data TDR" approach results in the least OAT among networks generated by the considered design approaches (see Figure 5), has low hardware overhead (compared with other networks with similar OAT numbers), and is the most robust among the studied approaches. In this approach, SCBs are placed in a TDR separate from the instrument shift-registers' TDR (see Figure 4).

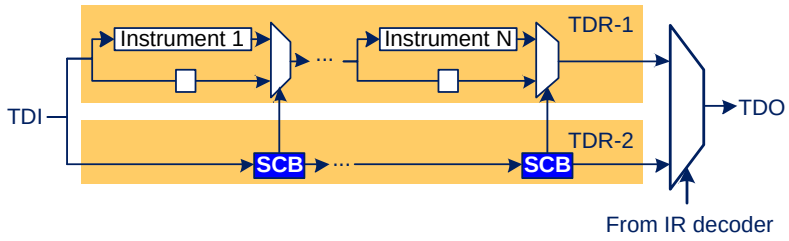

Figure 4. Using separate TDRs for instruments and SCBs

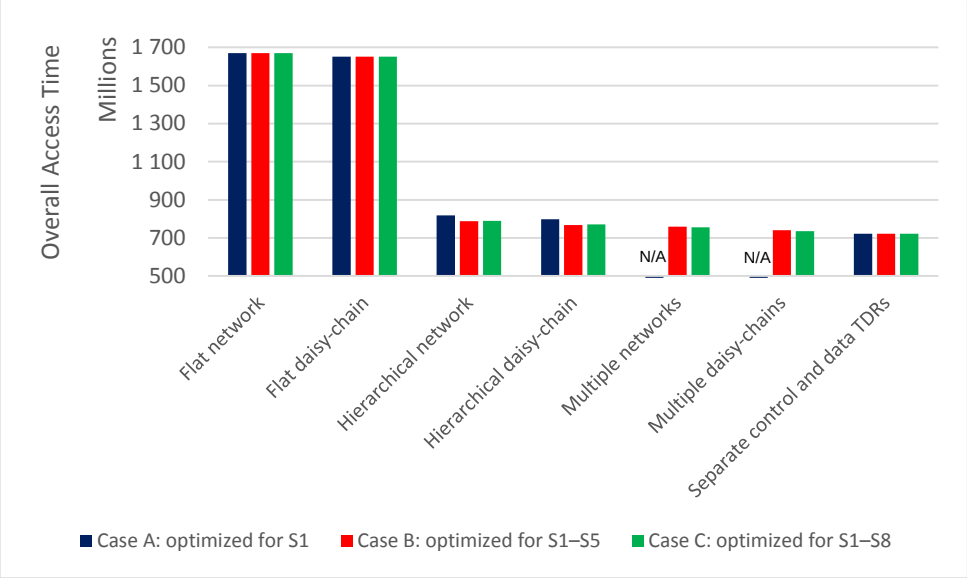

Figure 5. Results for the first set of experiments. The reported numbers are the weighted sum of the individual OAT numbers calculated in test clock cycles for each scenario.

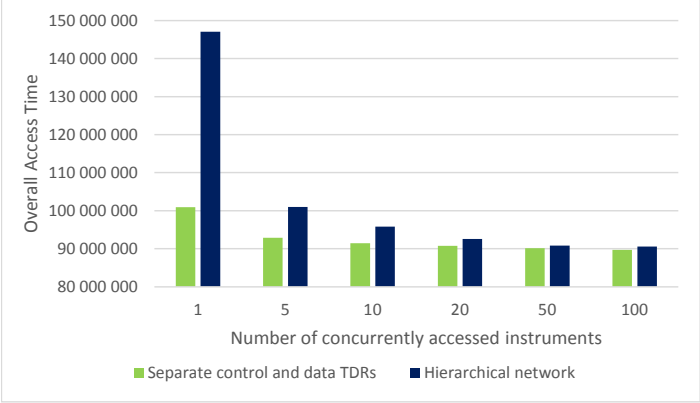

Figure 6. Results for the second set of experiments. Overall access time is reported in number of test clock cycles.

3.3.2 Race on Update Signal

A race hazard is a situation in which the correct operation of a digital circuit becomes dependent on the actual delays that occur on the signal transitions after the chip is being implemented. In this regard, it should be explained that in a hierarchical 1687 network, the control signals such as *ShiftEn*, *UpdateEn*, and *CaptureEn* that are applied to a component at a given hierarchical level, are gated by using an *enable* signal from the parent component–i.e., a component at a hierarchical level closer to the 1687's Gateway (see Figure 1). If as a result of an update, the transition of the *enable* signal reaches a component earlier than the update signal itself, that component might not see the change on the update signal which is gated by the enable signal. As an example, inFigure 7, U1 generates the "enable" signal. Assume that the current input to U1 is '0' and the current output of U1 is '1' (which has propagated to the input of the AND gate). The "UpdateEn" signal has two paths: a path to the clock of U1, and a path to the gating input of the AND gate. If the delays are such that a rising edge in the

UpdateEn signal reaches the AND gate later than the '0' on the U1 input has reached there (as indicated by the dashed arrows), this rising edge will be masked thus blocking the input signal of U2 to the output.

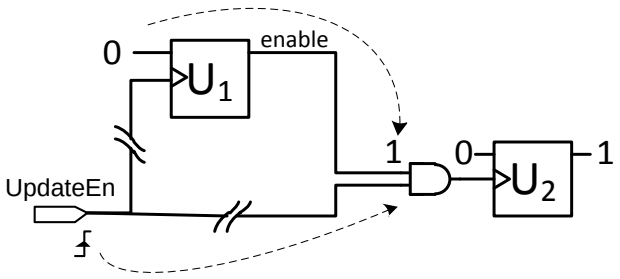

Figure 7. Race between two signals

In this example, the race can be eliminated by adding a delay element (e.g., one that acts on the negative edge of the UpdateEn signal) after the enable signal. In this work, we will investigate if this sort of race on the update signal is automatically addressed by the current tools used for synthesis, static timing analysis, and place and route. If it is not the case, a retargeting tool could prove helpful if it locates where delay elements should be inserted in order to avoid potential races. Moreover, since PDL could be written such that race is avoided by applying successive changes to the hierarchical layers (instead of simultaneous updates), an alternative solution to the problem of race on the update signal, would be that the retargeting tool breaks simultaneous updates (i.e., those in one iApply group) that can cause the such race situation, into multiple consecutive iApply commands.

3.3.3 IEEE 1687 Extensions for Fault Management

It has been shown that IEEE 1687 standard infrastructure can also be successfully and efficiently used or reused later in the field during the product's life cycle. One such application is fault management [4] – the usage that was hardly foreseen by the IJTAG standard development action group. IEEE 1687 standard allows to create an efficient and regular network for continuously handling fault detection information as well as to manage test and system resources as a system-wide background process during the system operation. The IJTAG framework matches especially well the requirements for supporting graceful degradation of the silicon under pressure of aging (which is the main topic of WP1), the problem that has been recently reported by the ITRS among a few most important research challenges. This sub-section describes the way the IEEE 1687 infrastructure can be efficiently extended to support fast emergency signaling for online-fault detection and localization as a background process in a running system. The extension for on-line diagnostics is based on two main components:

1. Hierarchy of status flag registers (IJTAG-compliant);
2. Asynchronous emergency network (non-IJTAG compliant).

In frames of BASTION we continue the research direction outlined earlier by us in [4] in the direction of error detection latency minimization and faulty resource localization speed acceleration.

3.3.3.1 Introduction to the IEEE 1687 based Fault Management

The aging failure resilience framework that we describe is based on fault tolerance and system health monitoring but goes beyond by localizing and classifying faults into different categories that should be handled differently: e.g. transient vs. permanent faults and critical vs. low-priority ones [4]. Independently of their priority, aged blocks with permanent faults are then either fully isolated or marked as reduced-capacity ones. In such a way a system health map is maintained that is used by an operating system (OS) to schedule new tasks. The main benefit of using IEEE 1687 IJTAG infrastructure for in-situ fault management is based on considerable reuse of existing test and debug infrastructure and instrumentation later in the field for the new purpose of resource management.

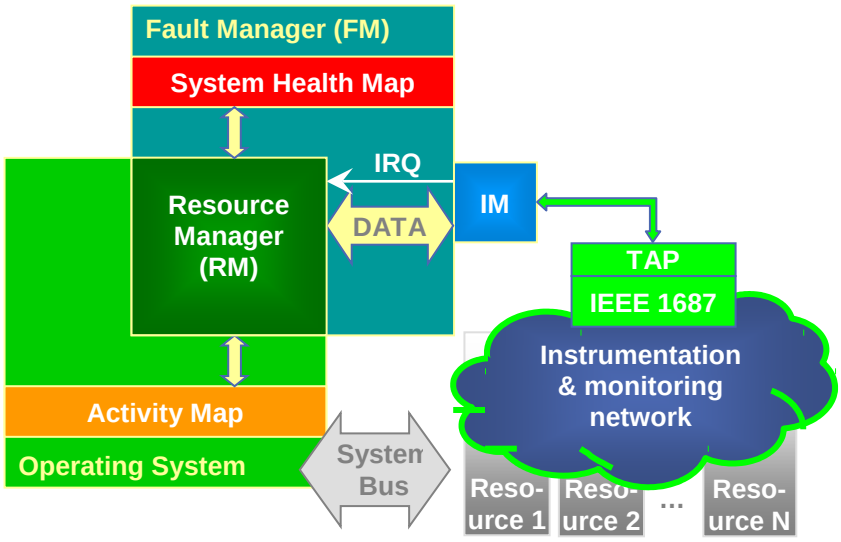

Figure 8. Failure resilient system with IEEE 1687 fault management network

The general concept is depicted in Figure 8. We assume that the fault management framework operates on the same SoC as the target system itself. Its parts - Resource Manager (RM) and Instrument Manager (IM) are closely coupled to the functional part of the system: RM is running on top of OS and exchanges data with IM which can be implemented as software on separate processor core or dedicated hardware. The rest of the SoC has an arbitrary structure but we assume that the target system contains heterogeneous or identical IP cores capable of fully or partially replacing one another, hence providing a room for graceful degradation. More relaxed requirements/scenarios (e.g., analysis on an external processor) are also possible with limited modifications of the basic concept described below.

The actual error detection is taking place in-situ by embedded instruments/monitors. The Diagnostic Instrumentation Network (DIN) should then immediately pass an emergency signal from the monitor to the OS so that the latter could reschedule the failed task immediately to another available resource. After detecting the fault, the failure has to be diagnosed and analyzed in order to update the system health map and isolate the resource in case of permanent fault detection.

The first priority requirements for an effective fault management system are error detection latency, faulty resource localization speed, reliability and scalability of the service infrastructure. They are followed by the possibility of design automation, reuse, and sharing of the IPs and/or instruments. The DIN has to provide unified interfaces to instruments from different vendors as a part of IP cores.

In [4] we proposed a flag-based error reporting system where each block or sub-module is provided with a dedicated flag, whereas all flags are collectively forming a hierarchical error indication and propagation structure tied with IEEE 1687 network. In BASTION, we elaborate and optimize the flag-based asynchronous signaling infrastructure (see Figure 9), elaborate on SIB design, consider DIN behavior in multiple fault scenarios, as well as study conflicting interrupts in the system.

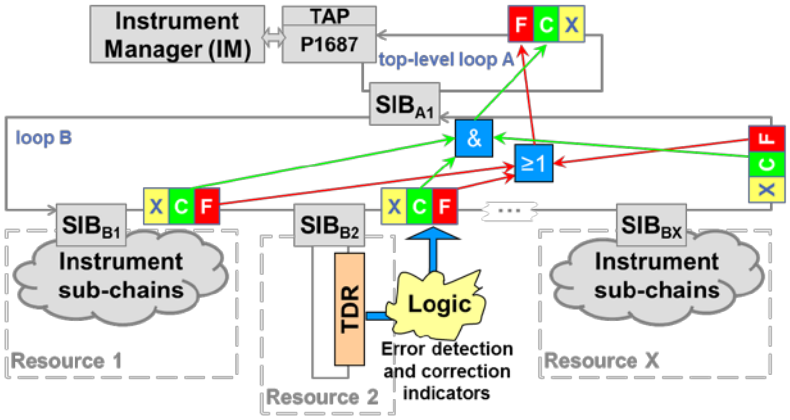

Figure 9. IJTAG network with asynchronous flag-based error reporting and localization system

3.3.3.2 Asynchronous Fault Detection

This subsection details respective requirements (design rules), which enables automatic seamless integration of heterogeneous diagnostic resources into a single homogenous system-wide DIN. As mentioned above, the rules describe two main extensions:

1. Hierarchy of status flag registers (IJTAG-compliant);
2. Asynchronous emergency network (non-IJTAG compliant).

Thanks to these rules and the IEEE 1687 standard, the DIN can be constructed in a way that allows creation of a single very small controller (instrument manager - IM) that handles all faulty resource localization tasks and instrument control for the whole SoC independently of its size [4].
The main part of our method is to propagate fault indication flags from instruments directly to IJTAG network controller by means of dedicated asynchronous signals. These signals carry information from instruments to higher levels and represent a significant contribution as the original IEEE 1687 standard does not consider such mechanisms at all. Yet, our proposal does not contradict the IEEE 1687 concept but rather appends it.
To be in line with hierarchical network structure with SIBs, asynchronous signals from a lower level of hierarchy are aggregated using logic gates to produce one signal for the higher level of hierarchy (see Figure 9).

There are two status bits and one mask bit for each SIB in the DIN. First status bit F (Fault) indicates the presence of a fault that occurred in an instrument in child segment of the SIB or somewhere further down in the hierarchy. The second status bit C (Correction) shows that the fault was automatically corrected. Mask bit X is used to disable fault detection for the underlying network segment, (e.g. to prevent the instrument from generating ”false alarms”). Figure 9 demonstrates an example of such instrumentation network implementation in a many-core SoC.
As described in earlier articles [6], the DIN may have IM which is responsible for controlling the instrumentation network and Resource Manager (RM) which is monitoring the status of instruments [10] and feeding the information to higher levels of system control, like OS scheduler.

3.3.3.3 Extended SIB Design vs. the Standard SIB

The IEEE 1687 standard defines the construction of instrumentation network as a serial scan chain divided into segments and connected together in a hierarchical manner. In such a hierarchical network the segments are connected in host-client (parent-child) manner where child segments must be accessed from parent segment through special blocks or SIBs. These blocks are controllable and allow concatenating the child segment into the scan chain. This allows manipulating the active configuration of the IJTAG network by including and excluding needed segments of the network.

