Evolutions of the Software Flow for Automated Testing

Michele Portolan

THÈSE

Pour obtenir l'Habilitation à Diriger des Recherches (HDR) de

L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : EEATS - Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)
Spécialité : Nano électronique et Nano technologies
Unité de recherche : Laboratoire TIMA (Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés)

Évolutions du flux logiciel pour les tests automatisés

Evolutions of the Software Flow for Automated Testing

Présentée par :

Michele PORTOLAN

Thèse soutenue publiquement le **19/03/2024**, devant le jury composé de :

Alberto BOSIO Professeur, Ecole Centrale de Lyon, INL (France)	Président du jury
Olivier SENTIEYS Professeur, Université de Rennes (France)	Rapporteur
Henri-Pierre CHARLES Directeur de Recherches, CEA LIST Grenoble (France)	Rapporteur
Arnaud VIRAZEL Professeur, LIRMM Montpellier (France)	Rapporteur
Matteo SONZA REORDA Professeur, Politecnico di Torino (Italie)	Examinateur
Régis LEVEUGLE Professeur, Laboratoire TIMA Grenoble (France)	Examinateur
Jeff REARICK Senior Fellow, AMD (USA)	Invité

Acknowledgments

First of all, I would like to thank the Reviewers and all the members of this HdR jury for their participation.

Thanks to Régis, Suresh and Brad: without your help, guidance and tutoring, I would never have become the professional I am now.

To my wife Nadia, thank you from the bottom of my heart for all your love and support and for helping me being myself.

A special thanks to my sons, Clovis and Cléante, for always keeping me on my toes with their questions and for showing me the world with new, enthusiastic eyes.

"The Road goes ever on and on,
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can,
Pursuing it with eager feet,
Until it joins some larger way
Where many paths and errands meet.
And whither then? I cannot say"
Bilbo Baggins, The Lord of the Rings

Table of Contents

Figure Index

Table Index

Introduction

During his career, a Researcher naturally follows different directions and topics. There are many reasons: for personal interests, to exploit collaboration opportunities, to learn new skills, etc.. There is no need, and neither strictly speaking any value, in enforcing coherence between the different topics as long as it is possible to correctly apply and expand one own's competences and be able to contribute to the field. And, why not, taking pleasure in it.

The way a Scientific Theme is addressed is well-known: first, it must be studied to understand its current State of the Art and, more importantly, its Open Questions and Evolution Potential. Second, a Theoretical analysis must be applied to find the common elements, the missing ones and elaborate a strategy to introduce new ideas. Last, the Abstraction must be Implemented to demonstrate its feasibility and its real contribution. Once some results are reached, the cycle starts from the beginning: well-known does not mean easy.

In this document, after giving a through description of my career in Chapter 1 "Curriculum Vitae", I decided to focus on one of my Research Themes, which I had the opportunity to build from the ground up over a period of almost 15 years, from my first Role after my PhD to the current time through different positions and employers : the Evolutions of the Software Flow for Automated Testing.

As its name states, Chapter 2 gives the "Motivations and State of the Art" of the topic, while Chapter 3 "The need for a New Test Flow" tackles the issue by providing an in-depth Theoretical Analysis of the current shortcomings. Chapter 4 "A New Automated Test Flow: Specification" provides and justifies the final Abstraction to improve the filed. Chapter 5 "A New Automated Test Flow: Implementation" provides a description of the different iterations done over the years to refine the Abstraction, Implement it and use to provide innovative solutions. Based on these results, Chapters 6 provides the "Short-to-Mid Term Perspectives" of these topics. Lastly, Chapter 7 draws "Conclusions and Long-Term Directions" from a more general point of view.

1 Curriculum Vitae

1.1 Curriculum Vitae

Name: Michele Portolan
Birth Date: 9/7/1979 in Trento (Italy)
Married
Nationality: Italian
Professional Address:
✉ : TIMA Laboratory
46 Av. Felix Viallet
38031 Grenoble
France
☏ : 04 76 57 48 55
@ : michele.portolan@grenoble-inp.fr

Current Position

Senior Associate Professor at Grenoble-INP Phelma, working at TIMA lab

Phelma Co-Head for the Major « Embedded Systems and Connected Objects »

Professional Experience

Oct 2021-Today	**Senior Associate Professor** at Grenoble-INP Phelma
Sep 2018- Sep 2019	**Invited Professor** at the Politecnico of Torino, Italy
Sep 2013- Sep 2021	**Associate Professor** at Grenoble-INP Phelma
2007 -August 2013	**Member of Techical Staff** at Alcatel-Lucent Bell Labs Ireland and France
2006-2007	**Lecturer** at Institut National Polytechnique de Grenoble

Academic Experience

2006	**PhD in Microelectronics**, defended on December 6th 2006
2003 September	**Italian Master Degree in Electronics Engineering**, awarded by the Politecnico di Torino, grade 110/110 Cum Laude.
	Master of Advanced Studies (DEA) in Microelectronics, with honors, delivered by Université Joseph Fourier, Grenoble
2003 June	**French Master Degree in Telecommunication Engineering**, with honors, delivered by the Département Télécoms of INPG

Language Skills

Italian	Mother Tongue
French	Bilingual
English	Bilingual
Japanese	Lower Intermediate

Competences

Test Standards, Hardware Design Flow, Automated Test Flow, Software Design, C/C++, Operating Systems, Compilation and Languages, FPGA Prototyping, Embedded Systems, Reliability and Dependability, Hardware Safety and Security, Hardware/Software mixed systems, Modeling and Simulation, Fixed Networks and 3G/4G.

Internantional Scientific Experience

- Secretary of the IEEE 1687 Renewal Working Group.
- Signing member of the IEEE 1687-2014 Standard.
- Valorization Liaison for the TIMA Laboratory
- Member of the standardization working groups IEEE P1687.1, IEEE P2654
- 30+ publications at International Conference, 7 Journal Papers
- 17 US and European Patents, 100% grant rate.
- APP (Agence pour la Protection des Programmes) filings for protection of original software
- Coordinator of several internal projects in Bell Labs.
- « Development of a safe and secure Embedded System», PhD Thesis

Teaching Experience

- Responsible for Industrial Liaisons for the SEOC Major of Phelma
- Supervisor for several Master Degree internship
- Joint Supervisor of two PhDs
- Development of teaching support materials
- Head of the SLE (Embedded Systems and Software) from 2014 to 2017
- Co founder of the SEOC Major in 2017 as the fusion of SLE and ISSC (Signals, Internet, Services and Connected Systems)

Awards and Promotions

- Promotion to Senior Associate Professor, October 2021. Reserved to the top ~10/20% of Associate Professors
- Prime d'Encadrement et De Recherche (PEDR) (French Research and Supervision Bonus, awarded following an evaluation over 5 years of activity, awarded to the top 20%) 2020-2024

International Experience

- Global Coordinator of TTTC's « E.J. McCluskey Best Doctoral Thesis Award » 2013 → present
- Program Chair for the "1st Test Access, Automation and Adoption Workshop", 2021
- Chair of « 3rdTest Standards Application Workshop (TESTA) », 2018
- Vice General Chair of « 2ndTest Standards Application Workshop (TESTA) », 2017
- Organiser of « 1st International Test Standards Application Workshop (TESTA) », 2016
- Program Committee Member of the Latin-American Test symposium (LATS) since 2016
- « Awards co-Chair » for the « Test Technology Technical Council» (TTTC) since 2012
- « Industrial Liaison co-Chair » for the « 2012 European Testing Symposium » (ETS12), Annecy, France, Mai 2012

- « Peer Reviewer » International Conferences (ITC, ETS, DDECS, etc…) and journals (IEEE Design and Test, Journal of Electronic Testing : Theory and Applications »)
- « Publication Chair » for the « 12th International On-Line Testing Symposium » (IOLTS06), Como, Italy, July 2006
- « Audio-visual Chair » for the « 11th International On-Line Testing Symposium » (IOLTS05), Saint Raphael, France, July 2005

Personal Interest and Hobbies

- Classical, Medieval and Contemporary Literature (English, French and Italian)
- Road Cycling, Mountain Hiking, Golf
- Analog Film Photography

1.2 Teaching Activities

Classes taught as PhD candidate and Lecturer (2004 à 2007)

Title	**University**	**Level**	**Year**	**Hours**	**Type**
Computer Architecture	Département Télécom	BAC+4	2007	13,5h	TD
Operating Systems	ENSERG	BAC+4	2007	32h	TD
SoC Project	ENSERG	BAC+5	2006	24h	Project
Computer Architecture	Département Télécom	BAC+4	2006	10,5h	TD
Digital Circuit Design	Département Télécom	BAC+3	2006	15h	TD
VHDL	ENSERG	BAC+4	2006	21h	TD
C Project	ENSERG	BAC+4	2004 à 2006	116h	Project
Computer Architecture	Département Télécom	BAC+3	2005	18h TP	Project
Computer Architecture	Département Télécom	BAC+3	2006	32h	Project
VLSI Design and Test	Département Télécom	BAC+4	2004	6h	TD

Classes taught as Associate Professor

Digital Design	**University**	**Level**	**Year**	**Hours**	**Type**
System Integration	Ensimag	BAC+4	2013-14-15-16	~15/an	CM
System Integration	Ensimag	BAC+4	2013	~18/an	TPTD
Design of Integrated Digital Systems	Ensimag	BAC+4	2017-19-20-21-22-23	~20/an	TPTD
Design of Integrated Digital Systems	Ensimag	BAC+4	2017-19-20-21-22-23	~15/an	CM
Implementation of a Embedded System U Case	Ensimag/Phelma	BAC+5	2013-14-15-16- 17-19-20-21-22-23	~40/an	Projet
Design of mixed integrated functions	Phelma	BAC+4	2014-15-16- 17-19-20-21-22-23*	~50/an	Projet
VLSI - ASIC + FPGA	Phelma	BAC+4	2014	12	TPTD
Analysis and integration of a complex integrated system	Phelma	BAC+5	2016-17-19-20-21-22	~30/an	Projet

** class scheduled in the second semester*

Reliability and Test	**University**	**Level**	**Year**	**Hours**	**Type**
Fault Tolenance	Ensimag	BAC+5	2013-14-15-16-17	~27/an	CM

Test of circuits	Phelma	BAC+5	2015-16-17-19-20-21-22-23*	~8/an	TPTD
Hardware Reliability and Security	Ensimag/Phelma	BAC+5	2019-20-21-22-23	~13/an	CM
Hardware Reliability and Security	Ensimag/Phelma	BAC+5	2019-20-21-22-23	~16/an	TP

** class scheduled in the second semester*

Computer Science	**University**	**Level**	**Year**	**Hours**	**Type**
Computers and Microprocessors	Phelma	BAC+3	2014	16	TD
Computer Science Project in C	Phelma	BAC+4	2014-15-16-17-19-20-21	~20/an	TPTD
Computer Science Projet (SEI)	Phelma	BAC+4	2014	35	TD+TP
Operating Systems and Parallel Programming	Phelma	BAC+4	2022	6	CM
Operating Systems and Parallel Programming	Phelma	BAC+4	2022	22	TD
Operating Systems and Parallel Programming	Phelma	BAC+4	2022	6	TP

Miscellanous	**University**	**Level**	**Year**	**Hours**	**Type**
Préorientation - SEI : Conception analogiq et numérique	Phelma	BAC+3	2013	8	TD
Préorientation - SLE : Circuits numériques	Phelma	BAC+3	2013-14-14	~20/an	TD
Préorientation - SEOC - Systèm Embarqués et Obj. Connectés	Phelma	BAC+3	2016-17-19-20-21-22-23*	~9/an	CM
Group Projects	Phelma	BAC+3	2014-15-16	~16/an	Projet
Group Projects	Phelma	BAC+3	2015	17	Projet

** : heures prévues pour le deuxième semestre*

Tutoring and Oral Defenses	**University**	**Level**	**Year**	**Hours**	**Type**
Retour d'Expérience (REX)	PHELMA	BAC+4	2013	1	TD
2nd Year Internship Tutor	Phelma	BAC+2	2013-14	6	Tutorat
3rd Year Internship Tutor	ENSIMAG	BAC+5	2013-2017-19-20-21-22-23	~6/an	PFE
3rd Year Internship Oral Defenses	ENSIMAG	BAC+5	2014-16-17-19-2021-22-23	~25/an	PFE

Tutoring for Apprentices	PHELMA	BAC+3	2013-14-15-16-17-18-19-20	~10/an	Tutorat
Industrial Projects	Phelma	BAC+4	2016	48	PFE

** : heures prévues pour le deuxième semestre*

Pedagogic Responsibilities	**University**	**Year**	**Hours**	**Type**
Responsibility for SLE Major	PHELMA	2013-14-15-16-17-19-20	24/an	Resp
Responsibility for SEOC Major	PHELMA	2021-22-23	~40/an	Resp
External Relationships with the “Grenoble University Space Center”	PHELMA	2015	12	Resp
Jury Pré sélection AP-CSI	PHELMA	2016	3	Jury
Jury Pré sélection AP-CSI	PHELMA	2017	6	Jury
Dossiers Admis Sur Titre (AST)	PHELMA	2016-17-19	6/an	Jury

Doctoral Level Classes

	University	**Level**	**Year**	**Hours**	**Type**
Test and Design for Test for Integrated Circuits	Ecole doctorale EEATS	Doctorat	2014,16	6	CM
Advanced Techniques for Digital Testing	Politecnico di Torino	Doctorat	2020,2021,2022, 2023	~8/an	CM

Summary of hours taught per year:

2013-2014	***2014-2015***	***2015-2016***	***2016-2017***	***2017-2018***	***2018-2019***	***2019-2020***	***2020-2021***	***2021-2022***	***2022-2023***
200	410,75	346,25	373,75	295,75*	192*	355*	344,75	372,75	348,5

****years taking part of a MOISE inter-annual modulation***

1.3 Career Synthesis

When I arrived at the TIMA laboratory in 2013, my Research group was in a "down" phase: several big projects were coming to an end, and classic themes were looking for a new ideas and directions. In this context, I immediately took the 50% supervision of Kais Chibani's thesis, and I committed myself to both integrate the group's open themes and develop my own topics. In the theme of Reliability, we have unsuccessfully submitted several ANR subjects over the years, regularly reaching Phase 2, which prevented us from launching any PhD theses in this subject. At the same time, my Test theme was gaining strength and allowed me to obtain my own funding. Unfortunately, the durations of these Projects were too short to finance PhD, and I had to fall back on short-term contracts to have the subject mature and develop. Thanks to these results, in 2017 I was able to participate in the European HADES Project, which allowed me to start the 50% supervision of Kalpana Senthamarai Kannan's thesis. The withdrawal of the German partners forced a budget reduction, which prevented me from supervising a second Phd student. During the Project, I also actively participated in the supervision of the second doctoral student in our group, Vincent Reynaud, even though this was not planned in advance and therefore I was not officially registered as a supervisor. My participation allowed Vincent to broaden the scope of his thesis towards my own themes (Standards and EDA for Test), which resulted in several high-end joint publications. The results of Kalpana Senthamarai Kannan's work allowed us to begin a collaboration with the DAUIN laboratory at the Politecnico di Torino: in this context we were able to launch an ERAMUS + exchange to welcome a doctoral student, Sandro Sartoni, which I supervised during his stay in Grenoble and resulted in a joint publication.

In 2023, I started a new collaboration with Emmanuel Simeu from the "Reliable RF and Mixed-signal Systems" (RMS) group to explore the synergies between Digital and Mixed Signal testing; through the PhD Thesis of Jules Quentin Kouamo, started in November through a "Thèse flechée" of the EEATS Doctoral School.

1.3.1 Project Participation:

- ICT Standardisation Observatory and Support facility in Europe 6th Call (StandICT.eu 2023, part de H2020),"Advance Design-for-Test standards for complex electronics systems", Budget ~10k€ over 6 Moth, Project Owner
- Eramus + Project EMNESS (European Master Network On Embedded System Security 394.5 K€) - Work Package Leader et Phelma Referent.
- Europen Project HADES (Hierarchy-Aware and secure embedded test infrastructure for Dependability and performance Enhancement of integrated Systems 15 M€), 2017-2020, Task Leader, PhD Co-Supervisor
- Technology Maturation project MAST financed by Linksium, 2015-2016, (138,7 k€) , Project Owner, Supervision of two Short-Term Contracts
- Technology Incubation project MAST financed by Linksium, 2017, (57,7 K€), Sceintific Advisor, Supersion of a Short-Term Contract

- IRS Project (Initiatives de Recherche Stratégiques) CADI, « Calcul Approché et Distribué dans les systèmes Intégrés », 2019, 10% research time, Supervision of two Master Degree Thesis
- IRS Project (Initiatives de Recherche Stratégiques) AVOCAM, « Analyse de durée de Vie pour l'Optimisation de Calcul Approché Matériel », 2020/2021, 20% research time, Supervision of three Master Degree Thesis
- IRT40 Cybersécurité, Development of lab classes dedicated to Hardware Security UGA/G-INP, années 2020-2021 (21k) 20% research time, Supervision of one Master Degree Thesis

1.3.2 Supervisions and Management

1.3.2.1 PhD :

1. Jules Quention Kouamo , PhD with the Université de Grenoble, started in November 2023, 50% supervision with Emmanuel Simeu, Professor at UGA
2. Senthamarai Kannan Kalpana, PhD with the Université de Grenoble, Defended on July 2015, 50% supervision with Lorena Anghel, Professor at G-INP → now FPGA Firmware Engineer at ASML, Netherlands
3. Chibani Kais, PhD with the Université de Grenoble, Defended in 2016, 50% supervision with Régis Leveugle, Professor at G-INP → now Sr. Digital Verification Engineer at ST Microelectronics, Grenoble

1.3.2.2 Erasmus + :

Sandro Sartoni, Doctorant au Politecnico di Torino, Italie, April 2022 -July 2022. Subject: Aging Prediction for a RISC-V processor in FDSOI 28nm

1.3.2.3 Master Thesis

1. Mert Arisoy, Politecnico di Torino, Italy;
2. Pierpaolo Iannicelli, Politecnico di Torino, Italy ;
3. Xavier Gros, Master 2 MISTRE, UGA, Grenoble
4. Atoine Cerf, Master 2 MISTRE, UGA, Grenoble
5. Provent Thomas, M2 - Université Claude Bernard Lyon 1 ;
6. Muller Meireles Assumpçao Joao Pedro, ENSIMAG ;
7. Josef Ahmad, Politecnico di Torino, Italy;

1.3.2.4 Short Term Contracts

1. Coulon Jean-Francois, 15 month over three contracts between 2016 and 2017 ;
2. Niels Grateloup, 3 months in 2016

1.3.2.5 PhD Jurys

I participated to several PhD Jurys, being also a Reviewer in Italy where the rules allowed me to. I have been proposed to be reviewer for Elshamy, but I had to decline because of my lack of HdR.

- 2021 : Jury Member for Mohamed Elshamy, Université de la Sorbonne, Laboratoire LIP6, France, July 2021
- 2021 : Reviewer for Davide Piumatti, Politecnico di Torino Doctorate School, Italy, February 2020
- 2020 : Reviewer for Marco Restifo, Politecnico di Torino Doctorate School, Italy, February 2020
- 2017 : Reviewer for Alejandro Velasco, Politecnico di Torino Doctorate School, Italy, February 2020
- 2017 : Jury Member for Riccardo Cantoro, Politecnico di Torino Doctorate School, Italy, February 2020
- 2015 : Jury President for the XXVII cycle of the « Phd in Computer Science and Information Systems "of the Politecnico di Torino Doctorate School, Italy

1.3.2.6 Recruitment Jurys

Member of the Recruitment jury for an Associate Professor (Maitre de Conférences Section 63) at the INP Toulouse, May 2019

1.4 Scientific Themes

Since my PhD, I worked on several scientific themes, of which I will give a brief summary in this section. However, in the rest of the document, I decided to develop in details my main Scientific Theme: Software Methods for Next-Generation Test Standards. It is a line of Research that I have been developing on my own since my arrival in Bell Labs Ireland in 2007 over a span of more than 15 years.

1.4.1 Software Methods for Next-Generation Test Standards

The complexity of today's electronic systems, the production volumes and the quality imposed by critical applications such as the automobile are putting traditional testing approaches under great pressure. To overcome this problem, the testing field is going through a period of profound evolution, dominated by new "Design for Test" techniques that push automation to the very heart of systems. This field has always been dominated by the simple and extremely effective "Scan Test" for structural testing, where the integrity of a circuit is checked by seeing it as a network of nodes. These approaches have difficulty following system scaling because of their combinatorial complexity. Moreover, they are not at all adapted to new design paradigms such as "IP-based Design" or to new issues such as security. Often seen by Academics as mere collections of already existing solutions, Standards are on the contrary very powerful tools for pushing new approaches towards manufacturers, who see in standardization a guarantee of quality and support. My direct experience in the development of the P1687 standard between 2007 and 2014 allowed me to foresee its impact on current practices and development flows. In particular, I identified a series of criticalities in the associated EDA tools, caused by the axioms and paradigms which are at their heart, and which cannot be resolved through simple incremental corrections. This research theme is therefore based on the analysis and theoretical abstraction of current practices to formalize new needs and develop the corresponding software suites. This line of research generated several publications, as well a Technology Transfer project, with several APP (Software Protection) filings and a Patent. It should be noted that this Patent is an integral part of the current proposal to the IEEE P1687.1 Standard Working Group and it was therefore the subject of a "Letter of Assurance for Essential Patent Claims" to the IEEE, which guarantees its possible commercial exploitation. future.

The PhDs of Kalapana Senthamarai Kannan and Jules Quention Kouamo, are part of this line of research as explorations of long-term impact and applications.

1.4.1.1 Kalapana Senthamarai Kannan: "Performance and Safety/Security Management in automotive and IoT Application"

Due to technology scaling and transistor size getting smaller and closer to atomic size, the last generation of CMOS technologies such as FDSOI presents important variability of several physical parameters. As a consequence, it becomes more and more difficult to guarantee circuit functionality for all Process, Voltage, Temperature (PVT) corners and in turn, to compensate for different sources of variability. Moreover, circuit wear-out degradation leads to additional temporal variations, potentially resulting in timing and functional failures. Under normal operation conditions, a transistor can be affected by various aging effects such as Hot Carrier Injection (HCI), Negative/Positive Bias Temperature Instability (NBTI,PBTI), and Time-

Dependent Dielectric Breakdown (TDDB). In advanced technologies, such as FDSOI, local and global variability, NBTI and HCI phenomena are considered as critical reliability issues. Hence, considering these phenomena as early as possible in the design steps (i.e. during the standard cells characterization step, or at the circuit and system design) are mandatory, especially for high reliable application such as automotive applications, or mixed critical applications.

Indeed, the above-mentioned reliability threats can severely degrade performances, and in the worst case, provoke system failures, affecting safety goals of critical reliable systems. Accurate simulations with physical degradation models of aging phenomena combined with actual silicon measures are, de facto, necessary to better understand and assess the reliability impact on complex digital designs. To handle such problems, one conventional method consists in providing bigger safety margins (also called guard bands) at design-time. Adding pessimistic timing margins (or their equivalent voltage margins) to guarantee all Operating Points under worse case conditions is not possible anymore due to the huge impact on design costs, with an upward trend as technology moves further. Therefore, the usage of delay violation monitors, usually placed at the end of potential critical paths, becomes necessary. Placing the monitors in a given design is a critical task: the designer has to select the endpoints that will age the most, as it may become a potential point of failure. Monitor warnings signals can trigger adaptive techniques, such as Adaptive Voltage Scaling (AVS) or Dynamic Voltage Frequency Scaling (DVFS). They are then used to adapt dynamically the frequency and the voltage according to the operating conditions and the application needs. In addition to the reduction of design margin, monitors also help compensate performance and power degradation. Sometimes, the circuit's lifetime can be extended. It is worth noticing that the area overhead induced by the monitor placement should be carefully considered and should remain reasonable. The number of selected endpoints for monitor insertion should be as small as possible, but still cover the most important critical endpoints of the design. However, endpoint selection is an extremely complex task which requires a deep knowledge of both the target technology and the final workload.

To alleviate these restrictions, in this PhD we explored Machine Learning approaches to find methods that starting from a limited set of technological parameters are able to efficiently predict the delay degradation of paths depending on a given workload and available Operating Performance Points (OPP) expressed in terms of Voltage and Frequency. The aim was to obtain a lightweight, embeddable solution that can be used in conjunction with delay violation monitors in order to alleviate monitor insertion complex task, but also and to identify the best OPPs following different optimization strategies. The ML framework obtained during this PhD has been validated and compared with the State-of-the-Art data with excellent results, and used as the base of innovative System-level applications to identify Aging-aware Path Slack Ranking and proposes an adaptative OPP strategy optimizing both performance and lifetime.

1.4.1.2 Jules Quentin KOUAMO: Software Methods and Architectures for the test of Mixed-Signal SoCs

The complexity of current electronic systems, the production volumes and the quality level required in critical applications, like for instance in automotive, challenge traditional approaches

to testing integrated systems. To overcome these difficulties, the field of testing is currently undergoing a period of profound evolution, dominated by new "Design for Test" techniques which push automation to the very heart of systems. This field has long been dominated by "Scan Test", which is simple and very effective for structural testing, which consists of verifying the integrity of the circuit as a network of nodes. However, these approaches have difficulty in keeping up with the scaling of systems due to their combinatorial complexity. In addition, they are not at all adapted to new design paradigms such as "Design by IP" or to systems that integrate analog modules, the tests of which are often functional and interactive. Often seen by Academics as collections of already existing solutions, Standards are on the contrary very powerful tools for transferring new approaches to industry, where standardization is seen as a guarantee of quality and support. This is particularly true for the IEEE 1687-2014 standard, also known as "Internal JTAG", which for the first time integrates functional testing and dynamic architectures at the very heart of its proposal. Unfortunately, the most innovative and disruptive elements of the standard are not integrated into current CAD flows, which are rather focused on incremental developments with high ROI (Return Of Investment). The importance and weight of Analog in modern System-on-Chip (SoC) continues to grow, but its complex interaction with Digital has not yet been fully explored. Several initiatives are underway, including an attempt to extend IJTAG to analog testing thanks to the IEEE P1687.2 standardization committee. While custom solutions to specific problems exist, the systematic consideration of the peculiar constraints of these areas for a unified and coherent solution has never been addressed. For example, analog testing is dominated by Built-In-Self-Test (BIST), where a hardware component is developed ad-hoc and integrated into the chip, with almost no interaction with the outside. This is of course efficient, but takes a long time to develop and is very resource-intensive.

The work proposed in this thesis aims to develop an infrastructure that will allow for hybrid software and hardware approaches optimized according to the needs of designers. Particular attention will be paid to the life cycle: a Mixed-Signal System must pass through multiple phases of Design, Validation and Test, and each has its own tools and constraints which make information sharing and solution porting almost impossible. An objective of the thesis work is to fill the technological "gap" between these stages and to evaluate the feasibility of a unified approach.

1.4.2 Early Reliability and Security Analysis

Today's computing is a true continuum that runs from IoT devices or smartphones to large, mission-critical data center servers, often performing crucial tasks. In this context, the security and reliability of microprocessor-based computer systems are therefore major challenges. The concepts are closely related: while reliability defines the probability that a system will not be subject to failures, safety guarantees that even in the presence of such failures, the system will not generate any dangerous results. While the first describes a characteristic of the system itself, the second is more focused on the interaction with the environment, and therefore its usage. Faults

affecting hardware components (e.g., microprocessor, memory,) are then propagated through the software and can induce failures such as loss of information, incorrect behavior, up to complete unavailability of the system. All this can be described in terms of reliability or security. These qualities are now mainly quantified through costly and complex Fault Injection or Radiation campaigns. This means that a new campaign must be performed for each software or hardware change. Additionally, at an early stage of design, the final architecture of the microprocessor may not yet be defined. These campaigns can therefore be very long and impact Time-To-Market, especially if reliability levels are not achieved and a correction of part of the system is necessary. This research theme is therefore focused on the research and formalization of alternative approaches allowing early analysis: light from a computational point of view but still precise, it is capable of coherently identifying critical elements of hardware and software. The goal is to obtain a set of tools that can be used repeatedly during the design phase to ensure that the final system will meet the target reliability and security constraints. This is one of the driving themes of my TIMA research group that I have pursued since my thesis work, and which has enabled significant scientific advances and also an industrial impact, evidenced by the APP deposit of the EARS software resulting from this thematic and the thesis of Kais Chibani which I co-supervised with Régis Leveugle.

1.4.2.1 Kais Chibani "Robustness analysis of Integrated Digital Systems"

Many applications are today concerned with soft errors, i.e. spurious bit modifications occurring in a circuit at runtime. Such errors can be provoked by environmental disturbances, without any physical defect induced in the circuit. In some cases, they can also result from malicious attacks. No matter their origin, a designer must consider the potential consequences of such errors. It is well known that not all soft errors lead to application failures; the probability of failure strongly depends not only on the target circuit's architecture, but also on the application characteristics, the induced usage of the hardware elements and the execution scenario. The real sensitivity of a circuit (defined here as the probability of failure, assuming a soft error occurred) must therefore be evaluated with respect to a given situation in order to avoid large over-estimations.

The first type of analysis required at design time is an evaluation of the intrinsic sensitivity. If the probability of failure is too large with respect to the application requirements, mitigation techniques can then be applied on the most critical parts. In this work, we focus on the intrinsic sensitivity evaluation, before any specific method is implemented for fault tolerance. However, the analysis must allow a designer to identify the most critical parts for selective hardening. The analysis must also be done early in the design flow in order to reduce the cost of rework or mitigation insertion, when necessary. Preliminary analyses can occur very early, based on the pure behavioral descriptions of the circuit, obtaining what is usually called an Architecture Vulnerability Factor. However, these analyses are usually extremely conservative and tend to over-estimate the sensibility of the system, resulting in high overheads which are not always acceptable. In order to obtain accurate quantifications, the registers actually implemented in the final circuit and potentially subject to soft errors must be known – a Register-Transfer Level (RTL) description is therefore often the earliest used in the studies. In this work, we will assume that

such a description is available, as well as a testbench defining an execution scenario representative of the use of the circuit in the final system. The precise sensitivity is usually evaluated by means of fault injection campaigns, which require a specific set-up, can be very long and expensive and can be performed only when the final circuit is available. As a result, if a major weakness is found, a redesign iteration can be extremely long and directly impact the whole project's timeline.

In this PhD, the focus is to avoid fault injections and the need for specific equipment or skills. The aim is to obtains tools and strategies that can be applied early in the design flow to obtain robustness estimations which can help the Designer's choices. The results are not meant to give a precise value of the system's robustness, but rather to identify the components which are more critical, so that they might be selectively hardened. This loop evaluation→hardening should be lightweight enough to be repeatable several times in the Design phase to obtain incremental gains and increase the confidence in the system's robustness. To estimate the "criticality" of an element, this work relied on the most classical metric, i.e. lifetime of data in the registers. At a given moment in time, not all flip-flops in a circuit contain useful data, while data re-used after a large number of clock cycles has a higher probability to be corrupted by random disturbances than some piece of information used only during a few cycles. So, the more often a register is "alive", i.e. it contains data that will be reused later, the more critical it will be.

The work started by making an architecture analysis of an existing open source processor, the Leon 3: we built a model of its Pipeline with a particular emphasis on the data transfer between registers (both visible to the User or hidden in the Architecture), and used traces from an execution to compute the Lifetime of each one of them. This proved the feasibility of the approach and its capability to provide estimate coherent with the State of the Art. We then moved to main part of the PhD, where we extended the approach to generic Digital Circuits expressed in RTL (VHDL). The resulting tool, called EARS, was able to give precise estimations of the criticality of each Register in the circuit with no a-priori information on its architecture by analyzing the simulation traces of a given payload against an internal model of the circuit. As opposed to simulation-based fault injections, only one simulation has to be run, with an important performance gain. The approach was run on the Leon 3 VHDL description for several workloads and consistently provided result coherent with emulation-based fault injections, but at a fraction of the computational effort. The EARS tool has been the subject of an APP depot, and has been reused several times in the context of TIMA's research in this subject.

1.4.3 Design Exploration for Approximate Computing

Approximate computing is a design methodology aimed at increasing the efficiency of electronics systems. As its name suggests, it involves accepting a result which is not necessarily exact, on condition of gaining in terms of energy consumption, calculation speed and/or complexity of the implemented system. Of course, an exact calculation must not be essential for correct operation. The loss of precision must then remain within a "reasonable" margin of error. To situate some of the areas of application, it is of course possible to mention image processing for which, for example, the exact value of a certain percentage of pixels might no impact on the finale result (ex: "light green" or "dark green" in a traffic light recognition). Major application areas currently include classification problems, artificial intelligence, etc. In addition to the relative novelty of

this type of approach (it is still considered "a new calculation paradigm"), the main obstacle to its application comes from the difficulty for a designer to be able to evaluate early enough and effectively which part of the system can be approximated without unacceptably degrading the service provided to the user. This new theme, coming directly from my Visiting Professor period of 2018-2019, is a synergy between TIMA's own themes (most notably, the EARS tool developed by Kais Chibani) and the research of the Politecnico di Torino focused on the statistical study of the effects of local approximations on the overall result. In the literature, the choice of elements to be approximated is always left to the expertise of the developer through qualitative and ad-hoc analyses: if the error measured is unacceptable, the only choice is to start from scratch, with a considerable cost. In this theme, possible thanks to the complementary skills of TIMA and Politecnico, we reverse the problem and aim at the formalization and development of automated methods and tools capable of analyzing a system and identifying the circuit areas that can be approximated with the least impact on the final result, with considerable savings in time and resources during specification and design.

1.4.4 Autonomous deployment of Small Cells

When deploying dense cellular networks, Quality of Service (QoS) depends not only by the Area Coverage, but from a multiplicity of dynamic parameters, such as for instance the user density, their usage behavior or their movement patterns. These conditions are extremely difficult to estimate through a priori models. Moreover, in the early 2000's a new deployment paradigm started being developed by actors such as Alcatel-Lucent: Small Cells. The correct term would rather we "a base station inside a small box": a PC-sized cabinet able to support a small number of users (usually a few dozens) in a restricted area. The concept has been pushed even further with the introduction of "femto cells", small boxes able to provide 3G coverage in a really small area (usually no more than 20-30 meters) to half a dozen users, and using the ADSL Box as backhaul. These have been commercially distributed under the name of either "femto cells" or "range extenders". As they are supposed to be plug-and-play into unknown environments (usually in close spaces) their centralized management and optimization is close to impossible. Upon me joining Bell Labs Ireland, a lot of work had been done in developing autonomous QoS genetic optimization algorithms, but all were based on centralized Matlab simulations. In this topic, I focused on the porting of these approaches to real-world scenarios by providing an abstraction from the Matlab simulation based on measurable distributed values, and ported a demonstrator on an actual Alcatel-Lucent product: a femto cell based on a Montavista Linux and IBM Rational Rose, commercialized by Vodaphone. Even though the results were promising, I was forced to interrupt this line of research when I moved to TIMA and lost access to the Alcatel-Lucent Intellectual Property.

1.5 Publications

ORCID profile: ***https://orcid.org/0000-0002-8284-3823***

- *Journal Publications*

[J.1] L. Anghel, R. Cantoro, R. Masante, **M. Portolan**, S. Sartoni and M. S. Reorda, "Self-Test Library Generation for In-field Test of Path Delay faults," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2023.3268210.

[J.2] **M. Portolan**, E. Valea, P. Maistri, G. Di Natale, "Flexible and Portable Management of Secure Scan Implementations Exploiting P1687.1 Extensions", IEEE Design & Test on 30/9.2021, DOI : 10.1109/MDAT.2021.3117875

[J.3] K. Kannan, **M. Portolan**, L. Anghel ,“ Activity-aware prediction of Critical Paths Aging in FDSOI technologies”, Microelectronics Reliability Volume 124, September 2021, 114261, https://doi.org/10.1016/j.microrel.2021. 114261

[J.4] **Portolan M.**, “Automated Test Flow: the Present and the Future”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (IEEE-TCAD), December 2019, DOI:10.1109/TCAD.2019.2961328

[J.5] **Portolan M.**, Goyal S., Van Treuren B.,, «Executing IJTAG: Are Vectors Enough? » IEEE Design & Test, vol.30, no.5, pp.15,25, Oct. 2013

[J.6] **Portolan M.**, Goyal S., Van Treuren B., Chiang C_H., Chakraborty T. and Cook T.B., « A Common Language Framework for Next-Generation Embedded Testing » IEEE Design & Test of Computers Volume: 27 , Issue: 5 , Pp: 36 – 49, 2010

[J.7] **Portolan M.**, Leveugle R., « A Highly Flexible Hardened RTL Processor Core Based on LEON », IEEE Transactions on Nuclear Science (IEEE-TNS), Volume 53, Issue 4, Part 1, Aug. 2006 Page(s):2069 - 2075

- *Standards:*

1687-2014 - IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device, Electronic ISBN: 978-0-7381-9416-5, DOI: 10.1109/IEEESTD.2014.6974961, 5 Dec. 2014

- *Proceedings in International Conferences*

[1] **M. Portolan**, V. Reynaud, P.Maistri, R. Leveugle, G. Di Natale “Security EDA Extension through P1687.1 and 1687 Callbacks”, , 2021 International Test Conference (ITC21), November 2021, ISBN: 978-1-6654-1695-5, ISSN: 2378-2250

[2] Maistri P., Reynaud V., **Portolan M**., Leveugle R. " Secure Test with RSNs: Seamless Authenticated Extended Confidentiality ", 19th IEEE International New Circuits and Systems Conference (NEWCAS), June 2021

[3] H. -M. von Staudt, B. Van Treuren, J. Rearick, **M. Portolan** and M. Keim, "Exploring and Comparing IEEE P1687.1 and IEEE 1687 Modeling of Non-TAP Interfaces," 2021 IEEE European Test Symposium (ETS), 2021, pp. 1-10, doi: 10.1109/ETS50041.2021.9465438.