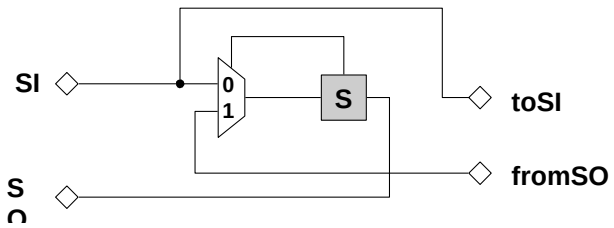

Figure 10. Normal SIB according to IEEE 1687 standard

A *standard SIB* is shown in Figure 10. Standard SIB has a client interface with SI (Serial Input) and SO (Serial Output) ports and a host interface with toSI (to Serial Input) and fromSO (from Serial Output) ports, a scan multiplexer and the SIB register. Depending on the value of SIB register, the scan multiplexer chooses either to:

- (value 0) connect its client signals SI to SO, thus skipping the child segment
- (value 1) connect the output of child segment fromSO to SIB’s output SO, effectively including child segment into the scan chain

The introduction of asynchronous fault detection requires some modification of the SIB or an add-on module which would accommodate additional fault management features integrated with SIB. These features support additional status bits (F, C and X) as well as propagation of their values to the upper levels of hierarchy.
The registers required to capture the value of Fault, Correction and mask flags can be added to the design of SIB, as well as synchronization circuit that will allow synchronous capture of flag values.
However, there are several possibilities of inserting these registers into SIB. While inserting a register into parent scan chain of the SIB will allow capturing the value of the registers without opening the SIB, it will increase the length of this scan chain. On

the other hand, inserting a register into the child scan chain of the SIB will not lengthen parent scan chain, but it will require opening of the SIB to gain access to the register.
We choose to keep all three registers in the parent scan chain to be able to monitor and control respective network segments as quickly as possible.

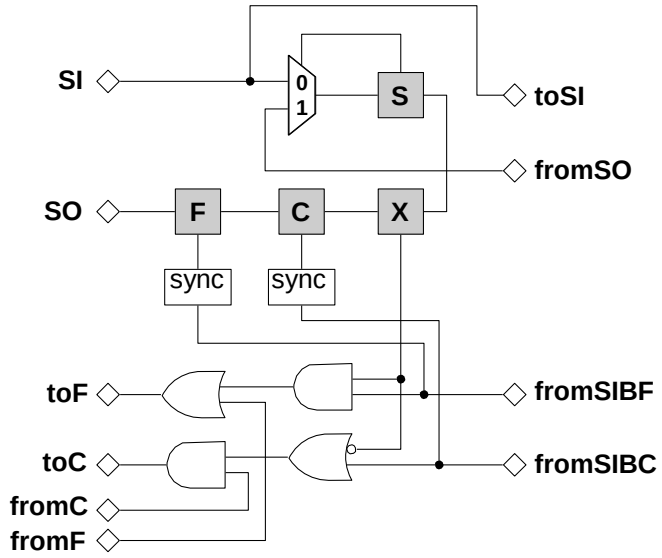

Figure 11. Extended SIB with asynchronous fault detection signals

The difference between the normal SIB and the extended SIB is evident when comparing Figure 10 and Figure 11 correspondingly. The extended SIB has 6 additional signals used for asynchronous signal propagation, 4 of which are on the client side:

- **toF**: Fault flag output. It is a result of logic OR operation between the Fault flag of previous SIB in the parent scan chain and Fault flag of child scan chain
- **toC**: Correction flag output. It is a result of logic AND operation between the Correction flag of previous SIB in the parent scan chain and Correction flag of child scan chain
- **fromF**: Fault flag input from previous SIB in the parent scan chain
- **fromC**: Correction flag input from previous SIB in the parent scan chain

Additional 2 signals are on the host side:

- **fromSIBF**: Fault flag input from child scan chain. It can be masked with value 0 in Mask bit
- **fromSIBC**: Correction flag input from child scan chain. It can be masked with value 0 in Mask bit

The extended SIB contains 3 additional registers (F, C, X) which are always included in the scan chain of SIB:

- **F bit**: it captures its value from synchronized Fault signal of child scan chain, which can be then shifted out to IJTAG network manager
- **C bit**: it captures its value from synchronized Correction signal of child scan chain, which can be then shifted out to IJTAG network manager

- **X bit**: it is updated by IJTAG network manager to mask Fault and Correction signals from child scan chain of this SIB so that they don't affect the operation of asynchronous fault detection system

Altogether, this additional hardware allows to connect the Fault and Correction signals hierarchically between networks segments (signals come from child segment to parent segment) and in daisy-chain fashion inside one network segment, i.e. similarly to the original IEEE 1687 routing concept.

3.3.3.4 Fault Localization and Diagnosis

Here, we briefly describe the way the IEEE 1687 DIN detects and localizes a faulty resource in event of fault detection. Detailed examples and scenarios will be elaborated in frames of WP3.

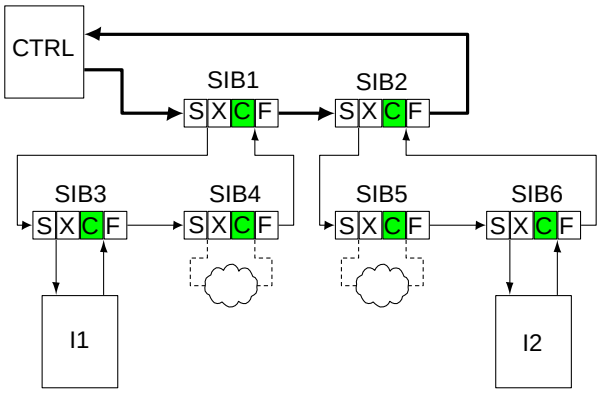

Figure 12. Example of IJTAG instrumentation network with asynchronous fault detection

The end point of the faulty resource localization is the identification of the particular instrument that detected the fault. An example DIN in its normal state is shown in Figure 12. The active scan chain is shown in bold while the inactive parts are shown in thin lines. Let's assume that the fault is detected by instrument *I1*. The fault is not automatically corrected and thus needs to be taken care of by the IM. The initial state of the DIN is with all SIBs closed.

At the time of fault detection, the instrument sets its Fault flag and clears Correction flag. These changes are propagated asynchronously to *SIB3* and then to *SIB1*, although *SIB4* is still in normal state (see Figure 13).

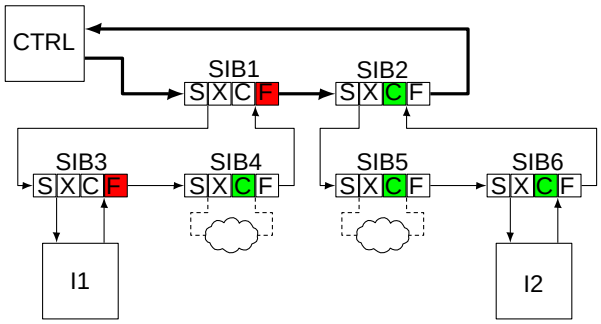

Figure 13. Detection of a single fault by instrument I1

When asynchronous fault detection signals reach the IM, the latter responds by performing localization actions, i.e. by opening *SIB1* and *SIB3* accordingly to corresponding indication FCX flag register. When the localization procedure is finished; the IM recognizes that the fault is situated in the child segment of *SIB3* which consists only of instrument *I1*. Then the IM will open *SIB3* and read out *I1* status. At this point the fault is localized and the control is given over to the FM for fine-grade diagnosis, fault classification and statistics data update.

Further research on fault localization scenarios is being performed in frames of WP3.

3.3.3.5 Interruption of Ongoing Access in Event of Fault

The normal work of IJTAG instrument network consists of read and write accesses to instruments, opening and closing the SIBs which are the results of normal requests from higher levels of the system (like OS). Since the IJTAG network can be constructed hierarchically to optimize access times, it may take considerable time to finish ongoing access and switch to another request.
Since asynchronous fault detection is designed to be as quick as possible to detect and find the source of fault, its speed would be useless if corresponding accesses to faulty re- source were delayed until current normal access is completely finished. Therefore, it is reasonable for IJTAG network with asynchronous fault detection to interrupt ongoing access and service faulty resource localization as soon as possible.
The localization of faulty resource would probably require modifying the configuration of IJTAG network, i.e. closing some SIBs and opening other SIBs. Hence, the current status of IJTAG network must be considered and an optimal modification, i.e. action that leads to quickest access to faulty resource, should be taken.
We foresee two possible solutions that can provide fastest reconfiguration of the IJTAG network in different situations:

- Dynamic retargeting
- Reset of IJTAG instrumentation network

Both procedures will be considered in our ongoing research in frames of WP3.

3.3.3.6 Multiple Fault Scenario

During the lifetime of the system it may happen that multiple faults occur at the same time, or at least, their localization times could overlap. In this case, it is important to make sure that fault management architecture would allow detecting both of them and reacting accordingly.

According to our initial observations the overall status of the asynchronous fault detection scheme remains adequate in case of several faults occurring at the same time or one fault occurs when another is already being localized.
This case will also will be considered in our ongoing research in frames of WP3.

3.3.4 Safety and Security Aspects of IEEE 1687 Networks

The reuse of existing chip internal test instruments on board-level is one method to improve the overall system reliability and allows the screening for potential aging effects. The IEEE 1687 standard simplifies the access and usage of embedded test instruments by describing their integration and test features by the standardized descriptions languages “Instrument Connectivity Language” (ICL) and “Procedural Description Language” (PDL). Even if the usage of chip-level test instruments is beneficial for the quality and diagnosis of board-level tests, this reuse also leads to new safety and security issues.
Not all chip-level test instruments and debug features, that have been initially implemented for the chip-level production test, are intended to be reused on board-level by the customer. Inappropriate usage of these features might cause malfunctions or even damage the device. From the safety point of view there are three different classes of test and debug instruments:

1. Instruments that can be used on board-level without any risk
2. Instruments that are intended to be reused in the system, but require special care
3. Test and debug features not intend to be used in the system

The first category of safe to use instruments contains all kind of health monitors and sensors that allow only the non-intrusive reading of data. As long as the chip-internal parameters are only read and the data cannot be modified, the usage of these instruments can be considered as safe and the access is allowed at any time.
The second class contains active test instruments that can disturb the system functionality when they are triggered. Typical examples are all kind of built-in self-test (BIST) engines. In most cases BIST circuits cannot be used while the application is running as they are modifying chip-internal data. BIST operations are usually performed at the start-up or shut-down of the device. An execution while the device is active is only possible, if the component to be tested is not functionally used at this moment. In addition, the BIST must not disturb the surrounding system. This usually requires some kind of isolation. After finishing the BIST the initial functional state needs to be restored. As a consequence, the user starting a BIST operation needs to be very aware of the system state and the potential impact of the BIST execution. In addition, there are usually other constraints when starting a BIST or using other active test instruments. For instance, running multiple BISTs at the same time might lead to power issues disturbing the test execution or even damaging the device.
The third groups of test and debug instruments contains features not intended for the customer usage. For the production test it is often possible to adjust chip internal voltages, timings and frequencies through the IJTAG network. These features should be usually hidden from the customer to prevent misuse affecting the device stability.
Giving an example for these three different instrument classes e.g. for an embedded memory, it is always safe to read out some ECC statistics monitoring the data corrections. Using an implemented MBIST to test the memory in detail might be

allowed from the board-level, but special care is required when and how to execute the test. Access to memory internal test features e.g. modifying memory internal timings is usually restricted for the customer.
Even if the user intends to access only the public test instruments in the appropriate way, there is a certain risk that accidently a wrong IJATG sequence might lead to a critical system operation. Especially when using the IJTAG network not only for dedicated tests, but also for online monitoring this is a risk to be considered.
Allowing the board-level access to the chip-internal IJTAG network causes not only safety issues as described above, but also leads to security risks. Some components accessible through the IJTAG network might contain or give access to confidential information that needs to be protected from unauthorized access. In addition, the IJTAG network must not enable prohibited circuit manipulations e.g. unauthorized chip tuning. Therefore, it is desirable to hide certain IJTAG functionality and to restrict the access.
The traditional way of accessing chip-internal test registers before the introduction of IJTAG was to use multiple so-called "private instructions" of the IEEE 1149.1TAP controller. The Boundary Scan Description Language (BSDL) specified in the IEEE 1149.1 standard supports these instructions. The content of the user-defined test registers is not described in the BSDL file, only the instruction opcodes of the private instructions are mentioned. This information is needed for board-level Boundary Scan pattern generation tools to prevent the usage of these instructions. This means in IEEE 1149.1 BSDL there is a clear separation of public Boundary Scan instructions and restricted private instructions accessing chip internal test data registers. In the ICL introduced with IEEE 1687 there is no methodology foreseen to distinguish between public and restricted parts of the IJTAG network.
Of course the description of private instructions in the BSDL does not prevent the access to the related test data registers. Even if it is clear that it is not a good idea to access these registers without knowing their functionality, this would not hinder attackers to do so. For this reason, the chip internal test data registers are often secured to prevent an accidental or attacking access. Typical measures to protect the activation of certain test modes are additional passwords needed for the access. In many cases these access restrictions are combined with electrical measures like under- or over-voltage detection at some pins at some time. While the test mode can be easily entered during production test of the chip, the activation on the functional board would be restricted.
There are few publications dealing with the safety and security aspect of the IEEE 1687 IJTAG network. In [11] so-called Locking Segment Insertion Bits (LSIB) are introduced replacing standard IEEE 1687 SIBs by an enhanced version with an additional key input required to "unlock" the SIB. The branch of the IJATG network controlled by the LSIB can be entered only when the condition of the key input is met. This method is extended with so-called trap bits intended to significantly increase the effort for attackers to unlock the LSIBs.
Another proposed solution to restrict the access to IJTAG test instruments is a sequence filter placed between the TAP controller and the IJTAG network [12]. The sequence filter is a Finite State Machine (FSM) operated using the JTAG signals TDI, capture, shift and update as input. An "allow" signal is only asserted for allowed input sequences. In case of a forbidden access sequence the "allow" output becomes 0 and the FSM enters a trap state. The "allow" signal is used to gate the update signal of the IJTAG network preventing any changes in the update registers when not allowed.

In [11] it is assumed that the attacker does not have any initial knowledge about the architecture of the implemented IJTAG network. When allowing an IJTAG based reuse of chip-level test instruments on board-level, the user would require at least a partial ICL description containing the structural information how to access the public instruments. On the other hand, this partial ICL description could be at least a starting point for an attacker. If the public and restricted test instruments would be mixed on various hierarchy levels of the IJTAG network, it would be difficult to hide the restricted parts in the ICL while still providing a valid file that can be understood by tools for an automated pattern retargeting. The main drawback of the method proposed in [12] is the need to recreate the sequence filter hardware each time when the IJTAG network or the allowed access sequences are changed. As both methods are not yet supported by commercial EDA tools an alternative approach has been evaluated in the next section.

The basic idea of the proposed access protection is the separation of the IJTAG network into two parts. The first part of the IJTAG network contains the test instruments to be reused by the customer and is directly accessible by one JTAG instruction. The restricted functions to be hidden from the public are grouped together in a completely independent IJTAG network accessible by another JTAG instruction (see Figure 14). The restricted IJTAG network can be secured by standard measures as done for normal private JTAG instructions.