[4] M. Laisne , A. Crouch, **M. Portolan**,; M. Keim, H.M. Von Staudt , M. Abdalwahab, B. Van Treuren, J. Rearick,, “Modeling Novel Non-JTAG IEEE 1687-Like Architectures”, 2020 International Test Conference (ITC20), November 2020, Washington DC, US

[5] L. Anghel, R. Cantoro, D. Foti, **M. Portolan**, S. Santoni. M. Sonza Reorda, “New Perspectives on Core In-Field Path-Delay Test”, 2020 International Test Conference (ITC20), November 2020, Washington DC, US

[6] **Portolan M**. et al., “A Comprehensive End-to-end Solution for a Secure and Dynamic Mixed-signal 1687 System”, 2020 International Symposium on On-Line Testing and Robust System Design (IOLTS 2020), Naples, Italy

[7] **Portolan M**., Reynaud V., Maistri P., Leveugle R., “Dynamic Authentication-Based Secure Access to Test Infrastructure”, 2020 European Test Symposium (ETS 2020), Tallin, ESTONIA, 25 mai au 1 juin 2020

[8] **Portolan M**., Rearick J., Keim M., Linking Chip, Board, and System Test via Standards”, 2020 European Test Symposium (ETS 2020), Tallinn, ESTONIA, 25 mai au 1 juin 2020

[9] A. Damljanovic, A. Jutman, **M. Portolan**, E. Sanchez, G. Squillero, A. Tsertov, “Simulation-based Equivalence Checking between IEEE 1687 ICL and RTL”, 2019 International Test Conference, November 2019

[10] **M. Portolan**, R. Cantoro, E. Sanchez, “A Functional Approach to Test and Debug of IEEE 1687 Reconfigurable Networks”, 2019 European Test Symposium, May 2019

[11] A. Savino, **M. Portolan**, S. Di Carlo and R. Leveugle, “Approximate computing design exploration through data lifetime metrics”, 2019 European Test Symposium, May 2019

[12] R. Leveugle, **M. Portolan**, S. Di Carlo, A. Savino, G. Di Natale and A. Bosio, “Alternatives to Fault Injections for Early Safety/Security Evaluations”, Embedded Tutorial at the 2019 European Test Symposium, May 2019

[13] **M. Portolan**, M. J. Barragan, R.Alhakim, S. Mir , “Mixed-signal BIST computation offloading using IEEE 1687”, 2017 22nd IEEE European Test Symposium (ETS), Year: 2017, Pages: 1 - 2, DOI: 10.1109/ETS.2017.7968222

[14] G. Di Natale, M. Kooli ; A. Bosio, **M. Portolan**, R.Leveugle, “Reliability of computing systems: From flip flops to variables”, 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS), 2017

[15] **Portolan M**., “Accessing 1687 systems using arbitrary protocols”, 2016 IEEE International Test Conference (ITC),Year: 2016, Pages: 1 - 9, DOI: 10.1109/TEST.2016.7805839

[16] K. Chibani; **M. Portolan**; R. Leveugle, “Evaluating application-aware soft error effects in digital circuits without fault injections or probabilistic computations”, 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS),Year: 2016, Pages: 54 - 59, DOI: 10.1109/IOLTS.2016.7604672

[17] **M. Portolan**, R. Rolland, “Student-driven development of a digital tester”, 2016 11th European Workshop on Microelectronics Education (EWME), Year: 2016, Pages: 1 - 3, DOI: 10.1109/EWME.2016.7496479

[18] K Chibani, **M Portolan**, R Leveugle, “Application-aware soft error sensitivity evaluation without fault injections-Application to Leon3”, European Conference on Radiation and its Effects on Components and Systems (RADECS'16), 2016

[19] **Portolan M**., “A novel test generation and application flow for functional access to IEEE 1687 instruments”, 21th IEEE European Test Symposium (ETS), Year: 2016, Pages: 1 - 6, DOI: 10.1109/ETS.2016.7519302

[20] K Chibani, M Ben-Jrad, **M Portolan**, R Leveugle, “Fast accurate evaluation of register lifetime and criticality in a pipelined microprocessor”, Very Large Scale Integration (VLSI-SoC), 2014 22nd International Conference on, October

[21] K. Chibani, **M Portolan**, R Leveugle, “Fast register criticality evaluation in a SPARC microprocessor”, Microelectronics and Electronics (PRIME), 2014 10th Conference on Ph. D. Research in, June 2014

[22] K. Chibani ; S. Bergaoui ; **M. Portolan** ; R. Leveugle, “Criticality evaluation of embedded software running on a pipelined microprocessor and impact of compilation options”, 2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2014

[23] Cherubini D., **Portolan M**., “Automatic Equivalent Model Generation and Evolution for Small Cell Networks”, Fourth International Workshop on Indoor and Outdoor Small Cells 2013 (WiOPT), Tsukuba, Japan, Mai 2013

[24] **Portolan M.,** « Packet-based JTAG for remote testing », 2012 International Test Conference (ITC12), Anaheim CA, 4-9 November 2012

[25] **Portolan M**., Goyal S. and Van Treuren B., « Scan chain Securization through Open-circuit Deadlocks », Poster for the 2010 International Test Conference (ITC10), Austin TX, November 2010

[26] **Portolan M**., Goyal S. and Van Treuren B., « Scalable and efficient integrated test architecture », 2009 International Test Conference (ITC09), Austin TX, November 2009

[27] Vanhauwaert, P.; **Portolan, M**.; Leveugle, R.; Roche, P., « Usefulness and effectiveness of HW and SW protection mechanisms in a processor-based system», 15th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2008), 2008

[28] **Portolan M**., Goyal S., Van Treuren B., Chiang C_H., Chakraborty T. and Cook T.B., « A New Language Approach for IJTAG», 2008 International Test Conference (ITC08), San Francisco CA, October 26-30 2008

[29] **Portolan M**., Leveugle R., « A Highly Flexible Hardened RTL Processor Core Based on LEON », 8th European Conference on Radiation and Its Effects on Components and Systems (RADECS 05) – 2005

[30] **Portolan M**., Leveugle R., « Towards a Secure and Reliable System » – 2005 IFIP International Conference on Embedded and Ubiquitous Computing (EUC'2005) – 2005

[31] **Portolan M**., Leveugle R., « On The Need for Common Evaluation Methods for Fault Tolerance Costs in Microprocessors » 11th International On-Line Testing Symposyum (IOLTS05) – 2005

[32] **Portolan M**., Leveugle R., « Operating systems function Reuse to achieve Low-Cost Fault-Tolerance », 10th International On-Line Testing Symposyum (IOLTS04) – 2004

[33] **Portolan M**., Leveugle R., « A Context-Switch Based checkpoint And Rollback Scheme » –XIX Conference on Design of Circuits and Integrated Systems (DCIS 04) – 2004

- *Workshops et Posters*

[W1] "Targeting Approximation through Data Lifetime: A Quest for Optimization Metrics", A. Savino, **M. Portolan**, S. Di Carlo and R. Leveugle, AxC 2019 : Fourth Workshop on Approximate Computing, May 2019

[W2] **M.Portolan**, R.Cantoro, E.Sanchez,M. Reorda, "A Functional Approach to Test and Debug of IEEE 1687 Reconfigurable Networks", 2018 International Test Conference, October 2018

[W3] K. Kannan, **M. Portolan**, L. Anghel ,"Run-Time Aging Prediction Through Machine-Learning", ", 2018 International Test Conference, October 2018

[W4] **M. Portolan**, M. J. Barragan, H. Malloug, S. Mir, "Interactive Mixed-Signal Testing Through 1687", First International Test Standards Application Workshop (TESTA'16)

[W5] **Portolan M.**, Goyal S., Van Treuren B. « A New Execution Model for Interactive JTAG Applications », 2013 European Test Symposium (ETS13), Avignon, France, May 2013

[W6] **Portolan M.**, Goyal S., Van Treuren B., Chiang C_H., Chakraborty T. and Cook T.B., « A new description language for SoC testing », 2008 European Test Symposium (ETS08), Verbania, Italy, May 25-29, 2008

[W7] **Portolan M.**, Goyal S., Van Treuren B., Chiang C_H., Chakraborty T. and Cook T.B., « A Novel Hardware Description language for efficient debug and diagnosis of digital circuits », 2008 IEEE International Workshop on Silicon Debug and Diagnosis (SDD2008), San Diego, CA , April 27- May 1st, 2008

- *National conferences*

Portolan M., Leveugle R., « Réalisation d'une Tolérance aux Fautes à Bas Coût dans les SoCs en Utilisant le Système d'Exploitation » – Actes des Journées Nationales du Réseau Doctoral de Microélectronique – 2004

- *Invited Presentations*

Anghel L., **Portolan M.**, Managing Wear out and Variability Monitors: IEEE 1687 to the Rescue, Keynote talk, East West Design and test Symposium, Yerevan, ARMENIA, 13 au 16 October 2016

Portolan M., "Standards: Can they co-exist for System Level Test?, Invited Talk", VLSI Test Symposium, Las Vegas, NE, UNITED STATES, 25 au 27 avril 2016

Portolan M., "Flexible and Extendable System-level JTAG Manager", Invited Talk, International Test Conference, Anaheim, CA, USA, October 2015

1.6 Patents and Software Filings

Only the original filings are listed: for a complete listing of international extensions, please referrer to the Lens aggregator. (102 items in October 2023) *https://www.lens.org/lens/profile/305697555/patent*

Patents for the MAST Software

[P1] Application : FR1754491A·2017-05-19 Publication : FR3066606A1·2018-11-23

Patents Granted in the field of Testing

[P2] “Method And Apparatus For Describing And Testing A System-On-Chip”, US Patent 7,958,479, June 2011
[P3] “Method And Apparatus For Describing Components Adapted For Dynamically Modifying A Scan Path For System-On-Chip Testing”, US Patent 7,962,885 June 2011
[P4] “Method And Apparatus For Describing Parallel Access To A System-On-Chip”, US Patent 7,949,915 May 2011
[P5] “Apparatus And Method For Isolating Portions Of A Scan Path Of A System-On-Chip”, US Patent 7,958,417 June 2011
[P6] “Apparatus And Method For Controlling Dynamic Modification Of A Scan Path”, US Patent ,7,954,022 May 2011
[P7] “Method And Apparatus For Providing Scan Chain Security, US Patent 8,495,758 July 2013
[P8] “Method And Apparatus For System Testing Using Multiple Instruction Types”, US Patent 8,533,545, September 2013
[P9] “Method And Apparatus For Virtual In-Circuit Emulation”, US Patent Number 8,621,301 December 2013
[P10] “Method And Apparatus For System Testing Using Multiple Processors”, US Patent 8,677,198
[P11] “Method And Apparatus For Position-Based Scheduling For JTAG Systems”, US Patent Number 8,775,884, 8 Jul 2014
[P12] “Method And Apparatus For Deferred Scheduling For JTAG Systems”, US Patent Number 8,719,649, May 2014
[P13] “Packet-Based Propagation Of Testing Information”, US Patent number 9,341,676, May 17, 2016
[P14] “Systems and methods for dynamic scan scheduling”, Michele Portolan, Suresh Goyal , Bradford Van Treuren, US Patent Number 9,183,105, November 10, 2015

Patents Granted in the field of Telecommunications

[P15] “A Telecommunications Network, And A Method Of Configuring Nodes Of A Telecommunications Network, EU Patent EP2346209, Mars 2013
[P16] “Device and Method for transmitting samples of a digital baseband signal, EU Patent Number EP2683102, Avril 2014
[P17] “Apparatuses, Methods And Computer Programs For A Remote Unit And A Central Unit”, EU Patent Number EP2720429A1,

[APP1] **APP filing for the NeSLT/MAST softwares**

- V1 : IDDN.FR.001.260016.000.S.P.2015.000.10600

- V2 : IDDN.FR.001.260016.001.S.P.2015.000.10600

- V2.1 : IDDN.FR.001.260016.002.S.P.2015.000.10600

- V3 : IDDN.FR.001.260016.003.S.P.2015.000.10600

APP filing for the EARS software

[APP2] IDDN.FR.001.530007.000.S.P.2016.000.10600, Décembre 2016

2 Motivations and State of the Art

The world of Testing is under constant pressure: arriving at the very end of the Design cycle, the Testing phase has a direct impact on Time-To-Market and the final quality of the product. This pressure to deliver is one of the most fascinating aspects of the domain, but also one of its limiters. Actors tend to "use what works" and are extremely conservative toward new approaches. There is little time and desire to try and understand "why" something is done, and "what if" things were done differently. Similarly, failure is not an option for a Test Engineer: faced with a problem, he/she will always find some way to solve it, and stick to this workaround even if it is cumbersome or unstable "because it does work".

In my work, I focused exactly on these "why"s and "what-if"s to find abstractions and solutions that could be applied as widely and generally as possible without custom workaround or patches. In this Chapter, I will provide an analysis of the current State of the Art with a particular emphasis in the missing pieces, which will then be covered in terms of both Abstraction and Implementation in Chapters 3, 4 and 5.

2.1 The Automated Test Flow

Automation is at the core of testing: the sheer size of modern systems, as well as the need to guarantee quantifiable quality in a reasonable test time led to the development of a rich, codified ecosystem, depicted in Figure 1

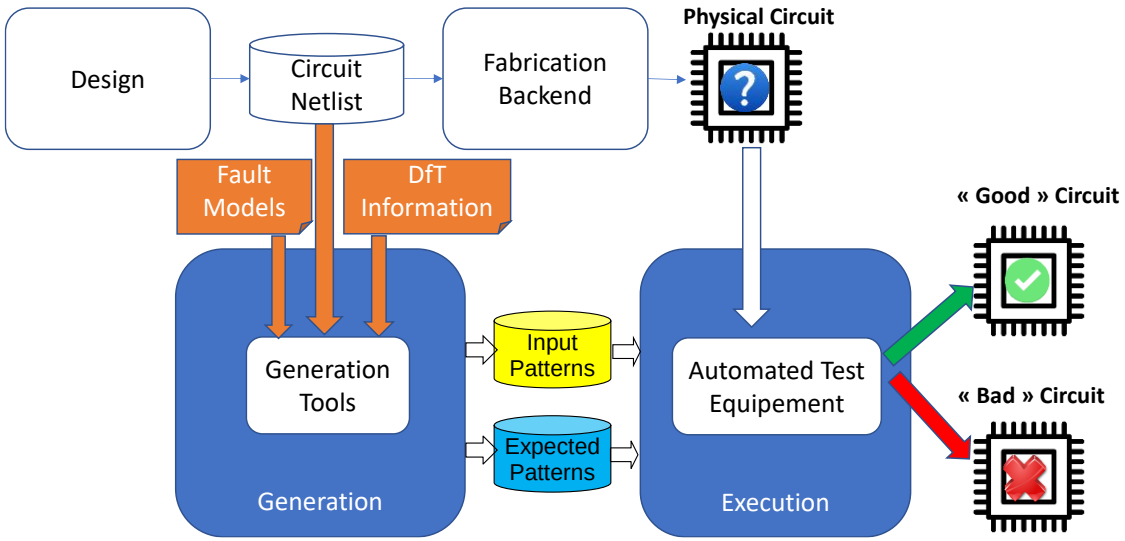


Figure 1: The Ecosystem of Automated Testing

In the upper part, a simplified view of the Implementation flow going from Design to the Physical Circuit is given, highlighting the intermediate Circuit Netlist. Coming after Synthesis, it is the first step where the system is seen as graph of interconnected logical nodes (typically, flip-flops and Boolean gates). It is the abstraction level where Fault Models can be defined: they are a logical representation of physical defects, expressed as "faults" in their Boolean functionality. The most famous and widely used is the "Stuck-At" model: a certain node is stuck to a logical value ('0' or '1') and is therefore unable to perform the desired operation. An automated tool applies Fault

Models to the System Under Test and computes the set of operations needed to both Activate and Detect those faults through a process named Automated Test Pattern Generation (ATPG). In this context, a "Pattern" or "Vector" is a set of binary Input values that can be applied to the SUT, and the expected Output values that a "sane" system should produce. These sets of patterns are then applied to the SUT through dedicated Automated Test Equipment (ATE). In this scheme, all intelligence is regrouped in the Generation phase, while Patterns are simply a collection of static vectors to be applied by the ATE. This process is optimized for factory testing, where the Key Performance Indicator is speed: testing time must be minimized to reduce costs.

The greatest value of ATPG is its quantitative nature: it can precisely compute the number of possible Faults a system might encounter, which one are "covered" by a set of Patterns and which one are "untestable". These precise coverage metrics are precious because they provide a direct measure of the effectiveness of the ATPG algorithm, something that is not possible with functional testing. However, pure ATPG rapidly hit a computational limit: Test Point Erosion [iNEMI09]. Following Moore's law, the density of circuit has been growing exponentially over the last decades, but access capabilities (i.e. the number of possible Input and Output pins) has been growing at a much slower scale. As a result, there is a big Controllability and Observability problem: it is more and more difficult to properly set the value of deeply integrated nets and/or to observe their values from the Functional Inputs and Outputs. This directly impacts ATPG algorithms, with coverage rates dropping and Pattern set size exploding. The solution is what is called Design-for-Test (DfT): the design is modified to boost its testability while maintaining the same functionality and impacting as little as possible its performances. Among the wide range of DfT solutions, we will focus on the most widely used: Scan Testing and Built-In-Self Test (BIST).

Scan Testing, represented in Figure 2, is a direct solution to Test Point Erosion [AGRA84]. A traditional circuit, depicted in the upper half is composed to both Combinatory and Sequential logic, and is accessed through its primary Inputs and Outputs. In order to Activate a fault in the combinatorial logic, a value must be loaded into one or more Sequential memorization points (represented as D Filp-Flops in the Figure) from the Primary Inputs PI. Similarly, to Detect a fault the result of the Combinatorial logic must be saved in the FF and read from the Primary Outputs PO. These routings PI→FF and FF→PO are what cause Test Point Erosion as they can be extremely difficult to compute and might require pipelining through several cycles.

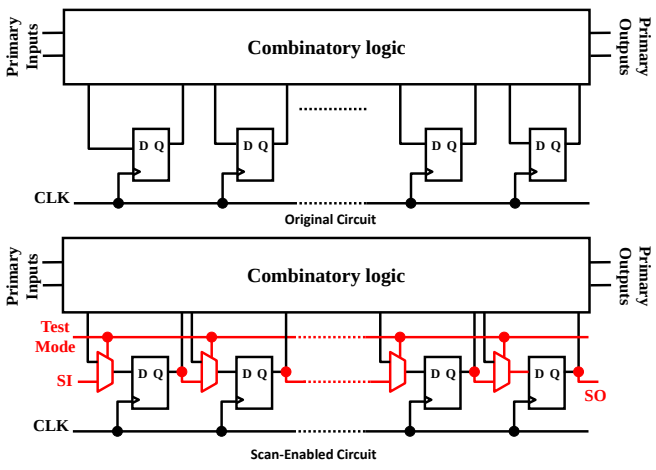


Figure 2 Example of Scan Insertion

To solve the issue Scan Testing, depicted in the lower half, applies a simple principle: through the introduction of some multiplexer and a control signals, all Flip-Flops can be connected serially to form a shift register. The strengths of this approach are that its cost is low (a few muxes and additional pins) and that both controllability and observability are total, as any FF can be directly accessed without going through the Combinatorial logical. The downside is performances: a "full-scan" of a modern design might be composed of thousands of FF, requiring a huge number of cycles to completely shift it. Moreover, as the Scan Control signals need to cover the whole circuit, timing closure is difficult and Scan Clocks are usually significantly slower than the Functional Clock. To boost performances in factory testing, several shorter Scan Chains are implemented in parallel: this requires adding specific Test Pins, but this is usually not an issue at die-level, where the access capabilities are higher than after packaging. Solutions like Scan Compression [KAPU08], which allow accessing several chains in parallel though a reduced set of scan interfaces are now commonplace.

On the other hand, Built-In-Self-Test tackles the issue of Scan Test Erosion in the opposite way: instead of enhancing access capabilities, new components are added inside the system so that it is able to test itself without the need of external communication. The most typical example is Memory Testing: instead of having thousands of vectors writing and reading back values [GOOR90], a Memory BIST (MBIST) is a component able to generate and compare them on the fly [WEST81]. Similarly, Logic BIST (LBIST) [KELL90] allows testing of embedded logic. These approaches allow for high fault coverage with little or no data exchange: they just need to be initialized (when needed), triggered and then their results can be collected.

Scan Testing and BIST are two faces of the same medal are complementary: any modern system will implement both to achieve its coverage requirements.

2.2 The Role of Standards and Patents

The world of testing is composed by a variety of actors, each with his own specialization. For instance, DfT insertion might be done by a team, test generation by another and the final testing on actual circuit by yet someone else. Each one's contribution might come at different times and places, due to the long time taken for projects and to the distributed nature of the of the electronics supply chain, and with different EDA Toolchains. In reality, things are even more complex: the final system is nowadays composed by several third-party IPs, each with its own Design, DfT and EDA Toolchain choices, but each (supposedly) inter-operable. The only way to make such a complex setup work is for everyone to speak the same language: standardization. Defined in the Shorter Oxford English Dictionary as "A document embodying an official statement of a rule or rules […] having a recognized and long-lasting value" a Standard is an irreplaceable tool in Engineering: by agreeing on a common set of rules and best practices, the actors are guaranteed inter-operability of their proprietary solution. By being standard-compliant, a company can have a reasonable guarantee of market acceptance for a new product, and therefore justify the investment to develop it. But the impact goes far beyond marketing: a successful standard can shape its technological domain for years, guiding not only Product development but also Research directions. When working in technologies at their mid-TRL stages [TLR-EU] [TLR-NASA], both Companies and Academia will be more willing to invest time and resources to

develop standard-related technologies whose acceptance is almost guaranteed. As such, a standard is usually the synonym of a whole ecosystem of IPs, Software and best practices: Users tend to avoid non-standard technologies that would force them to leave the comfort and safety of such an environment. The effort of using custom solutions is usually so high to nullify its potential advantages.

In all technological fields, there are solutions and best practices that are so effective and widespread that become de-fact standards. The most famous example in testing is Scan Testing: virtually any digital circuit includes one or more scan chains because it is the only way to reach a satisfying coverage rate. Each Toolchain implements it in equivalent ways, but each solution will be slightly different and not necessarily compatible. For instance, a Scan Chain inserted by Synopsys's Design Compiler [SYNO] will not necessarily be exactly the same as one inserted by Siemens EDA's Tessent [SEDA]. However, this is not necessarily an issue: to reduce the complexity of the flow, a Designer will tend to use the same Toolchain from DfT Insertion to Pattern generation. The need for inter-operability comes after: one the one hand if the Design is self-contained and is fabricated, the Patterns must be accepted by any ATE. On the other hand, if the Design is just an IP to be inserted in a bigger system, the two DfT must be compatible. Traditionally, this meant that the DfT for all IPs in a system was developed using the same Toolchain. While this is reasonable for a system where all IPs are developed in-house, the widespread usage of IP-based design is putting a serious strain on this usage model for two reasons. The first is that as IPs grow in size and complexity, Third-party Providers tend to implement their own DfT. This allows for better testability, and also allows the Provider to reveal as little as possible about its IP internal implementation. Strictly related, the second is that there is no guarantee that the IP Provider will have used the same Toolchain as the System Integrator. Unfortunately, the de-facto standard paradigm is not valid in this situation.

In the domain of Testing, formal Standards are defined by Standardization Bodies, i.e. entities that handle the Standard Development process and publish the Reference Documents. In the world of Electronics, the main standardization entity is the IEEE Standard Association [IEEESA]: its role is to centralize efforts by providing a process to propose, develop and publish reference standards. The development itself is carried out as a volunteering work by contributors, which can come from both Industry and Academia. The process is well documented and controlled by a set of bylaws, the most important assuring that no individual or company might use a standard as a way to obtain an unfair advantage over the competition. The usual steps are:

- People interested in developing a new Standard can ask the IEEE to open a Study Group. This is an informal group where members can discuss their ideas and objectives. The goal is to come up with a PAR (Project Authorization Request) to the IEEE. It must "define the scope, purpose, and contact points for the new project." [IEEESA].
- Upon approval, a Working Group (WG) is formed, and the project is assigned a number. This can either be a sequential number (ex: P1687) for a completely new standard, or a "dot" number for a new member of an existing standard family (ex: P1687.1). The "P" stands for "Progress", meaning that the standard has not been approved yet.

- A PAR is valid for a given time, usually two to three years. The WG can ask for one or more Extension if they can justify the reasons for the delay (for instance, new problems encountered or the need to finalize editing).
- The WG is supposed to write a Draft of the Standard document. Following a precise template and editing rules, a Draft is composed by Explanatory parts to help understanding the contents and by the Rules to be followed.
- A WG is led by three elected officers: the Chair and Co-Chair, who lead the discussions and are responsible for interfacing with IEEE, and the Secretary, who has to produce the official Meeting Minutes and maintain the Attendance Record and Voting Rights. Optionally, there can also be an Editor to coordinate the writing of the Draft. The mandate for Officer is two years, with no limits for re-election.
- When finished, the Draft is submitted to IEEE SA to be put under Ballot. The Ballot is a group of volunteers not related to the WG who can vote the acceptance of the Draft, and who usually ask for corrections and clarifications.
- When the balloting process ends and all corrections have been considered, the Standard is released. The "P" is dropped and the publication date is added, obtaining names such as IEEE 1687-2014.
- A Standard is valid for 10 years. After this period, it can either become inactive if it has not been successful, or it can be Renewed for another 10 years. The renewal process is similar to the Proposal: a PAR is issues and a WG formed to propose amendments and corrections, and then the new draft is put to ballot. The new standard will have the date changed: for instance, IEEE 1149.1-1991 has been renewed twice as 1149.1-2001 and 1149.1-2013 [1149.1]

The actual content of a Standard changes of course greatly depending on its subject. However, one important point is often overlooked: their innovation potential. Researchers, especially from Academia, tend to look at Standards as a collection of existing best practices, and as a way for Companies to push their solutions as "the" solution for the market. Even though there is a part of truth in that statement, that is only one small part of the big picture. First of all, Standards are supposed to shape their field for at least one decade: they need to provide solutions not only for today, but more especially for the future. As such, there is often the need to compare existing technologies, find their strength and weaknesses and come up with a common, future-proof solution. This can demand an important effort: for instance, the IEEE 1687-2014 [1687] standard took more than nine years in the making, from the first mentions [REA05] to the final publication. Second, Standard entities have safeguards against companies gaining unfair advantages by including proprietary technology. Working Group members are required to disclose the existence of any Patent they might be aware of, and the Patent Holder have then to file a Letter of Intent (LoI, for standards in their Draft stage) or a Letter of Assurance (LoA, for active standards). This document engages the Patent Holder to provide licensing to the protected technology in "Reasonable And Non-Discriminatory"" (RAND) term, i.e. with reasonable fees and without using them as a lever for excluding competitors. Failure to comply result in the technology being removed from the standard, regardless of its technological value. In fact, companies have two complementary interests in having patents included in a standard:

- One of the most difficult and expensive parts in Patent protection is proving infringement, as the burden of providing proof is on the plaintiff. This might require a lot of time and effort,

and sometimes might even result in the original patent being declared null. It is a high-budget and high-risk operation, which is why patent infringement fees are usually extremely heavy. On the other hand, when a patent is included in a standard, any compliant product will automatically be using it: the infringement is implicit, with no need to prove it. The more the standard is successful, the more potential infringements. This means that the Holder can be satisfied with low royalties, but repeated over a huge number of infringers. A famous example is Qualcomm, whose patents are at the heart of the Code Division Multiple Access (CDMA) technology, which is the foundation of the 3G cellular network standard from [3GPP]. As any chip and company involved with 3G Cellphone had to pay royalties, the small fees added up to a significant amount, so much that someone even doubted the fairness of Qualcomm's interpretation of "RAND" [CHA20]

- When a company files for a patent, it means that they are expert enough in that field to develop something new. So, apart from the direct gains from royalties, having a patent in a standard gives the Holder a strategic advantage: they already know and have potentially implemented the technology, so they can directly use it from Day One of the Standard's release. This is probably the main reasons for Companies to be involved in these activities, so much that they are willing to accept more stringent patent policies that in the past.

To conclude this section, we will give a small introduction to the most important Standards in the field of Testing.

2.2.1 JTAG

The most famous scan-based Standard is undoubtedly IEEE 1149.1 [1149.1] that is best known as JTAG, the acronym of the "Joint Test Action Group". Its full name "IEEE Standard Test Access Port and Boundary-Scan Architecture" perfectly resume its goals: provide an easy and portable way to access the Boundary of devices. It has been developed in the 1990's, when board-based systems were rapidly growing in size and new mounting technologies such a Ball Gate Array (BGA) [KAP99] becoming widespread. Engineers were confronted with what is commonly called "Test Point Erosion": as system became larger, it was becoming more and more difficult to get physical access to the pins to test that their soldering was correct and the PCB traces were operational. BGA is the best example: connectors are hidden between the chip and the board, as can be seen for instance in Figure 3, making physical access impossible.

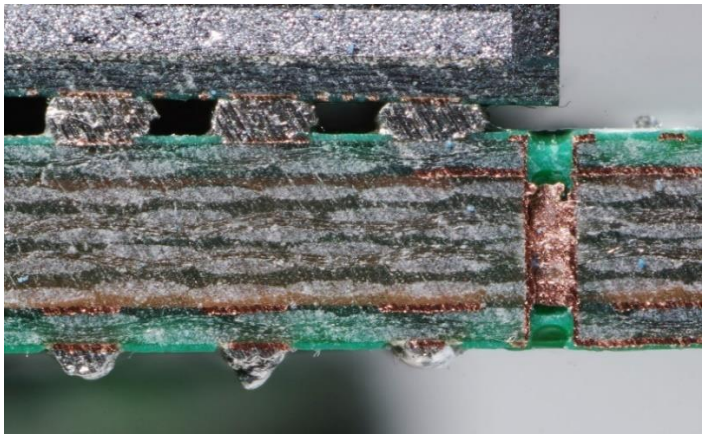


Figure 3 Cross-cut section of BGA mounted circuit (from Wikipedia)

JTAG solved the problem by providing logical access to the connection pins from the inside of the device thanks to a standardized Boundary Scan Chain, as depicted in Figure 4

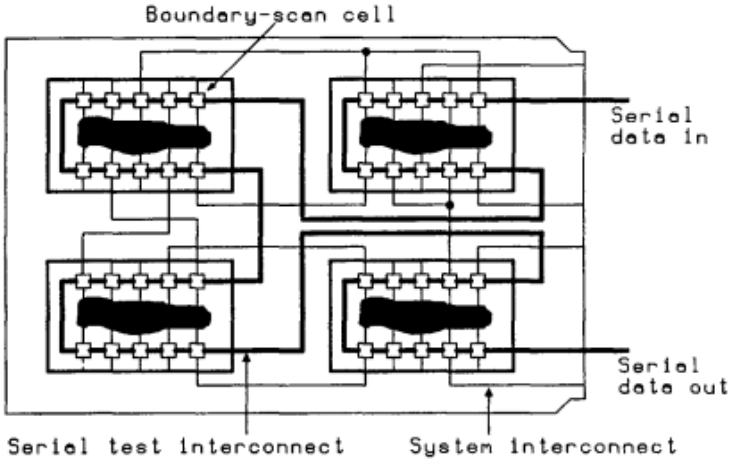


Figure 4 A boundary-scannable board design", from [1149]

2.2.1.1 JTAG Internal Architecture

The idea is simple and yet effective: instead of physically accessing the pins, JTAG allows the user to drive logical values '1' or '0' on output pins and read them back on input pins. Inside a device, a JTAG architecture look like Figure 5, and it composed by four main elements: the Boundary Scan Register Cell (BSC), the Instruction Register, the Test Access Port (TAP) and the TAP Controller.

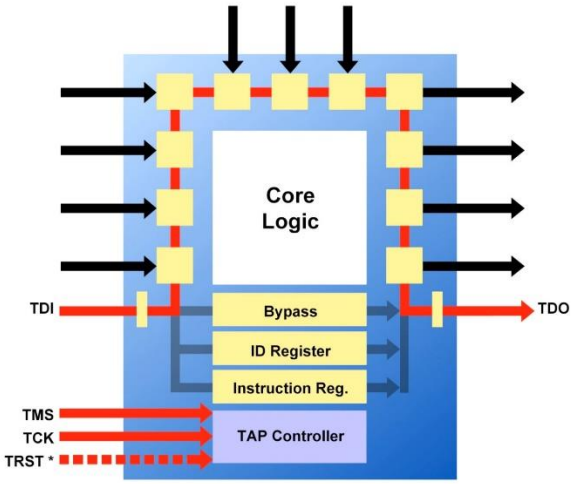


Figure 5 JTAG standard architecture (source JTAG Technologies)

The BSC, depicted in Figure 6, is the key element of JTAG: it applies the same principle as internal Scan Testing but with a key difference: the cell is actually composed by two registers. The first one is responsible for either Loading data from the Signal-In or Shifting data received from the Scan-In. The second one's role is to provide a stable value to Signal Out when the test Mode is selected. In Figure 6, the value of the two registers is updated through a rising edge of ClockB,

but that is just an example. Several other designs of a BSC are possible, for instance having just one clock using a Capture and an Update signal to control the loading of the first and second register respectively.

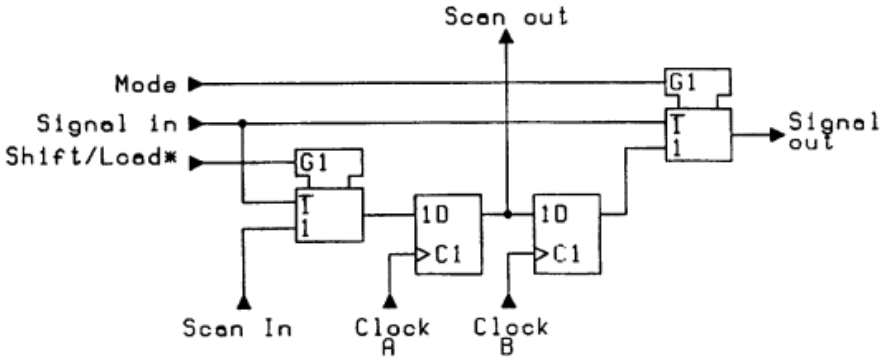


Figure 6 An example of boundary-scan register cell, from [1149]

The Test Access Port is the interface with the outside world and is it composed by 4 compulsory pins and 1 optional one, depicted in red in Figure 5:

- Test Data In (TDI): Input port for the Data
- Test Data Out (TDO): Output port for Data
- Test Clock (TCK): the clock source used for the JTAG infrastructure
- Test Mode Signal (TMS): the signal used to operate the JTAG infrastructure
- Test Reset (TRSTn): optional reset signal, active low.

The Instruction Register is a Scan Register like the Boundary Register, but it has a special purpose. This register is used to operate a JTAG system: depending on its value, the system will behave differently, as explained later in this section.

The TAP Controller is the real heart of JTAG: it is composed by a Finite State Machine (FSM) which is controlled by the TMS and by a decoding circuit connected to the Instruction Register. By driving the FSM with TMS, the User can read/write values in either the Instruction Register or the current Data Register, as shown in Figure 7

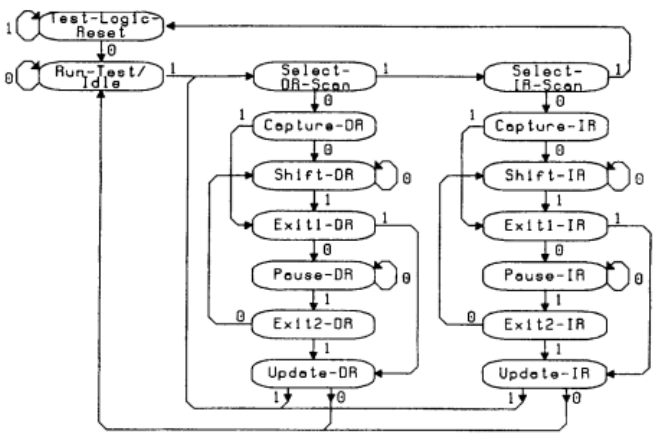


Figure 7 The JTAG Finite State Machine, from [1149.1]

The FSM is divided in two branches: the DR and IR branch, which allow access to the Data Registers and Instruction Register respectively. The operation is the same and it is done in three steps:

- first data is Captured at the input of the Register,
- then this data is Shifted out through TDO while the new data is shifted in through TDI
- last, the register is Updated as with the new value.

2.2.1.2 Operations

Operations are done following the scheme of Figure 8: the JTAG-Wrapped Design Under Test (DUT), is connected to the outside thanks to the TAP. On the Test Host, a JTAG controller implements a "Master" FSM whose role is to generate the right sequence of TAP signals (TCK and, most importantly, TMS) to drive the "Slave" FSM of the DUT through the desired states.

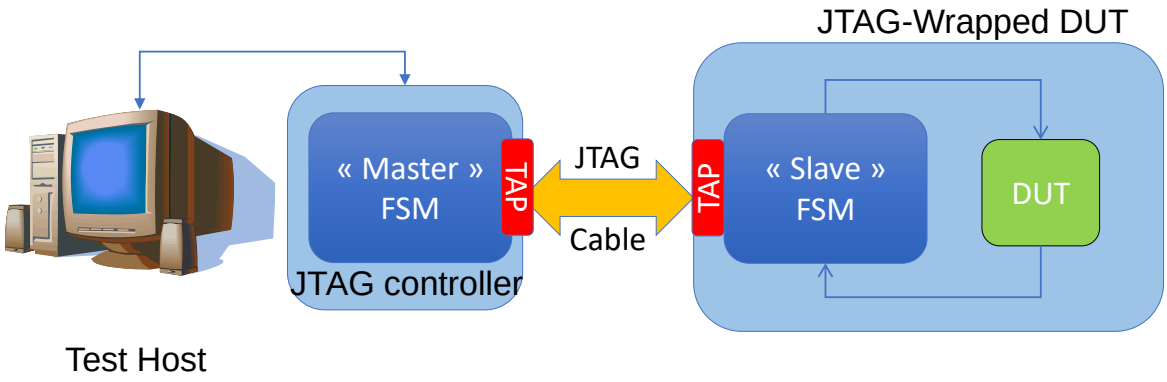


Figure 8 JTAG Usage Setup

The Test Host can be not only an ATE, but also a Desk-Top computer using a JTAG controller (cheap USB dongle cables are now commonplace [FTDI]) or an Embedded Controller. The synchronization between Master and Slave FSM is the key of operating a JTAG system: the Controller must at all times know the state of the DUT to drive it correctly. The Standard does not offer any type of introspection in the DUT state: if by any chance the synchronization is lost the only solution is to reset the system to bring it back to a known state. To reduce the occurrence of such situations, JTAG imposes several restrictions to the DUT's architecture, the main being: Registers must have a constant length, known in advance and documented. This greatly simplifies the DUT state tracing, but it turns out to be one serious limitation, as will be explained in the following sections.

Basically, a JTAG infrastructure can do two things: shift data to/from Data Registers or perform Instructions. The first is done through the DR branch: by exploiting the ShiftDR state, any number of bits can be shifted through the active Data Register using the Capture-Shift-Update (CSU) protocol. A JTAG system usually has several Data Registers, selected through the IR, the most important being:

- The Boundary Scan Register, BSR, selected by several instructions such as EXTEST or INTEST;

- A one-bit Bypass Register, selected by the BYPASS instruction. Its role it to reduce the length of the total scan chain when the current device is not under use, while guaranteeing that there are no unbounded paths between devices (which could happen with a direct TDI-TDO connection
- The ID Register, selected by the IDCODE instruction, which contains a unique Identifier Code. This allows identification for the current device without visual inspection
- Optionally, any number of USER registers can be defined and connected. They are not part of the JTAG operations, but can be used to add new non-fully compliant features.

Apart from selecting the current DR, the Instruction Register can also be used to perform operations in the DUT or system. The most important are:

- EXTEST : The BSR is selected, and it is connected (input and outputs) to the external pins of the DUT. This allows testing the PCB connections between JTAG-enabled devices.
- INTEST : the BSR is selected, and it is connected (input and outputs) to the internal DUT. This allows testing patterns to be applied to the device to test that it has not been damaged during assembly
- RUNBIST : This instruction allows the usage of internal BIST components, most notably by generating an internal clock. If several BIST are present inside the DUT, they might need to be selected/activated first using other Instructions.
- USER INSTRUCTION: these care custom instructions that the User can define to either access User Registers or trigger specific actions.

Please note that the Standard only defines the Instruction names and their role, but not their actual binary mapping, which is up to the Designer.

2.2.1.3 Boundary Scan Description Language (BSDL)

As seen in the previous paragraph, a JTAG infrastructure can be quite complex and have numerous parameters. For this reason, the Standard comes with its own Domain Specific Language, the Boundary Scan Description Language (BSDL). Its roles are multiple, the most important in this context being the description of the BSR and of the TAP. As a language, BSDL has been developed using the syntactical rules of VHDL: it can be parsed with no error by a VHDL parser, but its semantics are different and need special processing. Figure 9 show a BSDL snippet describing a Boundary Scan Register:

```
attribute BOUNDARY_REGISTER of diff : entity is
-- num cell  port      function safe [ccell disval rslt]
   "9 (BC_1, CLK,       input,   X)," &
   "8 (BC_1, OC_NEG,    input,   X)," & -- Merged input/control
   "8 (BC_1, *,         control, 1)," & -- Merged input/control
   "7 (BC_1, D_Pos(1), input,   X)," &
   "6 (BC_1, D_Pos(2), input,   X)," &
   "5 (BC_1, D_Pos(3), input,   X)," &
   "4 (BC_1, D_Pos(4), input,   X)," &
   "3 (BC_1, Q_Pos(1), output3, X,   8, 1, Z),"&-- Also bussable
   "2 (BC_1, Q_Pos(2), output3, X,   8, 1, Z),"&
   "1 (BC_1, Q_Pos(3), output3, X,   8, 1, Z),"&
   "0 (BC_1, Q_Pos(4), output3, X,   8, 1, Z)";

end diff;
```

Figure 9 Example of a BSR description from [1149]

Syntactically, the rule is incapsulated into a VHDL "Attribute" rule as a character string. Inside the string, each Boundary Cell (BC) is assigned a number and its parameters, defined in the standard document, are set. Semantically, BSDL is most of the times just an enumeration of options and parameters that are defined in the Standard Document. This is even more flagrant when looking at the description of the TAP, depicted in Figure 10

```
attribute INSTRUCTION_LENGTH of My_IC:-- Must be first
                                entity is 4;
attribute INSTRUCTION_OPCODE of My_IC:-- Must be second
                                entity is
      "EXTEST (0011), " &
      "EXTEST (1011), " &
      "BYPASS (1111), " &
      "SAMPLE (0001, 1000), " &
      "PRELOAD(1001, 1000)," &
      "HIGHZ  (0101), " &
      "SECRET (1010) ";
attribute INSTRUCTION_CAPTURE of My_IC:-- Must be third
                                entity is "0001";
attribute INSTRUCTION_PRIVATE of My_IC:-- Optional
                                entity is "Secret";
```

Figure 10 Example of TAP Instruction Mapping in BSDL, from [1149]

The binary mapping of each instruction is encapsulated into the string of the INSTRUCTION_OPCODE attribute: a separate parsing will be needed to interpret it.

To resume, BSDL is a simple yet effective language whose role is to provide a direct description and parametrization of a JTAG system, whose hardware implementation is fixed by the Standard.

2.2.1.4 System-Level Architecture

As shown in Figure 4, the main strength of JTAG is the capability of accessing all the Devices in a board without needing physical access apart from the TAP. In its 30-year history JTAG has been applied in a variety of topologies: the most important and widely used is daisy-chaining, where devices are added in a serial fashion as shown in Figure 11. In such a setup, all IRs are chained together and accessed in one operation, as are all active DRs.

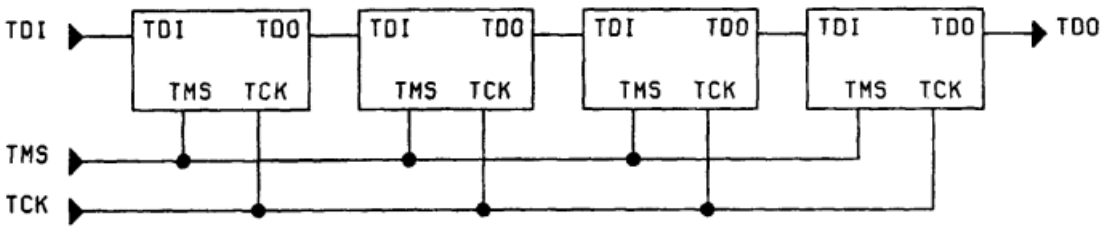


Figure 11 Daisy-Chain JTAG Topology, from [1149]

Upon reset, all Devices are put in Bypass mode, so the Controller can use IR writes to only turn on and select the desired DUTs. This simple solution is in fact quite fragile: if one of the DUTs is faulty or powered off, no access it possible. JTAG also offers the possibility of using a Star topology, where the same TAP is used to access multiple DUTs in parallel, as in Figure 12.

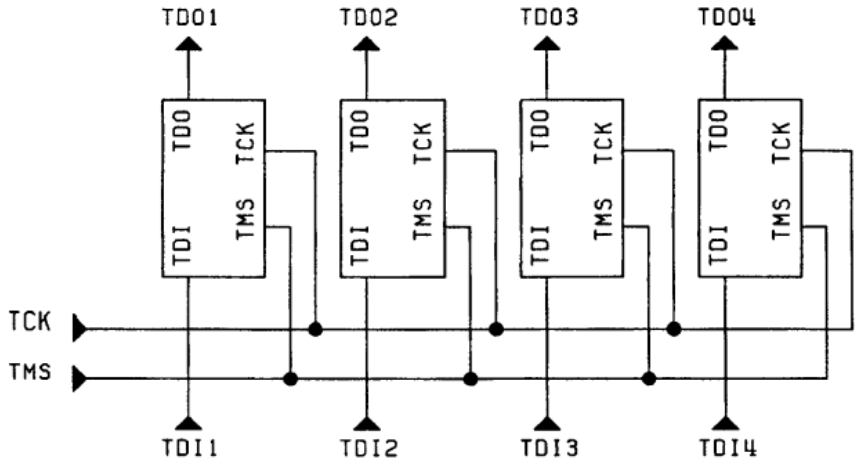


Figure 12 JTAG Star topology, from [1149]

This solution is more robust, but less efficient: each DUT needs its own TDI/TDO. Moreover, the devices all receive the same commands and data: if they are not identical, only one can be active at a time. Of course, all combinations of Daisy and Star topologies are also possible.

The main interest of having standard-supported topologies is reuse: in Figure 1 we showed how Patterns are obtained from the Fault List and the DfT description of a given device. In a System, each JTAG-compliant device will have its set of pre-computed patterns, and the BSDL description of its wrapper. By combining this information is therefore possible to compose Device-Level patterns to obtain System-level one. This process, depicted in Figure 13, is called Retargeting.

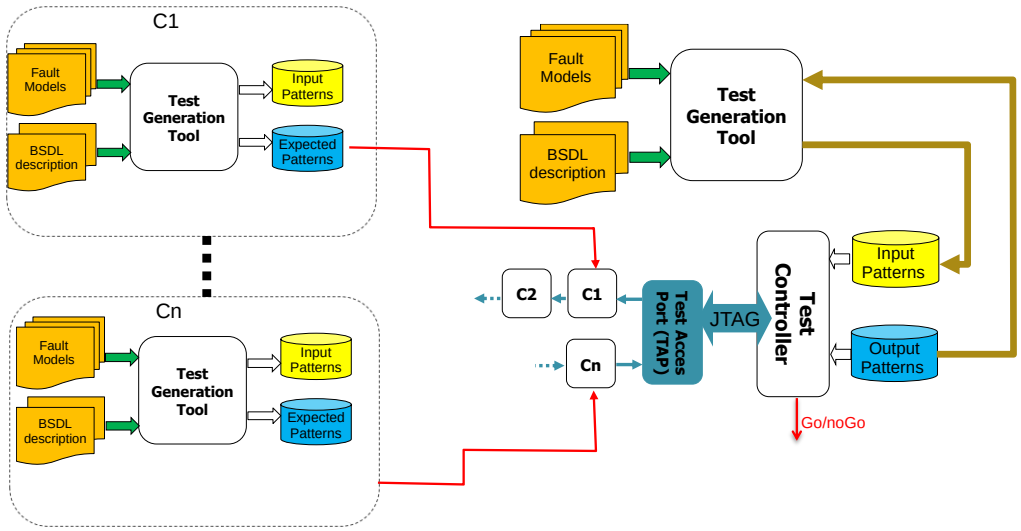


Figure 13 JTAG Retargeting

Please note that JTAG is an interface made for easy accessibility, not for performances: the TAP signals need to travel along the whole PCB board, with important delays. Even inside a chip, control signals need to reach the whole design, down to its boundaries. As such, usual TCK speed are around 10 to 100 Mhz, even in systems where the functional clocks are in the Ghz range.

2.2.2 Domain Specific Languages

As introduced in the previous section, Standards are needed when different actors have to exchange information. As such, a really important part is played by Domain Specific Languages (DSL), which are used to encode information. A complete survey is impossible and of little interest, so we will focus here on the two main languages used to exchange digital Test Patterns: STIL and SVF.

The Standard Interface Test Language (STIL) [1450] is an IEEE standard that is used primarily to provide ATPG patterns to ATE for their execution. As any given SUT can implement a great variety of DfT architectures, test patterns are more than just binary information: they must also describe the protocols needed to apply and interpret the data, in terms of sequences, waveforms, success conditions, error handling, etc... STIL provides this flexibility, while maintaining a simple-enough structure to be easily parsed and interpreted by ATEs. Its usage model is depicted in Figure 14.

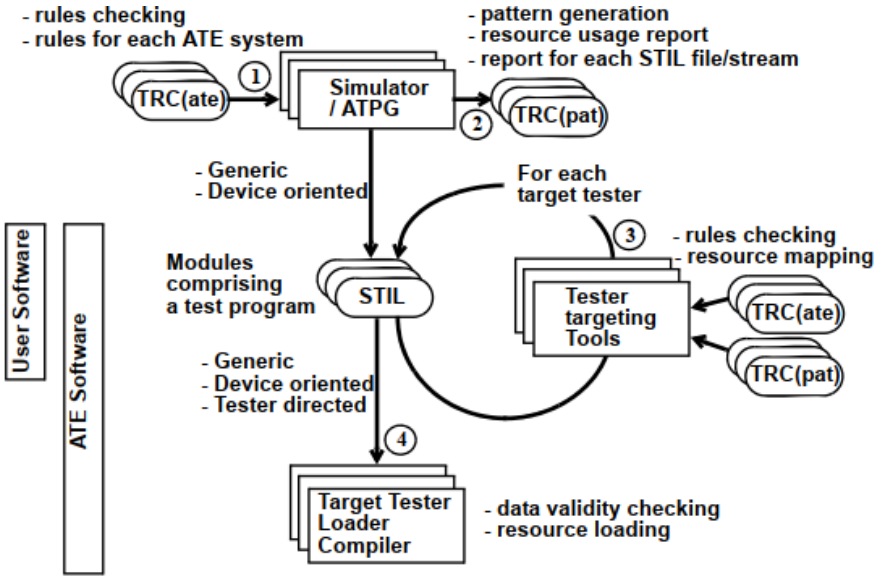


Figure 14 STIL Usage model, from [1450]

In the context of this document, STIL can be considered as the exemplary output format: regardless of the complexity of the internal Design for Test, the final pattern set is supposed to be expressed in STIL. We will see in Sections 5 and 6that this is not necessarily true, even though few people anticipated it.

The Serial Vector Format (SVF) is a de-facto standard language developed by Asset Intertech [SVF99] and universally used to express JTAG operations, of which it provides a one-to-one representation, as shown in Figure 15. It is extremely easy to read, and allows a JTAG expert to immediately understand what the Test is supposed to do. It is the reference Vector language for Board and System testing, where JTAG is the most used interface.

```
!Begin Test Program
TRST OFF;                                   !Disable Test Reset line
ENDIR IDLE;                                 !End IR scans in IDLE
ENDDR IDLE;                                 !End DR scans in IDLE
HIR  8  TDI (00);                           !8-bit IR header
HDR 16  TDI (FFFF) TDO (FFFF) MASK (FFFF);!16-bit DR header
TIR 16  TDI (0000);                         !16-bit IR trailer
TDR  8  TDI (12);                           !16-bit DR trailer
SIR  8  TDI (41);                           !8-bit IR scan
SDR 32  TDI (ABCD1234) TDO (11112222);      !32-bit DR scan
STATE   DRPAUSE;                            !Go to stable state DRPAUSE
RUNTEST 100 TCK ENDSTATE IRPAUSE;           !RUNBIST for 100 TCKs
!End Test Program
```

Figure 15 Example of an SVF program from [SVF99]

However, its lack of flexibility is an issue: SVF is only able to configure the JTAG TAP, push vectors and do bit-wise comparisons. When a User needs to do something slightly more complex, like composing and comparing vectors, he needs to add a lot of custom processing and infrastructure [VTB03][VTB05]. More sophisticated alternatives do exist like the Standard Test and Programming Language [STAPL], but they are seldom used apart in the niche applications they have been developed for. So far, no solution is flexible and comprehensive enough to adapt and cover all the needs.

2.2.3 Evolutions, limitations and new usages of JTAG

JTAG is undoubtedly one of the most successful IEEE standards: it is present in virtually any digital circuit of reasonable size. But it is also extensively used far outside of its original scope and is therefore clearly showing some limitations.

The first strain came from Topology: boards and systems can rapidly become extremely complex and difficult to handle efficiently with just daisy chains or stars. Solutions like [BSCAN2] allows multiplexing using a sort of "TAP of TAPs", while the Brocade selector [LIHN06] uses I2C (often preferred for configuration and setup [I2C14] because of its simplicity) for multiplexing different TAPs. Both solutions are conceptually simple and efficient, but they share the same issue: they are not JTAG-compliant and so they need custom software in addition to standard EDA Tools. What might seem a minor issue has become over the years a big problem: most companies ended up having reals "flows over the flows" with set of custom scripts and languages making pre and post-processing over EDA tools. This code base is not only intrinsically unstable as it depends on assumption on third-party tools, but also extremely complex and time consuming to maintain and debug. As such, in the later decade we have assisted to a big push toward standardization by the designer wishing to free themselves form the burden of maintaining in-house tools. A typical example is the IEEE 1687 Standard, which will be discussed in details in Section 2.2.5.

The second big evolution problem comes from Instrumentation: as detailed in Section 2.2.1.2, JTAG defines a "RUNBIST" instruction that can be used to run internal BIST instruments. This implicitly implied that any DUT would have just a few embedded instruments which could be controlled from the outside: if this was true in the 1990ies, evolution far surpassed the most optimistic expectations. Modern circuits can have dozens of instruments, and this trend can only continue as designs become bigger and technology nodes smaller. As such, having only one "RUNBIST" instruction was not enough, and designer were extremely creative in either defining

new Custom instructions or defining sequences of IR/DR operations to select and run specific subsets of instruments. Although theses sequences can be described as Scan Operations in SVF or STIL, their intent is outside of the scope of test DSLs which can push and receive data but not analyze it apart from basic mismatch. So, the portability of these solutions is extremely limited. Moreover, adding so may complex Custom Instruction and Data Registers can seriously impact the TAP Controller, which can become too big and slow for practical applications.

Last but not least, ironically the biggest problem for JTAG comes from its success itself: being virtually omnipresent in any circuit and having an important Software support, JTAG started being seen as the ideal "entry point", and being applied to usages far removed from its original scope. It is for instance the main interface used for Firmware programming, but is has also been used to interface and test Analog and RF systems. All these usages are completely non-JTAG compliant and need their own software infrastructure. This sometimes resulted in new Standards (for instance IEEE 1532 for FPGA programming [1532]), but more often in proprietary solutions like for instance the "Hierarchical Boundary Scan Standard Language" from Asset Intertech [HSDL] which extends BSDL's topology description capabilities, or the CASLAN language from Goepel Electronics [EHR09] for instrument operations. Some of their features were incorporated in revision of existing standard (for instance, IEEE 1149.1-2013 added to BSDL several HSDL ideas) but most notably gave rise to the two most influential Testing Standards of the 2010s : IEEE 1500 and IEEE 1687.

2.2.4 Core Testing : 1500

The "IEEE Standard for Core Testing" was published in 2005 [1500] and it has been a huge success: a quick search on scientific publication database will immediately show hundreds of hits. And this is without considering the thousands of Designs which simply implemented it. So, what is the reason for such a success?

In certain terms, 1500 is the perfect answer to the perfect storm: at the turning of the century, the principle of IP-based designs became predominant. By dividing a big System into a set of independently-developed IPs, the cost of design is lowered thanks to the reuse of existing IP or by the integration of third-party IP. However, this raised an issue for testing. Traditionally, ATPG is done at the end of the flow on the whole design. This means that it cannot take advantage of IP-based "divide and conquer", and is therefore takes the brunt of increasing complexity. Moreover, running ATPG means having a complete knowledge of the Design: Third-Party companies are willing to license their IP and provide enough information for them to be used, but not to be replicated. As such, providing full netlist for ATPG might be delicate for them.

The new 1500 brought a solution by replicating at IP-level what JTAG had been doing at circuit level: IPs can be Wrapped using a standardized DfT architecture, and the ATPG vectors defined as these boundaries. The similarities between the IEEE 1500 wrapper, depicted in Figure 16, and the JTAG wrapper are undeniable and completely intentional.

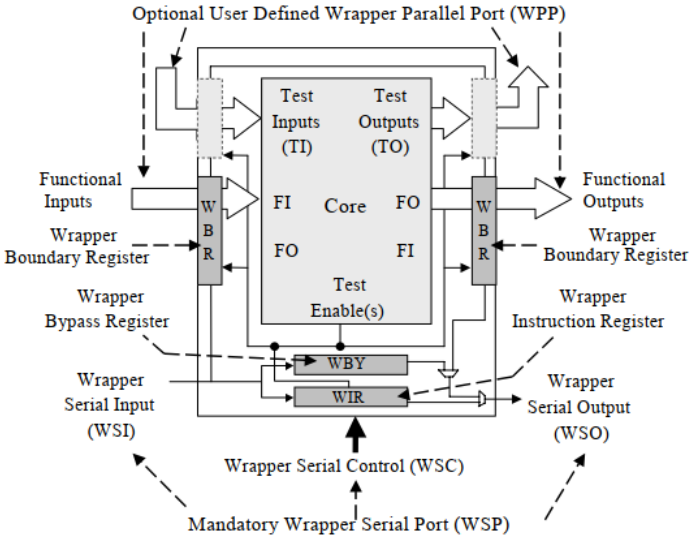


Figure 16 Standard IEEE 1500 Wrapper components, from [1500]

The IP is surrounded by a Wrapper Boundary Register (WBR), and can be accessed through a Wrapper Serial Port (WSP). A Wrapper Instruction Register (WIR) decides whether the BSR or a Wrapper Bypass Register (BPY) is selected. The WIR can also be used to load operational instructions, such as for instance "WS_EXTEST". The access to the WIR is done in the same way as for the WSP our WSP, with the difference that a "select WIR" signal (part of the WSP) must be asserted. The parallel with TAP operations is complete. However, the specific needs of Core-based design necessitated specific additions, as for instance the optional Wrapper Parallel Port (WPP) to accommodate high-bandwidth data transmission, or the definition of the Core Test Language (CTL) to express describe both the Wrapper and the vectors applied to it. This language became soon so complex to become a Standard by itself and was eventually incorporate into the STIL family [1450.6]. Even though a CTL description is formally required in the Standard document for compliance [1500], it is not unheard-of having IP wrapped in hardware but not having a CTL description, especially when they are developed and used in-house.

The IEEE 1500 Wrapper offers several benefits, one of the most important being the possibility of using retargeting: test patterns can be defined at the Wrapper level and then be composed to obtain the system-level vectors. Third-Party Providers can therefore deliver the Test Patterns as part of their IP, and when the same IP is replicated multiple times (as, for instance, the different Cores of a CPU), pattern generation can be done only on one instance and then replicated for the others [MCLA12].

But apart from these similarities, the two standards are in fact quite different. An IEEE 1500 Wrapper is not supposed to be directly connected to an IEEE 1149.1 for two main reasons. First, the wrapper is not JTAG compliant: the "Select WIR" is not part of the TAP, and necessitates therefore some "glue logic" to be generated. And even if this was done, the bandwidth offered by JTAG is much too limited for time-critical tasks as high-volume factory testing. But more importantly, the 1500 Wrapper allows the selection of several registers (WBR, WIR, etc…) of different lengths: this is formally prohibited by JTAG, making it indescribable in BSDL and non-compliant with its Tooling.

For all these reasons IEEE 1500 did not specify a specific interface, but rather left the user to define his own Test Access Mechanism (TAM). In practice, TAMs are usually provided by an EDA Company as part of its commercial package to speed up integration. For instance, Synopsys proposes its own STAR Hierarchical Subsystem [STARSY] to deploy 1500 Wrappers. While these solutions are usually effective and powerful, over the year this has become a serious limitation to inter-operability: while 1500 Wrappers are standardized, the final solution is based on the vendor's TAM, limiting the freedom of the designer to change EDA provider or to accept third-party systems based on different toolchains.

2.2.5 IEEE 1687 or IJTAG

As explained in the previous Section, the IEEE 1500 provides a solution for Core Testing, but sidesteps the Topology issue by leaving the TAM implementation dependent. Moreover, a 1500 Wrapper is not adapted to control internal Instruments such as BISTs: the wrapper is too bulky, and CTL can express rich vector operations but is not able to convey the "test intent" of using an instrument (i.e. it can express "how to run it" but not "what it does").

The "Internal JTAG" or "Instrument JTAG" initiative started in around 2005 to tackle these two issues [REA05], and resulted in the IEEE 1687-2014 Standard [1687]. It has been defined as a "paradigm shift" [REA12] because the solution hinged on two major innovations:

- The topology is not modeled in a top-down approach (i.e. the parametrization of a fixed solution), but rather as a bottom-up composition of base elements;
- For the really first time, the operation is based on a Functional approach: instruments can be operated by read/write operations in custom procedures, while JTAG has always been agnostically shifting bits.

The Standard purposefully decided to be descriptive and allow users to compose their own solutions rather than be prescriptive and define one fixed solution for all. This choice allows 1687 to be extremely flexible and propose a huge number of applications and innovation, several of which unforeseen by their authors. To achieve this goal IJTAG proposes the Use Model of Figure 17, based on two Domain-Specific Languages: ICL and PDL

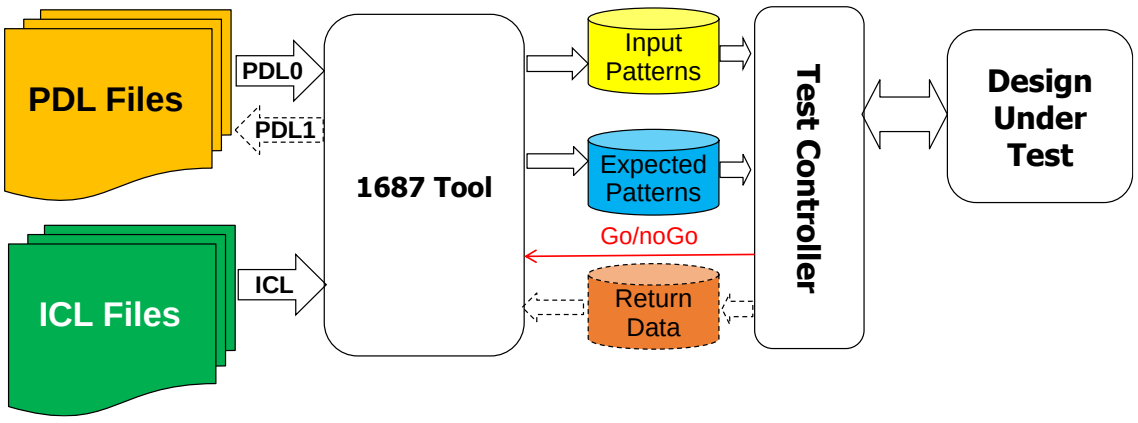


Figure 17 IEEE 1687 (IJTAG) Use Model

The parallel with the ATPG flow of Figure 1 is intentional: IJTAG aims at transporting the know-how of system-level test generation to the IP level: by taking a description of the topology (in ICL) and of the "Test Intent" (in PDL), a Tool will be able to generate top-level Patterns that can be applied to the DUT in the usual way. One of the novelties is also the possibility of a Return path, where the actual Data received from the DUT can be propagated back to the original PDL flow. In this Section we will provide a brief introduction of the main features and novelties.

2.2.5.1 ICL : Dynamic Topologies

IJTAG hardware is designed to be IEEE 1149.1 compliant, and it is focused on proposing a Reconfigurable Scan Network (RSN) that can be accessed as one of the user TDR registers. For the first time, IJTAG embraces dynamic topologies as the key of its architecture, as depicted in Figure 18. The main elements of the Standard are present:

- A JTAG TAP controller as the Access Mechanism, in the top left-hand corner;
- The Instruments to be accessed, in the right-hand side
- The TDRs used to access each Instrument.
- A Mux (called ScanMux in the standard) that selects which TDR(s) are active
- A Scan Register (labelled S1) which controls the Mux

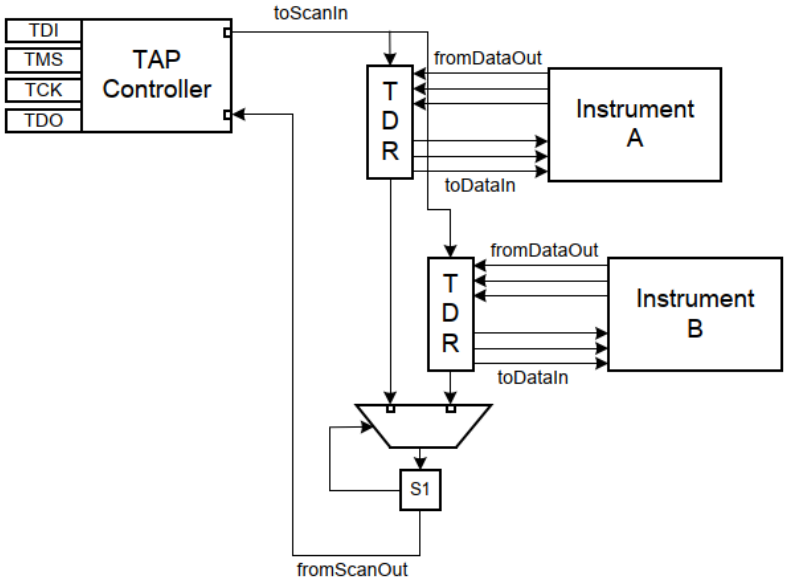


Figure 18 Example of an IJTAG Reconfigurable Scan Network, from [1687]

While it might seem trivial at first sight, the presence of the ScanMux is a small revolution in JTAG terms: the length of the active Scan Chain can vary depending on the values of the Scan Registers themselves, without the need to modify the IR. IJTAG calls each TDR section whose inclusion is controlled by a ScanMux a "Segment". Muxes can be freely instantiated in the topology, but the standard provides a reference setup, the Segment Insertion Bit (SIB) depicted in Figure 19, with a Mux and its controlling register side-by-side.

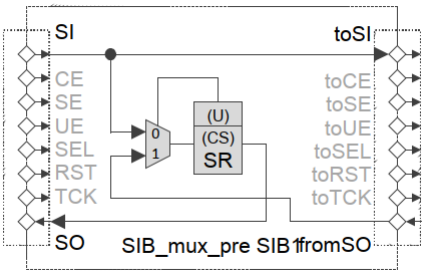


Figure 19 Example of a SIB, from [1687]

Thanks to this innovation, IEEE 1687 is able to support complex topologies of arbitrary hierarchical depth, as for instance in Figure 20, while optimizing test application time by keeping the active scan chain as short as possible.

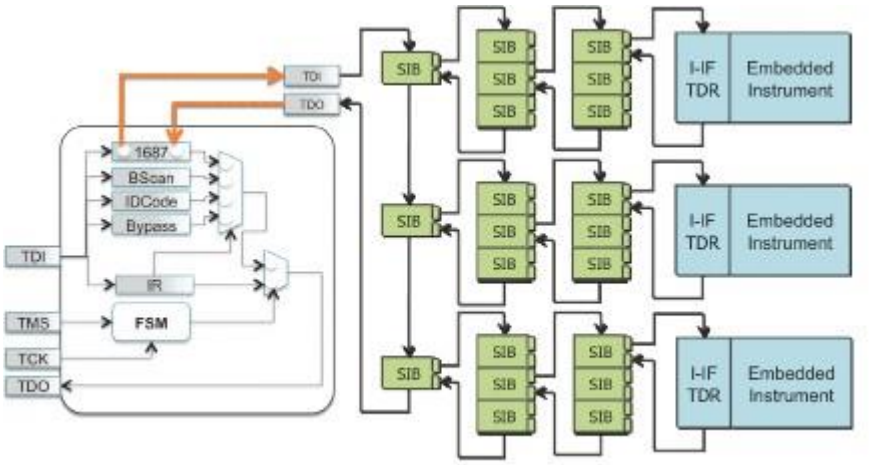


Figure 20 Example of a SIB-enabled hierarchy [DWO13]

In fact, SIBs are often the preferred choice by DfT Designers to create regular and easy-to-understand topologies. However, the standard is much richer than this and allows for more complex and even extravagant topologies, as explored for instance by the BASTION benchmarks [TSE16] [BAST19]. In order to provide Tool support for such rich topologies, IEEE 1687 proposes its own DSL: the Instrument Connectivity Language (ICL) [1687].

This document is not the place for an in-depth tutorial on the complexity of ICL, so we will just focus on its main elements: it is a "light" structural language whose role is to describe how "Raw Instruments", defined by their inputs and outputs, are connected to a scan-based infrastructure ending in an IEEE 1149.1 Test Access Port. While avoiding the complexity of a full-fledged HDL, the language allows the precise description of the connectivity of both data and control paths, so that an IJTAG-compliant tool (often referred to as "Solver" or "Retargeter") can understand how the topology can be configured and how data can be delivered to/collected from the Raw Instruments. The standard document [1687] provides several examples in its Annex E, as for instance the description of an IEEE 1500 wrapper reproduced in Figure 21

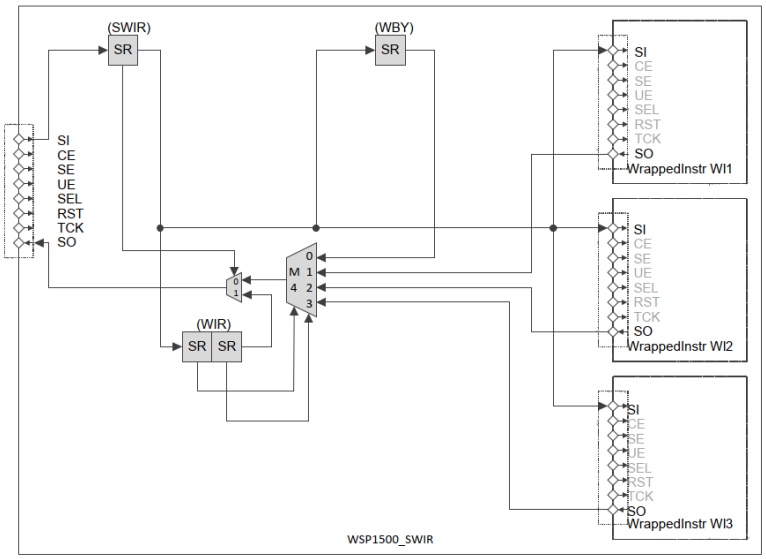


Figure 21 Example of a 1500 Wrapper from section E.20 of [1687]

Its description in ICL reproduces the hierarchy, as well as the different connections, as reported in Figure 22.

```
1.    Module WSP1500_SWIR {
2.      ScanInPort SI; CaptureEnPort CE; ShiftEnPort SE;
3.      UpdateEnPort UE; SelectPort SEL; ResetPort RST;
4.      TCKPort TCK; ScanOutPort SO { Source IR_MUX; }
5.
6.      ScanRegister SWIR {ScanInSource SI; ResetValue 1'b1;}
7.      ScanRegister WBY {ScanInSource SWIR; ResetValue 1'b1;}
8.      Instance WIR Of SReg { InputPort SI = SWIR; Parameter Size = 2;}
9.
10.     Instance WI1 Of WrappedInstr { InputPort SI = SWIR; }
11.     Instance WI2 Of WrappedInstr { InputPort SI = SWIR; }
12.     Instance WI3 Of WrappedInstr { InputPort SI = SWIR; }
13.
14.     ScanMux DR_MUX SelectedBy WIR[1:0] {2'b00 : WBY;
15.                                         2'b01 : WI1.SO;
16.                                         2'b10 : WI2.SO;
17.                                         2'b11 : WI3.SO;
18.     }
19.     ScanMux IR_MUX SelectedBy SWIR {1'b0 : DR_MUX;
20.                                     1'b1 : WIR[0];
21.
22.     }
23.   }
```

Figure 22 ICL Description of Figure 21, from [1687]

There is a crucial point in ICL: connectivity is always described backwards, from the output backtracking to the input. This algorithm applied to Figure 22 would execute as this:

1. The starting point is the ScanOutPort SO, which in line 3 is connected to the ScanOutPort of IR_MUX (implicit reference).
2. IR_MUX is declared in lines 19. It can be either connected to WIR or to DR_MUX
    a. The Selection is decided by the UpdateValue of SWIR (line 19):
    b. When '1', the next connection is WIR (line 20)
        i. After WIR, the next connection is SWIR (line 8)

```
        ii.  SWIR is connected to ScanInPort SI: connection is complete
c.  When '0', the next connection is  DR_MUX (line 19)
         i.  DR_MUX is selected by the Update value of WIR (line 14)
        ii.  When "00", the next connection is WBY (line 14)
                  1.  WBY is connected to SWIR (line 7)
                  2.  SWIR connection already resolved in 2.b.ii
       iii.  When "01", the next connection is WI1 (line 15)
                  1.  WI1 is connected to SWIR (line 10)
                  2.  SWIR connection already resolved in 2.b.ii
        iv.  When "10", the next connection is WI2 (line 16)
                  1.  WI2 is connected to SWIR (line 11)
                  2.  SWIR connection already resolved in 2.b.ii
         v.  When "11", the next connection is WI3 (line 17)
                  1.  WI2 is connected to SWIR (line 12)
                  2.  SWIR connection already resolved in 2.b.ii
```

The topology of Figure 21 is unambiguously described, in terms of both Scan and Control paths. As the name ICL clearly states, the focus is put on the Connectivity, which must be tracked through the arbitrary deep hierarchy. While undoubtedly powerful, ICL is quite difficult and error-prone to write. Moreover, its coherence with the actual HDL description of the circuit is not assured and needs to be verified [9]

After constructing a model of the SUT for the ICL description, a Solver must also be able to track the state of each Mux to construct the active scan chain at any moment. It must also be able to identify the state of the SUT which allows the access to a given segment, and construct the sequence of operations to achieve it.

2.2.5.2 PDL : Test Intent

The second great innovation of IJTAG is the possibility of describing the functional behavior of an instrument directly inside the standard thanks to its second DSL: the Procedural Description Language (PDL). The aim of this language is to go beyond pushing binary patterns and rather be able to provide "Test Intent". This means that PDL must be able to describe the operations that are supposed to be executed on one or more instrument in a formal way which is not necessarily executable: the Test Tool is supposed to parse PDL files and process them in order to obtain the operations corresponding to their description. To better explain this process, Figure 23 provides a detailed description of the internal setup of a 1687 Tool.

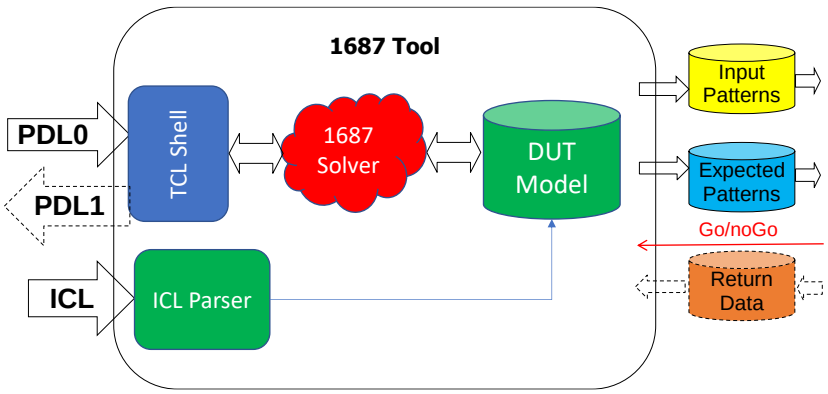


Figure 23 Internal Setup of the 1687 Tool of Figure 17

The ICL parser builds an Internal Model of the Design Under Test, which is not only used for topology resolution, as explained in the previous section, but also as the target of PDL operations.

The language has been built to be syntactically compatible with TCL, which is universally used to provide Interactive Shells in EDA tools: by interpreting a PDL file line-by-line, an EDA tool will therefore be able to produce JTAG-level patterns, expressed for instance in STIL or SVF. The language is divided in two "Levels":

- PDL Level 0 is supposed to express the procedures to operate an instrument "in terms of stimuli and expected responses for the ports and/or registers described in the ICL module for the instrument" [1687]. It provides the same functionalities of traditional ATE testing (writing data, setting expected outputs, etc…), but can also use TCL to write rich test static routines.
- PDL Level 1 targets "instruments whose functionality requires a more complex representation or in environments where interaction with the device determines the flow of the test, a high-level programming language is essential" [1687]. The main difference with PDL 0 is the possibility of collecting the actual data from the SUT and use it to dynamically modify the behavior and flow control of the test procedure.