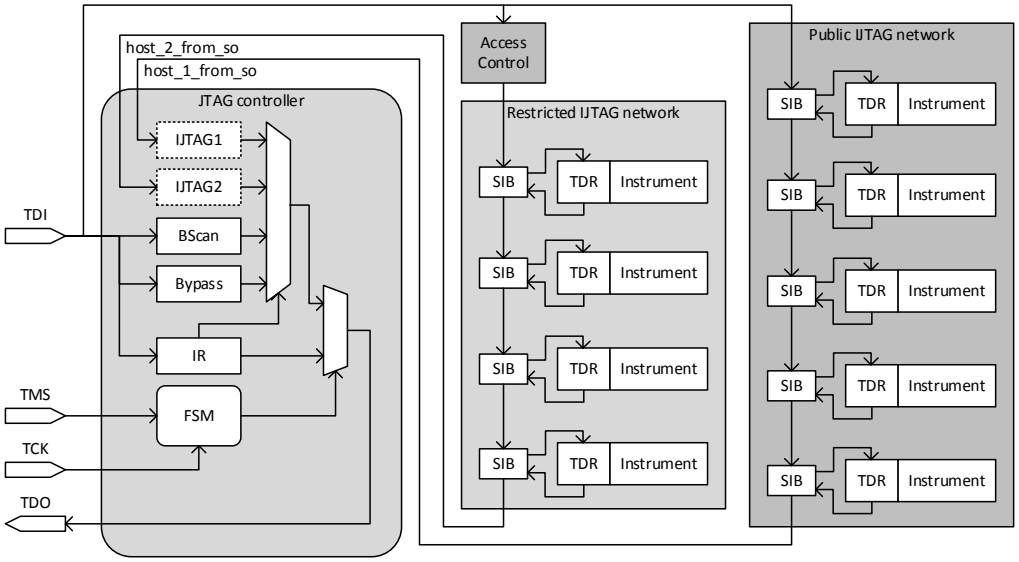

Figure 14. Overview of the proposed IJTAG network architecture

The overall design flow using the commercial IJTAG tool “Tessent IJTAG” [13] and the compatibility with the IEEE 1687 standard have been evaluated for a small test case design. The test case is based on a small processor design with four identical CPU cores. Each CPU core is equipped with a LBIST solution allowing a self-test of the digital logic. The LBIST initialization is done by several setup registers. The tool generated RTL code of the LBIST controller contains already an IJTAG interface and several SIBs grouping these configuration registers. The required ICL and PDL files describing the LBIST controller implementation and functionality are automatically generated together with RTL code by the Tessent LBIST tool. The CPU cores are implemented supporting several different scan test modes. The related scan configuration signals are only routed to the core boundary for a later integration by the

IJTAG tool. The test case design contains some more test modes on the top-level that also need to be considered for the IJTAG network generation.
The insertion of the IJTAG network has been done on the gate-level netlist using the "Tessent IJTAG" tool. The existing ICL files of the LBIST controller have been read together with some manually written ICL files describing other test functionality e.g., test clock multiplexers required for the scan based delay test. For the core-level scan configuration pins IJTAG attributes have been defined directly in the tool to trigger their integration into the IJTAG network. In a first step the tool has been free to build an optimized IJTAG network directly connected to a JTAG controller which has been also generated by the tool. The resulting design is used as reference implementation.
In the second phase the IJTAG network specification proposed by the tool has been manually edited to enforce a structure as shown in Figure 14. For the test case it is assumed that the test instruments of two CPU cores should be available for the user, while the test features of the other two CPU cores should be hidden and placed in the separate ICL network. To do so, the proposed IJTAG network specification of the first step has been duplicated using a copy and paste feature of the tool. Afterwards the IJTAG components of the two CPU cores that should be hidden have been removed from the first IJTAG network specification. In the specification of the second IJTAG network only the components of the two restricted cores have been kept. Due to the initial duplication of the IJTAG specification afterwards some component names have been made unique e.g., by simply adding the suffix "_1" and "_2" for the first and second IJTAG network. Based on this specification the tool has automatically created and integrated the intended IJTAG architecture into the test case design.
The automatically generated top-level ICL file contains the complete description of both IJTAG networks and their connection to the JTAG controller. Before providing the ICL file to the customer the restricted components need to be removed. For the test case design this has been done simply by editing the file. For more complex designs this could be done by some scripts, especially when the components to be removed have unique naming extensions as mentioned above. There is one part in the remaining ICL file that needs to be patched. The output from the restricted IJTAG network, which has now been removed in the ICL description, can no longer be used as input signal going into the TAP controller (see also Figure 14). To have a legal ICL file the original connection has been replaced by a constant X-value as shown in Figure 15. Of course the original (red) line could have been removed to hide traces to the removed ICL components.

```
Instance chip_tap_main_inst Of
    chip_tap_main {
   InputPort host_1_from_so = chip_sib_sri1_inst.ijtag_so;
// Patch replacing the original connection from hidden IJTAG by constant X
// InputPort host_2_from_so = chip_sib_sri2_inst.ijtag_so;
   InputPort host_2_from_so = 1'bx;
   InputPort tck = tck;
   InputPort tdi = tdi;
...
```

Figure 15. Required change in generated ICL file disconnecting the output of the protected IJTAG network

After the deletion of the restricted components there are only very few traces remaining in the ICL description indicating the hidden second IJTAG network. Of

course, the TAP controller description still contains the private instruction accessing the hidden IJTAG network, but based on the patch of Figure 15 it is no longer visible what kind of network and data can be accessed. Only the names used for the private JTAG instruction and the related scan interface indicate the existence of a second IJTAG network (see Figure 16), but these names could be easily changed.

```
Module chip_tap_main {
...
   ScanInterface host_ijtag_2 {
      Port host_2_from_so;
      Port host_2_to_sel;
   }
...
   Enum instruction_opcodes {
      BYPASS = 4'b1111;
...
      HOSTIJTAG_1 = 4'b0111;
      HOSTIJTAG_2 = 4'b1000;
   }
...
   LogicSignal host_2_to_sel_int {
      instruction == HOSTIJTAG_2;
   }
...
```

Figure 16. Locations in generated ICL indicating the existence of the protected IJTAG network

After the integration of the split IJTAG network into the design the structure has been verified by using the automated test pattern retargeting flow and simulating the resulting patterns. At first the complete ICL description has been used to retarget the core-level LBIST execution for all four CPU cores on the top-level as it could be done during the production test. The IJTAG tool is able to generate the appropriate JTAG sequence to start and evaluate all BISTs making use of both IJTAG networks and the two related JTAG instructions. The generated test patterns have been simulated without any mismatches.

Afterwards the customer application use case accessing only the two “public” LBIST instances based on the reduced ICL description has been verified. Also this pattern retargeting step and the following patter simulation have been performed without any issues.

From the design flow and tooling point of view the proposed separation of the overall IJTAG network into a public and restricted part can easily done using today’s available EDA tools. Only the removal of the hidden parts from the overall ICL description requires some manual editing or simple scripting. The proposed solution has basically no impact on the IJTAG network area when not considering the special hardware for the protection of the restricted JTAG instruction. The area of this protection hardware depends on the protection mechanism, but it is independent from the size of the protected IJTAG network as the complete private JTAG instruction is secured.

The split of the IJTAG network into two parts can have some impact on the test time. Instead of reconfiguring one SIB it is required to change the JTAG instruction to access the different IJTAG network parts. The test time increase is highly design

specific and depends on the required interaction between the two network branches. Assuming that both IJTAG networks themselves are optimized for performance, the overall test time increase can be considered as low as long as there is only limited interaction required between the two network parts. For the analyzed test case the split of the IJTAG network into two parts resulted in 268 additional test cycles compared to the reference implementation for a scenario of configuring and executing two different LBIST runs for each of the four CPU cores. This runtime increase is negligible compared to the overall test execution time. Of course this number is not really representative as it highly depends on the used instruments and access scenarios.

3.4 Summary

In this section BASTION contributions in the field of analysis and optimization of different 1687 network topologies with respect to OAT and robustness are discussed. It has been shown that the overall access time varies significantly depending on the applied topology while applying different application scenarios. Next the impact of race hazards in 1687 networks has been investigated. Furthermore, this section provides a discussion how 1687 networks could be extended in order to efficiently provide fault management capabilities such as emergency signaling for online-fault detection or the fault diagnosis capabilities during runtime. Finally, safety and security aspects in 1687 networks are discussed and an application example is described.

4 Verification and Validation of IEEE 1687 Networks

This section provides an overview of BASTION contributions in the area of verification and validation of IEEE 1687 networks. First, the topic is introduced and the state-of-the-art is discussed. Next the modelling of IEEE 1687 networks for the verification and validation is described in detail. In the next part of the section specific aspects of the verification and the validation of 1687 networks are discussed and their application are discussed, such as ICL compliance, smarter and optimal retargeting as well as the co-verification of ICL and PDL. Finally a new retargeting approach is described which guarantees minimal access time for a set of re-targeted PDL-instructions.

4.1 Introduction

A scan network typically consists of a multiple scan segments (also referred as test data registers – TDRs or scan registers) where each TDR is used to communicate with one or more embedded instruments. In the most cases, scan segments are not directly connected with each other. Instead of this, a typical scan network implements some kind of multiplexing logic that enables selection of only part of the segments at a time. Complex reconfigurable scan architectures capable for accommodation of hundreds of instruments normally include multiple levels of multiplexing logic that results in a large variety of different possible network configurations. The particular configuration is determined by the current state of multiplexers that are controlled via internal control signals. A control signal in its turn may be defined as a result of a logic operation on current state of scan segments of the network. Within a hierarchical scan-network, establishing of particular configuration turn out to be a complicated procedure because some of control signals may depend on configuration of scan segments located in different levels of hierarchy. As a consequence, the resulting scan-network may have very complex structure which verification and validation is problematic.

A single scan segment of the network normally consists of two parts: one is used for scanning through the test data during the shift operation (shift part), while the second part (update part) holds the previously scanned data (similar to data registers defined in IEEE 1149.1 standard). Two scan segments can either be directly connected into one scan-path or a multiplexer can be used to allow individual access to one of the segments. A single scan configuration defines the state of all multiplexers controlling the concatenation of scan-segments. There is also exists a default scan configuration that determines the initial state of scan-network after reset. An active scan-path is a number of scan-segments consecutively concatenated together by current scan configuration, starting from primary scan input (SI) and ending at primary scan output (SO). The standard enforces that the active scan-path should always be determinable and should not contain loops.

The access to the scan network (reading or writing a particular TDR) comprises of the following stages: (1) setting up the corresponded scan configuration that will place the scan segment of interest into the active scan-path and (2) performing capture-shift-update operation to carry out data exchange with this scan-register. The first step may

require sequence of capture-shift-update operations in case if inclusion of scan register into active scan path demands to change values of number of internal control. The overall performance of the target network is determined by the average number of test clocks needed to execute read or write operation on the network's scan registers.

4.2 Related Work

Efficient verification and optimization of reconfigurable IEEE 1687 networks is one of the key necessities to enable broad application of IEEE 1687. State-of-the-art approaches offer solutions to solve verification and retargeting challenges. In [14] the authors proposed a first approach to model 1687 networks in order to proof specific properties of the network, for example reachability of specific instruments, and to retarget PDL commands. In [15] the authors extend the earlier proposed method by pseudo-Boolean optimization in order to minimize the number of scan access operations and hence the time to perform PDL commands on the 1687 network.

4.3 BASTION Contributions

In this section, the contributions of the BASTION project regarding verification and validation of IEEE 1687 networks are presented: Section 4.3.1 presents an approach to model IEEE 1687 networks for verification and validation. The following section proposes a detailed concept for the ICL compliance and Silicon validation. The Sections 4.3.3 and 4.3.4 discuss aspects of smarter retargeting in IEEE 1687 networks and challenges on the co-verification of ICL and PDL, respectively.
Finally Section 4.3.5 presents a novel approach for optimal retargeting in generalized IEEE 1687 networks.

4.3.1 Modeling of IEEE 1687 Networks for Verification and Validation

In order to effectively represent complex scan structures as well as to discard implementation-specific details not needed for verification and validation tasks we describe IEEE 1687 networks using formal graph-based model. For that, the source description of scan network (in ICL language) has to be converted to abstract graph model by using special algorithm described below. This model is thereafter used to carry out verification and validation tasks. In the following paragraphs the description of a graph model of IEEE1687 networks is presented.

We model the structure of a scan network with a partially ordered directed acyclic graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. The set of vertices V consists of scan segment nodes V_S , auxiliary nodes V_A and two special nodes v_{SI} and v_{SO}. The set V_S embraces all scan cells in the network, while elements of V_S are addressable objects of the scan chain such as scan registers or SIB registers consisting of one or many scan cells each. Essentially every node $v_i \in V_s$ has different weight w_i equal to the number of scan cells in that node. Auxiliary nodes V_A are used to model converging/fanout points between two adjacent multiplexers (e.g. node 8 in Figure 19). The weight of any auxiliary node $v_a \in V_A$ is zero. In addition, each graph contains two special auxiliary nodes v_{SI} and v_{SO} that denote primary scan input SI and primary scan output SO.

The set of edges $E \subseteq V \times V$ represents connections between scan network elements such that an edge $e = (v_i, v_j) \in E$ if and only if the two elements corresponding to $v_i, v_j \in V$ are either directly connected together or can be concatenated through a multiplexer to form a single segment of a scan-path. If a node has more than one successor or in other words subsequent elements, each outgoing edge e_k is labeled with a set of activation condition C_k, which also corresponds to the select signal *sel(v)* of the corresponding scan segment, which in its turn is issued in agreement with the current state of the corresponding multiplexer. A particular outgoing edge e_k is activated in the current state of the network if and only if each condition $c_l \in C_k$ is satisfied.

A *scan path* in G is defined as a directed path from the primary scan-in node v_{SI} through zero or more directly connected nodes $v_i \in V$, to the primary scan-out node v_{SO}. A graph node $v \in V_S$ is defined *active* if and only if the select signal of the corresponding scan segment is asserted: *sel(v)* = 1. Otherwise the node is *inactive*, *sel(v)* = 0. An *active scan path* is a scan path in G that contains only active nodes.

An example scan network and the resulting graph are shown in Figure 17 and Figure 19 correspondingly. They illustrate the main ideas of constructing the graph from a scan network. Scan registers are shown as rectangles in Figure 17 and labelled as TDR or SIB. SIB is a special kind of scan register that contains an integrated multiplexer. Standalone multiplexers are shown as trapezoids labeled with M.