PDL follows a queuing scheme: users can request to write a value to a register (iWrite) or set an expected value to it (iRead). These commands are queued by the 1687 Solver and converted into Patterns when an iApply command is received. This scheme allows the Solver to collect requests to different Segments and resolve them together at the iApply synchronization edge, as shown in Figure 24

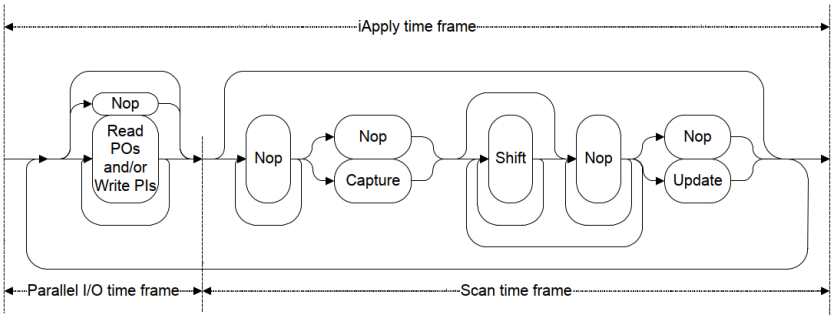


Figure 24 Sequence of Operations during an iApply , from [1687]

The iApply Time Frame reproduces at the Instrument Level the same type of Vector operations that can be done at a JTAG TAP or in an IEEE 1500 Instrument: the sequence of "Capture-Shift-Update", preceded by Parallel Input and Outputs. While the sequence in fixed to ensure predictability and coherence, each step in the frame can be skipped going through a "Nop" node. This is to take into account ICL connectivity: the Select, Capture and Update signals might be gated or inactive in certain situations, to the Solver can take into account when retargeting. This also leaves some optimization freedom (for instance, not Capturing data in frame if it is not used in the following one).

The PDL language, presented in Table 1 and Table 2, is quite rich as it presents commands to both provide retargeting information and configure the JTAG interface, as well as commands aimed at writing more user-readable code.

Table 1 First part of PDL Commands list, from [1687]

Command	Type	Parameters	Purpose
iPDLLevel	Setup	('0' | '1') '-version STD 1687 2014'	Identify PDL flavor
iPrefix	Setup	instance_path	Specify hierarchical prefix
iReset	Action	'-sync'?	Reset the network
iWrite	Setup	(register | port | alias) value	Queue data to be written
iRead	Setup	(register | port | alias) value	Queue data to be read
iScan	Setup	scanInterface_name '-ir'? length '-si' siData '-so' soData	Queue data to be scanned
iOverrideScanInterface	Setup	<scanInterfaceList> -captureEn (on) | off -updateEn (on) | off -broadcast on | (off)	Indicate the capture, update, and broadcast behavior to be imposed on a list of scan interfaces
iApply	Action	[-together]	Execute queued operations
iClock	Setup	ClockPort	Specify a system clock, which is required to be running
iClockOverride	Setup	ToClockPort '-source' clockPort -freqMultiplier mult -freqDivider div -period int [unit]	Override definition of system clock when it is generated on-chip
iRunLoop	Action	(cycleCount ('-tck' | '-sck' port)? | '-time' tvalue)	Issue a number of clocks
iProc	Setup	procName '{' arguments* '}' '{'commands+'}'	Wrapper for a PDL procedure
iProcsForModule	Setup	[namespace::]moduleName – iProcNameSpace nameSpace_name	Identify the module in the ICL with which subsequent iProcs are associated

Please refer to [1687] for a full description of PDL. In this document, we will focus on three command subsets.

Table 2 Second part of PDL Commands list, from [1687]

Command	Type	Parameters	Purpose
iUseProcNameSpace	Setup	nameSpace_name	Use namespace for subsequent iCalls
iCall	Setup	hierProcName (arguments)*	Invoke a PDL procedure
iNote	Action	-comment | -status text	Send text to runtime
iMerge	Setup	-begin | -end	Allow merging (concurrent evaluation) of iCalls
iTake	Setup	instance | register | port	Disallow other merge threads from modifying a model resource
iRelease	Setup	instance | register | port	Re-allow other merge threads to modify a model resource
iState	Action	reg|port value -LastWrittenValue | -LastReadValue | -LastMiscompareValue	Document the current state of the network

The first group is the subset related to vector handling:

- "iWrite $target $value": queues $value to be written to $target (can be a register, a port or an alias of one of the two);

- "iRead $target $value" queues expected $value to be compared with the data read from $target (can be a register, a port or an alias on one of the two);
- "iScan $ScanInterface -si $siData -so $soData". Queues both $siData to write and $soData to be compared from a $ScanInterface. It can be used to express the result of retargeting (see next section) or to drive black-boxes
- "iApply": executes all queued operations. It is used as a synchronization edge in PDL code
- "iRunLoop "$cyclecount" : Issues a number of clock cycles. It is useful to run internal instruments which are clocked by TCK, in the same way RUNBIST does at the TAP level.

The second subset includes some utility commands to gain programming capabilities, such as:

- 'iProc $name": defines a wrapper for a PDL procedure that can be called elsewhere in the code;
- 'iCall $name": invokes the iProc of the same name

The third subset is composed by the commands related to retargeting: they are iMerge, iTake and iRelease, whose behavior will be analyzed in the next section.

All these commands are part of "PDL Level 0" or "PDL 0", as they define a static behavior: the result of their retargeting can be directly expressed in terms of Input Patterns and Expected patterns, as depicted in the upper right-hand corner of Figure 23. Dynamic Behavior is achieved in "PDL Level 1" thanks to the 4 commands listed in Table 3.

Table 3 PDL Level-1 commands, from [1687]

Command	Arguments and options	Purpose
`iGetReadData`	`register | port | alias | ScanInterface [-bin | -hex | -dec]`	Return (as a string in the specified unsized number format) the value from the most recently applied iRead operation on register or output port (or an alias consisting of either or both). May contain x-values.
`iGetMiscompares`	`register | port | alias | ScanInterface [-bin | -hex | -dec]`	Return (as a string in the specified unsized number format) the XOR of the value from the most recently applied iRead operation on a register or output port (or an alias consisting of either or both) and the value expected for that iRead operation. May contain x-values.
`iGetStatus`	`[-clear]`	Return the decimal number of iApply miscompares that have occurred since the last time that iGetStatus –clear was issued. Clear the count afterwards if directed.
`iSetFail`	`message [-quit]`	Return the message string to the controlling program to indicate an unexpected condition, with the optional directive to abort execution.

These commands allow a PDL program to query the Solver for the Return Data that need to have been stored in the DUT Model.

2.2.6 Putting it all together: retargeting

The combination of ICL and PDL allows the tool to perform retargeting optimization and therefore reduce the overall test system. To better explain the process, we will use the "Instrument Example" provided in the standard document and reproduced in Figure 25 as a base Use Case and provide a step-by-step unrolling of an IEEE 1687 retargeting operation.

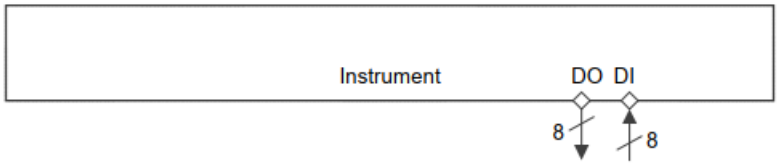


Figure 25 Example generic Instrument from Annex E.2 of [1687]

It is the simplest setup possible in IJTAG: a "raw instrument" having an Input and an Output port. Its ICL description is straightforward:

```
Module Instrument {
  DataInPort  DI[7:0];
  DataOutPort DO[7:0];
}
```

The DI and DO ports can be used as targets for PDL commands, as in this example:

```
iWrite DI 0b01010000
iApply
iWrite DI[7] 0b1
iApply
iWrite DI[7] 0b0
iApply
iRead DO[1] 0b1
iApply
iRead DO[0] 0b1
iApply
iRead DO[7:2]
```

The "Test Intent" of this sequence is easy to understand:

1) Apply the binary value to "0101000" to the Input port DI
2) Apply the binary value '1' to bit 7 of the Input port DI
3) Apply the binary value '0' to bit 7 of the Input p ort DI
4) Check that the value of bit 1 of output port DO is 1
5) Check that the value of bit 0 of output port DO is 1
6) Capture the value of bits 7 to 2 of output port DO

This sequence represents a typical usage of an Instrument such a BIST. Aliases can be defined in the ICL file to boost PDL readability, obtaining for instance the following, more meaningful code:

```
iWrite mode blue
iWrite enable No
iWrite data 0b100
iApply
iWrite enable Yes
iApply
iWrite enable No
iApply
iRead done Yes
iApply
iRead okay Pass
iApply
iRead count
```

Regardless of this last "cosmetic" change, the Solver can covert this PDL sequence in a pattern set on ports DI/DO, expressed for instance in the following pseudo-STIL code:

```
V {DI = 01010000}
V {DI = XXXXXX0X}
V {DI = XXXXXX1X}
V {DO = XXXXXX1X}
V {DO = XXXXXX0X}
V {DO = XXXXXXXX}
```

The application of this sequence on the DI/DO ports realizes the Test Intent expressed in the PDL file: Retargeting is finished.

Using ICL, it is possible to reuse this IP (both the register and its PDL Test Intent) in more complex topologies : the Solver will continue the retargeting until the edge of the 1687 Network. For instance, the Instrument might be connected to Scan Register, as in Figure 26, obtained by combining figures E.2 and E.3 of [1687].

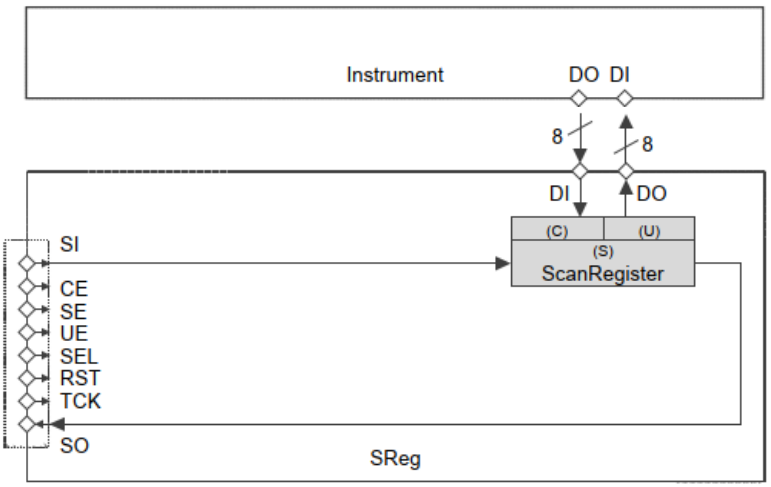


Figure 26 Raw Instrument connected to a Scan Register

In this case, the Solver needs to move the retargeting scope to the Scan Interface on the left-hand-side: the values for DI and DO need to be propagated through the scan chain. A pseudo-STIL of the retargeting result would look like this ('T' meaning "Toggle"):

```
 #shift value "01010000"
 V {SEL  = 1, CE=1, SE=0, UE=0, TCK=T}
 V { SEL = 1, CE=0, SE=1, UE=0, SI=0, TCK=T }
 V { SEL = 1, SE=0, SE=1, UE=0, SI=0, TCK=T }
 V { SEL = 1, SE=0, SE=1, UE=0, SI=0, TCK=T }
 V { SEL = 1, SE=0, SE=1, UE=0, SI=0, TCK=T }
 V { SEL = 1, SE=0, SE=1, UE=0, SI=1, TCK=T }
 V { SEL = 1, SE=0, SE=1, UE=0, SI=0, TCK=T }
 V { SEL = 1, CE=0, SE=1, UE=0, SI=1, TCK=T }
 V { SEL = 1, CE=0, SE=1, UE=0, SI=0, TCK=T }
 V { SEL = 1, CE=0, SE=0, UE=1, SI=0, TCK=T }

#shift value "XXXXXX0X"
 V {SEL  = 1, CE=1, SE=0, UE=0, TCK=T}
[…]
```

This pattern set might be applied by an ATE on the Scan Interface to realize the same Test Intent as the original PDL code. Please note that the same pattern could also be expressed as iScan Operations in the ScanInterface:

```
iScan -si 0b01010000
iApply
iScan -si 0b XXXXXX0X
iApply
iScan -si 0b XXXXXX1X
iApply
iScan -so 0b XXXXXX1X
iApply
iScan -so 0b XXXXXX0X
iApply
iScan -so 0b XXXXXXXX
iApply
```

This representation can be used to retarget PDL locally to a hierarchical element, to be re-instantiated in a bigger topology.

This example is purposefully basic so that the work of the Solver can be easily explained, but it is not so far from reality. In the face of an extremely complex system, or in an IP-based design paradigm, the EDA Tool might choose to partition the retargeting in smaller and easier-to-compute subsystems, and then compose the final result as in Figure 27

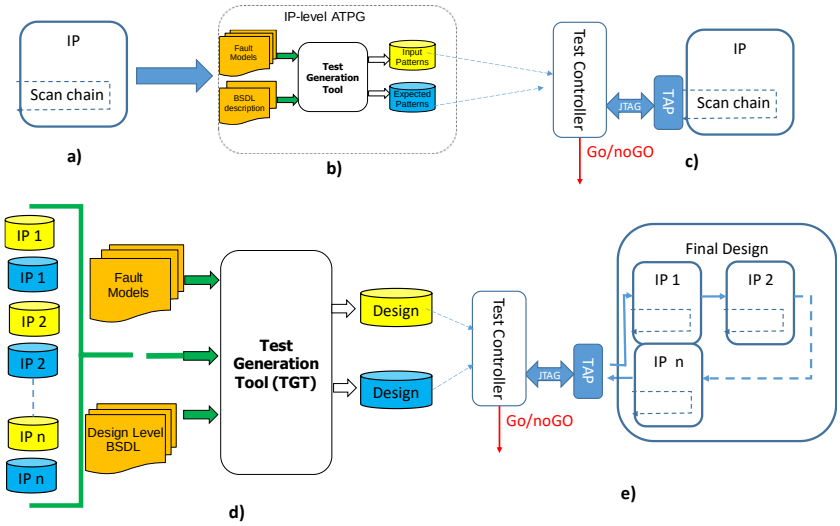


Figure 27 Partitioned Retargeting [J.4]

In a real system the Solver will need to both configure the Dynamic Topology and compose PDL operations on Registers in the same active scan chain. We called these two operations "Vertical Retargeting" and "Horizontal Retargeting" in [J.4].

Horizontal Retargeting is the process of assembling PDL sequences defined on instruments belonging to the same Scan Chain, as depicted in Figure 29. The assumption is that the PDL associated to a given instrument will be regrouped in one or more procedures ("iProc" in PDL terms), and a top-level PDL routine will call them using the appropriate "iCall" command. When handling these iCalls, the tool cannot simply sequentially flatten operations, because it would

incur in serious timing penalties. It will rather need to examine the code execution and dependencies to identify and extract potential parallelism. The "iMerge" command (Table 2) aims at helping this task by explicitly exposing concurrency, but there is no clear method on how to handle it: it is simply a markup/pragma. The usual solution, shown in the examples of the standard document, is to perform a static scheduling, where PDL operations happening during the same cycle on segments belonging to active chain are regrouped in a flattened top-level PDL operation (usually an iScan). Static Scheduling is a well-known problem with a vast literature, but its computational complexity and difficult setup limits its application field to specialized fields such as, for instance, real-time or high-performance systems [PARHI91].

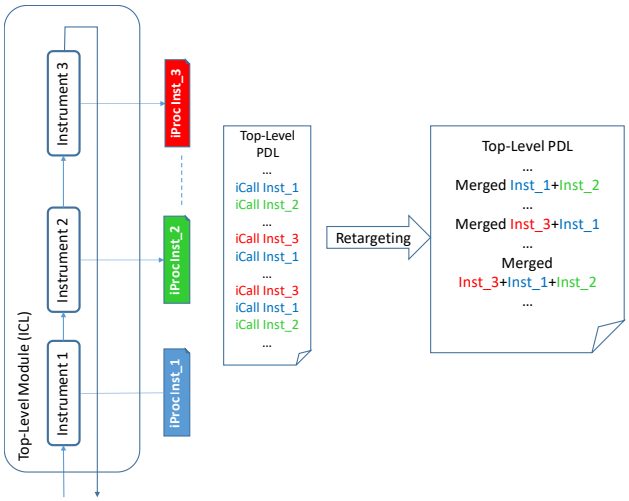


Figure 28 Horizontal Retargeting Merging for a 3-instrument 1687 system [J.4]

This "top-level sequential execution" paradigm is a general assumption of the traditional Test Generation Flow, which always aims at computing the "final test program". IEEE 1687 amplifies the problem because while introducing concurrency it also completely changes the scale of the application. While traditional JTAG considers a moderate number of components, a fully-fledged 1687 System on Chip could easily be composed of hundreds or even thousands of instruments.

In Vertical Retargeting, the Solver must consider a hierarchical topology needing the configuration of one or more ScanMuxes. Figure 29 depicts the most typical example: a register targeted by the PDL code behind an SIB.

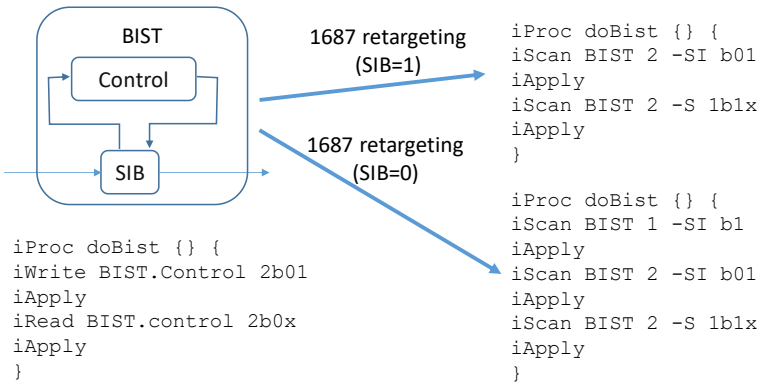


Figure 29 Vertical Retargeting of a SIB, from [J.4]

The result of the retargeting operation is different depending on the value of the SIB control register: it is up to the Solver to track the status of the DUT in its internal Model (ref. Figure 23) and modify it as needed. This means that the functional operation of the retargeted vector depends on the actual status of the Design Under Test: if the retargeting is partitioned and the sub-system of Figure 29 is pre-computed, the Solver needs to keep the two possibilities and choose which one to use at the time of composition. Please note that while the SIB is just a two-way selection, the standard allows for multiplexers of any size: this is an exciting and promising feature for DfT designers, but makes the retargeting problem even more difficult. In such a setup, a Solver would be forced to compute and retarget for all possible ScanMux state combinations.

Vertical retargeting is also extremely problematic in terms of concurrency: static PDL merging cannot work in presence of dynamic topologies, as the scheduling would need to be adapted to the current state of each dynamic elements. The only solution so far has been to force static topology resolution, which is sub-optimal, difficult to reuse and incompatible with interactive execution.

2.3 Open Standards

Almost immediately after the release of 1687, early adopters started investigating the possibility of extending the standard to support other interfaces than pure JTAG. This led to the creation of the IEEE P1687.1 Working Group [P1687.1].After an initial analysis phase, the WG started converging towards a possible solution [3][4][8]. So far consensus is going towards a direct extension of the 1687 standard, both in terms for hardware requirements and descriptive languages, while maintaining the *status quo* as far as Generation and Execution flow, obtaining the hypothetical flow of Figure 30. The general expectation is that thanks to some kind of standardized "adapter" and its description in an extended version of ICL, vectors computed for a JTAG-based system (a) will be translated to the new interface (b).

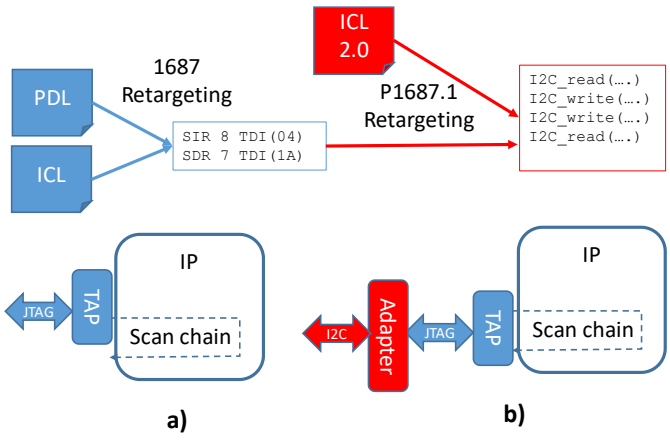


Figure 30 Hypothetical P1687.1 Retargeting Flow, from [J.4]

In such a setup the conversion would be done between pattern languages (in the example, from SVF to some I2C scripting language). Text-based conversions based on macro expansions and/or templates are theoretically possible, but they all have the same problem: they are completely agnostic of the system. In the retargeting example Section 2.2.6 of we had to infer some

information from the BSDL to move from chain-based PDL to interface-based SVF. Similarly, a conversion like the one in Figure 30 would require I2C-specific information (address table, register mapping, etc..). This information is completely lost once the retargeting is done: the generation flow would need each time to start from scratch, negating the advantages of retargeting. Another open point is debug: as vectors are transformed from one interface to another, it becomes more and more difficult to keep track of the original design. We have been contributing to the WG since its establishment, ad proposed several key innovations, which will be detailed later Sections.

As already stated, the move toward SoC-based architectures is blurring the boundaries between Architecture, Chip, Board and System: problems that once were restricted to one domain are now occurring at several abstraction levels, regardless of the physical boundaries. Another long-standing standardization Group called "System JTAG" (SJTAG) was facing similar problems: the need to test chips in a board/system having different access interfaces in a coordinated way. Roughly at the same time as P1687.1 they filed a PAR to establish the "System Test Access Management" standard, labelled as IEEE P2654. As several members are shared between the two WG, a special attention is being given for the two future solutions to be compatible, as can be seen in a series of common publications [3][4][8].

Another domain has been looking with interest to IEEE 1687: Analog and Mixed Signals. Modern SoCs are never either fully-analog or fully-digital, but rather a mixture of the two, usually in terms of "Small A – Big D " (a little bit of Analog for a lot of Digital) of "Big A – Small D". Analog testing is in several ways the polar opposite of digital testing: little or no automation, and a lot of interactive functional testing. The complexity of testing Analog circuits comes from several factors [MIL94], like for instance the huge variability of test targets and the difficulty in coming up with quantitative Fault Models, but one of the most important in undoubtedly the lack of a standardized and automated framework [SUN09]. IJTAG promise to provide interactive functional testing has been taken up but several actors, reunited in the P1687.2 Working Group [SAR17]. The aim of the Working Group is to develop an overset of the existing 1687 standard family, where Analog testing can benefit from the same automation capabilities, as shown for instance in Figure 31

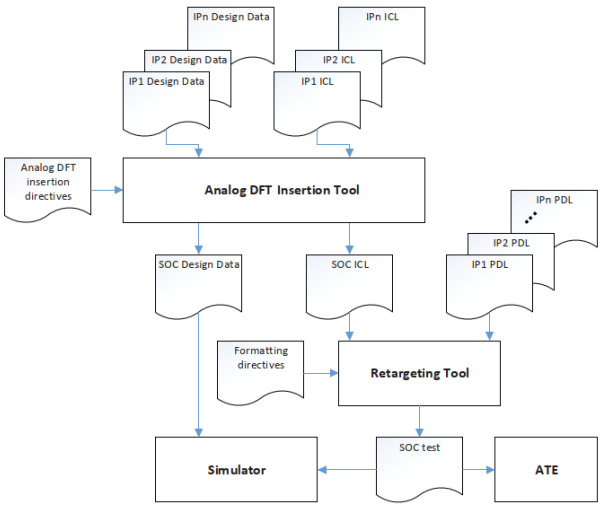


Figure 31 Envisioned EDA ecosystem to support structured analog DFT and testing, from [SAR17]

2.4 Security Issues

The omnipresence of electronics devices, along its diverse and fragment Design and Test flow raised in recent year a new concern: security. While using DfT to grant access into a circuit is unavoidable for test quality, these same facilities may create an important security backdoor into the circuit, which may be used by malicious users. Possible outcomes may consist in leakage of sensitive and critical data [SKSU13], illegal tampering of circuit behavior [BAR17], or theft of Intellectual Property. Therefore, to seal this security breach it is mandatory to implement a protection layer over the test infrastructure. Over the years, two solutions families have been proposed: Scan Authentication and Scan Encryption.

2.4.1 Scan Authentication

The principle of Scan Authentication is to limit the access to certain portions of the scan path to protect sensitive information. Few solutions currently exist [DWO13] [RAFA15] [MERA19], where the access to the test infrastructure is granted only after the User has successfully performed some kind of authentication thanks to at least one secret key. The security potential of the SIB has been recognized quite early: the Locking Segment Insertion Bit (LSIB) [DWO13], depicted in Figure 32.a, needs a specific binary condition to unlock access to the scan segment.

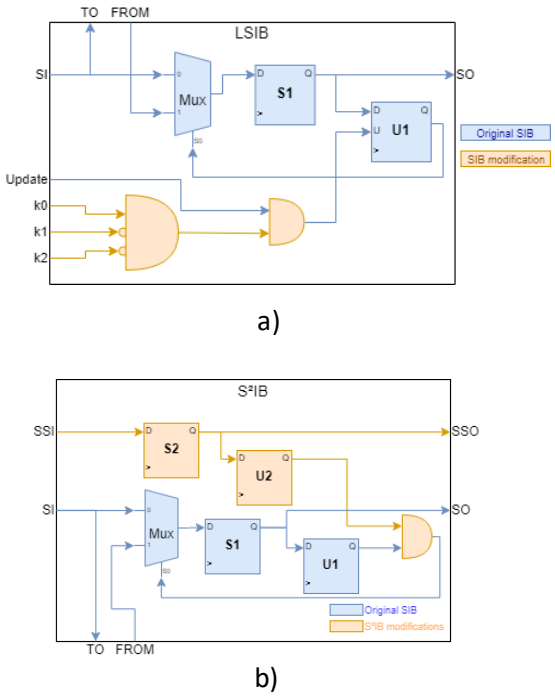


Figure 32 Locking SIB and Secure SIB Implementations

Access to the segment is therefore not automatic, but needs a secret Boolean condition. This key is spread and hidden inside the DfT infrastructure so that, in principle, an attacker would need to know its value and exact position to unlock the LSIB. In practice, however, the key will be stored in plain text inside the Pattern Set: this setup is vulnerable to Replay or Man-in-the-Middle type of attacks, where the key can be inferred from the data.

In a secure setup, when using an insecure channel like a scan chain, the key should never be transferred in plain text. The Fine-Grained-Access (FGA) solution [RAFA15] uses a challenge-response protocol to perform the authentication before granting access to a sensitive scan chain segment. It is a classic scheme, where the two parties share a secret used to generate a one-time transaction token, while the secret keys themselves are never transmitted on the insecure medium, making replay attacks ineffective.

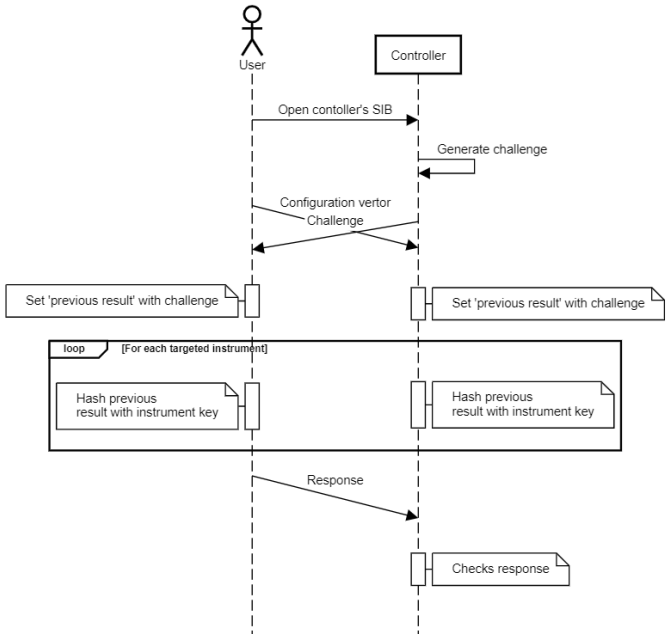


Figure 33: FGA Challenge response protocol, from [7]

Furthermore, FGA is also able to provide a personalized access to the reconfigurable scan network by using instrument-level keys: users have to know the key for each instrument they want to access. The authorization controller checks if the user's response is correct; it then allows the user to access the secure scan chain by propagating the authorization through the specially introduced Secure Segment Insertion Bit (S^2IB) visible on Figure 32.b. [RAFA15]

In [MERA19], the authors propose some modifications to the FGA approach with the objective to allow usage of reprogrammable memory for the secret key and reduce the average authentication time. To obtain a faster authentication, the instrument keys are replaced by configuration keys or Segment Set Authorization Keys (SSAK). To avoid the need of storing the secret keys on the System, the authors propose the procedural generation key mechanism presented in Figure 34, where all the configurations keys of a circuit instance are dynamically generated from the unique secret key of the circuit and a configuration vector containing the list of targeted instruments. The generation consists in an encryption of the configuration vector using the circuit key as encryption key. The security provider can distribute credentials composed of configuration and SSAK to the users. On the other side, only the Circuit Key is securely stored in the reconfigurable memory of the Authorization Controller.

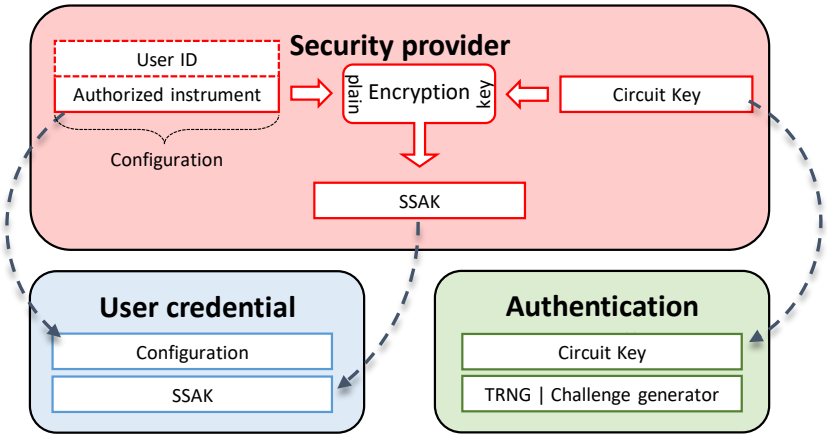


Figure 34 SSAK's procedural key generation and distribution, from [7]

Concerning the authentication protocol, only the challenge resolution is performed. The controller receives the configuration vector from the user, and then thanks to the procedural key generation, it is able to compute the associated SSAK, using its embedded encryption processor. Then, still using the same encryption hardware, the controller can resolve the challenge with this SSAK. On the user side, the process is easier as the SSAK is already known, so the user only needs to encrypt the challenge with the key contained in the credentials. Once the authentication is done, the controller needs to unlock the S²IBs targeted by the user. Figure 35 shows the scheme of the SSAK solution architecture: the different S²IBs are linked together by the so-called secure scan chain, driven by the authentication controller.

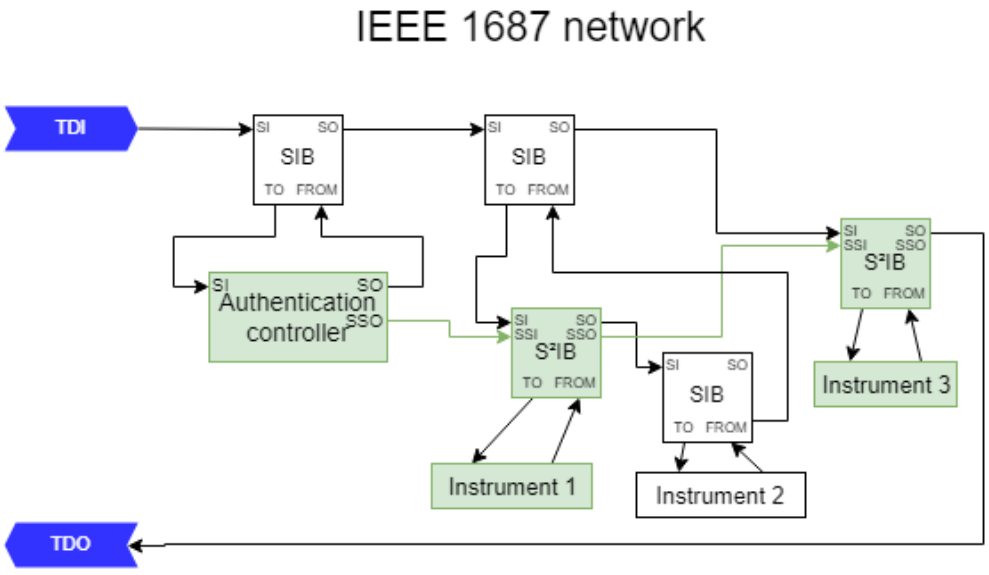


Figure 35: SSAK Authentication architecture, from [7]

In terms of hardware overhead, the LSIB approach is flexible and inexpensive in case of small implementations; it is also compatible with the traditional Test Flow as its usage can be expressed in terms of plain vector operations. However, its overall security level may be insufficient due to the plaintext key exchange. FGA and SSAK solutions have both a more secure protocol for authorization. In addition, they are quite efficient even for a large number of protected

instruments. As the number of instruments increases, SSAK becomes clearly the more efficient and secure alternative, thanks to the constant authentication overhead. From this data alone, FGA and SSAK seem to have an overwhelming advantage.

However, this comparison does not consider a fundamental point: the dynamic nature of the challenge/response protocol cannot be expressed in terms of a Pattern Set, making these approaches incompatible with the legacy Automated Test Ecosystem. As a result, their actual implementation requires a significant and custom development by the user, effectively limiting their applicability.

2.4.2 Scan Encryption

When accessing a Scan Chain, all the data exchanged on the JTAG interface is accessible by anyone in both directions, allowing Snooping or Mad-in-The-Middle attacks. The principle of Scan Encryption is to obscure the data being transmitted, as depicted in Figure 36 : a Scan Cypher is inserted before the TAP and is charged to the Decryption of the Vector sent to the SUT, and of the Encryption of its output, so that all communication is obscured. This means that the Plaintext Vectors computed by an EDA Tool must be Encrypted before transmission, and similarly the Outputs from the SUT must be Decrypted before processing.

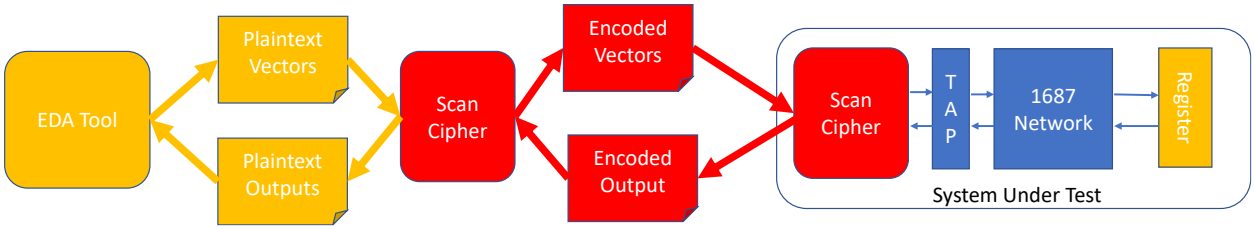


Figure 36 Principles of Scan Encryption

The scan encryption technique sits on the cryptographic foundation of the symmetric encryption schemes. These primitives provide confidentiality to the communication between two parties: in the test scenario, the two communicating parties are the tester and the device under test. Both of them share a secret key, which the circuit manufacturer properly deploys to authorized testers and, at the same time, stores it inside the target device in a secure memory. All data that is exchanged between the tester and the device is encrypted/decrypted thanks to cryptographic modules that are placed at the TDI/TDO interfaces of the device.

When scan encryption is implemented, the typical test procedure is executed through the following steps, performed by the red elements in Figure 36:

1. All data produced by the tester is fully encrypted with a secret key that is known being associated to the device under test;
2. Encrypted data is shifted through the TDI interface of the device, where it is decrypted through a decryption module that is located at the interface with the rest of the test infrastructure;
3. All results of the test procedure, which are sent through the TDO interface of the device, are first encrypted by a cryptographic module;
4. Encrypted results are retrieved by the tester, decrypted, and analyzed.

Scan encryption can be classified according to the kind of cipher that is employed, namely *stream ciphers* or *block ciphers*. Scan encryption can be based on block ciphers, or block-based scan encryption, as for instance in [DASIL19] in with a negligible area overhead (less than 0.5%) when implemented on complex SoCs. However, block ciphers need to process test data in blocks of a fixed size (usually 128 bits), thus requiring a careful parsing of test data and its subsequent padding in order to fit a multiple of the block size. Even if it does not cause a significant overhead on the test time, it complicates the interface between the serial test interface and the encryption/decryption modules. In scan encryption, based on stream ciphers [ROS10], encryption is performed serially when processing the data. This property conveniently fits with the serial structure of test infrastructure interfaces, leading to perfectly transparent encryption/decryption operations that do not impact on the performance of the test procedure. Moreover, the capability of stream ciphers to adapt inherently to any input length makes them ideal candidates for IEEE 1687. A good example is the TRIVIUM stream cipher which is often exploited as reference stream cipher implementation because it is a "trade-off soft spot" which provides good security properties for a very limited hardware overhead [VAL19].

While its lightweight hardware implementation is undoubtedly Scan Encryption strongest point, its great Achilles's heel is its software part. Coding and decoding of vectors has to happen outside of the traditional Test Flow, demanding had-hoc post-processing and imposing constraints on the design. For instance in [THIE19] the authors need to impose a specific hardware architecture (a PUK for every protected segment) and an extremely heavy software infrastructure that needs to modify both the input PDL/ICL files and post-process the retargeted vector, completely outside any standardized EDA flow (ex: by manually modifying the shift cycle count). These implementation choices limit its applicability, scalability and portability.

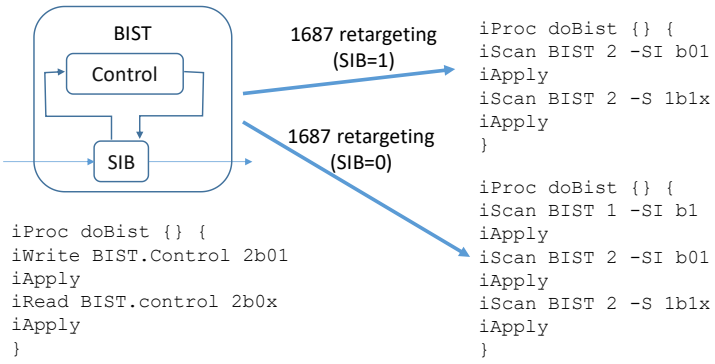
3 The need for a New Test Flow

In the previous Section, we introduced some of the novelties the Testing field is going through, exemplified by the emergences of new standards like IEEE 1687. While most of the actors in the field have been considering them as simple incremental updates to solve contingent problems, they actually have much deeper roots and wider ramifications and might usher in a profound change in how Testing is done, and its relationships with the other steps of the Electronics Design cycle. In this Section, we will highlight the true potential of these evolutions through an in-depth theoretical analysis followed by the presentation of the results of our research. The reason why IEEE 1687 is so important and disruptive is that instead of simply codifying a specific solution for a given problem like most standards (like, for instance JTAG), it is rather an "enabler": it provides strategies and tools for users to tackle new problems and try to put some order in a mess of workaround and custom solutions. For this reason, its implications go much further than what was originally intended by its developers. As mentioned in the previous Section, there are three points that put the traditional flow at risk: Topology Resolution, Concurrency and Interactive behavior.

3.1 Limitations of the Legacy Automated Test Flow

3.1.1 Topology resolution (vertical retargeting) limitations

First introduced in Section 2.2.6 and Figure 29, the aim of Vertical Retargeting is to identify the topology state that allows access to one or more registers which are the target of PDL operations. Usually, scan based system apply a top-down approach: starting from a top-level command, an algorithmic method identifies the low-level operations that perform it. Figure 29 is a perfect example: from the top-level need of writing a value to the "control" register, the Retargeter computes the operations needed to access it, i.e. the SIB configuration.



Replica of Figure 29: Vertical Retargeting of a SIB, from [J.4]

Top-down approaches are easy to understand and to implement, but have difficulty in scaling. Complex SoC might have several layers of hierarchy: take for instance the system of Figure 37 : in order to reach "Control", the Retargeter needs to configure a big number of SIBs. The combinatorial complexity of finding the Shortest Path quickly explodes: a traditional Dijkstra

algorithm as an $O(X^2)$ complexity (with X being the number of nodes, i.e SIBs), and optimized versions are usually no better than O(XlogX).

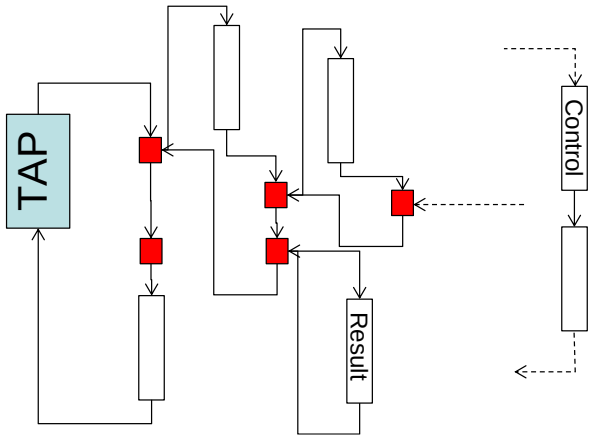


Figure 37 Example of a deeply hierarchical system

The problem is even more complex when considering multiple registers: still in reference to Figure 37, let us consider a test that needs to write both register "Control" and read register "Result" : the configuration needs either to find a Network State NS(Control,Register) where are both accessible, or if it is not possible the two states NS(Control) and NS(Register) and the Transition between the two NS(Control)→NS(Register): several NS might be needed to correctly configure the network. Of course, the Transition it not reversible:

NS(Control)→NS(Register) ≠ NS(Register)→NS(Control)

In more general terms, any Transition is dependent on the initial Network State, NS(Start). This means that the Retargeter cannot make one "absolute" retargeting for PDL codes defined at a register level, but must always consider the initial state. This is a serious limitation to code reuse: the retargeting of an IP cannot be fully resolved until the whole 1687 Network is known.

There are no real solutions to this problem: to reduce its impact industry tools usually perform retargeting using a reference Safe state: NS(Control) is in reality NS(Safe → Control →Safe), so that:

NS(Control→Register) = NS(Safe → Control →Safe) + NS(Safe → Register →Safe) =

NS(Safe → Control → Safe → Register → Safe)

Of course, most tools won't stop at this suboptimal solution but will run additional optimization steps to end up with NS(Safe → Control → Register → Safe). However, the combinatorial cost of these additional steps can be computationally high, with serious scaling issues which are difficult to overcome while keeping a top-down approach.

3.1.2 Concurrency (horizontal retargeting) limitation

The base element of IEEE 1687 is the Instrument: however, it is never used alone. In a real system the user will have N Instruments, over which N PDL procedures might be defined. For these, a Top-Down Retargeter has to compute the top-level PDL operation which corresponds to the execution of the N PDL subroutines, as depicted in Figure 38.

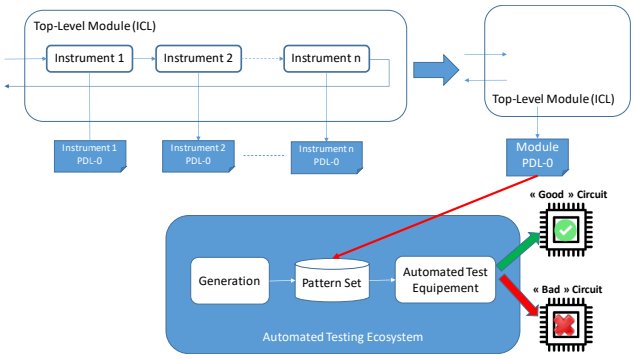
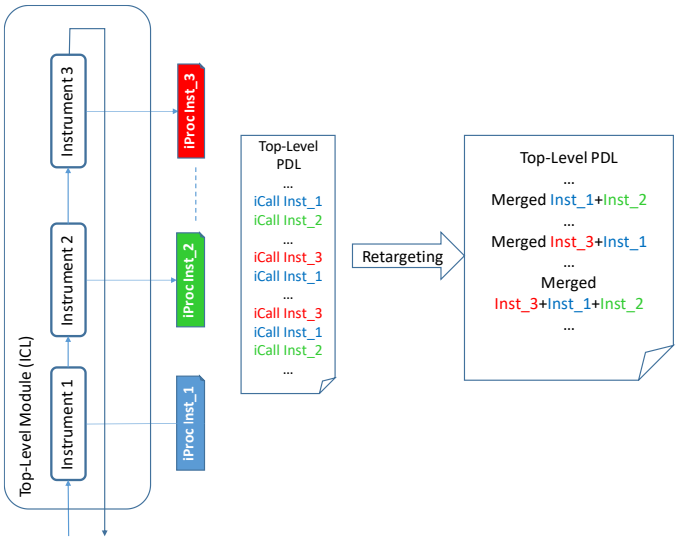


Figure 38 IJTAG Legacy Top-Down Retargeting Backend for PDL-0, from [J.4]

In most Tools, and in the 1687 Standard document itself, the assumption is that the User will write a Top-Level PDL file which will iCall the PDL subroutines defined at the Instrument level. From there, the Retargeter will try and merge the individual iApply groups defined in the called iProcs to obtained a flattened Top-Level PDL. It is the flow already documented in Figure 28, reproduced here for easier reference. The "iMerge" instruction is used to "expose potential concurrency"[1687] to the Tool, .i.e mark the PDL sections that might be parallelized.



Reproduction of Figure 28 Horizontal Retargeting Merging for a 3-instrument 1687 system [J.4]

This scheme is intellectually easy to understand, but it actually computationally extremely intensive: extracting parallelism from a sequential program is a complicate and well-known problem in computer science [GIRK92], and its application is usually limited to specific use cases due to its complexity. In the case of IJTAG the problem is even more complicated because of the need to consider the dynamic topology of the network connecting the Instruments.

3.1.3 Interactive behavior

As stated in the previous Chapter, one of the big novelties of IEEE 1687 is the native support of Interactive Behavior thanks to the PDL-1 instruction set. The most important is "iGetReadData", which returns the last data value read from the target Register. As PDL has been developed to be syntactically compatible with TCL, the natural implementation of PDL-1 as described in the Standard is an interpreter linked to a Retargeting tool executing an algorithm composed of both TCL and PDL-1, obtaining the setup of Figure 39.

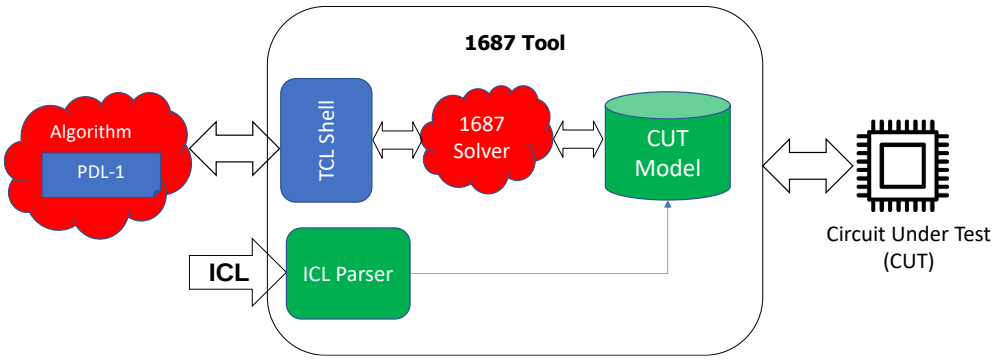


Figure 39 Interactive IEEE 1687 Tool Setup

The combinatorial complexity of Vertical and Horizontal retargeting, as described in the previous sections, seriously limits the applicability of this setup: the Tool needs a serious amount of resources to run the algorithms, so it cannot be run on constrained setups such as embedded controllers. Moreover, there is no clear path for portability and retargeting: how can such a setup be compatible with the traditional retargeting flow, as depicted in Figure 40 ?

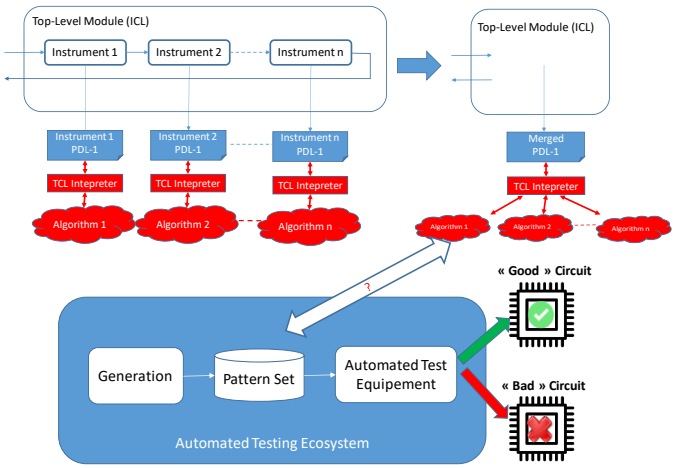


Figure 40 IJTAG Legacy Retargeting Backend for PDL-0, from [J.4]

Pattern languages cannot express interactive behavior when Flow Control decisions (ex: if-then-else) depend on data retrieved from the SUT. A direct solution would be to execute PDL-1 on the fly, but TCL interpreter are notoriously resource-hungry and difficult to support in Execution Backends other thank PC, like for instance ATE.

As of now, support for Interactive IEEE 1687 from Industrial Tool is tenuous at most.

3.1.4 Interface Domains

Usually, EDA Tools work in a well-specified Domain: Retargeters work on Scan-Chains, JTAG tools on JTAG systems, etc… By Domain we mean a system that can be accessed or controlled only through a specific set of Primitive Operations. For instance, a JTAG system can only perform either operations that are possible in the standardized FSM (most notably, going through the Capture-→ Shift →Update (CSU) sequence on either the IR or DR scan chain), or Instructions defined in the Standard (INTEST, EXTEST,etc…). On the other hand, a Scan Domain will have a richer set of data Primitives (not only CSU, but also CS, SU, S, etc…), but no Instructions. There is a lot of variability: domains such as I2C will rather have primitives focused on Writing and Reading to specific addresses, etc… This allows each Tool to exploit the specifications of its domain to process the required operations and express them in the Primitive of its domain. However, a complete system is seldom composed by a single Domain, so the Tool will have to Translate between them. Sometimes it is easy, as for instance in Figure 41 : Scan and JTAG domains are purposefully extremely similar, so Tools can easily translate between them thanks to a-priori knowledge of the domains (obtained from their Standard specifications) and store the resulting primitives in a DSL file, such as for instance SVF.

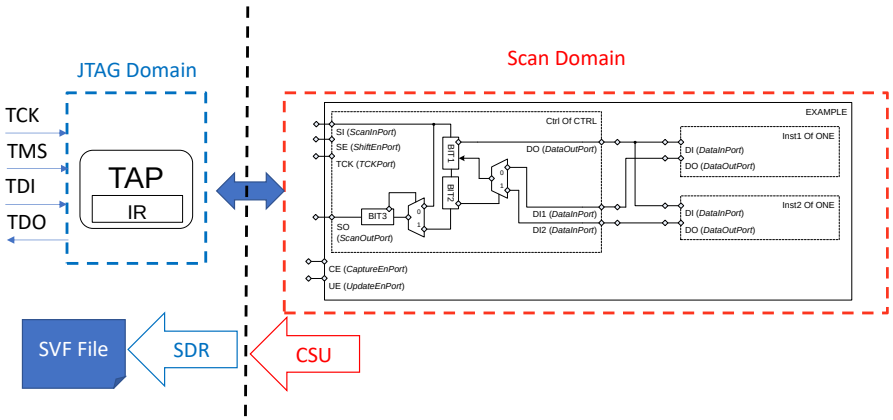


Figure 41 Domain Translation between Scan and JTAG

The generic case is much more complex: take for instance the system of Figure 42 : the connection between I2C and JTAG is not standardized, so the Tool has no easy way to make the translation.

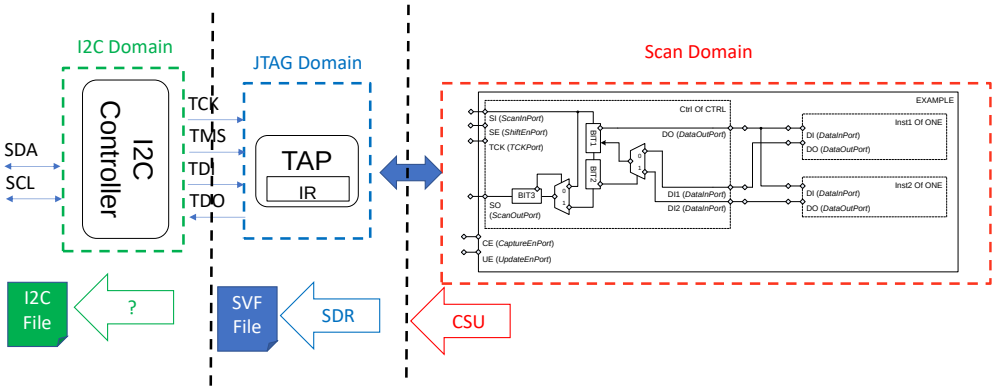


Figure 42 Domain Translation between multiple interfaces

Such setups are in fact quite common, and most of the times users need to rely on custom routines which parse the DSL files and convert among them exploiting the built-in knowledge of the Translation. This is usually done through in-house tools, whose validation and maintenance can require important resources. Efforts like the [P1687.1] Working Group are trying to develop a standardized Domain Translation methodology [3] [4], but so far, no solution have been officially ratified yet.

4 A New Automated Test Flow: Specification

As we pointed out in the previous Sections, the main limitations come from legacy features in the Automated Test Flow. The most important and limiting one if the dichotomy between "Smart Generation" and "Dumb Application" [J.4]: the need of resolving all computation offline imposes a series of "a priori" decisions to obtain a static Patten Set based on a reference setup. When the setup changes, undoing the computations and adapting the set demands a lot of effort and actually makes the operation much more difficult that it could have been if done correctly the first time.

4.1 High Level Requirements

It is useful to make a comparison with Computer Science, where a Source Code written in a high-level language (ex: C) is compiled into a Binary program which can then be stored in a Memory to be finally executed by a Processor. The direct, naïve flow is depicted in Figure 43-a): The High-Level program is first converted into low-level Assembler instructions, which are then mapped into the Memory. This requires the knowledge of the exact start address of the program in the memory to solve all addressing in the Assembler instructions (the absolute Jumps, but most importantly the location of Data Variables).

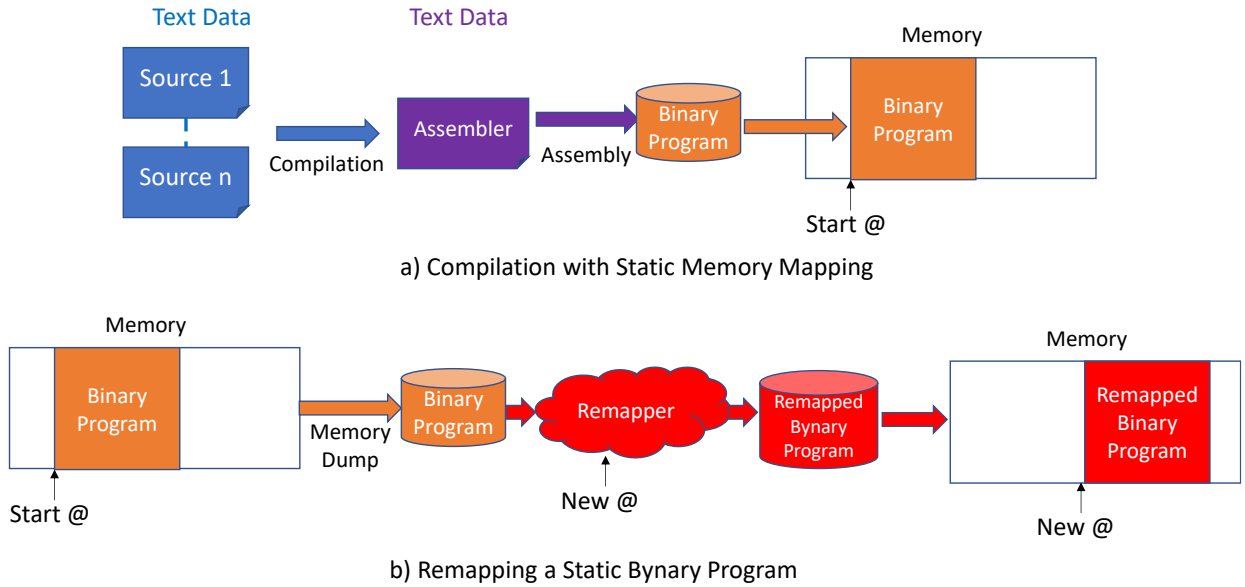


Figure 43 Software Compilation with Static Memory Mapping

If the program needs to be executed in another memory location, it must be remapped, as depicted in Figure 43-b): first the Binary Program must be reconstituted, potentially dumping it from the Memory, and then remapped to the new Address. This requires parsing the whole binary data, identify the variables from the Assembler operations accessing them, compute their new locations and substitute all the references in the binary code. This is of course not impossible, but extremely time-consuming, suboptimal and not scalable. This problem is solved by the concept of Relocation, depicted in Figure 44. The Assembler code is not directly converted in pure

Binary, but in an intermediate format where some placeholders (called "symbols") are set in place of the addresses. In a BareMetal approach, these symbols are resolved when the final Memory Mapping is available, to obtain a static binary representation. When an Operating System is present, the linking is done at execution time by a specific component, the Loader. Each compilation toolchain and Operating System has its format: in Linux-Gnu, for instance it is the "Executable and Linking Format" [ELF95], in Windows the "Portable Executable" [PE15]. This procedure is also what allows the inclusion and usage of Static and Dynamic software libraries.

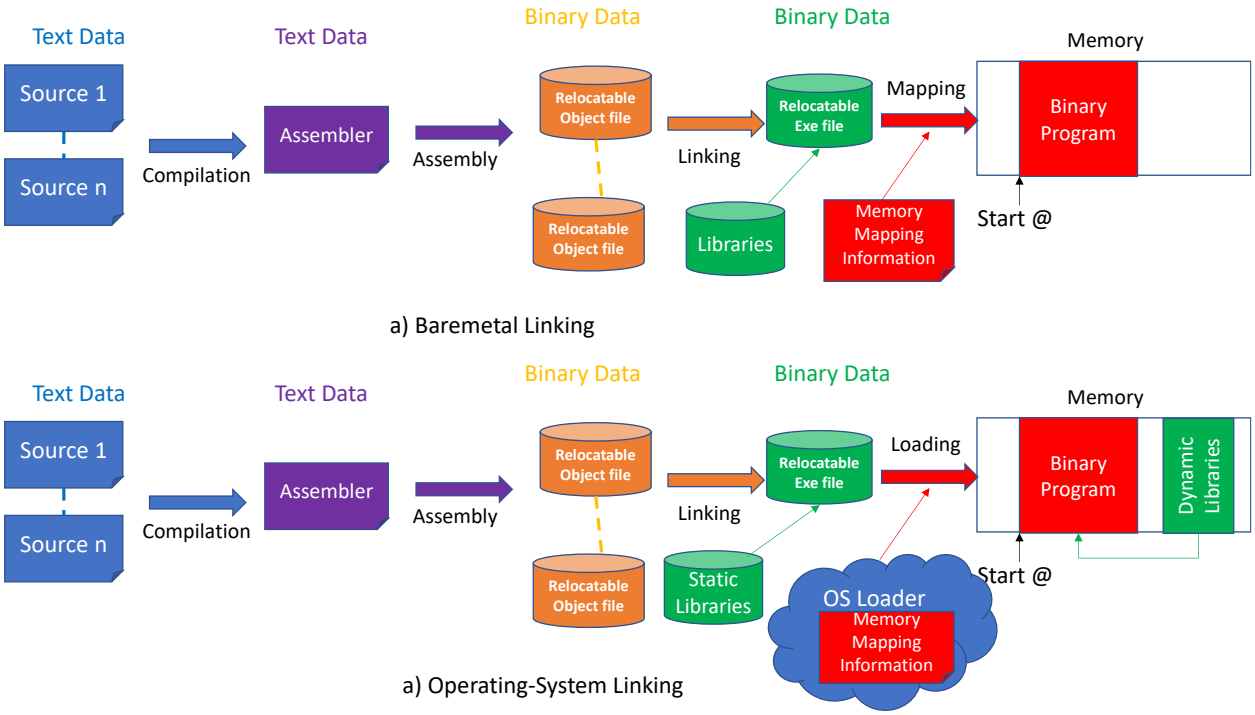


Figure 44 Principles of the Software Relocation Flow

The strongest point of Software Relocation is the capability of distributing processing during the whole flow: each piece of information is treated as soon as it is available, but only as far as it is possible. If some operation cannot be completely performed, some Symbols are placed into the binary code with all the available information, to be used later when new elements are collected.

If we apply the same analysis to the Automated Test Flow, we end up with the scheme of Figure 45-a). During Patten Generation, the Design and Domain Specific Language files (ICL/PDL) are the Source Files, and the Generation step has the same role as Compilation. Pattern files are the same as Assembler files, and Retargeting is functionally the same as Memory Mapping: it establishes the binary pin-level representation. Similarly, Figure 45-b) shows that Pattern Retargeting if functionally the same as Static Remapping: to change Interface (ex: going from JTAG to I2C), a Tool first need to extract the Test Instructions corresponding to the flattened Patterns, convert them to the new Protocol and then recompute the patterns.

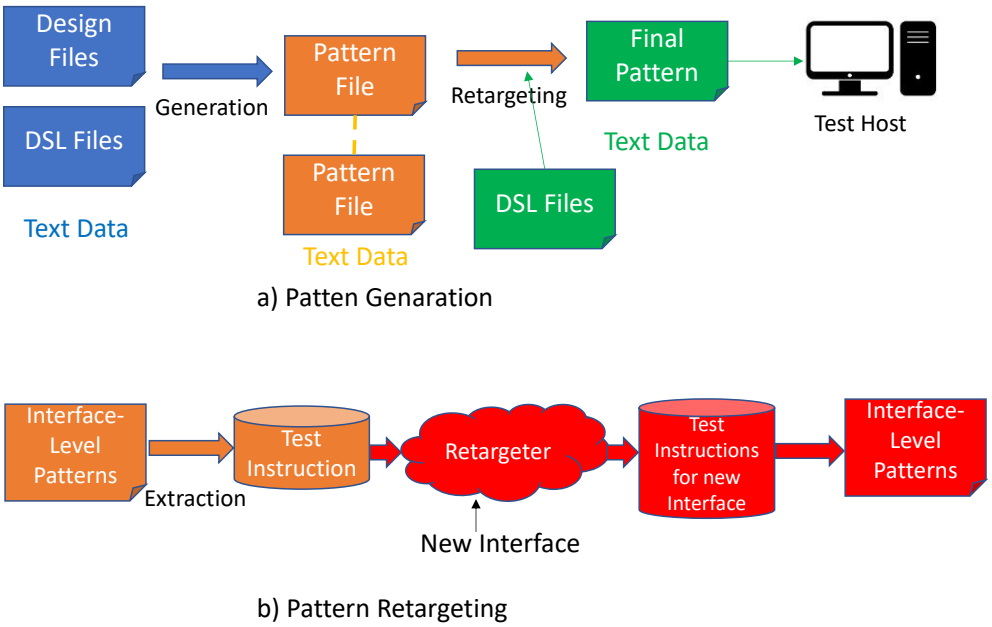


Figure 45 Information flow in the Automated Test Flow

From this analysis it is easy to understand the weak points of the current Automated Test Flow: because of the strict Generation vs Application dichotomy, all data and information processing is done in one step, ending up with static Patterns which have knowledge of neither the original Algorithms nor of the System Under Test. To retarget, a Tool needs first to extract the original Test Instructions from the flat Pattern files: this is often close to Reverse-Engineering as the Pattern files contain no Intermediate Information to help processing.

To resume, if we want to apply the Relocation Flow to Testing we have to avoid needless and counter-productive a-priori operations, resolving them only when all the necessary information is available. Based on the State of the Art of Chapter 2 and the analysis of Section 3.1, we can identify the critical elements are follow:

1. Vertical Retargeting: to resolve the Topology, the Tool needs knowledge of the system Network State
2. Horizontal retargeting: to resolve Concurrency, the Tool needs knowledge of both the Execution and Network Stage
3. Interactive Behavior: to resolve Flow Control, the Tools needs the data coming back from the SUT

These requirements can be satisfied by the Flow proposed in Figure 46 : the "Generation vs Application" dichotomy is broken, to obtain a more nuanced flow inspired by the Relocation Flow of Figure 44.

- Instead of being completely resolved, Test Algorithms (in the upper right-hand corner) are simply compiled, to obtain a Test Executable in a Relocatable Test Executable in a format such as ELF or PE, depending on the Test Host.
- Topology and Interface information contained into the DfT files (such as ICL, BSDL, etc..) is processed to compose a model of Circuit Under Test

- During Execution, a Test Manager on the Test Host runs the Test Executable. Both the algorithmic execution and the CUT Model can be updated depending on the data exchanged with the actual Circuit Under Test.

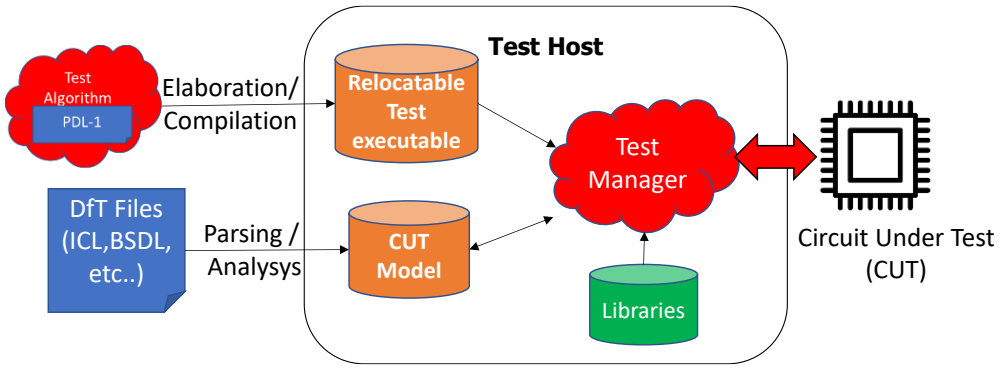


Figure 46 New Automated Test Flow

The heart of the approach is the Test Manager: it is the middleware that provides interaction between the Software (the Test Algorithms) and the Hardware (the CUT). It has therefore the same role as an Operating System.

In the following paragraphs, we will detail the specifications of the different components.

4.2 Relocatable Test Executable

The role of the Relocatable Test Executable is to collect both Pattern, Algorithmic and PDL information in a ready-to-use container. For the first two it is quite straightforward: Patterns are nothing more than Data, and Algorithms can be compiled to ISA instructions. Formats such as [ELF] have provisions for both. PDL instructions on the other hand have been developed to convey "Test Intent", and as such it is not possible to compile them. However, each instruction is precisely specified both in terms of input/outputs and expected behavior (See Tables 1 to 3 or Sections 7 and 8 of [1687]). This means that PDL is an ideal candidate for an API approach: at compilation time, PDL instructions are treated as external functions for which the compiler creates the appropriate Relocation Symbols. The Test Manager will implement the actual library, and therefore be able to eventually resolve them. The final setup is shown in Figure 47. Almost all programming languages have provisions for this flow: for instance, in C/C++ the header would be a .h/.hpp file.

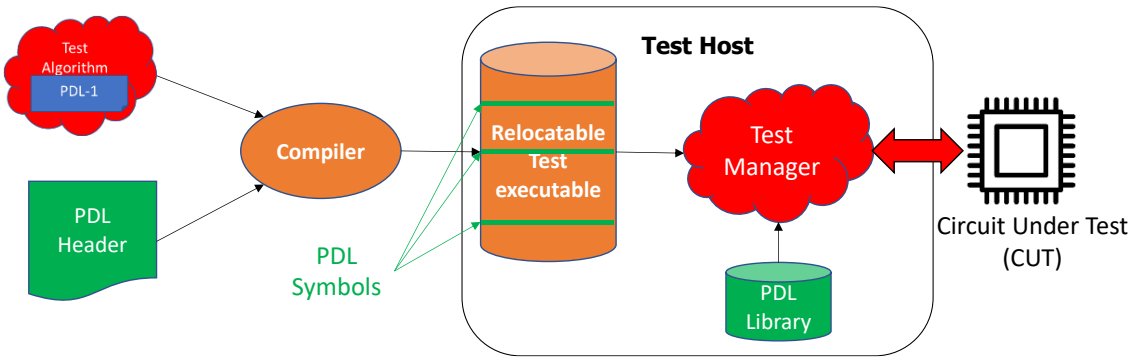


Figure 47 Details of the Compilation Step

There is however a limitation: this flow depends on a compilation step, while [1687] points TCL as the PDL overlay language. This requires an adaptation step to existing IJTAG programs, an example of which will be given in Section 5. This might not be necessary in the future, as the 1687 Refresh Working Group is discussing the possibility of opening up the Standard to other languages.

4.3 Circuit Model

The Circuit Model has several roles:

- Store all information needed for Retargering
- Track the current state of the CUT;
- Assist the Manager in handling the Dynamic Topology configuration;
- Make the connection between PDL operations and the corresponding Instruments
- Handle the connection with the CUT
- Convert scan-level bitstreams into Interface operations.

In this section, we will detail a Model able to satisfy all these requirements.

4.3.1 Retargeting

The choice of a correct Data Structure is of primary importance. As detailed in Section 2.2.5, in IJTAG the segments can either be connected serially, or selected through Multiplexers. This type of hierarchy is naturally described through a Tree. The aim is to represent the hierarchical relationships of the different IEEE 1687 elements through the shape of the Tree, so that we will be able to extract and modify the Topology state by analyzing the Data Structure through strategies such as Visitors or Iterators. In an Object-Oriented modelling, this can be achieved through three nodes:

- As the leaf, a Register node which stores both the value of the register inside the CUT, obtained from previous interactions, and the Future Value, which the System wishes the Register to reach. This can be the result of either PDL operations (ex: an iWrite which queues a value modification) or configuration algorithms (ex: wanting to open a closed SIB).
- A Chain node to concatenate Registers. The order of the children reflects the order in the scan Chain and can be extracted through a standard Transversal, as depicted in Figure 48

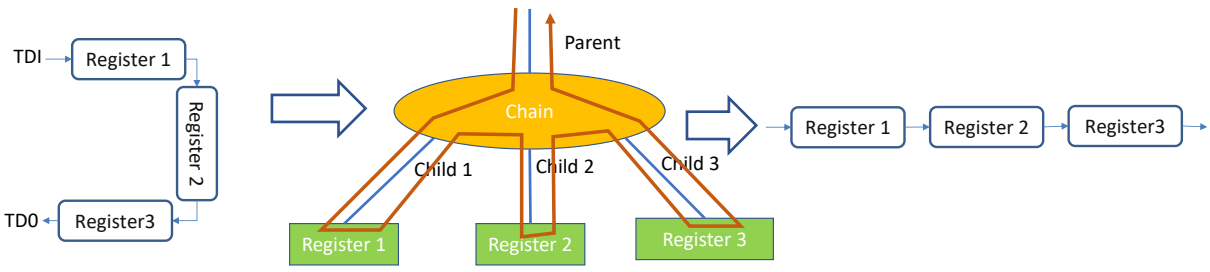


Figure 48 Tree Representation of a Scan Chain

- A Linker node which codifies the dynamic hierarchy introduced by a ScanMux, as depicted in Figure 49.

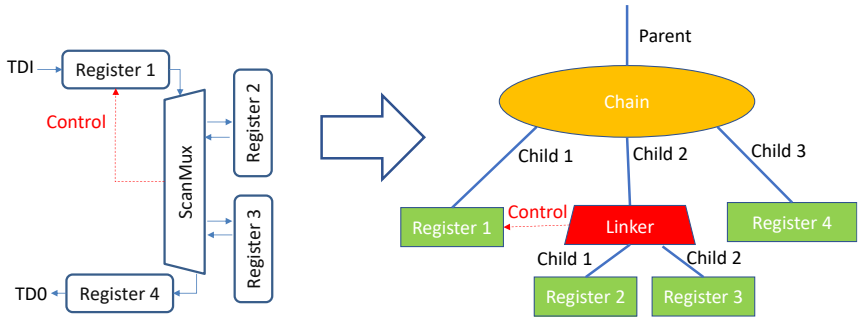


Figure 49 Tree Representation of a ScanMux

While for a Chain node all its children are always connected, for the Linker node the selection of each node depends on the value of the Control register. This can be used to guide a Depth-First Transversal of the tree, as depicted for instance in Figure 50.

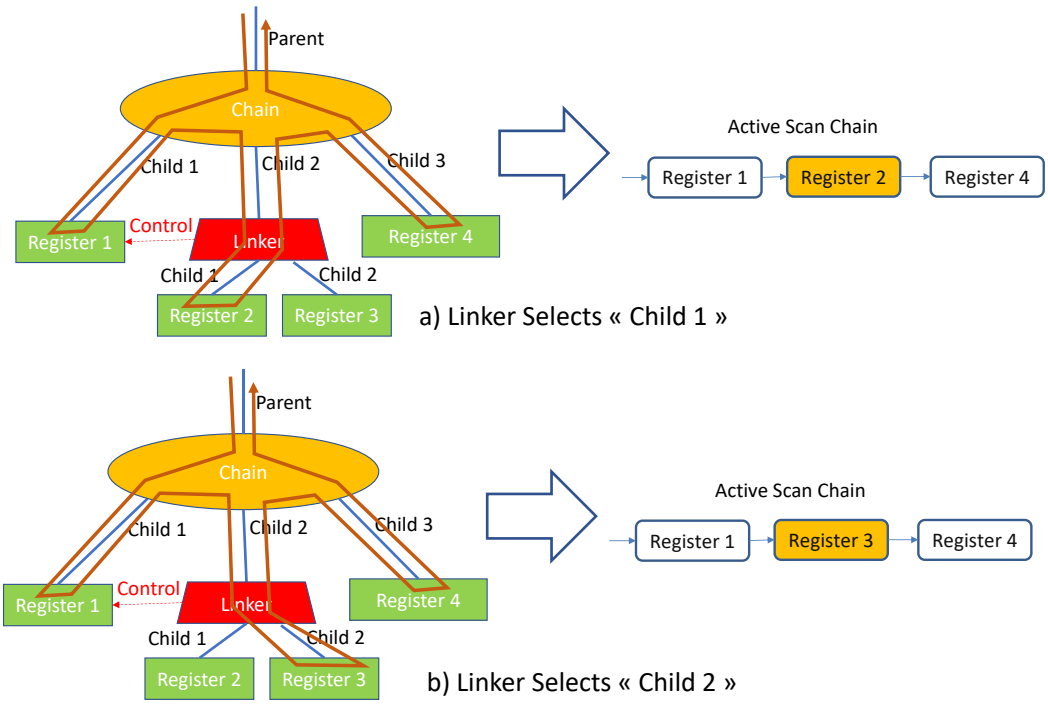


Figure 50 Extraction of Active Scan Path through depth-first tree transversal

In Figure 50-a), the Linker is selecting "Child 1", so when the Visitor reaches the node, the recursion in propagated to that child (therefore reaching "Register 2"), but no to "Child 2", which is not selected (therefore "Register 3" is not reachable). The inverse happens in Figure 50-b). The information about selection can be easily extracted from the ICL (rule "SelectedBy", Chapter 6 of [1687]). We propose to code it into the Linker node through these elements:

- an **is_active(n)** method, which returns True when child "n" is selected, and False otherwise;
- a **select(n)** method, that sets the future value of the Control Register to select Child "n";
- a **deselect(n)** method, that sets the future value of the Control Register to deselect Child "n";

The triplet is_active/select/deselect, which we will call PathSelector, is the key to the abstraction: regardless of the mux coding or even origin of the selection signals, a Test Manager will always

be able to manage the Linker without needed to change its internal Topology Configuration algorithms. The final modelling of the Linker was first presented in [1], and it is reproduced in Figure 51

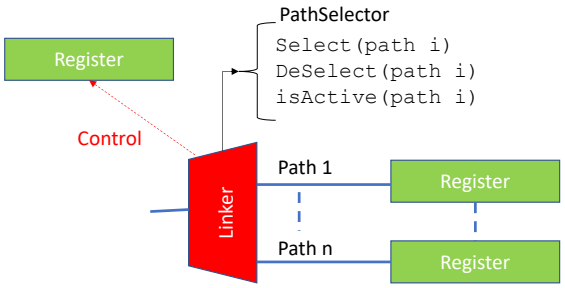


Figure 51 Complete Specification of a Linker Node, from [1]

4.3.2 Interfaces

The problem of Domain Translation needs an additional abstraction step: as depicted in Figure 42, the usual approach is to extract the Primitive Operations expressed in a DSL, process and convert them into the new Domain and express them once more in a new DSL. This solution is an extension of the legacy Static Retargeting of Figure 45, and suffers from the same limitation: by forcing a static resolution, useful information is lost and each step has to recreate it from scratch. Moreover, hardware description languages like ICL are based on a structural approach, which is ideal for ATPG or pattern generation, but not for Domain Translation. It is in fact extremely difficult to extract behavior from structure: for instance, if a Tool can easily parse ICL and use the connectivity information for Retargating, it cannot understand how to use a JTAG or I2C interface just from their AccessLink descriptions. Even doing it from an HDL such as VHDL and Verilog is not an easy task, as any Validation expert knows well. To avoid these issues, we propose the abstraction depicted in Figure 52.