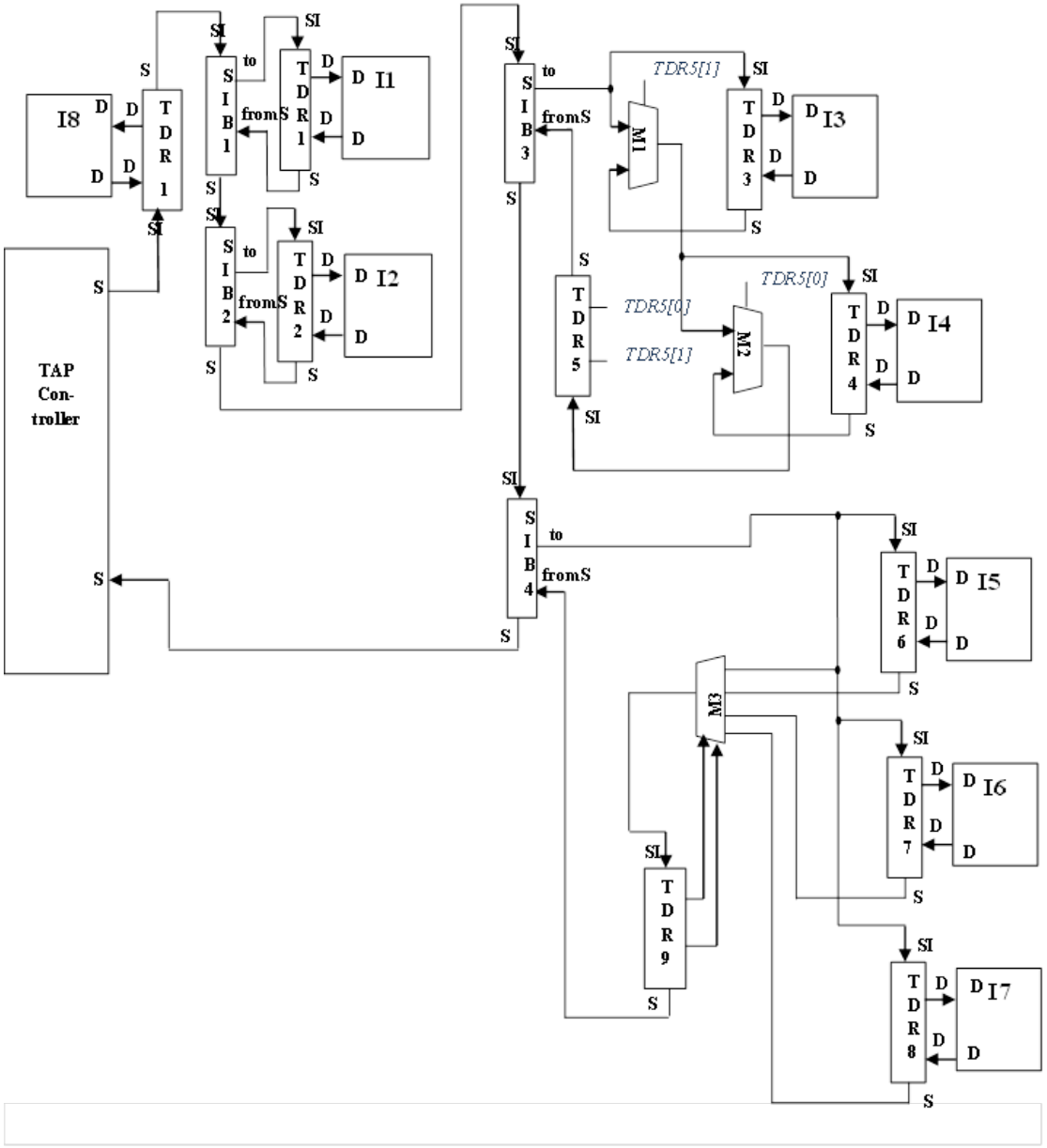

Figure 17. Example scan-network with SIBs and scan multiplexers

```
Module sib {
    ScanInPort SI;
        ScanInPort   fromSO;
        ScanOutPort SO { Source Mux; }
        ScanOutPort   toSI { Source SR; }

        ScanRegister SR { ScanInSource SI; CaptureSource SR; ResetValue 1'b0; }
        ScanMux Mux SelectBy SR { 1'b0: SR;       1'b1: fromSO; }
}

Module tdr {
    ScanInPort SI;
        ScanOutPort SO { Source SR[0]}
        ScanRegister SR[L:0] { ScanInSource SI; CaptureSource DI; ResetValue 1'b0; }
}

Module top {
    ScanInPort SI;
        ScanOutPort SO { Source SIB4.SO }

        Instance TDR10 of tdr { InputPort SI = SI; InputPort DI = I8.DO; }
        Instance TDR1 of tdr { InputPort SI = SIB1.toSI; InputPort DI = I1.DO; }
        Instance TDR2 of tdr { InputPort SI = SIB2.toSI; InputPort DI = I2.DO; }
        Instance TDR3 of tdr { InputPort SI = SIB3.toSI; InputPort DI = I3.DO; }
        Instance TDR4 of tdr { InputPort SI = M1; InputPort DI = I4.DO; }
        Instance TDR6 of tdr { InputPort SI = SIB4.toSI; InputPort DI = I5.DO; }
        Instance TDR7 of tdr { InputPort SI = SIB4.toSI; InputPort DI = I6.DO; }
        Instance TDR8 of tdr { InputPort SI = SIB4.toSI; InputPort DI = I7.DO; }

        Instance SIB1 of sib { InputPort SI = TDR10.SO; InputPort fromSO = TDR1.SO; }
        Instance SIB2 of sib { InputPort SI = SIB1.SO; InputPort fromSO = TDR2.SO; }
        Instance SIB3 of sib { InputPort SI = SIB2.SO; InputPort fromSO = TDR5[0]; }
        Instance SIB4 of sib { InputPort SI = SIB3.SO; InputPort fromSO = TDR9[0]; }

        ScanMux M1 SelectBy TDR5[1] {1'b0 : SIB3.toSI; 1'b1: TDR3.SO; }
        ScanMux M2 SelectBy TDR5[0] {1'b0 :M1; 1'b1: TDR4.SO; }
        ScanMux M3 SelectBy TDR9[1:0] {    2'b00 :SIB4.toSI;
                                    2'b01: TDR6.SO;
                                    2'b10: TDR7.SO;
                                    2'b11: TDR8.SO; }

        Instance I1 of Instrument {InputPort DI = TDR1.DO; }
        Instance I2 of Instrument {InputPort DI = TDR2.DO; }
        Instance I3 of Instrument {InputPort DI = TDR3.DO; }
        Instance I4 of Instrument {InputPort DI = TDR4.DO; }
        Instance I5 of Instrument {InputPort DI = TDR6.DO; }
        Instance I6 of Instrument {InputPort DI = TDR7.DO; }
        Instance I7 of Instrument {InputPort DI = TDR8.DO; }
        Instance I8 of Instrument {InputPort DI = TDR10.DO; }

        ScanRegister TDR5[1:0] {ScanInSource M2; CaptureSource 2'b00; ResetValue 2'b00; }
        ScanRegister TDR9[1:0] {ScanInSource M3; CaptureSource 2'b00; ResetValue 2'b00; }
}
```

Figure 18. Formal description of example scan-network in IEEE1687 ICL language

The corresponded ICL description of this scan network is presented in Figure 18. To make our example simpler we intentionally removed some of ICL constructs that are

required by the standard but are not directly related to the description of example scan network structure.

The structure of the resulting graph in Figure 19 directly corresponds to the structure of the initial scan network. The nodes $v_i \in V_S$ in the graph are labeled in accordance with the scan register labels (SIB or TDR in Figure 17). Auxiliary nodes $v_a \in V_A$ are labeled by the name of the corresponded multiplexer to which the signals from the

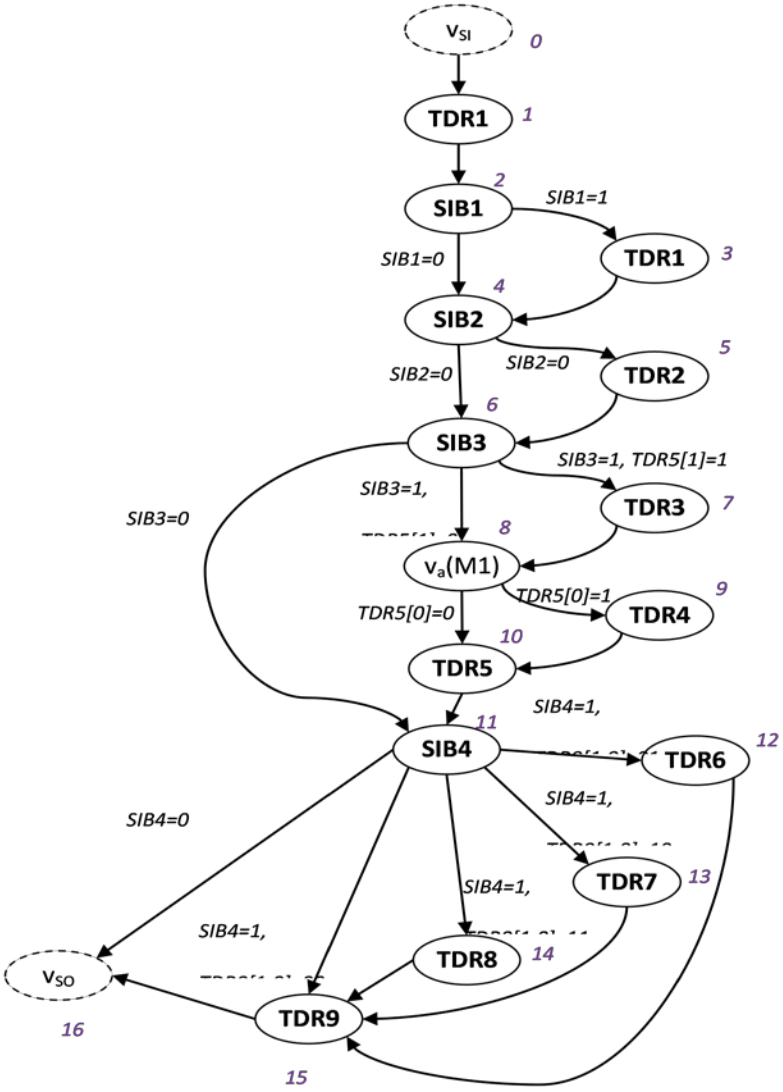

Figure 19. Graph representation of example scan-network

preceding nodes are connected. Multiplexers (including those inside the SIBs) define edges e_i of the graph and corresponding sets of activation conditions C_i. In case of SIBs, there are exactly two successors and edges with complementary activation conditions defined by a one-bit shift register inside the SIB. In case of multiplexers, the number of outgoing edges is defined by the multiplexer width. The activation conditions, in this case, could be any arbitrary collection of data bits of any scan register (depending on the scan network design).

It is important to mention that nodes in graph are partially ordered to improve the efficiency of the graph traversal algorithm. The order of edges is implied by the topological ordering of nodes in the graph. In the graphical representation the leftmost successor of a node always has highest rank than the ranks of other successor nodes.

In the context of scan-network, the ordering is related to the fact that scan multiplexers usually contain one or more register loops and one empty (direct) path. Thus the empty path is always placed first and shown as the leftmost edge in the graph. In rare occasions all branches of a multiplexer could be connected to a scan register (not a typical configuration of an IEEE1687 scan-path). Then either no edge is getting a special priority or the edge leading to the node with lowest weight (shortest register) is put first. The sorting helps to identify the shortest active scan path in the graph for each scan segment and, hence, to speed up the validation algorithms. Graph sorting is also the main differentiator of our approach compared to the one reported in [5].

The second important aspect is the way of introducing auxiliary nodes. Such nodes are needed to mark sections in the graph where two or more adjacent multiplexers are serially connected. In order to be able to distinguish them in the graph, an empty (zero-weight) auxiliary node is added at every such connecting point. In Figure 17, multiplexers M1 and M2 illustrate this case. Both multiplexers are controlled from TDR5 (by two separate bits). In order to reflect the structural composition of the scan network (two multiplexers are actually two consequent components on the scan path, and not a single cluster based on larger multiplexer), an empty auxiliary node 8 is inserted (see Figure 19).

Next we will discuss the algorithm, shown in Figure 21, for construction of graph model out of scan-network description. As the source for graph construction we use a description of scan-network in ICL format (part of IEEE1687 standard) which is at first being preprocessed (flattened) to obtain a set of atomic elements of scan-network:

$SN = SR \cup SM \cup SO$. Here, SR is a set of scan registers (correspond to ScanRegister statement in ICL description), SM is a set of multiplexers (correspond to ScanMux in ICL) and SO denotes primary output.
For each $sr_i \in SR$ we keep the following properties:

- Name of scan-register
- *ScanIn* – Refers to a name of an atomic element *el* of scan network *SN*, $\in SR \cup SM \cup SI$. Scan output of *el* is connected to serial scan input of sr_i.
- *value* – a value which is captured into scan register during capture operation. This property is optional and because the captured value may be not defined. By default, it is equal to the reset value.
- *StartBit, EndBit* – refer to the first and the last flip-flop of the register in the scan chain; register length can be derived from these values, as well as register direction (i.e. big endian vs. little endian).

For each $sm_j \in SM$ we keep the following properties:

- *SelectBy* – Denotes a controlling signal sc_j that controls this multiplexer

- A set consisting of pairs $(sv_s \rightarrow ss_s)$ where ss_s refers to the name of an atomic element of scan network SN, $el \in SR \cup SM$ whose scan output is connected to one of the inputs of the multiplexer sm_j and sv_s defines the state of sc_j under which this input becomes active.

SO refers to a name of an atomic element of scan network SN whose scan output is directly connected to primary scan output of the network.

```
ScanRegister     top.TDR10.SR (ScanIn: top.SI)
ScanRegister     top.TDR1.SR (ScanIn: top.SIB1.SR)
ScanRegister     top.TDR2.SR (ScanIn: top.SIB2.SR)
ScanRegister     top.TDR3.SR (ScanIn: top.SIB3.SR)
ScanRegister     top.TDR4.SR (ScanIn: top.M1)
ScanRegister     top.TDR6.SR (ScanIn: top.SIB4.SR)
ScanRegister     top.TDR7.SR (ScanIn: top.SIB4.SR)
ScanRegister     top.TDR8.SR (ScanIn: top.SIB4.SR)

ScanRegister     top.SIB1.SR (ScanIn: top.TDR10.SR)
ScanMux     top.SIB1.Mux (SelectBy: top.SIB1.SR {0 → top.SIB1.SR, 1 → top.TDR1.SR})

ScanRegister     top.SIB2.SR (ScanIn: top.SIB1.Mux)
ScanMux     top.SIB1.Mux(SelectBy: top.SIB2.SR {0 → top.SIB2.SR, 1 → top.TDR2.SR})

ScanRegister     top.SIB3.SR (ScanIn: top.SIB2.Mux)
ScanMux     top.SIB3.Mux(SelectBy: top.SIB3.SR {0 → top.SIB3.SR, 1 → top.TDR5})

ScanRegister     top.SIB4.SR (ScanIn: top.SIB3.SIBMux)
ScanMux     top.SIB4.Mux(SelectBy: top.SIB3.SR {0 → top.SIB4.SR, 1 → top.TDR9})

ScanMux     top.M1 (SelectBy: TDR5[1] {0 → top.SIB3.SR, 1 → top.TDR3.SR})
ScanMux     top.M2 (SelectBy: TDR5[0]  {0 → top.M1, 1 → top.TDR4.SR})
ScanMux     top.M3(SelectBy: TDR9[1:0]
    {00 → top.SIB4.SR, 01 → top.TDR6.SR, 10 → top.TDR7.SR, 11 → top.TDR8.SR})
ScanRegister     top.TDR5 (ScanIn: top.M2)
ScanRegister     top.TDR9 (ScanIn: top.M3)
SO (SIB4.Mux)
```

Figure 20 Intermediate description of flattenned scan-network

The flattening into a set of atomic elements is performed by using recursive inspection of ICL description starting from top module. On each step, the algorithm processes contents of a module and extracts ScanRegister and ScanMux elements. The ScanIn or SelectBy property of the extracted element is then assigned with the corresponding reference of other element of the network with respect to hierarchy. In case if currently processed module contains instances of other modules, the algorithm descents into processing of the nested modules recursively. The resulting intermediate description of our example network in a flattened form is presented in Figure 20.

This intermediate description of scan-network is then used by the graph construction algorithm which is shown in Figure 21.

// Add primary nodes
Foreach ScanRegister $sr_i \in SR$ **do**
    Add primary node v_i representing sr_i to set of nodes V of G
    Assign the respective label l_i and weight w_i to v_i
End foreach
// Add auxiliary nodes
Add node v_{SI} and v_{SO} to G that represent primary scan input and output
Foreach ScanMux $sm_j \in SM$ **do**
    Foreach multiplexed input ss_k in a set of conditional inputs of sm_j **do**
        If ss_k refers to another ScanMux $sm_l \in SM$ **then**
            Add auxiliary node v_a to set of nodes V of G
            Assign respective label $l_a = ss_k$ and weight $w_a = 0$ to v_a
        End if
    End foreach
End foreach
// Add edges and conditions
Foreach element $sr_i \in SR$ and SO **do**
    If *ScanIn* property of sr_i or SO references to $v_j \in V$
        Add edge $e = (v_j, v_i)$ or $e = (v_{SO}, v_i)$ with empty condition set $C = \{\emptyset\}$ to G
    Else If *ScanIn* property of sr_ireferences *ScanMux* $sm_j \in SM$ **then**
        Foreach multiplexed input ss_k of scan multiplexer sm_j **do**
            Select node with $v \in V$which has label $l = ss_k$
            Add edge $e = (v, v_i)$
            Add condition $c = (sc_j, sv_k)$ to edge e,
            where sc_j is a signal referenced by *SelectBy* property of ScanMux
                and sv_k is value on this signal that makes k-th input be active
        End foreach
    End if
End foreach
// Sort nodes of graph
Assign order to nodes $v \in V$ using topological graph sorting
// Propagate conditions to nodes matching to fan-out points
Foreach $v_i \in V$ starting from node of the highest rank **do**
    Define set S that contains edges outgoing from node v_i
    Define set L that contains all conditions of edges $e \in S$
    Foreach condition $c_k \in L$ **do**
        If all edges $e \in S$ contain condition c_k **then**
            Remove condition c_k from all edges $e \in S$
        End if
        Foreach edge $e_l = (v, v_i)$ **do**
            Add condition c_k to set of conditions of edge e_l
        End foreach
End foreach

Figure 21. Graph construction algorithm

4.3.2 ICL Compliance, Verification and Silicon Validation

The main objective of IJTAG verification and silicon validation is to prove the correctness of initial specification (i.e. description of IJTAG network in ICL language) and to validate silicon implementation against this specification.

The task of ICL verification begins with simple syntax and semantics checks for source ICL description and is followed by the examination of its compliance to the rules of IEEE 1687 standard. In addition, design rule violation checking (DRC) can be carried out in order to determine if the implementation of the described network will meet the restrictions set for particular design. For instance, design rules may restrict the maximal length of scan-path in any of network's states or put limits on depth of multiplexing logic.

Silicon validation is also performed at different levels. On the lowest level we only validate infrastructure of IJTAG network to ensure that every instrument is accessible. The next level already involves validation of instruments functionality that implies sending of commands to and receiving responses from an instrument. For this type of validation, a PDL description that contains command sequences for communication with the particular instrument is also required. Finally, we plan to develop techniques that would attempt to inspect IEEE1687 network for presence of un-documented or specially hidden structures.

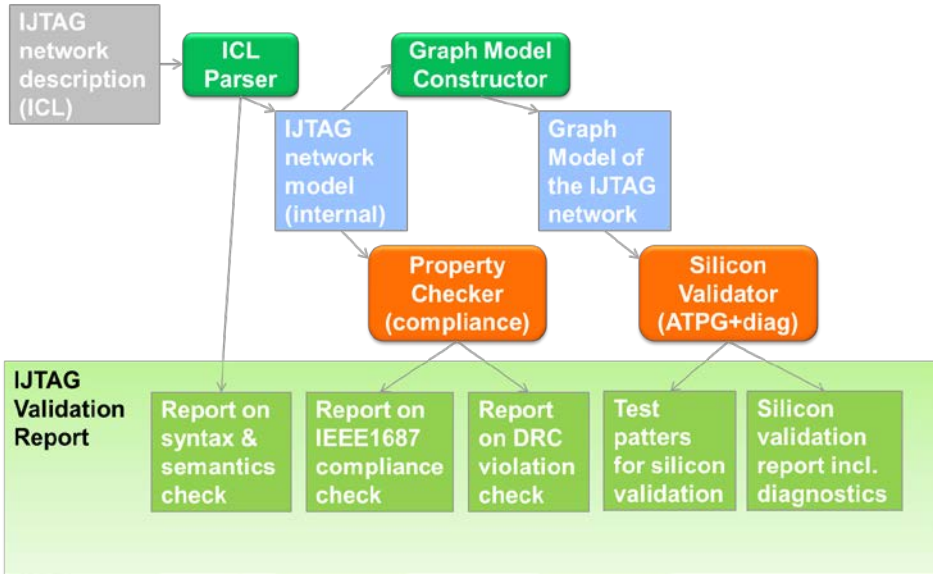

Figure 22. Overall scheme of validation of IEEE 1687 scan-network

Details about the level of silicon validation are provided below:

Level 0: ICL-based chain-length analysis (instrument access and conflict check)
Purpose: *simple check of infrastructure (similar to Boundary Scan infrastructure test) for ICL or standard compliance (check that infrastructure is usable).*

Requirements/methods: *a) check that all TDRs (instruments) are accessible and b) have correct length; c) check for overall chain length violations; d) check capture values.*

Level 1: instrument correctness check (combined ICL + PDL)
Purpose: *a) correct reaction of instruments on PDL-defined actions; b) distinguish between equivalent transformations (e.g. detect swap of TDRs of the same length).*

Requirements/methods: *a) execute PDLs and check results; b) generate additional test sequences (PDLs) in case the transformations/instruments remained indistinguishable (improve diagnostic resolution if the test fails);*

Level 2: phantom detection (presence of undocumented structures)
Purpose: *a) detect hidden features – relatively easy; b) detect hostile features (Trojan horses) – extremely hard.*

Requirements/methods: *a) undocumented features could be found by toggling bits in the scan chain and checking if the length changes unexpectedly; b)Trojan detection seems to be impossible from the first impression;*

4.3.2.1 Level-0 Silicon Validation

The ultimate goal of level-0 silicon validation is to perform a simple check of IJTAG infrastructure, ensure that all the segments of on-chip scan network can be accessed. The validation procedure consequently places target scan network into different possible states and then verifies that the actual length of entire scan-path in the particular state matches to the expected (specified by ICL description). The actual length of scan-path can be validated by shifting an arbitrary sequence of bits (signature) into the primary input of scan-network and then expecting the same sequence to appear at primary output. The length of active scan-path is then determined by number of test clock cycles required for this sequence to pass through the scan network. At the same time, level-0 validation test verifies that the capture values for each of scan-register (optionally specified in ICL) match to actually captured data.

Below we will describe the process of generation test patterns for level-0 silicon validation which effectively corresponds to the traversal of graph model of scan-network that was presented above. The aim of the graph traversal algorithm is to consequently access all scan registers using the shortest possible active scan path that includes the currently accessed register. As a side effect, such an algorithm allows to evaluate minimum access time needed to process one register in the network and

hence calculate average minimum processing time for the diagnostic network infrastructure as a whole.
The same graph traversal principle leads to subsequent activation of all branches in every scan multiplexer in the network, hence offering a good validation strategy of implemented scan network against the initial specification or a structural model.
The partially ordered edges in the graph enable an efficient implementation of such an algorithm as a recursive process. The algorithm starts with building the minimal (shortest) path through the graph by choosing the leftmost edge in every node that has several successors. Then, starting from the primary scan output node SO one step back on the path is taken (step back in recursion). In the current node, until there are more non-traversed edges left, a new successor is taken, from which the shortest path to SO is constructed. Otherwise, if all edges of the current node have been traversed, another step back is taken. The algorithm continues when reaching back the primary scan input node SI and traversing all its edges if any. A detailed graph traversal algorithm is presented in Figure 23.

After finishing the traversal, the resulting set S will contain all the traversed paths $P_i \in S$ where each path P_i contains one or more traversed nodes $v_j \in P_i$. The condition for activating a particular scan-path P_i of scan-network is obtained by Boolean conjunction over activation conditions of all nodes v_j belonging to the corresponding path P_i. The resulting activation condition can then be directly converted to a shift sequence (test pattern) that has to be applied to primary input *SI*.

After finishing the traversal, the resulting set S will contain all the traversed paths $P_i \in S$ where each path P_i contains one or more traversed nodes $v_j \in P_i$. The condition for activating a particular scan-path P_i of scan-network is obtained by Boolean conjunction over activation conditions of all nodes v_j belonging to the corresponding path P_i. If resulting activation conditions can be activated on the current path then they are directly converted to shift sequences (test patterns) that have to be applied to

```
                // P′ will contain nodes belonging to the shortest possible path in G
P′ = {v_SI}
Until not v_SO ∈ P′ do
    Select node v_i ∈ P′ with maximum i   // Node with the highest rank
    Select maximum j so that ∃e = (v_i, v_j) ∈ E    // Go along the leftmost edge
    P′ = P′ ∪ {v_j}
End until

// Set S will contains all the traversed paths
S = {P′}              // Add the shortest path
A = P′                    // Set A contains nodes traversed so far
Foreach v_i ∈ P′ do
    Call RecursiveProcessing(i)
End foreach

RecursiveProcessing(i)
Begin
    Select maximum k so that ∃e = (v_j, v_k) ∈ E       // Leftmost edge
    Foreach j so that ∃e = (v_i, v_j) ∈ E and i ≠ k and not v_j ∈ A do
        P = P′ ∪ {v_j}

        // Traverse shortest way form this node back to the path P′
        Until not v_j ∈ P′ do
            Call RecursiveProcessing(j)
            // Go along the leftmost edge
            Select maximum k so that ∃e = (v_j, v_k) ∈ E
            P = P ∪ {v_k} , j = k
        End until

        // Add newly traversed path to set of paths
        S = S ∪ {P}
        A = A ∪ P       // Add newly traversed nodes
    End foreach
End
```

Figure 23. Graph traversal algorithm

primary input *SI*, otherwise a sequence of paths is found that enables a step by step activation of required constraints.

Level-0 validation is then facilitated by comparison of actual length of every path P_i(determined by observation on SO) against to the sum of weights of nodes $v_j \in P_i$ (which in its turn corresponds to the length of this scan-path specified in source ICL).

4.3.3 Smarter Retargeting

The flexibility that IEEE 1687 brings to scan-path allows for description of complicated scan networks such as the one shown in Figure 25. In the shown network, the shift-and-update register of the sensor instrument is accessible through two different segments of the scan-path: a path through the hierarchical port of SIB_1, and a path through the hierarchical port of SIB_2. The *OneHotScanGroup* and *LogicSignal* components are building blocks in ICL. OneHotScanGroup is used to describe a mux to which only one input is active (enabled) at a time. LogicSignal can be used to describe operation of logic gates.
The SCB allows for exclusive access to either of the SIBs. However, it is possible to create a configuration in which both SIBs contain a logic value '1'—a situation in which both inputs to OneHotScanGroup are active at the same time and can disrupt the correct operation of the network due to having multiple drivers for the shift-register. Such a configuration can be created by first accessing one SIB, enabling its hierarchical port by writing logic value '1' into the SIB, and then accessing the other SIB and activating its hierarchical port.

From the above, it can be seen that for the shown network the retargeting tool faces these challenges:

- How to choose the path to the sensor's shift register from any possible starting configuration
- How to recognize invalid configurations (such as having both SIBs open at the same time)

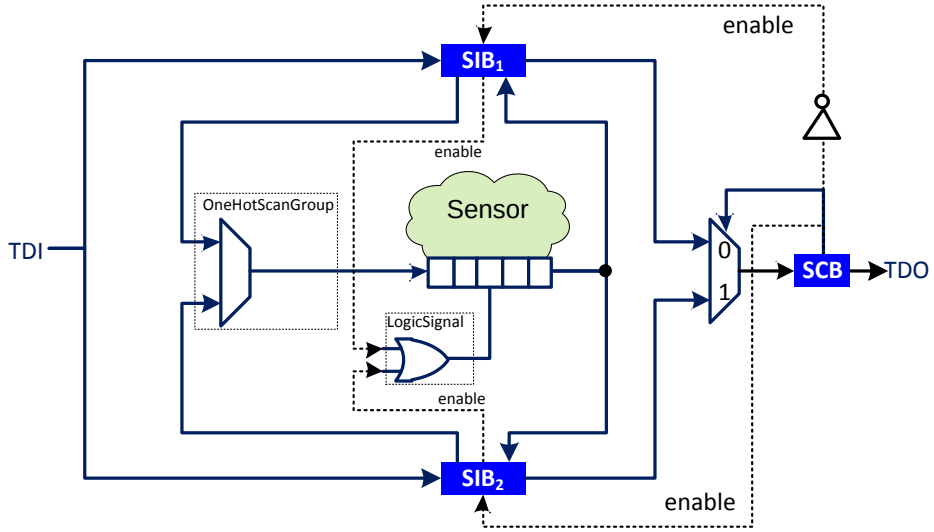

Figure 24. 1687 network in which there exist two paths to the sensor instrument

4.3.4 Co-verification of ICL and PDL

An essential part of the retargeting process is making sure that the description of the 1687 network in ICL is correct w.r.t. the connectivity of instruments (including both control signals and data-path). For example, the tool should verify that all instruments on the scan-path are accessible. That is, for each required combination of instruments, there is a reachable configuration that puts those instruments on the scan-path at the same time [14]. Moreover, the tool should also check for invalid configurations. For example, in the network shown in Figure 26, if both SIBs are programmed to be open at the same time, the resulting scan-path will contain a loop (which is an invalid configuration). This invalid configuration, however, might not be reachable when taking the PDL into account. Therefore, to verify the correctness of a network, the tool can perform a co-verification of a given set of ICL and PDL files.