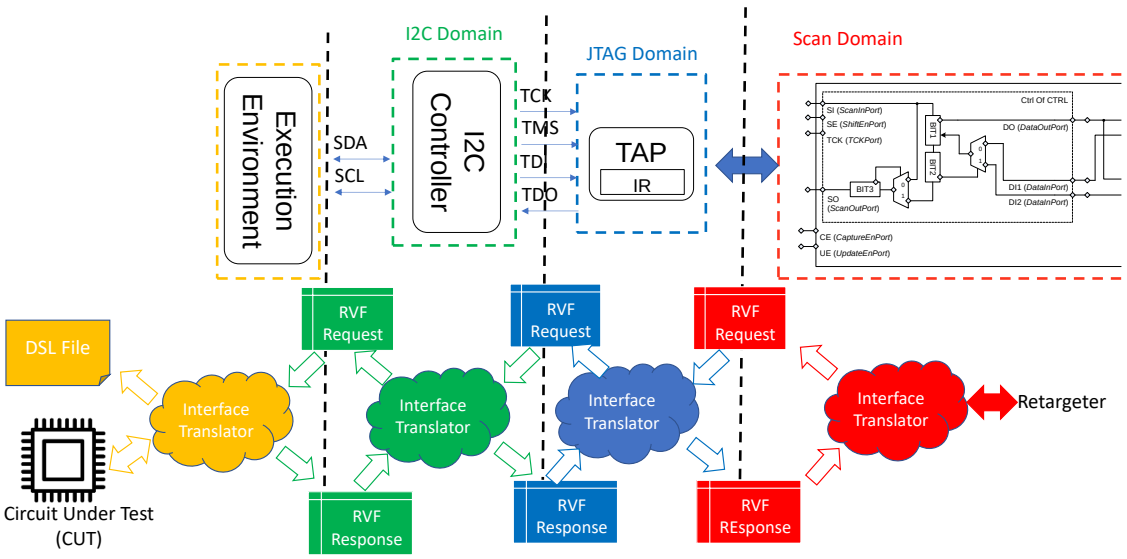


Figure 52 Domain Crossing through the Relocatable Vector Format

The principle is to exploit a new piece of data, the "Relocatable Vector Format" (RVF), detailed in Table 4, to express the Primitive operations in each domain, and use it to cross the boundaries

through some Interface Translation. As it name clearly states, the principle is to reproduce the Software Relocation Flow. The RVF format was firstly introduced in [J.4] and is based on a Request-Response protocol: for each Domain, the Tool generates an RVF Request to perform a certain Primitive, and passes it to next Domain. There, the Request will be processed and if needed forwarded to the next Domain. At the end of processing, each Domain will generate an RVF Response which can contain both Status information and return Data (in an interactive approach).

Table 4 Specification of the Relocatable Vector Format (RVF)

RVF Request		
Field Name	Type	Description
Data	Binary	Binary representation of the vector to be sent to the SUT
Primitive_idf	String	Primitive identifier
Optional Data	binary	Primitive-specific data
RVF Result		
Field Name	Type	Description
Status	String	Information about the execution of the previous RVF Request
Data	Binary	Binary representation of the vector received from the SUT

The RVF format contains the Minimum Information Set which is needed to make a Translation between Domains as it expresses only the Primitive itself. The key abstraction points are:

- Data is stored in Binary terms, to avoid the complexity of expressing it to specific string formats (ex: SVF if right-aligned, while I2C usually prefers left-alignment);
- A Primitive is expressed as string. This is for both debug and inter-operability: using Binary Enumeration can be more optimized in terms of memory space, but it can lead to confusion especially when different Tool Providers are in play
- The "Optional Data" field allows RVF to vehiculate data which is proper to the interface and not directly expressible in ICL. For instance, I2C or SPI need the target addresses for Read/Write operations.

For instance, and "SDR 0x1234"command can be directly coded as ("SDR",0x1234), while an "I2C_Write(data=0x1234,address=0x27) command will become ("I2C_Read",0x1234,0x27).

The Interface Translators of Figure 52 are the final point in solving the issue: moving away from a purely structural point of view, the abstraction embraces a Functional approach. In coherence with the localized Linker abstraction of the previous section, each Interface is responsible for providing a Translator that is able to convert RVF requests coming from its right-hand side in a Protocol (i.e. a set of Primitive) into one or more RVF Requests in another Protocol on its left-hand side, and vice-versa for the Response path. Each Translator is based on the behavior of its

corresponding hardware, extracted rather from the Data Sheet and documentation than from the HDL. For instance, Figure 53 proposes an example of RVF Translation for a JTAG-to-I2C translator which expresses "SDR" operations as a sequence of a Write and a Read at address 0x12.

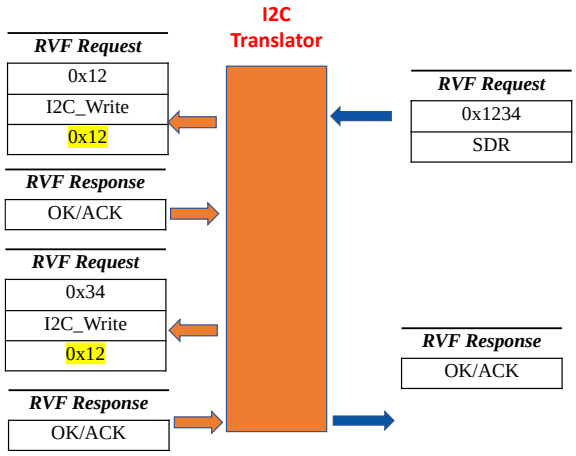


Figure 53 RVF Translation for a JTAG - to - I2C Translator

In terms of Circuit Model, this abstraction is summarized by the Interface Translator Node, depicted in Figure 54. The "Channel" models the different connections to the Interface. For instance, a JTAG TAP has two channels, connected to the IR and DR chains respectively. The "Translate" method is supposed to completely process one right-side RVF Request, if needed generating multiple Requests on its left-hand side and composing the left-hand side Responses, finally returning the result as a unique RVF Response on its right-hand side.

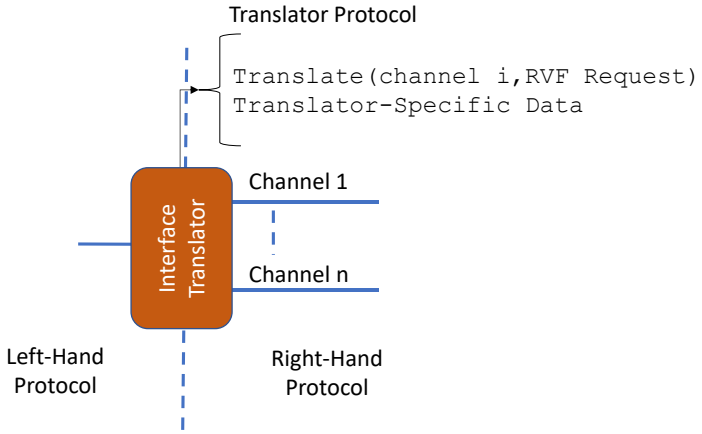


Figure 54 Complete Specification of the Interface Translator Node

The scheme of Figure 52 supposes that the Retargeter is able to issue RVF Requests and process RVF Responses. However, this might not always be the case, especially for legacy tools. To allow compatibility, we therefore propose the AccessInterface node, depicted in Figure 55, which instead of receiving RVF Requests uses the Primitive Identifier and Data computed by the Retargeter on its right-hand side to start an RVF flow on its left-hand side.

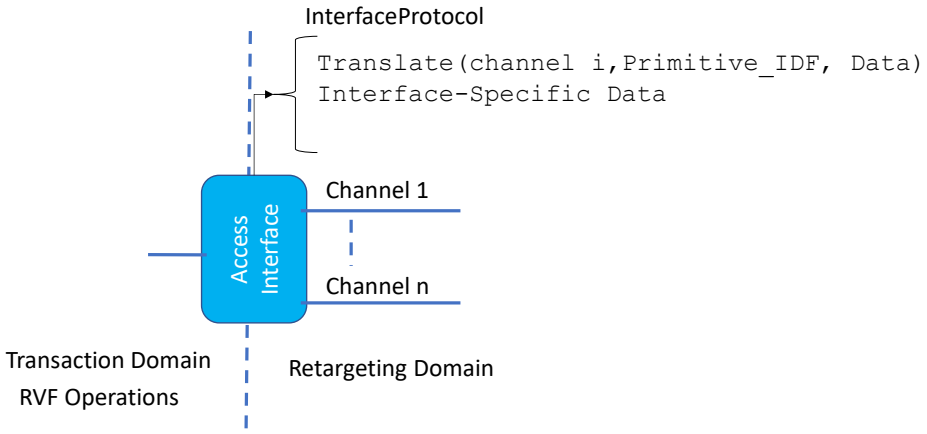


Figure 55 Specification of the Access Interface Node

4.3.3 Conclusions

To conclude, the different building blocks of the Circuit Model introduced in this Section are assembled in Figure 56. The Test Manager can interact with it thanks to the methods defined at each Node.

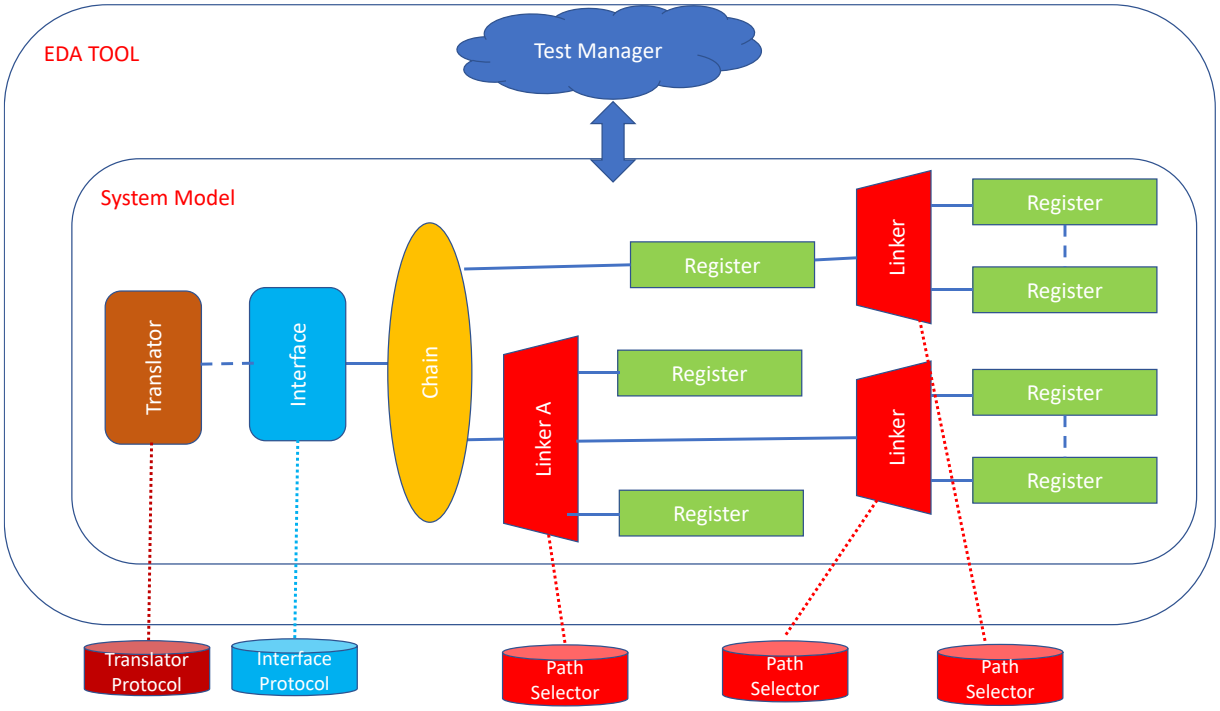


Figure 56 Complete Circuit Model Abstraction

Please also notice that the PathSelectors and Protocols are not necessarily part of the EDA Tool itself: as they are exclusively based on the RVF abstraction, they can be developed separately (typically, either by the IP Designer or the System Integrator) and included as separate modules. Section 6 will provide an example of this process.

4.4 Test Manager

As depicted in Figure 46, the Test Manager is responsible for the interface and coordination between the Test Algorithms and the Circuit Under Test. The actual algorithms and methods are implementation-dependent: the role of the Test Manager in the Abstraction if therefore of a place-holder/black box that is in the ideal place to collect the information provided by the other element and act upon it. In the following two sections we will show examples of how the Test Manager can fulfill its role to solve the issues of Vertical and Horizontal Retargeting, and how the Circuit Model Abstraction can provide the necessary information and data.

4.4.1 Vertical Retargeting

For Vertical Retargeting, the Test Manager must be able to link the PDL operations and the Registers they target. Figure 57 depicts the complete setup: a PDL algorithm, in the upper left-hand corner, requires access to a specific Target register. As explained earlier, this source code is compiled into a Relocatable Executable, which contains the API Calls for PDL. If we consider the Instructions specifications as reported in [1687] in Table 1 and 2, their implementation by the System Manager by exploiting the Circuit Model (CM) can be done in three ways:

- Modifications to the CM. For instance, an "iWrite" instruction modifies the Future Value its Target register (as defined in Section 4.3.1), having it effectively queued as required in the Standard. As a result, the State of the CM is no longer fully synchronized with the State of the CUT : S(SM)≠S(CUT).
- Queries from the CM. For instance, an "iRead" returns the last value read from the CUT and stored in the CM;
- CM/CUT Synchronization. Instructions such as iApply require queued modifications to be applied, therefore restoring the equality between S(CM) and S(CUT)

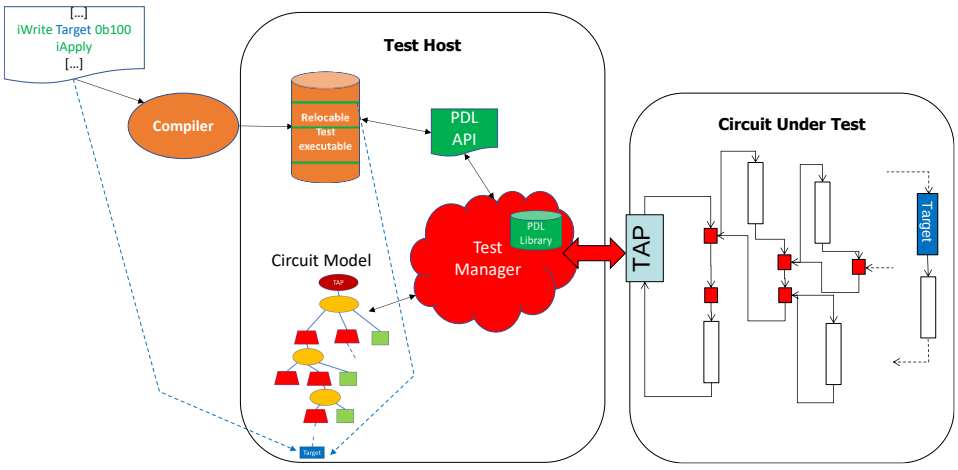


Figure 57 Complete Setup for Vertical Retargeting

The actual implementation of the PDL Library is not part of this Abstraction, so each Test Manager can implement it as it better suits its constraints and goal. The same goes for the CM/CUT synchronization Algorithm, which can exploit the Circuit Model abstraction to identify the S(CM) needed to satisfy the PDL instruction and guide the S(CUT) to it. However, the Circuit Model provides all the information and methods needed to manipulate both S(CM) and S(CUT).

To demonstrate its usage, we will unroll the solution of the example of Figure 57. First, we will call S^0(CM) and S^0(CUT) the states of CM and CUT respectively at the beginning of the execution of the provided PDL code. Figure 58 depicts this setup, where for simplicity, we considered that at that at time 0 all ScanMuxes are closed. The dotted line highlights the Active Scan Path in both CM and CUT, while the red arrow the execution position in the PDL code.

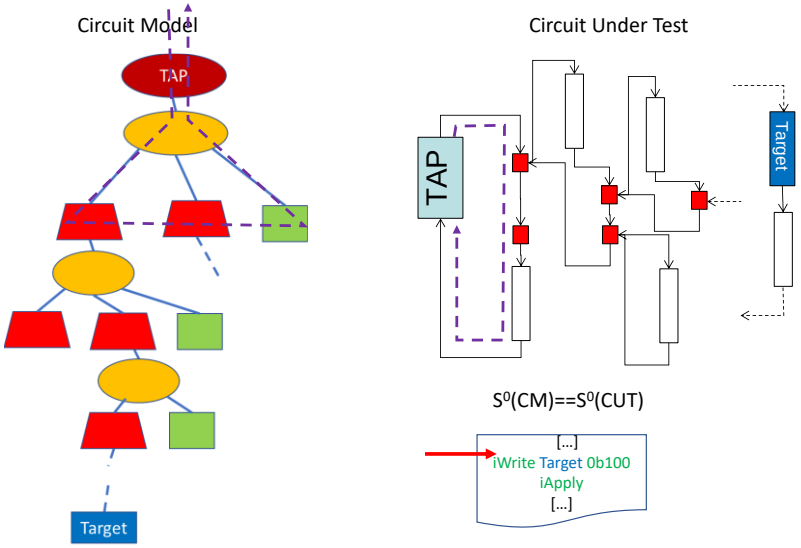


Figure 58 Circuit Model and CUT status at time 0

The result of the execution of the iWrite command is shown in Figure 59

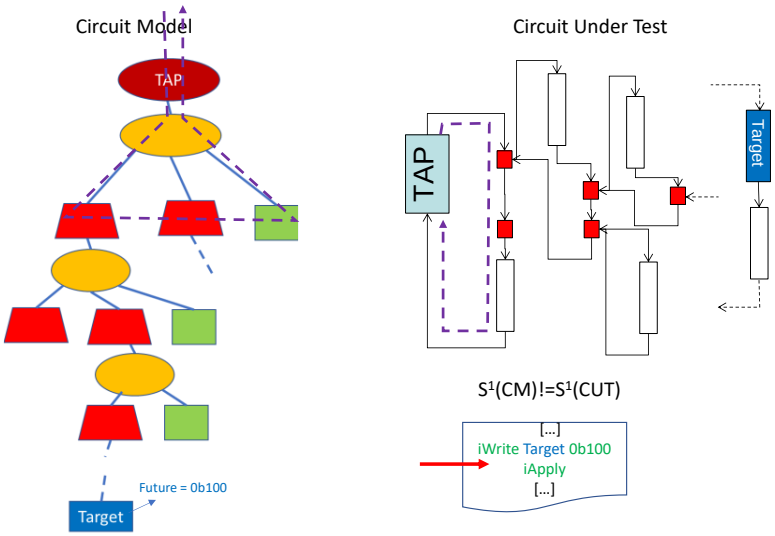


Figure 59 Status mismatch caused by a PDL instruction

The iApply instruction demands a synchronization between S(CM) and S(CUT). For this, the Task Manager needs to compute the S^{TARGET}(CM) for which the Target Register is part of the Active

Scan Chain, and the sequence "n" states which will bring the CM to the desired state.

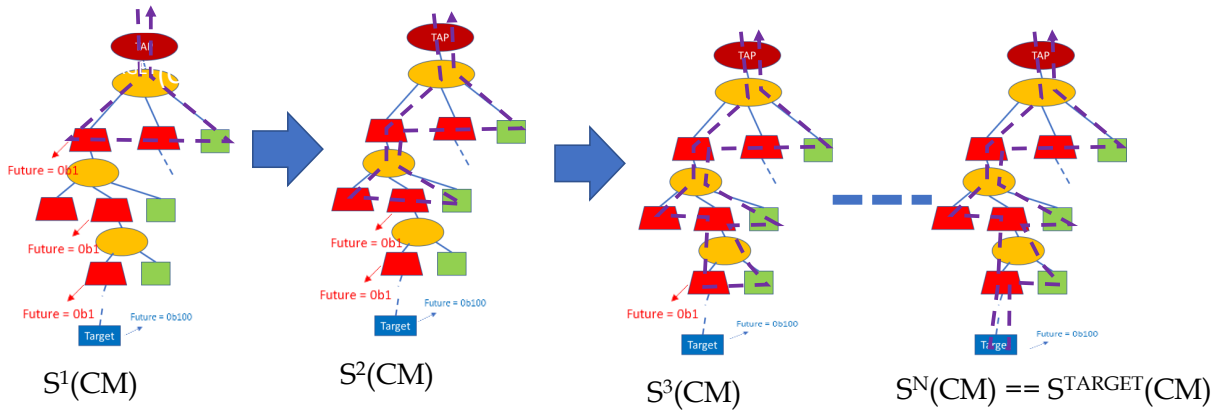


Figure 60 depicts one possible sequence. Please note how each Selection Request of a Linker queues a state modification for its controlling register, and how each S^i(CM) in the sequence can resolve at least one of these requests. For readability's sake this Configuration requests are marked on the Linkers in

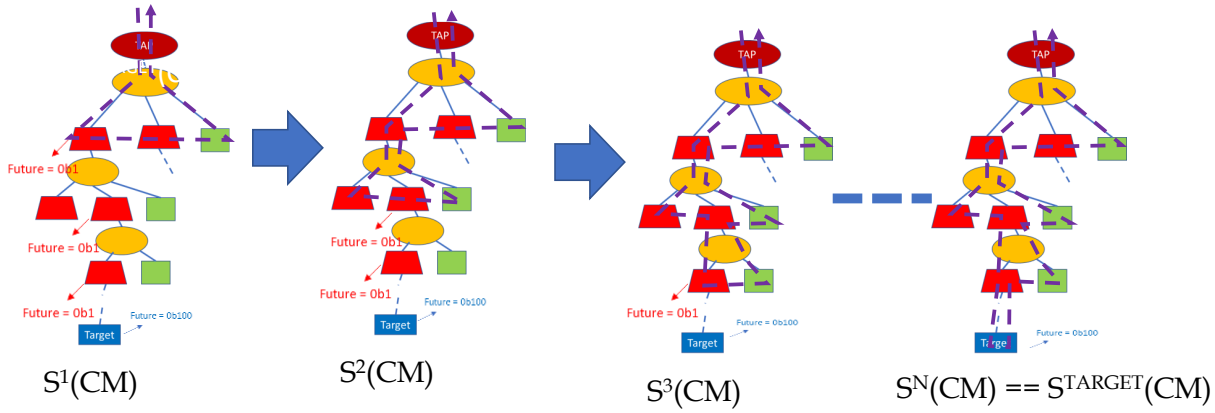


Figure 60, but in reality, they result as Future Values change requests in the controlling register(s), as explained in Section 4.3.1 and depicted in Figure 49 and Figure 50.

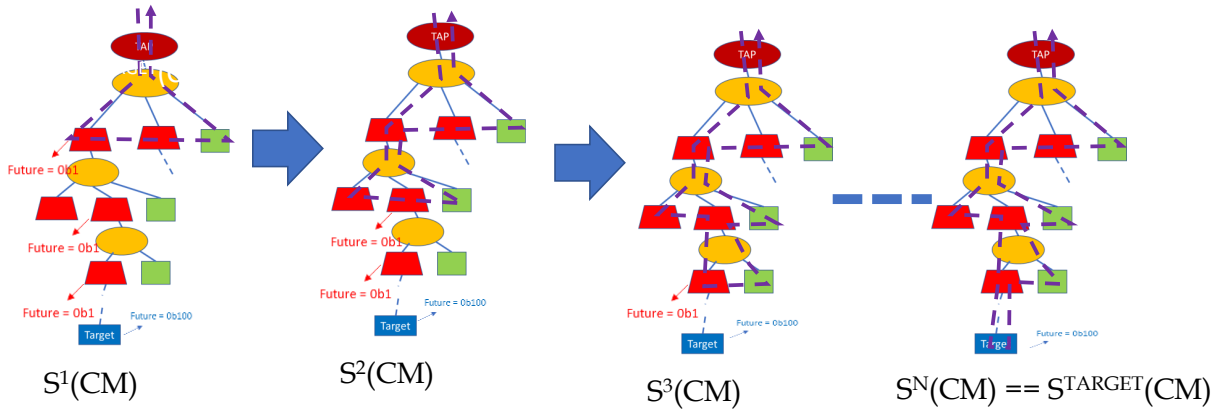
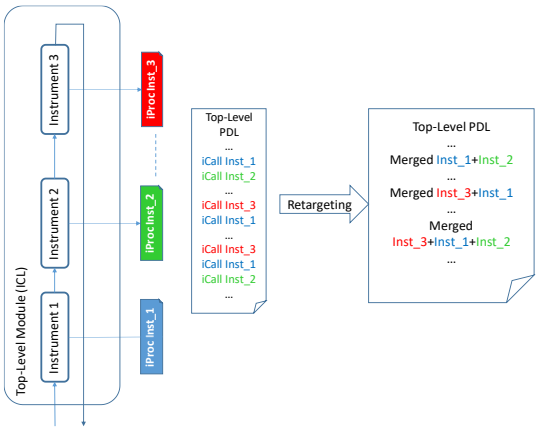


Figure 60 Sequence of n CM state reaching S^{TARGET}(CM) from S^0(CM)

The computation of both $S^{TARGET}(CM)$ and the sequence $S^{0\rightarrow N}(CM)$ is not unique and is left to the Implementation of the Abstraction. It could use classical solutions like Dijkstra, any modern shortest-path algorithms or even incremental algorithms as in the MAST Implementation that will be described in Section 5. Once the sequence of states is known, the Test Manager can apply them to the CSU until $S^{N}(CPU) == S^{TARGET}(CM)$ and the processing of PDL operations can continue. As this unrolling shows, a great part of the Task Manager is implementation-dependent, leaving a lot of freedom to each EDA implementation. However, all variants are only possible thanks to this Abstraction, in particular to the information stored in the Circuit Model.

4.4.2 Concurrency (Horizontal Retargeting)

In Horizontal Retargerting, the Test Manager must be able to make the connection between several PDL Instructions that need to be pushed to the SUT. In the traditional approach, depicted in Figure 28 and reproduced here for easier reference, the EDA Tool aims at producing a top-level PDL Stream by statically extracting concurrency from a PDL stream: the iMerge instruction is provided to highlight sections whose execution might be parallelized. The principle is that "everything is sequential if not told otherwise".



Reproduction of Figure 28 Horizontal Retargeting Merging for a 3-instrument 1687 system [J.4]

This is a direct evolution of the classical flow, where the User manually composes the top-level Test Strategy. However, the extraction of concurrency from a sequential program is a problem that is computationally extremely complex. In Computer Science, especially when an Operating System is present, the general assumption is the opposite: "everything is parallel if not told overwise". Processes and threads are executed in parallel thanks to the Scheduler, and provisions are given to force sequential behavior when needed (such as semaphore, rendez-vous, etc…) [TANE15]. This same principle can be applied to Testing: each IP comes with a set of Test Procedures, which must be executed in parallel to test the whole system. Our new Test Flow can support parallel execution in the same way, as depicted in Figure 61. The Test Manager has the same role of the OS: arbitrating the access requests from the Test Algorithms to the limited resources of the CUT, through the often-unique interface.

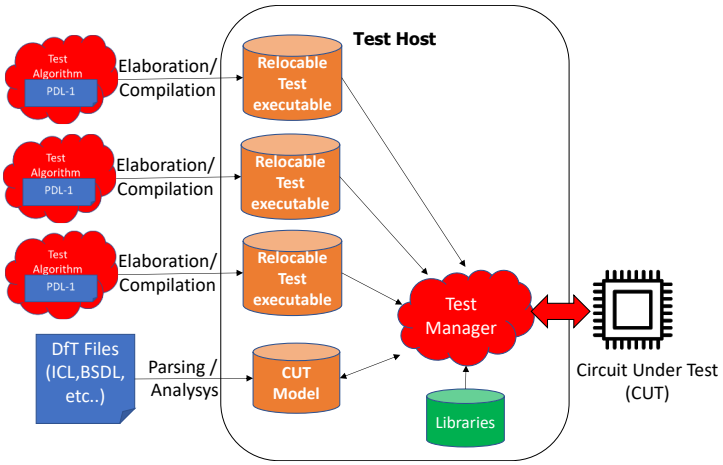


Figure 61 Horizontal Retargeting in the new Test Flow

Regardless of the concurrency principle chosen in a given implementation, once more, the Circuit Model provides the key to the resolution: thanks to the PDL API, each Test Algorithm will either iRead data from it, or iWrite modifications to it. As a result, the S(CM) loses its synchronization with S(CUT) while storing all the modifications as changes to the Future values of the target registers, as depicted in Figure 62.

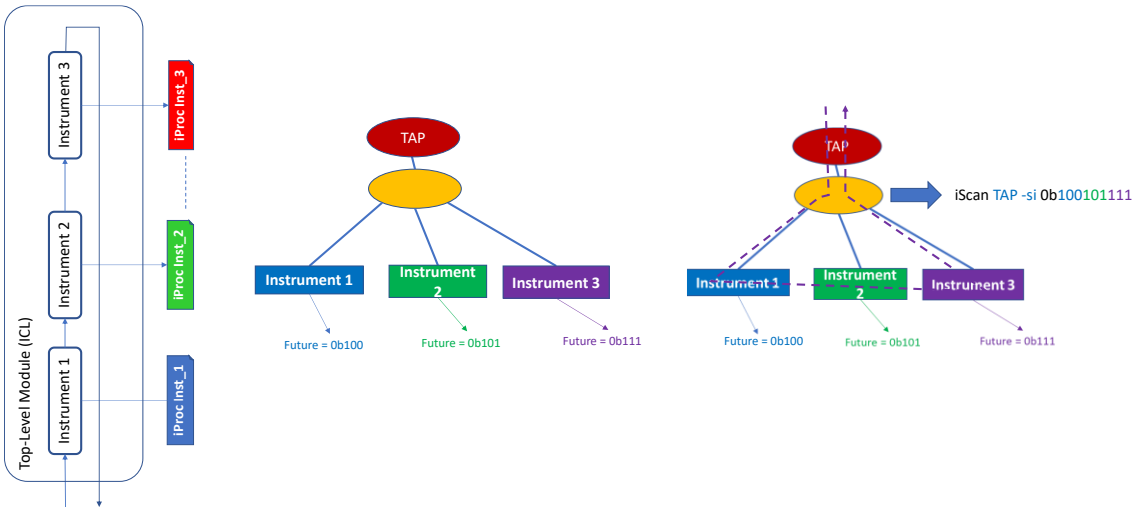


Figure 62 Circuit Model Abstraction for Horizontal Retargeting

Horizontal retargeting is once more solved thanks to a tree transversal that identifies the Active Scan Path, and can therefore service the queued PDL operations. On the other hand, the way these modifications are queued and how the concurrency of the Algorithm is handled is left to the Tool Implementation. In the following Chapter we will show the Implementation chosen in the MAST tool, and also some possible variations using the same Abstraction introduced here.

The clear advantage of this Abstraction is that both Horizontal and Vertical retargeting are solved through an analysis/transversal of the Circuit Model. It is therefore possible to solve the two at the same time, as shown in Figure 63

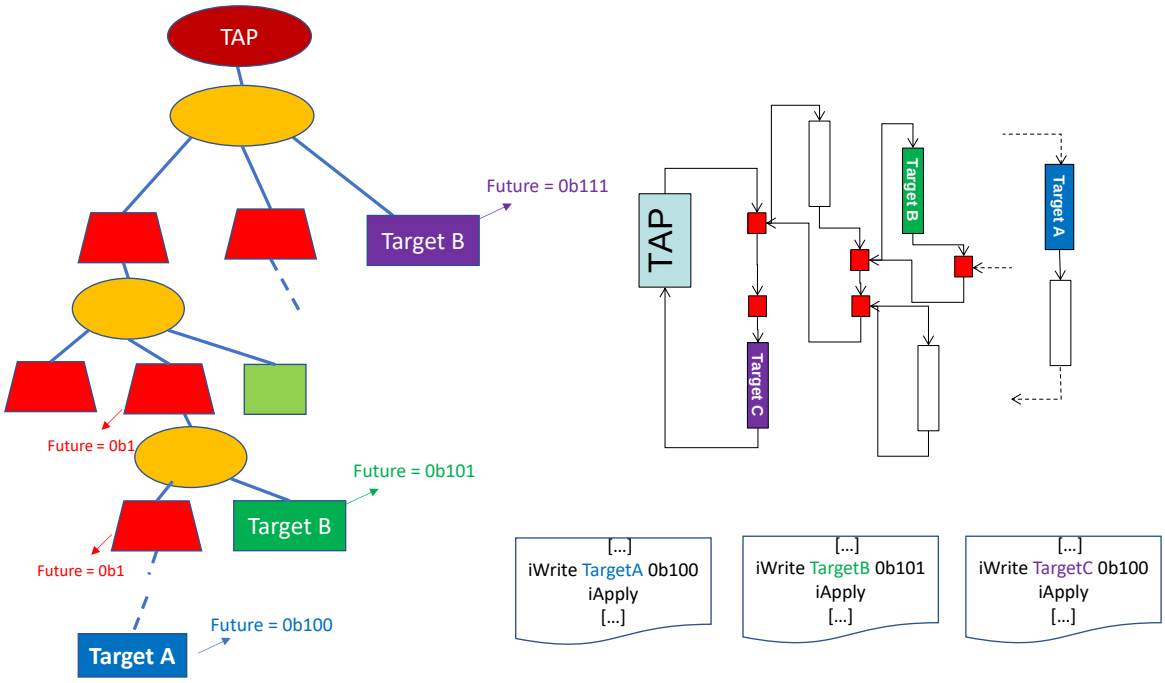


Figure 63 Circuit Model Abstraction for Horizontal and Vertical Retargeting

Both PDL Operations and Configuration actions result in Modifications Requests to S(CM) and in a desynchronization with S(CUT), so they can be solved in the same way through the algorithms and strategy presented in this Section.

4.5 Domain Crossing and RVF propagation

The last role of the Test Manager to handle the Domain crossing, so that the operations defined in a Protocol (i.e. a set of Primitives) can be translated to another, eventually obtaining Primitives for the interface with the CUT. This is done by propagating the Operations expressed as RVF Messages through the Interface and Translator nodes introduced in Section 4.3.2 through the Circuit Model nodes, as depicted in Figure 64. The transformation and forwarding of the messages are left to the of the Protocols defined into each node of the Abstraction, and introduced earlier in this Chapter.

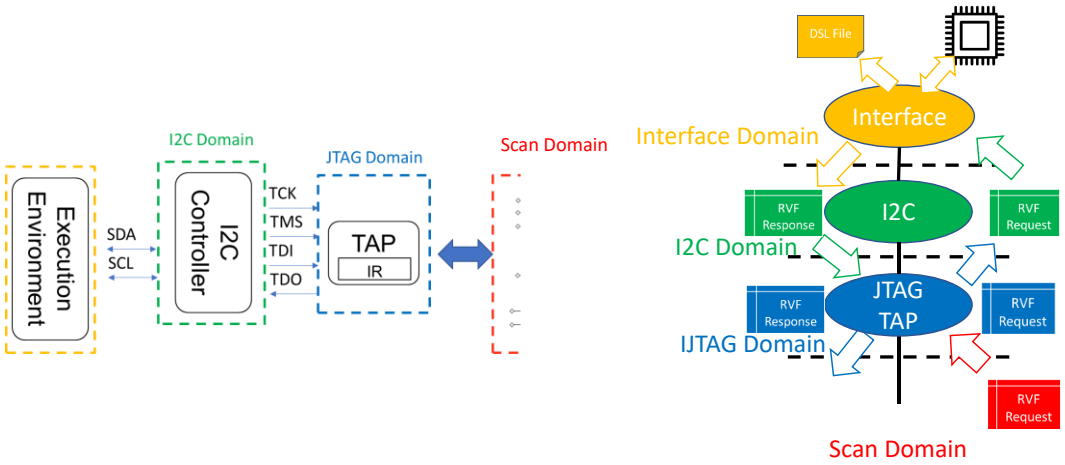


Figure 64 RVF Packet Propagation on the Circuit Model

Once more, the actual shape of the RVF Messages inside the tool and the Messaging Layer responsible of their propagation is left to the Implementation, of which an example will be given in the next section for the MAST Tool.

5 A New Automated Test Flow: Implementation

The Abstraction presented in the previous Chapter is the result of almost 15 years of work. In this Chapter, I will retrace the iterations between Analysis and Implementations that allowed not only to develop and refine the Abstraction itself, but also to focus the work to the actual needs of the Testing Field.

5.1 Early Developments : Test Instruction Set Architecture and NSDL: 2007 → 2013

I started working in the field of Automated Testing in 2007, upon my arrival in Bell Labs Ireland, by joining the IEEE P1687 Standardization Working Group. At that moment in time Alcatel-Lucent did not have any major ASIC project: rather than influencing the internal DfT strategy, the priority was to push into the Standard features that would be useful later in the Design flow. As a new member of its Research & Development Division, my manager Suresh Goyal put me in relationships with three Bell Labs Senior Members, Bradford Van Treuren, Tapan Chakraborty and Chen-Huan Chiang, each having more than 20 years of experiences in the historical locations of Whippany and Murray Hill in New Jersey, USA. Together we formed a distribute research group where my role was to collect their requirements and expectations and transmit them to the new Standard. The aim was to both understand the direction the standard was going and to influence its development in a 5-to-10 years window.

The first step was to develop an internal demonstrator for the P1687 proposals and testbench it against the most use Boundary-Scan tools used in Alcatel-Lucent, such as for instance ScanWorks from Assett Intertech. The result, depicted in Figure 65, comes from the feedback from the expert user and mimics the real flow used by Test Engineer in the company: the Circuit Under Test (CUT) in the right-hand side is a VHDL-based FPGA implementation of a P1687 network as it is was being discussed in the WG at that moment in time (registers and simple SIBs). Its TAP can be accessed from a Test Host using the Impact tool of the Xilinx ISE suite, which exploits their proprietary JTAG controller and a built-in IP which redirects the bitstream inside of the FPGA rather than to its configuration memory. Impact accepts SVF files, which are generated by the ScanWorks tool from the BSDL description of the ML505 board, provided by Xilinx. Inputs and outputs of the P1687 networks can be set/observed using onboard components (Leds and Dip-switch respectively). Last but not least, the Chipscope tool from Xilinx allows monitoring of internal signals of the P1687 network by means of a "virtual oscilloscope". The content of the BSDL and SVF files must be manually edited to add the new P1687 features.

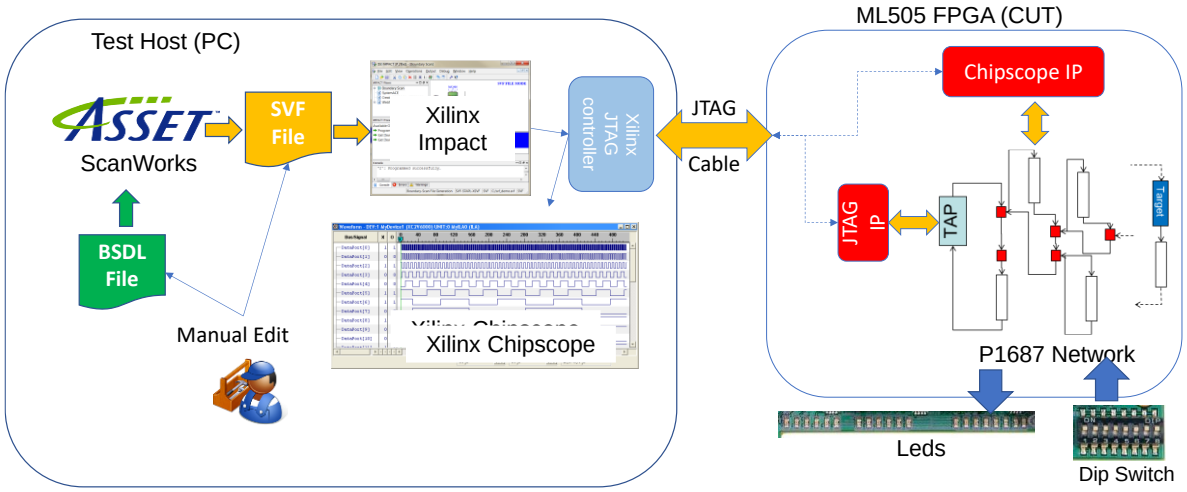


Figure 65 P1687 Demonstrator (2007-2008)

Extensive experiments on this platform highlighted two main criticalities:

- The inadequacy of BSDL to describe the dynamic SIB-based topologies of P1687
- The limitations of SVF to express functional operations over P1687 instruments

As the Standard discussions were focusing on the Language issue, and I had experience in both Grammars and Compilers, I decided to focus on it. Moving away from both the enumeration-based BSDL and the structure-based RTL, I specified and implemented NSDL, the "New Scan Description Language". For the first time in Boundary Scan, NSDL proposed a description based on the hierarchy of the system rather than on its signal-level connectivity, and allowed functional procedures to be freely mixed with the hardware description. This was achieved by leveraging the capabilities of VHDL, the base language for BSDL, which already contained most of the desired features (components, hierarchy, functions, etc…) and adding only the specificity of P1687. Through an informal collaboration with the University of Maynooth, Ireland, we modified a VHDL parser (the front-end of the Open-Source tool GAUT [MAR93]) to verify NSDL and produce and intermediate XML file that could be fed to EDA tools, therefore proving its usefulness in a real flow as the one in Figure 65

First presented at some workshops ([W6], [W7]), NSDL rapidly raised interest in the field and we were able to publish at conferences [28] and journals [J.6]. In parallel, following the aggressive IP policy of Bell Labs we filed several patents to cover the language, all of which were eventually granted [P2][P3][P4]. NSDL was presented to the P1687 Working Group and was for some time one of the candidates to become the Standard's language, but it was eventually discarded for several technical and political reasons. Technically speaking, NSDL had two shortcomings:

- It was based on VHDL, which is a language mostly used in Europe, while most (or all) of the WG member were American and had therefore little familiarity with it. VHDL's strict typing and grammar makes it difficult for the newcomers, and that difficulty was transferred to NSDL;
- The hierarchy-based abstraction was too high-level for the point of view of the WG members, which preferred to stick to a more classical structural view, which eventually became ICL.

Politically, the patenting policy of Bell Labs backfired: even though the company promptly produced Letter of Assurances (cfr. Section 2.2) the WG did not want to risk selecting proprietary solutions and risk a "patent blockade" (it was the period of the Apple vs Samsung controversy). Even if it was not finally retained, NSDL still had a deep impact on P1687 by pushing the limits of what was possible to describe, and influenced the final version of ICL.

In parallel with NSDL, I focused on the second P1687 shortcoming: the lack of a proper way to express functional operations over Instruments. This was caused by the limitations of DSLs like SVF, which could only express JTAG operations and nothing else. In strict collaboration with my New Jersey colleagues, in particular Brad Van Treuren, we decided to explore a completely new direction: instead of enriching the DSL with algorithmic capabilities as done for instance in [STAPL] and STIL [1450], we decided to enrich the processing capabilities of a CPU with Scan features. We therefore developed the "Test Instruction Set Architecture" (TISA) a set of processor instructions able to control a JTAG TAP in the same way an SVF file can. The actual implementation of the TISA instructions is proprietary to Bell Labs, so in this document I will focus on its high-level goals and abstractions which have been disclosed either in publications [J.5][26][W5] or in Patents [P6][P8][P9][P10][P11][P12][P14].

The principle of TISA is depicted in Figure 66:in order to shift a scan chain composed of several Segments, TISA Instructions can be used to reference each segment separately, instead of having a single top-level SDR command. Each instruction will activate the TAP signals (most notably the clock) to shift enough bits for the length of the Segment it is referencing. The execution of the sequence of TISA Instructions has therefore the effect of shifting the whole scan chain: Horizontal Retargeting is reduced to a simple sequencing problem.

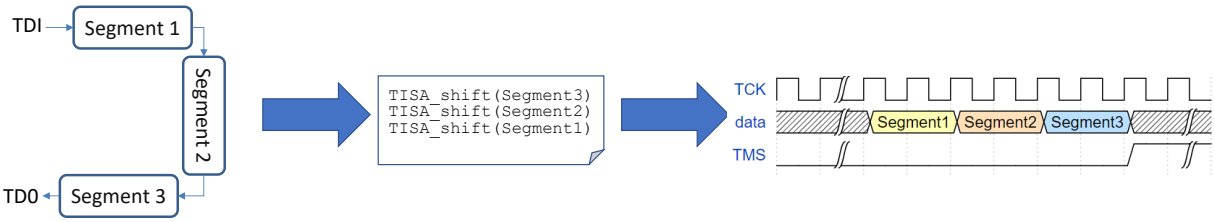


Figure 66 TISA Principles

TISA and traditional ISA instructions can be freely mixed inside an executable file such as [ELF95], so the execution of a TISA-enriched algorithm can be expressed using the Use Model of Figure 67.

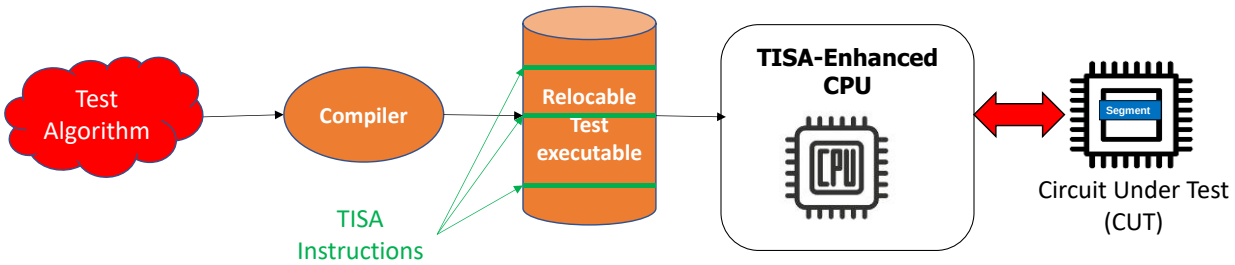


Figure 67 TISA Single-Algorithm Use Model

For the first time, it was possible to fully integrate Test and Software flows. However, the flow required the Compiler to accept TISA instructions, and the User to correctly (and manually) write the sequence of TISA Instruction, considering also the status of the SIB. To remove these locks, we developed thanks to the Master Internship of Josef Ahmad the setup of Figure 68.

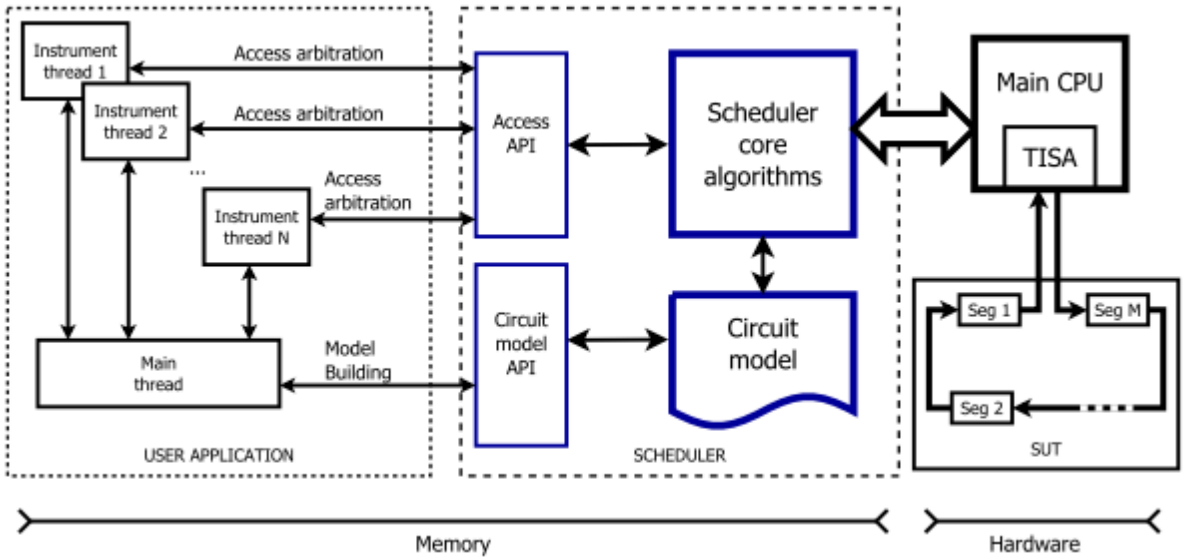


Figure 68 Complete TISA Setup, from [J.5]

First of all, we defined a target CPU for the TISA extension: the Leon2 processor from Gaisler Research [GAI]. Now replaced by its newer versions (such as Leon5), it was at that moment in time one of the few Open Source Softcore processors with a fully-functional software stack, and I had experience with it thanks to my PhD. I first defined a binary mapping for TISA instructions compatible with the Sparc-V8 ISA [SPARCV8] exploiting the "unimplemented" field values, and then the Master Thesis started by adding a modified Leon2 architecture target to GCC to support the new extension set in the ELF flow. Then we developed the system of Figure 68, which is the first embryo of the Abstraction detailed in Section 4. The TISA instructions were both implemented in a VHDL co-processor for simulation and FPGA emulation and in software to generate on-screen debug messages and traces.

There were however some limitations:

- The Circuit Model was incomplete and only supported a simplistic version of the SIB;
- The Circuit Model was to be built directly into the code, and the correspondence between nodes and Instrument threads done manually through named mutexes.
- The Instrument threads would not execute PDL commands, but needed TISA assembler instructions to be hard-coded as inline "asm" code;
- The arbitration mechanism was based on the over-mentioned mutexes, with each Instrument thread encasing its TISA instructions inside a Critical Section [TANE15], and communicating with the Scheduler thanks to Request/Acknowledge protocol, depicted in Figure 69

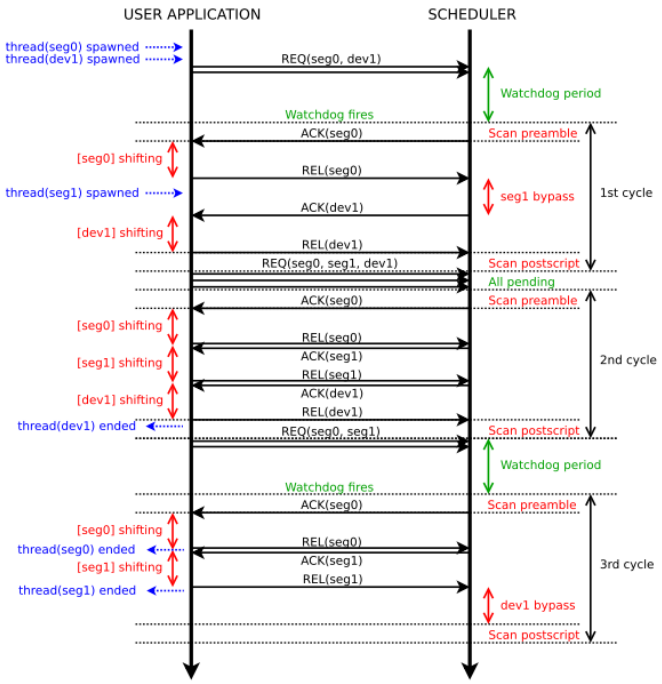


Figure 69 TISA Scheduler protocol, from [J.5]

Regardless of these limitations, by the end of the Internships we had a fully functional demo, which proved the feasibility of the TISA approach.

The next improvement of the TISA setup was boosted by un unexpected opportunity: upon me joining the Paris Bell Labs location in 2011, I had access to one of the first ZC702 cards available in France, [ZC702] : Xilinx has just released its new Zynq SoC concept joining the flexibility of an FPGA fabric and the performances of a dual-core Cortex A9 ARM hard processor, along with a set of 4G-oriented IPs, and was aggressively advertising it to Alcatel-Lucent, one of its main customers (most of the specialized hardware of Base Stations at that time was implemented using Xilinx PFGAs). I therefore decided to port the hardware version of TISA on the new platform to leverage the significant software performance gain: the hard-macro ARM processor had a working frequency configurable between 0,5 and of 1Ghz, while the Leon 2 soft core had a working frequency of 50Mhz. This meant also porting the Software TISA flow to the processor: luckily in contrast with the SPARC V8, the ARM v7a ISA contains some dedicated co-processor instructions [ARMv7] which can be used by an implementer for his own hardware accelerator. I was therefore immediately able to cross-compile and use the complete TISA setup of Figure 68 without needing any custom change in the GCC cross-compiler. JTAG being notoriously slow, performances were not an issue so I opted for a classical co-processor setup on the AXI bridge, depicted in Figure 70 : the TISA coprocessor is plugged to the AXI bus of the ARM core as a Slave, and interacts with outside thanks to one of the GPIO banks of the card. The TISA test algorithm can be directly compiled by GCC, but as the ARM core is a hard macro, it is not possible to modify its Decode stage: Coprocessor instructions will therefore trigger an Illegal Instruction Interruption. The Handler is used a driver: it decodes the instructions and performs the memory-mapped communication with the TISA processor to execute it.

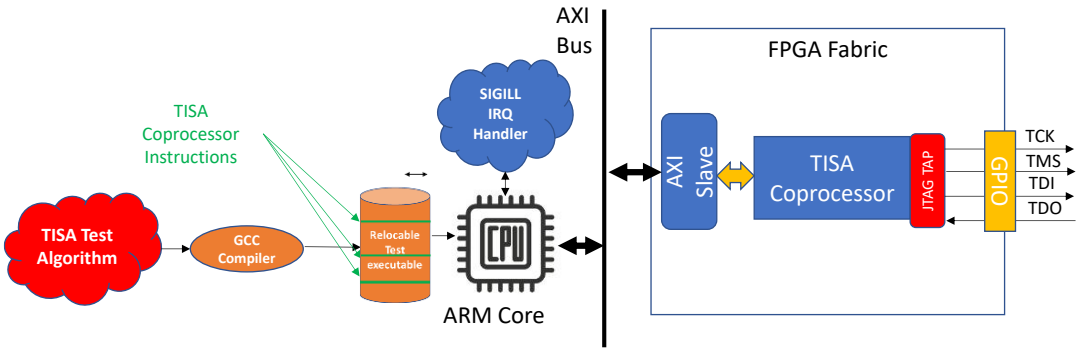


Figure 70 TISA Hardware demonstrator on Zynq

With this demonstrator the TISA ecosystem was finally complete and we could demonstrate its usage on a complete FPGA target. Figure 71 reports the experimental results: the chronograms are a direct replica of the waveform captured on an oscilloscope (which did not have a record function).

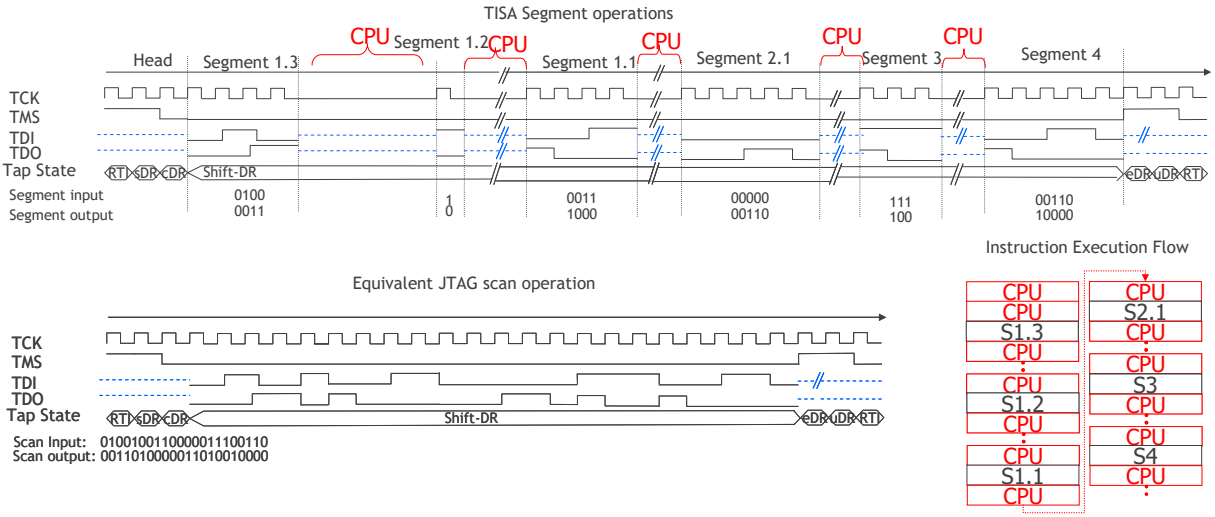


Figure 71 Execution of the TISA Hardware demonstrator, from [J.5]

We also measured TISA scheduling time depending on the number of parallel threads, and demonstrated its capability to scale linearly. At the moment of the publication of [J.5], no other solution could boast similar performances.

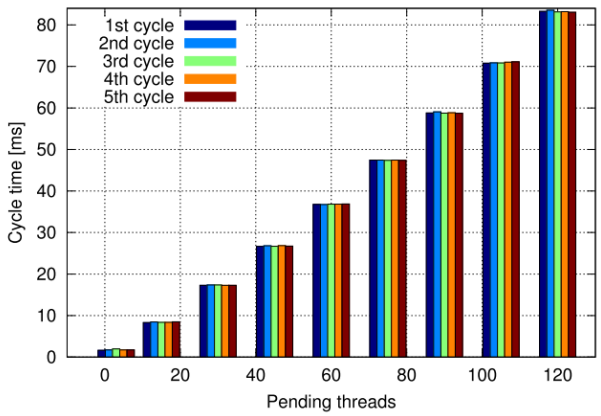


Figure 72 TISA Scheduling Performances, from [J.5]

5.2 First Abstractions: New System-Level Test (NeSLT : 2013 →2015)

In 2013, I decided to leave Alcatel-Lucent and join Grenoble-INP Phelma as an Associate Professor. This resulted in the impossibility to access the TISA technology that remained propriety of my previous employer. Through Grenoble-INP Valorisation Department, in particular thanks to the efforts of Wahiba Robert and Isabelle Chery, I launched a licensing project with Bell Labs to achieve Research access rights to the technology, but the promising discussions were halted by internal problems. Alcatel-Lucent entered a Restructuring phase which ultimately ended up with the closing of Bell Labs Ireland. All my contacts were either fired or moved to different positions, effectively ending the project.

I took this setback as the opportunity to come back to the drawing board and develop a completely new setup which would be independent from by Bell Labs' work and able to overcome its limitations. In fact, upon a closer inspection a practical usage of TISA was limited by these factors:

- the execution of the setup strictly relied on the correct sequence of TISA instructions belonging to different Segments. In a real setup, each Test Algorithm might come from different actors and EDA tools suites, which would be responsible of the translation of PDL commands into TISA instructions. Any ambiguity or difference in the translation algorithm would result into non-compatible sequences, with unpredictable behavior;
- the Request/Acknowledge protocol had to be directly implemented in each thread. An error in one of them (ex: a thread not releasing its mutex or executing TISA instructions outside of a Critical Section) would also result in unpredictable behavior or deadlocks;
- The Circuit Model was simplistic and relied a simplified model of the SIB, making extensions to other types of Topologies close to impossible;
- The need of a specialized Co-Processor restricted the application domain to embedded controllers, cutting out the most used Execution Environments such as ATE or Desktop.
- Last but not least TISA was completely JTAG-centric and unable to support any type of alternate Interface.

To solve these issues, I developed two fundamental bricks of the Abstraction of Section 4:

- The Linker Abstraction, which reduces SIB as a simple example;
- The Test Manager and the PDL API, which replaced the Scheduler.

The second point is probably the most important: in contrast with the TISA Scheduler, which simply decides in which order each Thread can communicate with the CUT through the TISA instructions, the Test Manager is the single access point to the CUT. This allows a solution for each of the previously-mentioned weaknesses:

- Threads can queue their modification requests inside the Circuit Model, but only the Test Manager can propagate them to the real hardware. This excludes the possibility of unpredictable operations on the CUT.
- The PDL commands are directly implemented by the Test Manager, rather than by a third party. This means that the Request/Acknowledge protocol is part of the Manager itself, making their usage predictable and excluding bad usage (in the absence of bugs).

- The Linker Abstraction makes the Circuit Model extendable to arbitrary topologies
- Interaction with the CUT is centralized by the Test Manager, which is therefore responsible for controlling the Interface. There is therefore a specific step where JTAG instructions are created, and which could be modified to support other Interface types.
- The translation between retargeted vectors and Interface operation is done purely in software, without the need for specialized hardware

Following the usual method, I developed an Implementation of this Abstraction: the "New System-Level Test" (NeSLT). Written in C following an Object-Oriented approach [OOC], it implemented the new Abstraction from scratch, without reusing any line of code of the TISA Scheduler to guarantee independence. I quickly realized that for the complete implementation of the Abstraction I needed more manpower and expertise that I could manage by myself. I therefore looked for Financing opportunity: given its industrial value and potential impact, I engaged discussions with Grenoble-INP Valorisation department: after a first "APP" Software Depot [APP1], I was selected for a Maturation Project by Linksium, the Technology Transfer Accelerators (SATT, from the French "Societé d'Accéleration du Transfer Technologique") of the Grenoble Region. The project was coordinated by Christophe Poyet of Linksium, and the team was soon strengthened by Olivier Bolon, an expert of the EDA market looking for investment opportunities.

5.3 A General Solution: MAnager for Soc Test (MAST: 2015→2017)

As its name states, the aim of a Maturation project is to take a promising technology and make it mature so that is closer to industrial applications. This is usually measured by the TRL, or "Technology Readiness Level". The wording can be slightly different, but the general consensus reproduced in Table 5, is clear. From this definition, NeSLT was clearly a TRL 3: "Experimental Proof of Concept". The aim of the Maturation project was to bring it to TRL 5/6 and ideally up to TRL 7.

Table 5 Technology Readiness Level definition, adapted from Wikipedia

TRL	NASA usage [TRL-NASA]	European Union [TRL-EU]
1	Basic principles observed and reported	Basic principles observed
2	Technology concept and/or application formulated	Technology concept formulated
3	Analytical and experimental critical function and/or characteristic proof-of concept	Experimental proof of concept
4	Component and/or breadboard validation in laboratory environment	Technology validated in lab
5	Component and/or breadboard validation in relevant environment	Technology validated in relevant environment (industrially relevant environment in the case of key enabling technologies)

6	System/subsystem model or prototype demonstration in a relevant environment (ground or space)	Technology demonstrated in relevant environment (industrially relevant environment in the case of key enabling technologies)
7	System prototype demonstration in a space environment	System prototype demonstration in operational environment
8	Actual system completed and "flight qualified" through test and demonstration (ground or space)	System complete and qualified
9	Actual system "flight proven" through successful mission operations	Actual system proven in operational environment (competitive manufacturing in the case of key enabling technologies; or in space)

A quick inspection NeSLT made it clear it was impossible for it to mature. The Abstraction was Object-Oriented, but I wrote the code in "Object-Oriented C" (OCC) because I lacked the necessary Software Development skills in C++. While it is sometimes claimed that OOC is "more performing" than pure OO code, I never found any real data to support this assumption. However, the need to manipulate, convert and allocate a huge number of pointers to mimic Objects and Methods indisputably made the code base difficult to read and maintain. Moreover, performances were not a Key Performance Indicator in the Project: the priority for Linksium was the TRL enhancement with the intention of launching a Start-Up. For me, to have a code easy to maintain and expand even after the Project was finished. I therefore decided for a completely new development with the following goals:

- C++ as the language, to marry performances and maintainability
- Modern Project Management facilities, such as CMake for compilation, GIT for version control, Unit Testing for code quality, etc…
- Replicate NeSLT features from the ground up, without any code reuse
- Privilege code maintainability and readability with reasonable performances
- Extreme portability

Portability was the most important and constraining point: to avoid ending up in the same hyper-specialized solution as TISA, the coding style should follow strict constraints:

- Guarantee OS-independence (targeting Linux and Windows)
- Guarantee Architecture-Independence, notably in terms of Endianness;
- No IDE dependence in the build flow;
- Use only standard C++ library, avoiding solutions such as [BOOST];

The latter point might seem over-the-top, but it is actually key: while libraries like BOOST are widely used and are described as "portable", they are not always included in constrained installation such as Embedded Controllers or even ATEs, where streamlined distributions with the strict minimum of modules are usually preferred.

Through Linksium's channels I was able to recruit Jean-Francois Coulon, a C++11 Developer with more than 30 years of experience. This supervision was also the opportunity for me to raise my competences in C++11 and Software Engineering: we set up an Agile-like methodology, with regular meetings (at least once each couple of days) to validate developments and decide new

tasks. This timeline might remind a Scrum Project Management, but I took ownership mostly of the development guidelines and choice of tools and algorithms, while Jean-François proposed and actuated the low-level task partitioning. In the 10 months of contract we were able to develop a complete solution, the "MAnager for Soc Test" (MAST), which answered all the constraints. After the Maturation project ended, with Linksium we passed into an Incubation project, with another 5 months of contract for Jean-François. Olivier took on the role of Project Lead, and the Linkisum supervision was passed on to Luc Oba. At that moment in time, I was also able to recruit Niels Grateloup, a young Grenoble-INP engineer, for 3 months thanks to a budget transfer from another Incubation project in TIMA which was terminated by its Project Leader, Stéphane Mancini.

The final form of MAST, protected by 2 updates to the original [APP1] filing, is depicted in Figure 73. Maintained as GIT project on UGA's GitLab server, the project is composed of a total of 2473 files, of which 329 C++ Headers and 392 source files for more than 150K lines of code. The build process is handled by CMake, and all modules are doubled by a CTest infrastructure, not shown, with a rough total of 5000 Unit Tests. All code is extensively commented, with a Doxygen-generated HTML Documentation.

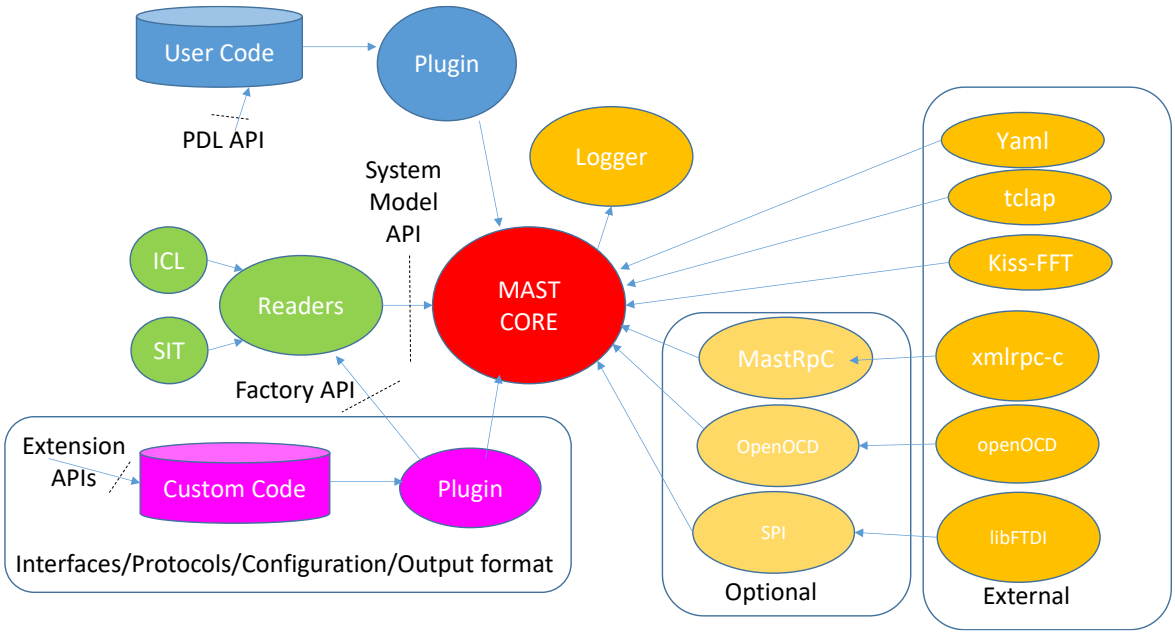


Figure 73 Final MAST Software Architecture

The main part of the software is the MAST Core, developed during the Maturation project, which by itself is responsible for 294 source files and 470 Headers for ~27K lines of code. This contains the Implementation of the Abstraction of Section 4, and is completely self-contained. Libraries that can be used for specific setups are regrouped into an Optional library set, whose inclusion can be controlled through build parameters. External libraries have their own set, and are included as source code to avoid dependencies on the host platform. The core can be extended through specific APIs, and a Factory Design Pattern [DEPA94] is implemented to allow run-time loading of custom extensions compiled as Shared Executables (.so in Linux, .dll in Windows).

The Circuit Model implemented into the Mast Core, shown in Figure 77, is a direct implementation of the Abstraction, and can be built from ICL thanks to a dedicated parser.

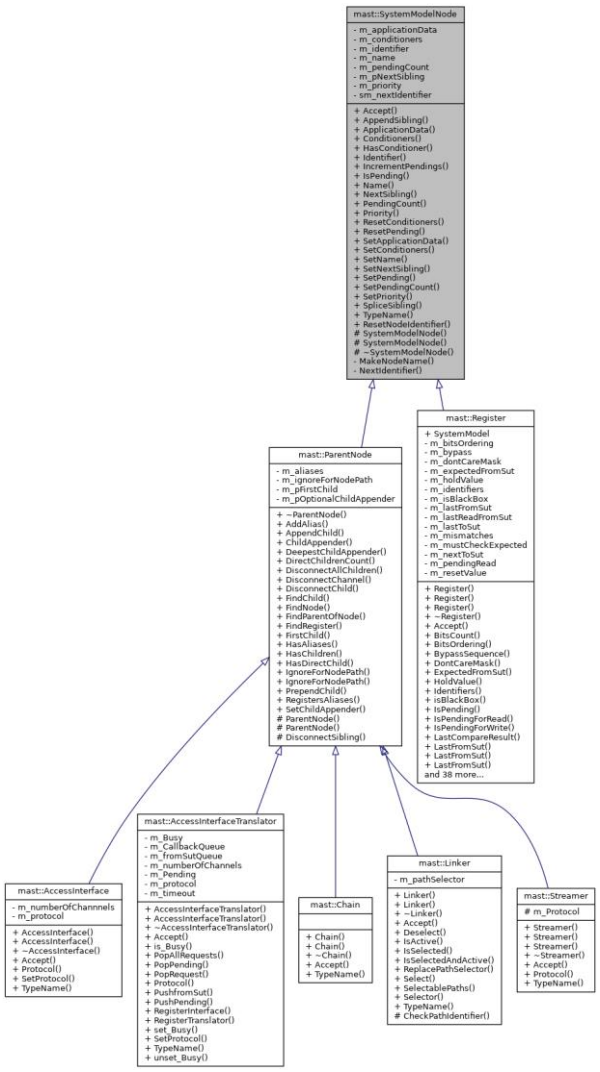


Figure 74 Doxygen-Generated UML Class Diagram for MAST System Model

However, ICL is an extremely detailed and error-prone language and it is not adapted to quick-iteration experimentations. Its parsing and elaboration is quite complex: experimenting new rules can be an extremely time-consuming task. For these reasons, MAST implements also its own DSL, called "Simplified ICL Tree" (SIT): born as a simple textual dump for MAST's System Model, it developed into a fully-fledged DSL able to describe complex hierarchical systems. Without going into syntactical details, we will show some usage example. The role of SIT is not to become "the new 1687 Language", but rather to be the sandbox to test new rules and approaches which should eventually make them way to ICL.

To demonstrate how MAST implements the complete abstraction of Figure 46 and Figure 47, we will provide a step-by-step description of its usage through a benchmark module, depicted in Figure 75. It is a simple 8-bit register called reg_8, over which we want to execute a Test Algorithm

called “Random”. For easier usage, it is wrapped into a chain called “base_example”, which can be used in the SIT syntax but which will not appear in the path of PDL module names.

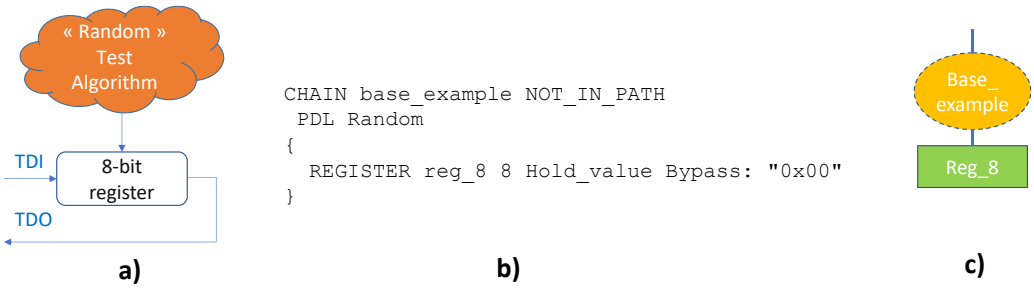


Figure 75 Benchmark Module a), its SIT Description b) and the corresponding abstraction c)

The “Random” algorithm is reproduced in Figure 76 : a set of “loopCount” iWrites to the target register of random values. Thanks to the C++ wrapping, we can freely use standard libraries as in a classical programming setup.

```
void Random ()
  {
   auto seed = chrono::high_resolution_clock::now().time_since_epoch().count();
   auto word_rand = std::bind(std::uniform_int_distribution<int>(0,1<<registerSize),
                              mt19937(seed));

   auto    loopCount    = 10u;
  auto     i    = 0u;
   while (i<loopCount)
   {
    iWrite(reg_8, word_rand());
      iApply();
     i++;
  }
}
```

Figure 76 "Random" Testbench algorithm

The compilation flow for the Testbench is detailed in Figure 77: as in the TISA setup, the test executable contains the Test information, but this time they are Relocation Symbols for PDL operations: the result is completely portable with no dependencies on specialized hardware. Of course, the actual symbols are mangled by GCC [HERY98], but as they are referenced through the C++ Header this is transparent from the user’s point of view.

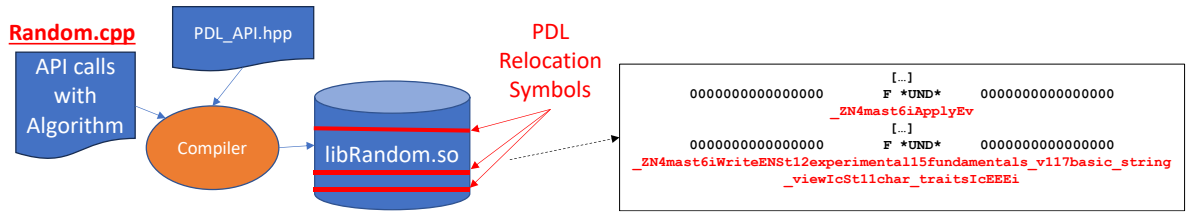


Figure 77 Details of the MAST Compilation flow for the Testbench

Execution, depicted in Figure 78, is a direct implementation of the Abstraction: the test executable is loaded at Runtime, and it is executed against the Circuit Model obtained from the SIT/ICL file to obtain the output patterns. Some implementation details are omitted for clarity’s sake (ex: the

usage of a Factory Pattern to link the C++ PDL function name with the SIT file reference), but the information flow is complete.

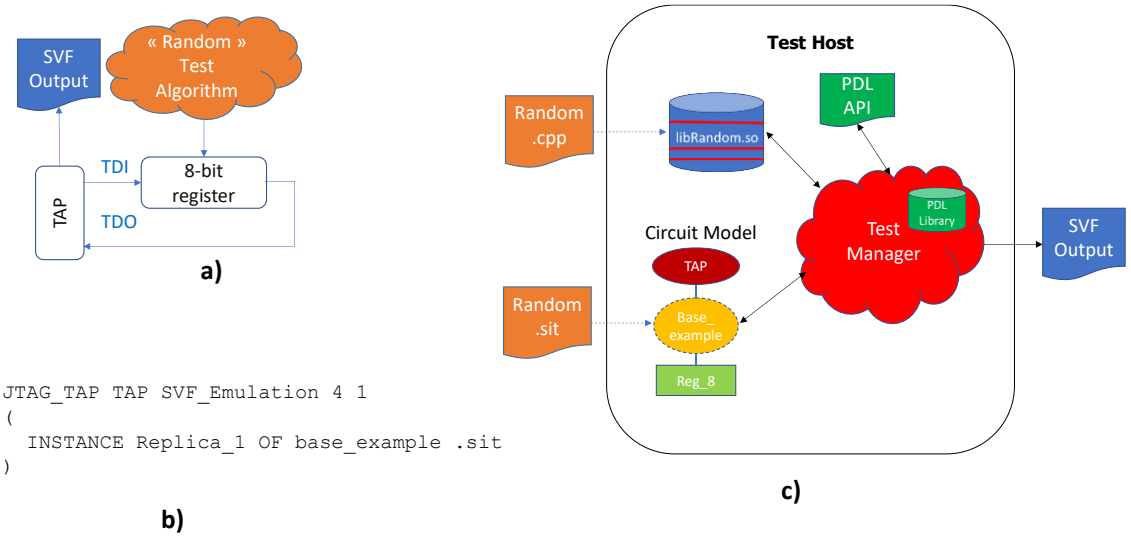


Figure 78 Execution of the Random testbench

Please note how in the SIT file, the base example can be directly instantiated from the TAP description. By acting on this top-level description we can measure MAST's performances for Retargeting, first published in [J.4]. By replicating the INSTANCE in the SIT file, we can benchmark the retargeting times for Horizontal Retargeting: the results are reported in Figure 79 for different "loopCount" values: 10, 100 and 1000.