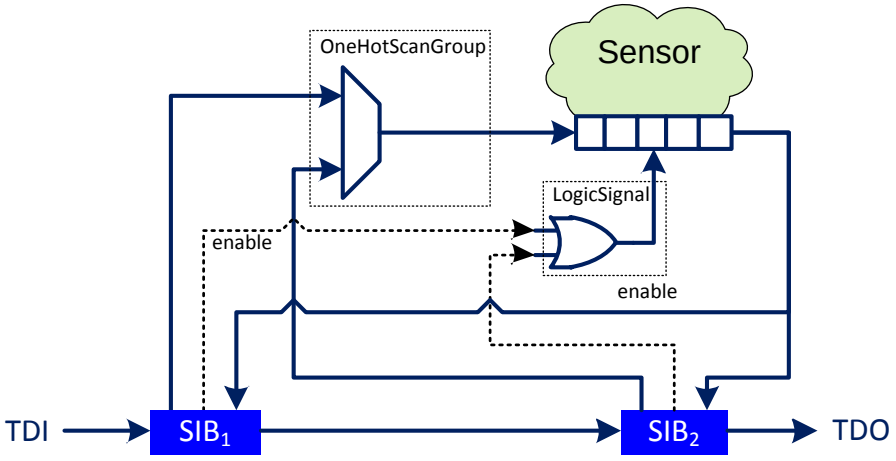

Figure 25. 1687 network in which the existing loop can be avoided through PDL

4.3.5 Access Time Minimization in IEEE 1687 Networks

As described above IEEE 1687 enables flexible access to the embedded(on-chip) instruments that are needed for post-silicon validation, debugging, wafer sort, package test, burn-in, printed circuit board bring-up, printed circuit board assembly manufacturing test, power-on self-test, and in-field test. At any of these scenarios, the instruments are accessed differently, and at a given scenario the instruments are accessed differently over time. It means the IEEE 1687 network needs to be frequently reconfigured from
accessing one set of instruments to accessing a different set of instruments. Due to the need of frequent reconfiguration of the IEEE 1687 network it is important to (1) minimize the runtime for the algorithm finding the new reconfiguration, and (2)
generate scan vectors with minimized access time. In this part of the report we describe a model the reconfiguration problem using Boolean Satisfiability of Problem (SAT). Compared to previous works we show significant reduction in run-time and we ensure minimal access time for the generated scan vectors for generalized reconfigurable scan networks.

4.3.5.1 Previous Work

In this part, the relevant hardware features of IEEE 1687 will be discussed, and a retargeting concept will be explained. Moreover, as we have modeled the IEEE 1687 retargeting as Boolean satisfiability problem (SAT), subsection C will give a brief introduction into SAT.

A. Instrument Access Infrastructure (Network)

A strong feature in IEEE 1687 networks is the possibility of dynamic reconfiguration, which allows for reduction of instrument access time by varying the length of the scan-path to include only those instruments in the path which are needed for current session. To enable variable-length scan-paths in IEEE 1687 networks, a ScanMux control bit is used, which is a shift-update register that can be placed anywhere on the scan-path to configure one or more scan multiplexers (ScanMux components). Figure 27(a) shows ScanMux control bits C_1 and C_2 used to configure a network of two instruments. To program the control bits to any desired configuration, the right values should be placed in their shift cells (denoted by S) during the Shift phase, and copied to their parallel latch (denoted by U) during the Update phase. We will use the symbol in Figure 27(b) to represent a ScanMux control bit in the rest of this paper.

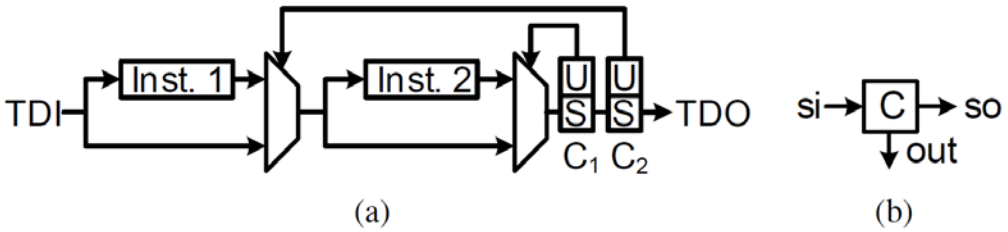

Figure 26. ScanMux control bit: (a) a network of two instruments, configured by ScanMux control bits C1 and C2, (b) corresponding symbol

To access the network of instruments from the chip boundary, IEEE 1687 specifies the JTAG TAP as the primary inter-face. Interfacing is performed by connecting the first level of the IEEE 1687 network as a custom TDR to the JTAG circuitry. Since the JTAG TAP FSM is primarily used to operate IEEE 1687 networks, performing each cycle of network configuration involves going through the capture, shift, and update states in the FSM, which is referred to as a CSU cycle [16]. As an example, Figure 28 illustrates a small IEEE 1687 network consisting of three instruments (namely a DFT instrument in the first hierarchical level, and a sensor and a debugging feature in the second hierarchical level) and six ScanMux control bits (C_1–C_6). The instruments are interfaced to the scan-path through shift-registers with parallel I/O. To access the instruments, ScanMux control bits should be programmed to include the required shift-registers into the scan-path. For example, to access only the DFT feature, C_1 and C_2 should be set to logic value "1", and C_3 should be set to "0" to bypass (via input 0 of mux M3) the network segment containing the Sensor and Debug instruments, as well as C_4, C_5, and C_6 components.
It can be immediately noticed from the network in Figure 28 that reconfiguring the network to the desired configuration might need several CSU cycles (CSUs). For example, assuming an initial configuration of $C_1 = \ldots = C_6 = 0$, accessing the Debug instrument needs two cycles of shift and update. In the first cycle, only C_1, C_2, and C_3 are accessible and by setting $C_2 = 0$ and $C_1 = C_3 = 1$, C_4, C_5, and C_6 become accessible. It is in the second cycle when C_4, C_5, and C_6 can be configured to the right values, i.e., $C_5 = 0$ and $C_4 = C_6 = 1$, so that the Debug instrument becomes accessible.

B. Description Languages and Retargeting

IEEE 1687 introduces two description languages, namely Instrument Connectivity Language (ICL) and Procedural Description Language (PDL). ICL is used to describe the network, that is, how the instruments are connected to the JTAG TAP. PDL is used to describe the operation of instruments at their terminals. PDL commands allow to perform read/write operations on the instrument shift-registers and configurable components, as well as to wait for an instrument (such as a BIST engine) to finish its operation.
Given the PDL of each instrument, a retargeting tool generates scan vectors to configure the network and transport the required data bits from the JTAG TAP to/from the instruments' shift-registers. A retargeting tool relieves the designer from dealing with network configuration (i.e., writing the PDL to configure ScanMux Control bits directly). For example, assuming that the goal is to read the value from the sensor instrument in Figure 28 the PDL developer might simply use a write command to

activate the sensor, a wait command to wait for the sensor to capture the value, and a read command to read the captured value out. It is then the task of the retargeting tool to generate one scan vector to configure C_1, C_2, and C_3, one vector to configure C_4, C_5, and C_6, one vector to write to the enable bit in the sensor's shift-register, a wait cycle of enough length, and finally one vector to scan the captured value out.
In its basic form, a PDL script is a sequence of iApply groups. In each iApply group, there are a number of read and write operations to the registers in the network (setup commands). These read/write operations take effect upon encountering an iApply command (an action command). A retargeting step will then be to generate a number of scan vectors to (1) change the configuration of the network (from its current state) to a configuration in which the specified registers are accessible, and (2) to perform the read/write operation. These vectors are then applied to the network through a number of CSUs. A complete retargeting flow will then be a number of such retargeting steps.

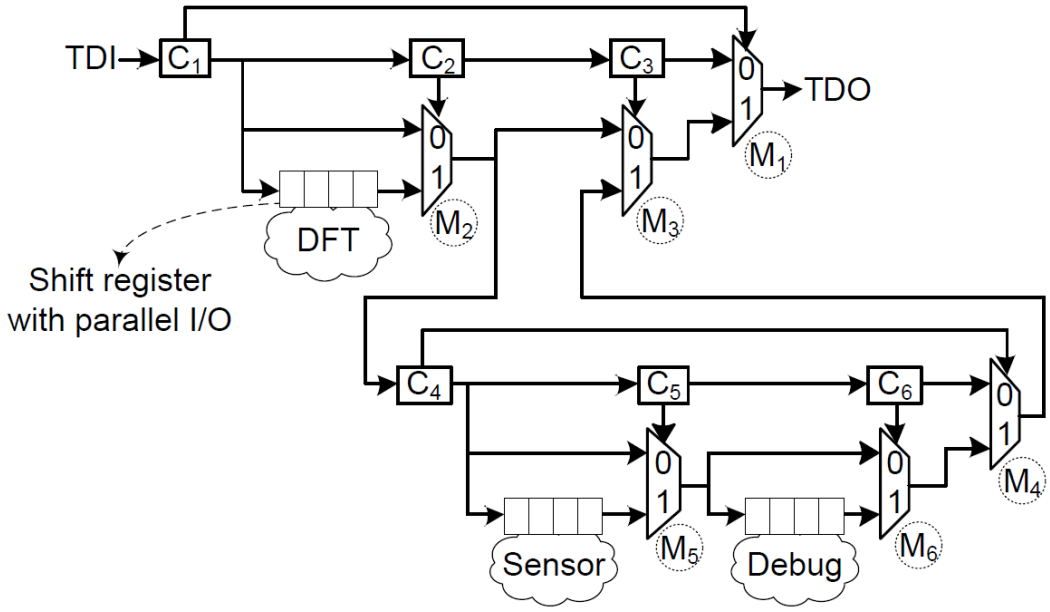

Figure 27. An IEEE 1687 network with three instruments inside a chip

For complex IEEE 1687 networks and especially for long PDL scripts, it becomes desirable to both speed up the retargeting process and to generate effective scan vectors which are optimal with regards to the application time. To achieve this goal, a first step would then be to optimize the basic retargeting step for both run-time efficiency and effectiveness of the generated vectors. There have been a number of works addressing retargeting for an IEEE 1687 network [14]- [17]. So far only [14]-[18] have addressed the issues of efficiency and effectiveness in retargeting. What distinguishes [18] from the other works is addressing the efficiency of retargeting when applying interactive PDL (PDL Level-1 which supports programming language constructs such as conditions and loops) with the help of hardware acceleration. Therefore, the only works that have so far addressed efficiency and effectiveness for a basic retargeting step are [14], [15].

C. Boolean Satisfiability Problem (SAT)

The retargeting approach proposed in this work employs a SAT-based reasoning engine [19]. The Boolean satisfiability problem is the problem of finding an assignment satisfying some given Boolean formula. Typically the Boolean formula is expressed in the conjunctive normal form (CNF), denoted Φ, which is the conjunction of a set of clauses C, where each clause $c \in C$ is a disjunction of Boolean literals, $c = l^c_1 \vee \ldots \vee l^c_n$.

The satisfiability problem is among the first problems proven to be NP-hard. Recent developments in the area of SAT, such as conflict-based learning, conflict analysis [19]- [20] as well as powerful preprocessing techniques [21], led to the integration of SAT-solvers in almost all areas of the electronic design automation (EDA) industry [22]- [23].

Further extensions of SAT-based reasoning engines enable its application to multi-valued problems [24] or 0-1-ILP problems [25] [26]. These extensions enabled further applications for example in the area of test set optimization [27] or for an optimized collapsing of cell-aware fault sets [28].

4.3.5.2 New Upper Bound Computation

As discussed above, retargeting is effective when the application of the generated scan vectors results in the least number of test clock cycles, which requires that the whole solution space be explored during retargeting. It was also mentioned that the number of allowed CSU cycles should be chosen such that neither any solution is removed from the solution space (which happens if not enough CSUs are allowed), nor is the run-time efficiency decreased (which happens if too many CSUs are allowed). Therefore, there is a need to find the upper bound on the number of CSU cycles that the search algorithm can use in order to reconfigure the network from any initial state to any target configuration, while considering all the possible solutions and choosing the one that results in the least access time in terms of test clock cycles. In this section, we present an upper bound calculation method for a subset of IEEE 1687 networks described in prior work [15], referred to as MUX-based networks. It should be mentioned that in [15], experiments are performed also on another architecture which is referred to as SIB-based architecture. However, the authors state that for SIB-based architecture, retargeting reduces to a simple decision problem, and therefore, in this work we only focus our attention on the more challenging MUX-based architecture.

Figure 29(a) shows part of an IEEE 1687 network (referred to as MUX-based in [15]), in which M_2 (controlled by C_2) is used to bypass an IEEE 1687 network segment S_1, and M_1 (controlled by C_1) is used to select between C_2 and the mux M_2. The select (enable) and control (capture, shift, update) signals are not shown in the figure. The select signal is used to gate the control signals so that at any time a unique scan-path is activated. In the shown network, the assumption is that the select signal is connected such that only the scan-path connected to the selected input of a mux is activated. For example, if the aim is to activate S_1, both C_1 and C_2 should be set to logic value "1" so that the active scan-path goes from TDI to TDO via S_1, M_2 , and M_1. In the illustrated network, no matter what values C_1 and C_2 are initially set to, any desired configuration of these two components can be achieved within at most two CSU cycles. For example, assume that initially $C_1 = C_2 = 0$ and that the aim is to access S_1. In this case, both C_1 and C_2 are on the active scan-path (C_1 is always on the active scan-path and C_2 is on the active scan-path since $C_1 = 0$) and can be programmed to any desired

configuration value with only one CSU cycle. If, however, initially $C_1 = 1$ and $C_2 = 0$, only C_1 is on the active scan-path (which goes from TDI to TDO via C_1, input 0 of M_2, and input 1 of M_1). Therefore, in order to achieve a configuration in which $C_1 = C_2 = 1$, first a CSU cycle is needed to set $C_1 = 0$ so that the active scan-path goes through C2, and then a second CSU cycle is needed to set $C_1 = C_2 = 1$. It should be noted that the maximum two CSU cycles for configuration of C_1 and C_2 components is independent of the topology of S_1. That is, no matter if S_1 is an instrument shift-register or a network, it is possible to activate S_1 within maximum two CSU cycles.

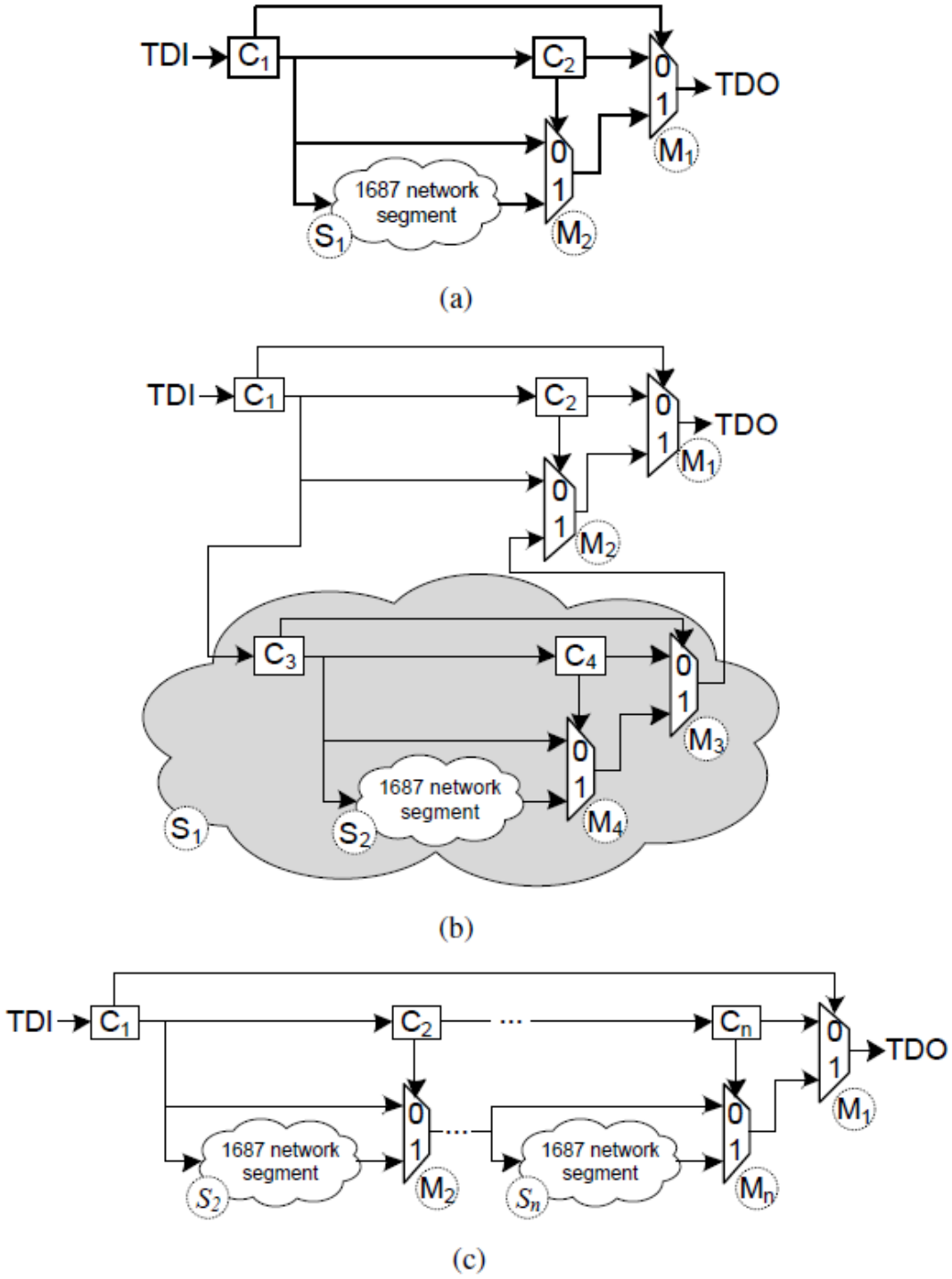

Figure 28. Examples of MUX-based network used for computation of upper bound: (a) a network where the "1687 network segment" (i.e., S_1) can be an instrument shift-register or a network. (b) a network based on (a) where the "1687 network segment" is replaced with entire network in (a). (c) a generic structure in which "1687 network segment" parts can be replaced by the network in (a) to create a hierarchical MUX-based network.

If in Figure 29(a), S_1 is replaced by a network similar to Figure 29(a), the resulting network is as shown in Figure 29(b). In this network, in order to activate S_2, all four ScanMux control bits C_1–C_4 should be set to "1". Similar to the argument in the previous paragraph, it can be argued that no matter what the initial configuration is, it

takes at most four CSU cycles to activate S_2. This can be proven by considering that C_1 is always on the active scan-path. Therefore, if $C_2 = 0$ and is not on the active scan-path (i.e., $C_1 = 1$), it takes at most two CSU cycles to set $C_1 = C_2 = 1$ (by first setting $C_1 = 0$ to access C_2, and then setting $C_1 = C_2 = 1$). Once $C_1 = C_2 = 1$, S_1 is activated and the same argument as the above paragraph can be used to say that it will take at most two extra CSU cycles to activate S_2—hence maximum four CSU cycles. If we again replace S_2 with another instance of the network in Figure 29(a), we will need at most six CSU cycles to activate the "1687 network segment" in the resulting network. Therefore, if we consider each of these replacements of "1687 network segment" with the network in Figure 29(a), as adding another hierarchical level, it can be concluded that for each additional level, two extra CSU cycles are needed.
Finally, Figure 29(c) shows a generic MUX-based IEEE 1687 network in which C_2 to C_n are used to bypass their associated segments (i.e., S_2–S_n) and C_1 is used to select between the scan-path going through C_2–C_n and the path that goes through the muxes M_2–M_n. Since in this case, C_2–C_n receive their select signals from C_1 (meaning that they receive capture, shift, and update signals simultaneously), and since there is no functional correlation between C_2–C_n components, all of them can be configured independently but at the same time. Therefore, the argument for Figure 29(a) applies to this network, too. That is, as C_1 is always on the active scan-path, it is possible to activate any of the segments S_2–S_n from any given initial configuration of the network, with at most two CSU cycles. By the same token, the segments S_2–S_n can also be independently and simultaneously configured. Now, from the argument in the above paragraph, we know that if any of S_2–S_n segments has multiple hierarchical levels, for each additional hierarchical level, two extra cycles are needed for configuration of the corresponding segment. Since we can access all the segments S_2–S_n at the same time, the upper bound on the total number of CSU cycles needed to explore all the possible configurations of the network, is the maximum hierarchical depth found in the generic network shown in Figure 29(c), multiplied by two.

4.3.5.3 Improved Modeling and Optimal Retargeting

In this section we introduce the notion of perfect networks (PNs) within reconfigurable scan networks. Functional and structural properties of PNs are described and examples are provided. Next we discuss how PNs can simplify the modeling of reconfigurable scan networks and in particular IEEE 1687 networks. Finally it is described how the reductions obtained by using PNs together with the observations formulated above can be applied to ease the retargeting process in IEEE 1687 networks in order to enable a minimum with respect to the number of CSUs and the overall number of shift cycles. In contrast to the approach presented in [15] our method ensures optimality with respect to minimal access time for a subset of IEEE 1687 networks (described as MUX-based networks in [15]).

A. Perfect Networks in IEEE 1687 Networks

A typical IEEE 1687 network is shown in Figure 30. The parts of the network labeled PN_1, PN_2, and PN_3 depict a reappearing network structure, which we refer to as a PN. The key properties of such a network structure are that there exists a single test data input, a single test data output and a set of control bits, such that every possible

assignment of these control bits establishes an active path between the test data input and the test data output.
Applying the notation introduced in Subsection 4.3.1 it is possible to apply structural properties to formulate rules defining the data input and data output as well as control inputs of a PN.
Assuming a reconfigurable scan network as described in Subsection 4.3.1 then the data input vertex of a PN is dominated by the data output of this PN with respect to the network output representing TDO. Graph dominators provide information about the origin and the end of re-converging paths in a network. A dominator $u \in V$ of a vertex $v \in V$ with respect to some output vertex $w \in V_P$ is a vertex, which is contained in every path starting from v to w. In other words, all data passing a PNs data input also pass the data output of the PN and vice versa. The first efficient algorithm on finding dominators in large graphs has been presented in [29]. In [30] it has been shown that graph dominators can be found in linear time.

The key properties of a PN can be formulated as follows:

1) the data output vertex of a PN dominates the data input vertex with respect to TDO,
2) every component within a PN is reachable from the PN's data input,
3) from every component within a PN there exists a path towards the PN's data output,
4) all ScanMux control bits controlling some multiplexer within a PN control that multiplexer exclusively, and
5) for every possible assignment of values in the MUX-controlling ScanMux control bits, there exists an active scan-path from the PN's data input.

Considering the components depicted in the PN_1-area of Figure 30 the data output of multiplexer M_1 dominates the data input of PN_1 since every path starting from the PN_1 data input leading to TDO is passing multiplexer M_1. Furthermore the select bit of M_1 is exclusively connected to C_1. Finally since M_1 is a 2-input multiplexer controlled by a single ScanMux control bit, denoted C_1, PN_1 is a perfect network. The rules formulated above are also true for PN_2 and PN_3 in Figure 30.
In the context of modeling a reconfigurable scan network and computing active scan-paths, the modeling of components within a PN is not required to establish an active scan-path passing through this PN since by construction there always exists a path starting from the data input of the PN reaching the data output of the PN, where every component contained in the path is active.
Furthermore it is possible to derive for every PN the minimum number of scan elements on a scan-path and the corresponding assignment of the relevant ScanMux control bits. This analysis is performed upfront to ease the subsequent reasoning process. Due to that all scan segments contained in a perfect network can be removed from the cost function. Hence the minimization problem can be significantly reduced. The derivation of the minimum number of scan elements and the setting of the corresponding control bits is performed independently of the actual re-targeting process.
Please note that SIB-based structures within scan networks fulfill the requirements of a PN by construction. Hence the proposed reduction approach can also be applied for SIB-based scan networks.

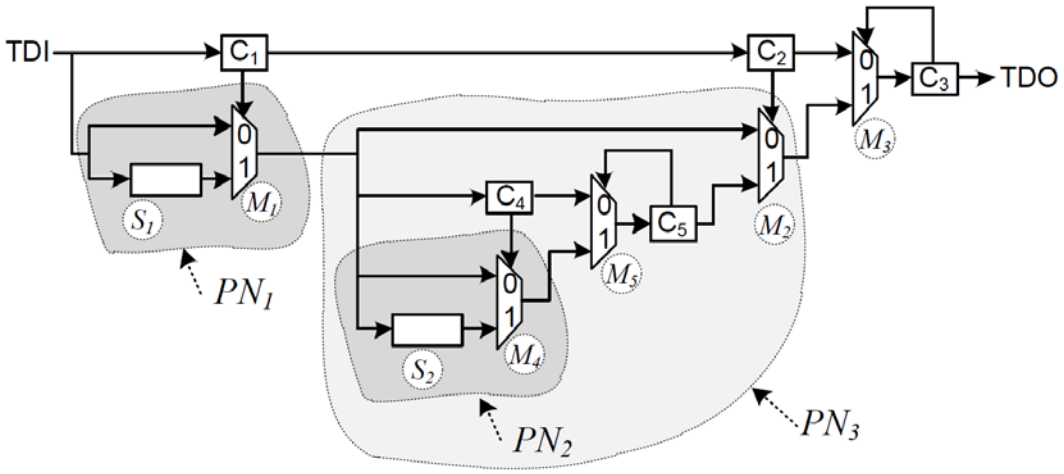

Figure 29. Perfect network example with three PNs (namely, PN_1, PN_2, and PN_3)

B. Optimized Reasoning in 1687 Networks

The focus of this subsection is to employ the new upper bound computation described earlier and the notion of perfect networks described above in order to propose an efficient and applicable approach to retarget sets of PDL commands in IEEE 1687 networks ensuring the global minimum with respect to the number of CSUs and with respect to the number of shift cycles.

Knowing the upper bound of a sequential problem with respect to the number of time frames enables the immediate computation of a global optimum. In contrast to the approach proposed in [15], where numerous runs of a SAT-solver are required to obtain the minimum of CSUs, our approach only requires a single run of the pseudo-Boolean optimizer MiniSat+ [25]. This is ensured by generating a network representation of the targeted IEEE 1687 network which is unrolled over the number of time frames as determined by the new upper bound computation. The cost function added to the pseudo-Boolean representation is formulated such that every time frame requiring some active scan-path increases the value of the cost function by one. Hence finding a solution with a minimum of activated scan-paths provides the minimum number of CSUs required to execute a set of PDL commands in an IEEE 1687 network.

As described in before this upper bound is also applicable for finding the global minimum with respect to the number of shift cycles to establish the required assignments within the scan network. In order to obtain the global minimum it is required to replace the cost function for the CSU minimization by a cost function as described in III-B, where it contains all scan elements which are possibly included in some active scan-path during some time frame.

Although the complexity of IEEE 1687 network controls is reasonably small, the sequential problem modeling the IEEE 1687 network over a number of time frames is significantly harder. However, the complexity of the corresponding optimization problems is mostly resulting from the complexity of the cost functions. In other words the solver-internal representation of the cost function is especially for the cycle-based optimization much larger than the representation of the unrolled scan network. The size of the cost function and its solver-internal representation is highly depending on

the number of modeled time frames and on the number of considered scan elements. This is due to the larger set of possible numerical solutions. The numerical solution space in the solver is modeled in form of an adder, a sorter, or a BDD-like representation. The applied representation is automatically chosen by a solver-internal heuristic during the reasoning process [25].

PNs reduce the size of the cost function and hence the complexity of its solver internal representation. Due to this reduction, the complexity of the subsequent reasoning process is also significantly decreased. In the following it is described how the concept of PNs is applied to identify redundant elements in the cost function. Consider the scan network depicted in Figure 30 and let us assume that scan segment S_2 should be activated, then the only path towards TDI passes through PN1. As discussed in above the minimum length of an active scan-path and the corresponding assignment of the control signals, in this case C_1, are known. Hence in the described case S_1 does not need to appear in the cost function. Only an additional constraint needs to be added to enforce $C_1 = 0$ during the time frame, where S_2 is supposed to be active. Furthermore all constraints ensuring the continuation of an active scan-path towards TDI, as required in the model proposed in [14], are also not required since by construction there always exists a valid path through a PN and PN_1 is directly connected to TDI.

Considering the case that scan segment S_1 in Figure 30 should be accessed, then every active scan-path starting from S_1 towards TDO is passing PN_3. Again it is possible to apply the above concept, firstly we can rely that there always exists a valid path through a PN and hence the representation of the components within PN3 is redundant. Secondly it is possible to remove the scan segments, S_2, C_4, C_5, from the cost function. The resulting cost function would only contain elements representing S_1, C_1, C_2 and C_3, in each modeled time frame.

If a set of PDL commands requires access to several instruments or scan segments, then only those scan segments can be removed from the cost function which are not required for any of those instruments or scan segments. Assuming a PDL command block would require access to S_1 and S_2 in Figure 30,then there are no reductions obtainable by our approach. It is important to emphasize that the reductions achievable by applying PNs depend highly on the set of instruments which need to be accessed. Hence the resulting cost function needs to be derived separately for every set of PDL commands. Furthermore it can be stated that the number and the complexity of applicable PNs is vital to the complexity of the reasoning process since a linear reduction of the size of the cost function leads to an exponential reduction of the solution space.

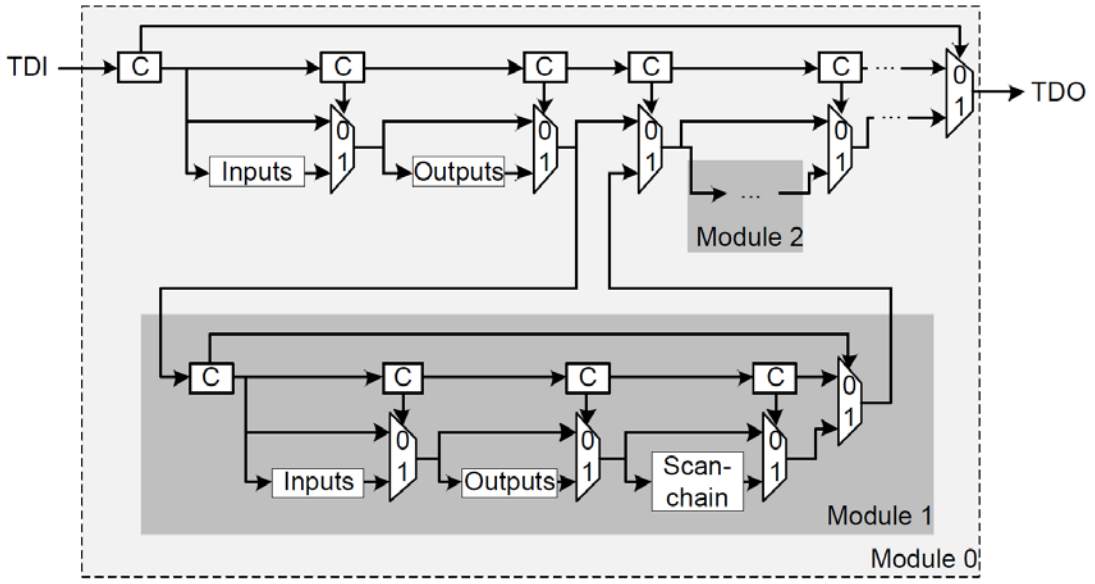

Figure 30. Part of the network generated for the P34392 benchmark circuit

4.3.5.4 Experimental Results

The objective of the experiments is to compare the proposed approach in respect to efficiency against the method described in [21]. As mentioned earlier, efficiency is given by the CPU time it takes to generate the scan vectors required to change the network from accessing one set of instruments to accessing another set of instruments. As for the effectiveness, it should be noted that the proposed method is guaranteed to find the optimal solution in terms of access time, due to the use of the presented upper bound calculation method.
For experimentation, we used a set of designs with networks implemented as the MUX-based architecture described in [15]. These designs are based on the 12 hierarchical circuits in the ITC'02 benchmark set [31]. For each circuit, the number of modules, hierarchical structure, number and type of ports, and number and length of internal scan-chains is available [31]. To create the networks, a number of shift-registers are considered for each module as follows:

- a shift-register with a length equal to the number of input pins,
- a shift-register with a length equal to the number of output pins,
- and one shift-register per internal scan-chain where the length of the shift-register equals that of the scan-chain.

The general architecture of the generated MUX-based networks has the style of the network shown in Figure 28, where the DFT instrument is placed in the first level of the hierarchy and the Sensor and Debug instruments are placed in the second level. Considering that the ITC'02 benchmark circuits are hierarchical designs, for each benchmark circuit, a corresponding network is designed such that the shift-registers extracted as listed above, are placed in a hierarchical level corresponding to the original hierarchy reported for the ITC'02 circuits. As an example, Figure 31 shows

how a MUX-based network is constructed for the P34392 benchmark circuit. The details of the designs are in the first five columns of Table I listed in columns for design, number of multiplexers, number of scan segments, number of total scan bits, and number of hierarchical levels in the design, respectively.

The modeling approach was implemented in C++. For solving the ILP-problems the pseudo-Boolean solver Minisat+ v1.0 [25] has been used. The experiments were conducted on an INTEL Xeon E5645 2.4GHz with 32GByte main memory running Linux Mint 13 64-bit.

For the experiments, two cost functions are used: one which finds the minimum number of CSU cycles needed for the retargeting step (i.e., the generated scan vectors use the least number of CSU cycles), and one which finds the minimum access time for the retargeting step (i.e., the generated scan vectors take the least number of test clock cycles for their application). The former cost function is used in [14], while the latter is used in [15].

For every benchmark 10 test cases were applied. Each test case randomly activates 10% of the scan segments in the design. For the benchmarks containing less than 100 scan segments, 10 scan segments were activated. In our experiments the activation of a scan segment S_i implies that the generated sequence of scan vectors, starting from an initial state, ensures that S_i is contained in at least one active scan path established within the test vector sequence.

The results from the experiments are listed in Table I. The benchmarks were translated into pseudo-Boolean (pB) constraints (clauses) and unrolled over a number of time frames determined by the new upper bound computation presented above. The results of the upper bound computation are reported in the sixth column of the table. The number of pseudo-Boolean constraints required to represent the unrolled scan network are reported under columns “number of pB clauses” for both previous model (column seven) and the proposed model (column eleven). The reduction of the proposed model compared to previous model in number of pseudo-Boolean constraints for all designs is around 40%. The average run-times to generate scan vectors requiring the minimal number of CSUs for the 10 test cases are listed under columns “t_{csu}” for previous model (column eight) and for the proposed model (column twelve). For all designs, the run-times are significantly lower for the proposed model. On average run-times are reduced by 33.4%.

The average run-time to compute minimal scan vector with respect to the number of shift cycles and hence minimal overall access time are listed under columns “t^{cycles}_{avg}” for the previous model (column nine) and the proposed model (column thirteen). The time-out limitation for these experiments was set to 300 seconds. The proposed model computed and proved the minimum of shift cycles within the time-out limit for 108 out of 120 test cases, while the previous model computed and proved the minimum in 63 out of 120 test cases. For several larger benchmarks all test cases timed out using the previous model. Please note that the presented upper bound results were applied to both the previous model and the new model. Hence the listed run-time improvements result only from the reductions in the proposed model.

The effectiveness of the retargeting process has been improved such that due to the proposed upper bound computation the generated scan vector sequences are proved to be minimal.

Design	Number of muxes	Total scan segm.	Total scan bits	Hierar-chical levels	Proposed upper bound (Section IV)	Previous model [15]				Proposed model			
						number of pB clauses	$t_{csu\ avg}$ [s]	$t_{cycles\ avg}$ [s]	number of time outs	number of pB clauses	$t_{csu\ avg}$ [s]	$t_{cycles\ avg}$ [s]	number of time outs
u226	59	99	1475	2	4	3342	0.060	1.340	0	2030	0.040	0.480	0
d281	67	117	3880	2	4	3816	0.068	1.620	0	2315	0.044	0.820	0
h953	63	109	5649	2	4	3584	0.065	3.066	0	2175	0.044	0.869	0
f2126	45	81	15834	2	4	2580	0.047	1.310	0	1565	0.034	0.460	0
a586710	47	79	41682	3	6	1710	0.060	1.231	0	1040	0.040	1.840	0
q12710	30	51	26188	2	4	1710	0.031	0.970	0	1040	0.020	0.460	0
g1023	94	159	5400	2	4	5322	0.091	1.966	7	3230	0.063	1.320	0
d695	178	335	8407	2	4	10239	0.168	-	10	6195	0.113	4.740	0
p34392	142	245	23261	3	6	9864	0.167	-	10	6002	0.112	1.117	0
t512505	191	319	77037	2	4	10780	0.187	-	10	6542	0.118	3.850	0
p22810	311	565	30139	3	6	21770	0.382	-	10	13230	0.243	9.691	2
p93791	653	1241	98637	3	6	46037	0.810	-	10	27945	0.537	-	10

Table 1. EXPERIMENTAL RESULTS ON ITC'02 BENCHMARK SET.

4.4 Summary

In this section different BASTION contributions in the area of verification and validation of 1687 networks are presented. After an introduction into the topic an efficient approach to model of 1687 networks is discussed, which is a base for applicable and scalable reasoning methods. Next specific verification and validation aspects with respect to 1687 networks and their application are discussed such as ICL compliance, PDL retargeting as well as aspects of co-verification of ICL and PDL.
Finally a novel retargeting approach for generalized IEEE 1687 networks has been presented which employs pseudo-Boolean solvers.

5 Conclusions

Design, analysis, verification and validation of 1687 networks are essential building blocks in order to successfully establish and spread the new IEEE 1687 standard. This report presents first results in the area of design and optimization of 1687 networks.
The first set of contributions focuses on the minimization of the overall access time under consideration of different application scenarios. In particular, the robustness of different network topologies with respect to the OAT is evaluated. Next possible extensions of 1687 are proposed to enable an efficient fault management, involving fault detection, fault reporting as well as fault diagnosis in future complex 1687 networks.
The second set of contributions focuses on aspects of the verification and validation of 1687 networks. The efficient modulation and representation of 1687 networks is described in detail. Furthermore, verification aspects with respect to ICL compliance in general as well as the verification specific 1687 structures are discussed. In connection to that an overall validation scheme of 1687 networks is proposed. Finally, two specific verification challenges regarding PDL retargeting in 1687 networks as well as future challenges in ICL-PDL co-verification are discussed.

The presented work concludes the BASTION activities regarding the task T2.3.

6 Bibliography

[1] F. Ghani 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.

[2] F. Ghani Zadegan, U. Ingelsson, G. Asani, G. Carlsson and E. Larsson, "Test Scheduling in an IEEE P1687 Environment with Resource and Power Constraints," in *Asian Test Symposium (ATS)*, Delhi, Nov. 2011.

[3] F. Ghani Zadegan, U. Ingelsson, G. Carlsson and E. Larsson, "Design Automation for IEEE P1687," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, Mar. 2011.

[4] A. Jutman, S. Devadze and K. Shibin, "Effective Scalable IEEE 1687 Instrumentation Network for Fault Management," *Design & Test,* vol. 30, no. 5, pp. 26-35, 2013.

[5] F. Ghani Zadegan, G. Carlsson and E. Larsson, "Robustness of TAP-Based Scan Networks," in *International Test Conference*, Seattle, Oct. 2014.

[6] M. Keim et al., "Industrial Application of IEEE P1687 for an Automotive Product," in *Euromicro Conference on Digital System Design (DSD)*, Sep. 2013.

[7] K. Yamasaki et al., "External Memory BIST for System-in-Package," in *International Test Conference*, 2005.

[8] A. Carbine and D. Feltham, "Pentium(R) Pro Processor Design for Test and Debug," in *International Test Conference*, 1997.

[9] Z. Conroy et al., "Board Assisted-BIST: Long and Short Term Solutions for Testpoint Erosion – Reaching into the DFx Toolbox," in *International Test Conference (ITC)*, 2012.

[10] Margulis et al., "Evolution of Graphics Northbridge Test and Debug Architectures Across Four Generations of AMD ASICs," *Design & Test,* vol. 30, no. 4, pp. 16-25, Aug. 2013.

[11] J. Dworak, A. Crouch, A. Zygontowicz and M. Thornton, "Don't Forget to Lock your SIB: Hiding Instruments using P1687," in *Proceedings of the IEEE International Test Conference*, 2013.

[12] R. Baranowski, M. A. Kochte and H.-J. Wunderlich, "Access Port Protection for Reconfigurable Scan Networks," *Journal of Electronic Testing Theory and Applications (JETTA),* vol. 30, no. 6, pp. 711-723, Dec. 2014.

[13] Mentor Graphics Corp., Tessent IJTAG User's Manual 2015.3, 2015.

[14] R. Baranowski, M. Kochte and H.-J. Wunderlich, "Modeling, Verification and Pattern Generation for Reconfigurable Scan Networks," in *International Test Conference*, Nov. 2012.

[15] R. Baranowski, M. A. Kochte and H.-J. Wunderlich, "Scan Pattern Retargeting and Merging with Reduced Access Time," in *IEEE European Test Symposium (ETS'13)*, 2013.

[16] IEEE, "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," p. 1–283, December 2014.

[17] Y. Fkih, P. Vivet, B. Rouzeyre, M.-L. Flottes, G. D. Natale and J. Schloeffel, "2D to 3D Test Pattern Retargeting Using IEEE P1687 Based 3D DFT Architectures," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI),* p. 386–391, July 2014.

[18] M. Portolan, B. Van Treuren and S. Goyal, "Executing IJTAG: Are Vectors Enough?," *IEEE,* vol. 30, no. 5, pp. 15-25, Oct 2013.

[19] J. P. Marques-Silva and K. A. Sakallah, "GRASP: a search algorithm for propositional

satisfiability," *IEEE Transactions on Computers,* vol. 5, no. 48, p. 506–521, 1999.

[20] N. Eén and N. Sörensson, "An extensible sat-solver," *Theory and Applications of Satisfiability Testing, 6th International Conference,* no. Selected Revised Papers, p. 502–518, 5-8 May 2003.

[21] N. Eén and A. Biere, "Effective preprocessing in SAT through variable and clause elimination," *Theory and Applications of Satisfiability Testing,* pp. 61-75, 19-23 June 2005.

[22] A. K. J. Baumgartner and J. Abraham, "Property checking via structural analysis," *Proc. 14 Intl. Conference on Computer Aided Verification (CAV'02).,* p. 151–165, 2002.

[23] D. Tille, R. Krenz-Baath, J. Schloeffel and R. Drechsler, "Improved circuit-to-CNF transformation for SAT-based ATPG," *IEEE European Test Symposium,* 2008.

[24] C. Liu, A. Kuehlmann and M. W. Moskewicz, "CAMA: A multi-valued satisfiability solver," *2003 International Conference on Computer-Aided Design (ICCAD'03),* p. 326–333, 9-13 November 2003.

[25] N. Eén and N. Sörensson, "Translating pseudo-boolean constraints into SAT," *Boolean Modeling and Computation,* vol. 2, pp. 1-26, 2006.

[26] M. Gebser, B. Kaufmann, A. Neumann and T. Schaub, "Conflict-driven answer set solving," *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence,* p. 386, 6-12 January 2007.

[27] S. Eggersglüß, K. Schmitz, R. Krenz-Baath and R. Drechsler, "Optimization-based multiple target test generation for highly compacted test sets," *19th IEEE European Test Symposium, ETS 2014,* p. 1–6, 26-30 May 2014.

[28] R. Krenz-Baath, A. Glowatz and F. Hapke, "Fault collapsing of multi-conditional faults," *16th IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems, DDECS 2013,* p. 42–47, 8-10 April 2013.

[29] T. Lengauer and R. E. Tarjan, "A fast algorithm for finding dominators in a flowgraph," *Transactions on Programming Languages and Systems,* vol. 1, no. 1, p. 121–141, July 1979.

[30] S. Alstrup, D. Harel, J. Clausen and M. Thorup, "Dominators in linear time," *SIAM Journal on Computing,* vol. 28, no. 6, p. 2117–2132, 1999.

[31] E. J. Marinissen and et. al, "A set of benchmarks for modular testing of SOCs," *Proc. ITC,* p. 519–528, 2002.

[32] F. Ghani Zadegan, U. Ingelsson, G. Carlsson and E. Larsson, "Reusing and Retargeting On-Chip Instrument Access Procedures in IEEE P1687," *Design & Test of Computers,* vol. 29, no. 2, pp. 78-88, Apr. 2012.

[33] K. Shibin, S. Devadze and A. Jutman, "Asynchronous Fault Detection in IEEE P1687 Instrument Network," in *NATW*, 2014.

[34] A. IEEE, "IEEE Std 1149.1-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture," 2001.

[35] A. Biere, A. Cimatti, E. M. Clarke and Y. Zhu, "Symbolic model checking without BDDs," *5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99),* p. 193–207, March 1999.

[36] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang and S. Malik, Chaff: Engineering an efficient SAT solver, Las Vegas, Nevada: Proceedings of the 38th ACM/IEEE Design Automation Conference, 2001, p. 530–535.

[37] N. Eén and N. Sörensson, "Temporal induction by incremental SAT solving," vol. 4, no. 89, 2003.

[38] K. Yang, K.-T. Cheng and L.-C. Wang, "Trangen: a SAT-based ATPG for path-oriented

transition faults," *Proceedings of the 2004 Asia and South Pacific Design Automation Conference,* p. 92–97, 2004.

[39] e. a. Zadegan F. G., "Reusing and Retargeting On-Chip Instrument Access Procedures in IEEE P1687," *IEEE,* vol. 2, no. 29, p. 79 –88, April 2012.

[40] J. Rearick and A. Volz, "A Case Study of Using IEEE P1687 (IJTAG) for High-Speed Serial I/O Characterization and Testing," in *Proc. IEEE Int'l Test Conf. (ITC)*, Oct. 2006.