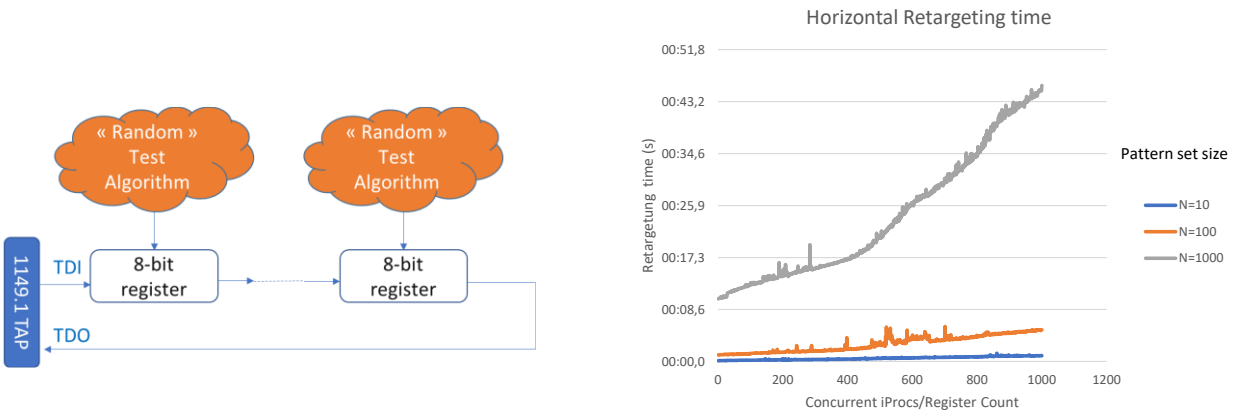


Figure 79 MAST performances for Horizontal Retargeting [J.4]

There are two takeaways: first of all, the retargeting times are extremely small: the most complex example (1000 iWrites replicated 1000) takes less than 50 seconds on a standard laptop. Second, apart from the OS noise (we measure wall-time execution) the complexity is clearly linear, the best possible result for scaling.

To measure Vertical retargeting, we follow the same scheme but we put a SIB before the "base example", which then we replicate "n" times to obtain the results of Figure 80: once more, MAST guarantees linear performances with limited absolute execution times.

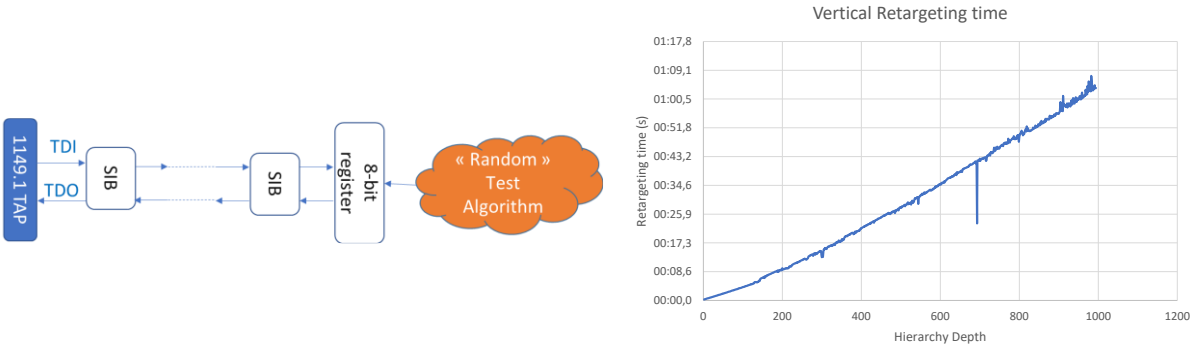


Figure 80 MAST performances for Vertical Retargeting [J.4]

Last but not least, one of the promises of the Abstraction is to efficiently handle both Horizontal and Vertical retargeting, as demonstrated in Figure 81

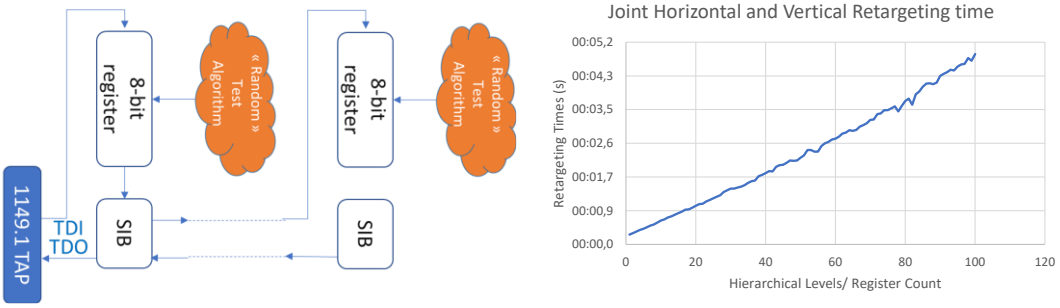


Figure 81 MAST performances for joint Horizontal and Vertical Retargeting [J.4]

In all the testbenches, the size of the computed Pattern Set was extremely close to the theoretical optimum.

Scientifically, the Maturation and Incubation projects are a success: the MAST software is a completely functional implementation of our Abstraction, and can boast performances no other commercial Tool can match, at least on published data. It is also the only complete implementation of PDL-1 interactive behavior.

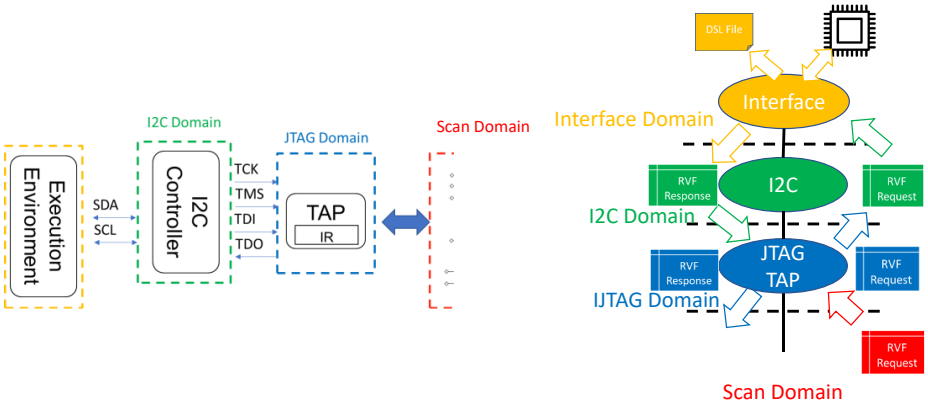
However, the Incubation project was stopped in 2018 in common accord with Linkisum and the Incubation Project Leader, Olivier Bolon, when the financing ran out. The projected Start-Up was not able to secure the initial, key First Customer to finance the final TRL improvement and gain a market share. This was due to several reasons, the most important being that the MAST software was probably still too advanced and "different" for the cautious Testing Market. We contacted several key actors in the Semiconductor, EDA and Instrumentation businesses and received an extremely positive feedback, but we were never able to secure a deal. While being "too advanced" is a flattering formulation in scientific terms, it also means that MAST did not answer to a clearly-identifiable industrial need: from a customer's perspective its market value was therefore too weak to justify an investment. In hindsight, during the Incubation phase we should have probably identified a Target Market Segment and focus development on it, but we lacked this feedback at the time.

5.4 New Perspectives with MAST: 2018→ Present

Even though the Incubation project did not result in a Start-Up launch, the scientific objective of developing a first Implementation of the Abstraction was successful. In this Section, I will resume the main innovations that I was able to build using MAST as a working base.

5.4.1 Interface Independence and P1687.1

The last part of the Abstraction, described in Section 4.5, is the handling of Domain Crossing through RVF Propagation, as depicted in Figure 64, reproduced here for easier reference.



Reproduction of Figure 62 RVF Packet Propagation on the Circuit Model

This information flow is the result two steps of Abstractions. The first was implemented in the Maturation and Incubation version of MAST, and is depicted in Figure 82 : it is a one-step Translation, where "To_SUT" vectors computed in the Scan Domain through retargeting are translated into top-level Operations on the scan interface. Similarly, the return "From_SUT" bitstream is extracted from these.

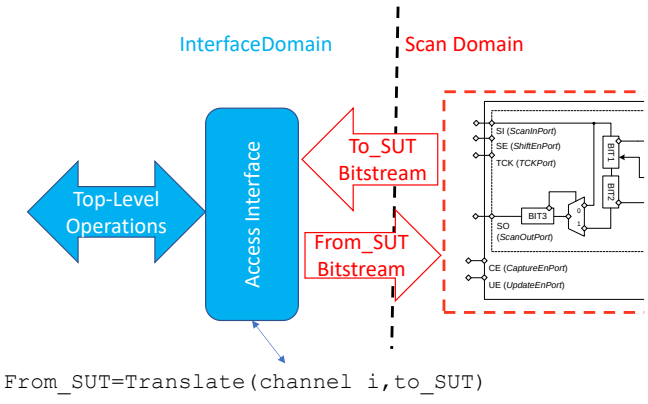


Figure 82 First Domain Crossing Abstraction

The merit of this Abstraction is to provide a single Translation function for each Access Interface. In this version, MAST could define any number of AccessInterface Protocols inside its Core thanks to an Inheritance Strategy. For instance, Figure 83 shows the Doxygen UML Class Diagram for the "SVF_Emulation" protocol used in the benchmarks of [J.4] described in Section 5.3.

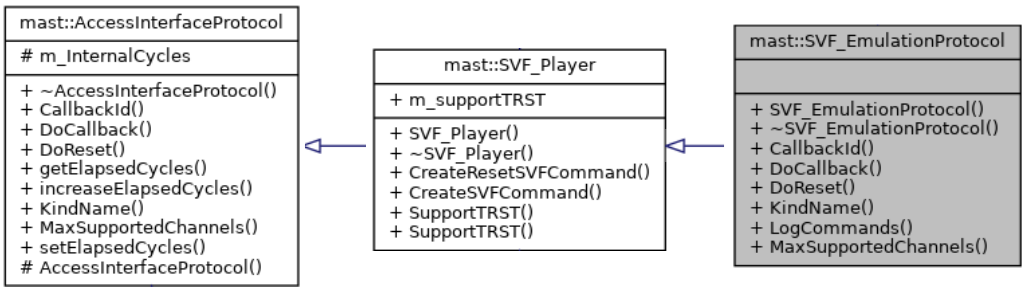


Figure 83 UML Class Diagram for the SVF Emulation Protocol

This solution did allow MAST to support any type of Interface thanks to the Protocol Abstraction, but it had two main weakness:

- It could only support one interface at a time. Translators such as an "I2C to JTAG" would need to be complexly incapsulated into the "Translate" function. This seriously limited flexibility and code reuse;
- To profit from the Inheritance and be included in MAST, a Protocol needed to be compiled as part of the Core. This effectively prevented Third-Party development of Translators

The first point was solved by developing the RVF abstraction: instead of solving everything in one step, each Translate function uses RVF packets as inputs and outputs. The flow of Figure 64 is realized through a series of Blocking Message Queues [TANE15], as detailed in Figure 84. Each Translator is blocked on its "To_SUT_Queue", where RVF Requests are deposited by the Parent translator. Upon reception, it the Translator function processes the RVF packet, and can Push one or more RVF Request(s) to the lower level translator. If a return is needed from the SUT it (being data or status information), it can Pop a RVF_Request from its "From_SUT" queue.

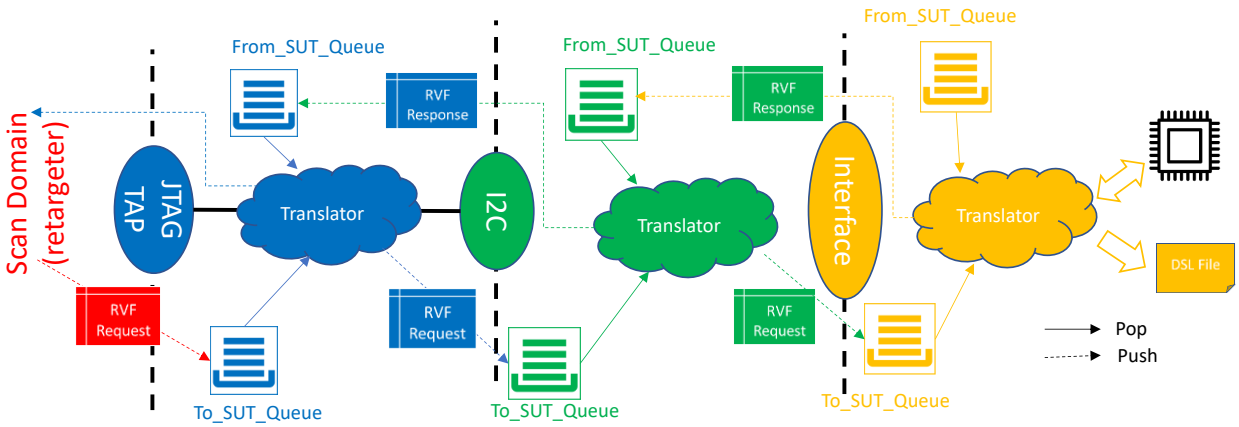


Figure 84 Propagation and synchronization of RVF packets inside MAST

Each RVF packet is therefore handled locally by its Translator. Please note that for this scheme to work the Translator must be able to generate RVF Responses in the format needed by its Parent, as part of an exchange contract. This behavior is usually part of the Documentation or Data Sheet of the Translator, and can therefore be easily implemented.

For instance, the example of Figure 53 presents a Transaction-to-Transaction I2C-to-JTAG translator: each JTAG instruction corresponds to an I2C Write/Read on a given address. In this

case, in the scheme of Figure 84, a single JTAG "Blue" RVF request would result in two "Green" RVF Request, as depicted in Figure 85 : one to Write the data to be transmitted to the SUT and one to retrieve the data coming back from the SUT. This would result in two Green SVF Result: the Green translator would use the first (related to the I2C Write request) to check for errors and the second (related to the I2C Read request) to retrieve the data for its own Blue Response.

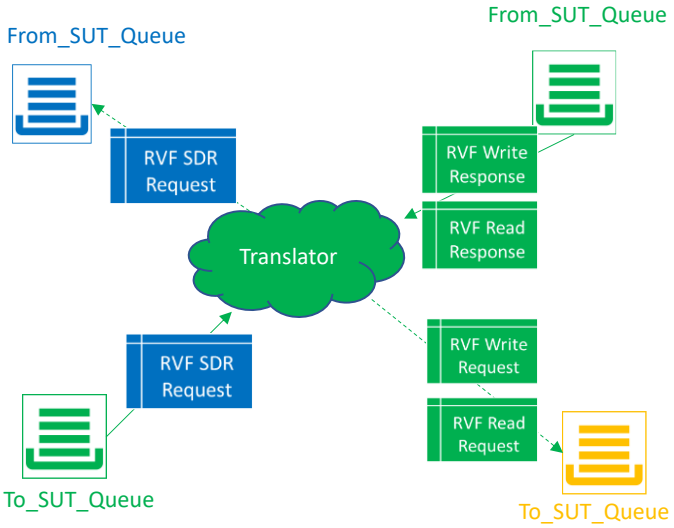


Figure 85 RVF Packet flow for a T-2-T I2C-to-JTAG Translator

However, this solution is not unique. A really common setup is the so-called bit-banging: the I2C controller writes to a 4-bit register, and each bit is used to generate a TAP signal (TCK, TMS, TDI, TDO). The TAP control chronogram is therefore generated through a set of write/reads on this register. Far from optimal, this solution is nevertheless extremely simple to implement in hardware and widely used. This means that a single "Blue" RVF request/response pair will result in a myriad of Green pairs, as in Figure 86.

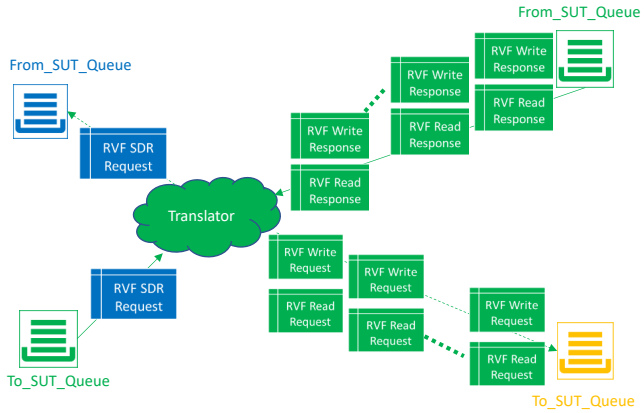


Figure 86 RVF Packet flow for a Bit-Banging I2C-to-JTAG Translator

The key takeaway is that the Abstraction is not impacted: the only modification is in the Translator function itself, while the flow remains the same. This is a major innovation: it is the first (and so far, unique) solution able to provide support not only for arbitrary Interfaces, but also for any number of Translations between them.

The second limitation of the original MAST solution was the need to develop Interface Translators as part of the tool's build flow. As introduced previously, we solved this issue by the systematic implementation of the Factory Design Pattern [DEPA94] in MAST, as detailed in Figure 87.

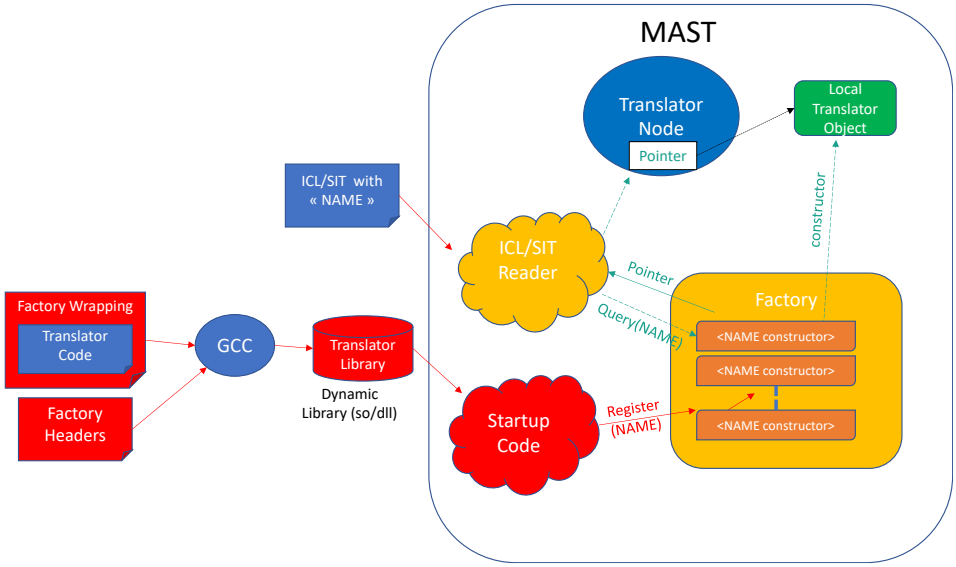


Figure 87 Factory Pattern applied to Translators

The problem of having the Translator code compiled separately is that its function names are mangled by the compiler [HERY98]: it is impossible to reference to them in a consistent way. It is, for instance, impossible to call a Constructor for a class compiled in an external module. The idea of a Factory Pattern is to avoid this reference: each Translator can register its Constructor in a Factory, where the internal pointer is mapped to a string (ex: "JTAG"). The External code is compiled against a Factory Header which provides the required API calls. Upon Startup, MAST loads the Shared Libraries and executes the Registrations through this same API, populating the Factory. Each time we need to use the constructor (for instance, during the System Model construction by the Parser), we query the Factory for it and use it to generate a Local object. The implementation of this process can be complex, but it is done only once inside the Tool. The result is quite powerful: any User can develop his/her own Translators, and have MAST automatically use it. The Factory Pattern is extensively used in MAST: apart from Translators, it is also used for Linker's Path Selectors, for loading the PDL functions referenced in the SIT/ICL and for the Topology configuration algorithms.

This solution has been developed outside of the Maturation and Incubation projects on a separate Branch, and is the subject of the third version of [APP1]. We also proposed it to the IEEE P1687.1 and P2654 Working Group, where it had a big impact. The principle of Primitive Domain Crossing, RVF Packets and their Propagation flow as described in Section 4 and [J.4] has been fully embraced and will be undoubtedly part of the Standard. However, the discussions are still ongoing and no definitive decision has been made on the final implementation inside of the Standard, which could be significantly different from solution presented in this Section.

5.4.2 Interactive IJTAG

As explained in Section 2, the usual setup for JTAG and its derivates is offline generation vs test application. IEEE 1687 has been the first to introduce the idea of functional behavior thanks PDL and the possibility of using TCL as a language superset, but the main Use Mode never really changed. The possibility of reading data back from the System Under Test is present thanks to the iGetReadData Instruction, but not really developed. For most Users, the principle is to be able to read configuration or identification data from a CUT (like, for instance, a serial number) and adapt the test execution accordingly. However, the possibility of having a completely interactive setup, where the whole test algorithm depends on the data received from the CUT was never considered a possibility, mostly because of the already mentioned technical limitations of the legacy Automated Test Flow. The Execution model was not directly mandated into the Standard document, but is it implicitly referred to what is usually called "ATE Bring-Up" or "ATE Debug", depicted in Figure 88 : the TCL script containing the PDL-1 code is executed by the shell of an EDA tool, which is able to communicate to an ATE in order to push to the SUT the vectors computed from the iApply operations, and provide to the iGetReadData commands the data captured from it.

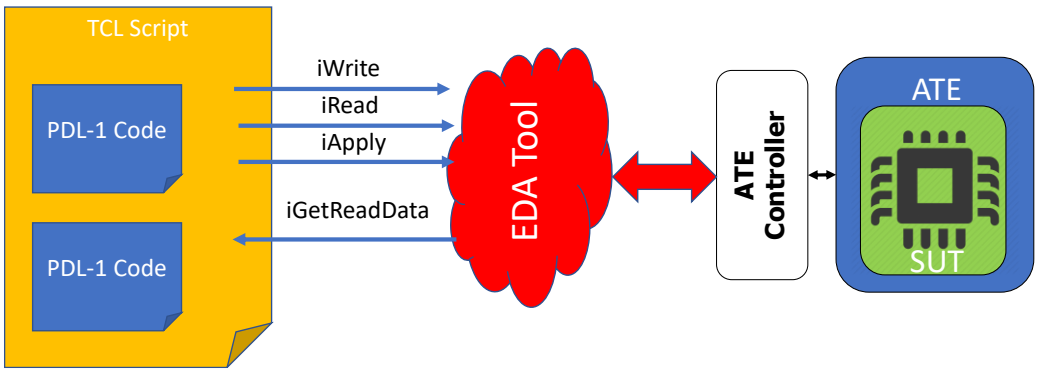


Figure 88 Implicit PDL-1 Execution Model

This is a complex setup that even if not impossible (some industrial vendors propose similar solutions [SEDA-IN]) is quite cumbersome and resource-hungry. The access to an EDA tool might not always possible (for instance, if testing is done by a third-party company) or its execution might be impossible by the target platform (for instance, when performing online testing through an embedded controller).

Thanks to the new Automated Test Flow we presented and its implementation through MAST, we were able to break the Generation/Execution barrier to obtain the completely interactive setup of Figure 89, which is a direct implementation of the new Abstraction. As the Text Executable still contains the Algorithms, interactive execution and debug can be done using traditional Computer Science tools, such as for instance GDB.

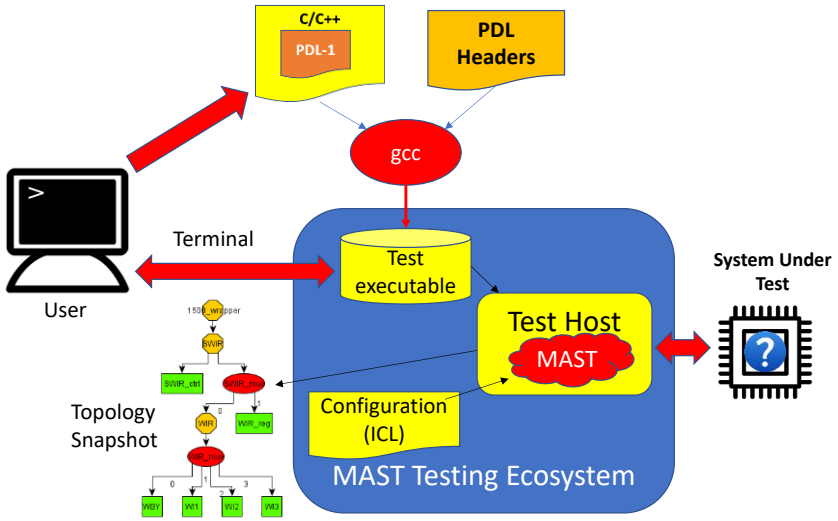


Figure 89 Fully Interactive Flow with MAST

To demonstrate these capabilities, we developed an FPGA-based demo representative of an interactive signal processing setup and visually compelling: music volume bars. The setup, depicted in Figure 90, is functionally quite simple: a 1687 Instrument samples music coming from an audio stereo source, and the digital samples are collected by a PDL-1 function which performs an FFT to extract the Volume level of the right and left channels, which are then sent to a visualization 1687 Instrument.

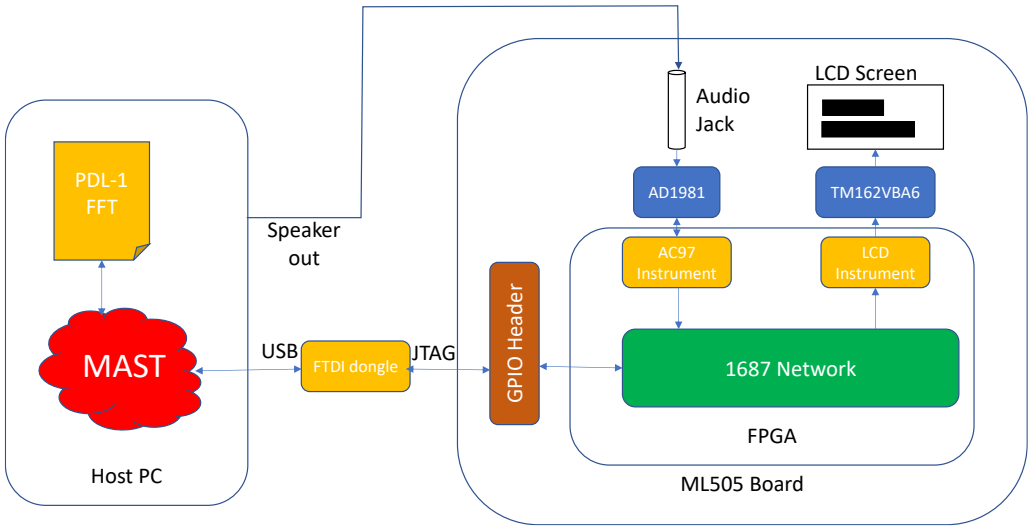
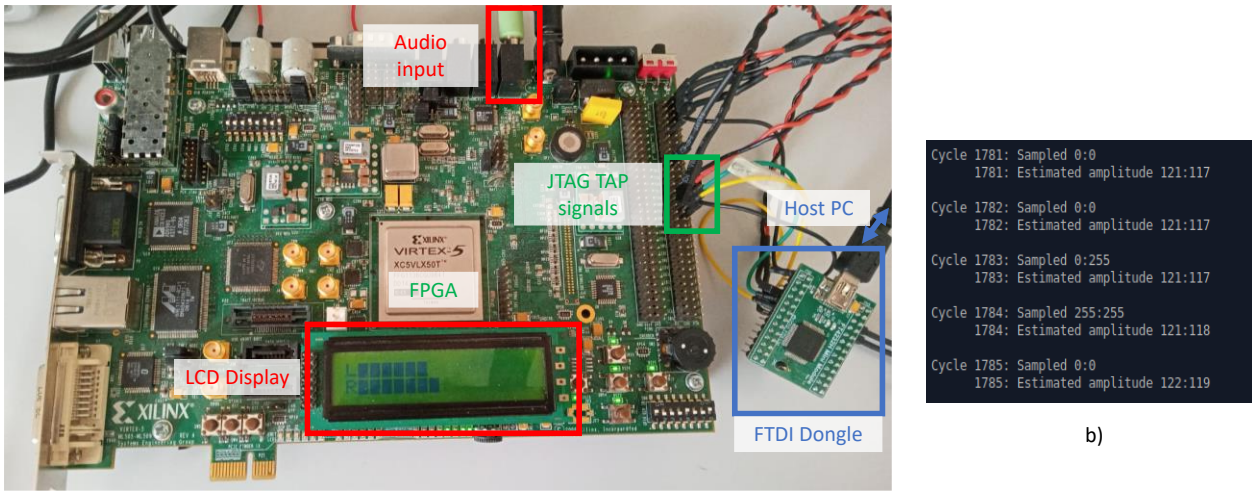


Figure 90 Top-Level Specification of the Interactive Demo

The first version has been developed by Niels Grateloup as part of the MAST Maturation project, using [OPENOCD] as the Interface library to communicate with the [FTDI] dongle, and custom VHDL IPs to command the on-board components of a Xilinx ML505 card. The final setup is shown in Figure 91-a). During the execution, the Left and Right volume bars move depending on the music's volume on each channel, while the actual audio samples are displayed on the user prompt for debugging, as shown in Figure 91-b)



a) b)

Figure 91 Experimental setup for the Interactive Demo.

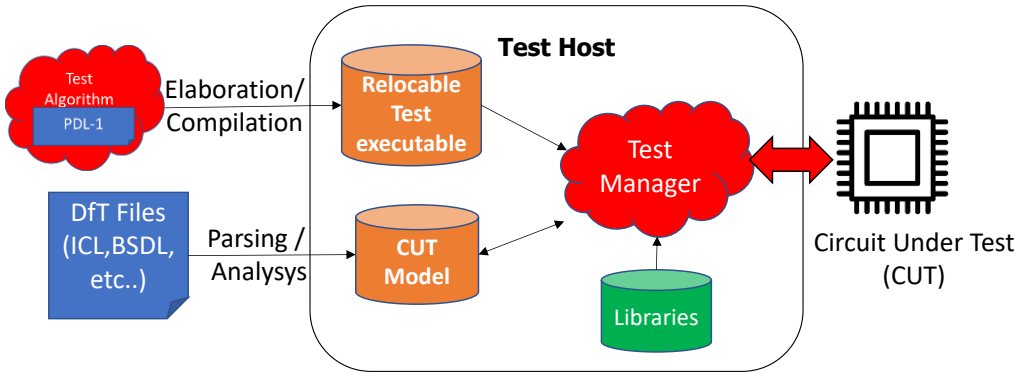
We later simplified the approach by replacing this complex Interface developed for an earlier version of MAST with a streamlined RVF-based Translator which directly communicates with the FTDI chip. This library was developed by Clement Tardy, a Phelma 2nd Year student as part of his "Assistant Engineer" internship.

Another innovative application of Interactive IJTAG has been published in [9] and [W2] : the complexity of 1687 Networks make their integrity Validation and Debug extremely difficult. This was one of the research subjects of my host at Politecnico di Torino during my Visiting period, most notably for Giovanni Squillero [DAM19] and Riccardo Cantoro[CAN18]. During my stay, we demonstrated how Interactive execution of IJTAG testbenches could be used both for boosting post-silicon debug of scan chain integrity [W2] and to verify the correctness of the ICL description of the 1687 network [9].

Such a solution would have been close to impossible using the Legacy Automated Test Flow, and it is still the only example of a real interactive 1687 flow.

5.4.3 Unified Test Middleware: "Test Operating System"

While in this document we focused on the limitations of the legacy Generation/Application duality, this solution also has some points that justified its success. Its greatest asset is undoubtedly its simplicity: by making no assumption on the Execution backend, a Pattern Set can be played on basically any type of platform. The price to pay for this portability is the lack of information conveyed in the format itself, as pointed out in Section 3. In our proposed new Automated Flow, the Pattern Player is replaced by the Test Manager, as depicted in Figure 46, reproduced here for easier reference.



Reproduction of Figure 44 New Automated Test Flow

The position and role of the Test Manager is quite similar to that of an Operating System: it provides an Abstraction of the Hardware (in this case, the CUT), which the Software can access and exploit thanks to a set of standardized System Calls (in this case, the PDL Instructions). The Translators take the role of Hardware Abstraction Level, responsible for interacting the actual hardware. This means that thanks to our Abstraction and MAST as the "Test Operating System", we are able to provide a unified solution to port and execute complex Test Algorithms over any platform, as for instance is depicted in Figure 92 : the Test Algorithm can be developed from the Specification, and executed on MAST using an "Emulation Translator", which generates debug logs and Topology Snapshots to verify the 1687 Network, while the Software debugging can be done using standard tools like GDB coupled with a Graphical Debugger.

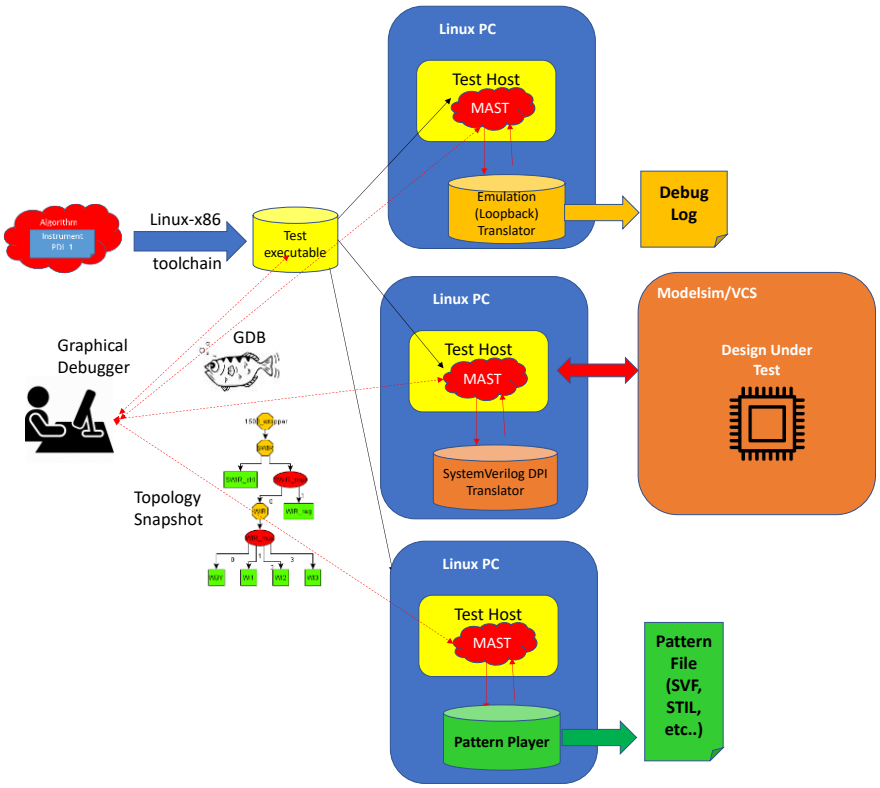


Figure 92 MAST Portability through the Design Cycle

By simply changing the Translator the setup can be ported to a new Execution setup, in the same way an OS can be ported to a new Processor by porting its HAL. In the middle of Figure 92, we

show in the middle co-simulation with an HDL thanks to Translator that can interface with the Design Under Test using the SystemVerilog DPI libraries. These two steps can be used both at IP and System Level, following the development of the DfT infrastructure to all its steps. After Sign-Off, a "Pattern Player" can be used to generate traditional Pattern Files (SVF, STIL, etc..) for backward compatibility with the legacy Automated Flow.

The big novelty of the new Flow is that this same setup can follow the Design to its prototyping phase, as shown in Figure 93. Thanks to an FTDI Translator MAST can directly communicate with a Circuit Under Test in a bench-top environment. Similarly, we developed a Proof-of-Concept with MAST being executed on an Advantest 93k and communicating with the SmarTest suite controlling the ATE.

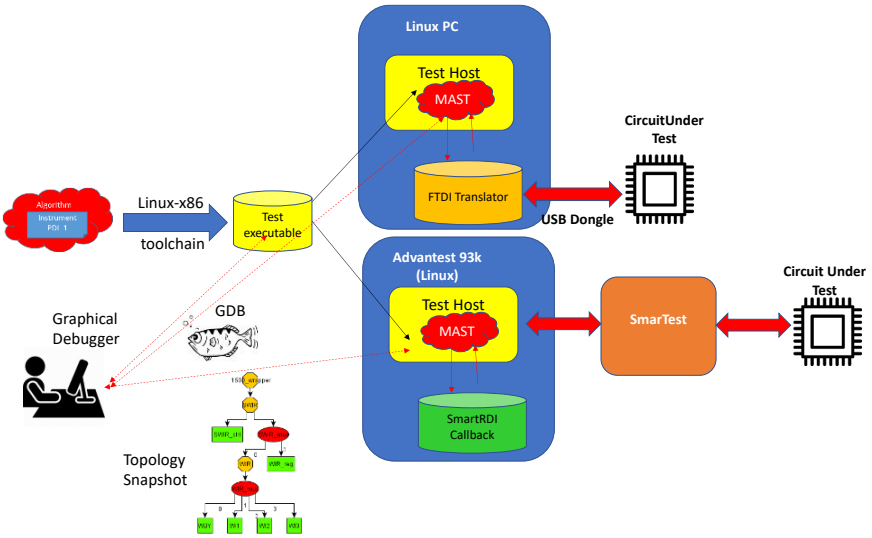


Figure 93 Circuit Prototyping and ATE Bring-Up with MAST

Last but not least, thanks to the portability constraints we imposed in MAST specifications, we were able to validate its portability on Embedded Processors, as shown in Figure 94, for both Big- and Little-endian targets. We ran our experiments on Embedded Linux for both targets, but theoretically MAST should also be able to run on BareMetal, as long as multithreading is supported.

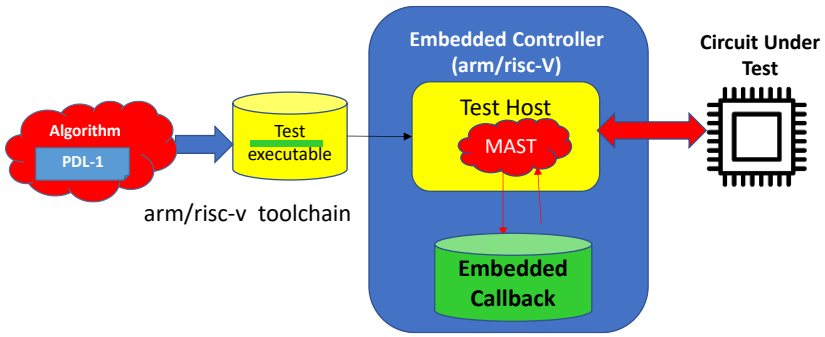


Figure 94 Embedded Test Controller with MAST

All the examples in this Section have been fully implemented and verified on real targets. In particular, the ATE Proof-of-Concept was developed and benchmarked thanks to the

collaboration of the [FMNT] in the person of Beatrice Pradavelli, who gave us remote access to the machine located in Montpellier, and the technical support of Advantest Grenoble in the persons of Brocheton Herve and Bruno Bourgeois.

This portability spanning the whole Design and Development cycle is absolutely unique for MAST, and will be the base of several research directions in the future, some of which will be detailed in the next Section.

5.4.4 Security as part of the Test Flow

In Section 2.4 we highlighted how several setups have been proposed to enhance the security of Scan Testing, but how regardless of their technical merit they remain had-hoc solution outside of any Flow, needing custom adaptations. However, the new Abstraction and its implementation using MAST allowed us to solve this issue, as published in [J.2][1][2][7][6].

As previously explained, the first family of Scan Security solutions is Scan Authentication, i.e. limiting the access to specific portions of the Scan Path by submitting the Opening/Closing of one or more SIBs to some credentials. The SSAK secure setup can be easily expressed in the new Abstraction by defining two Path Selectors which implement the Authentication algorithm, as shown in Figure 95

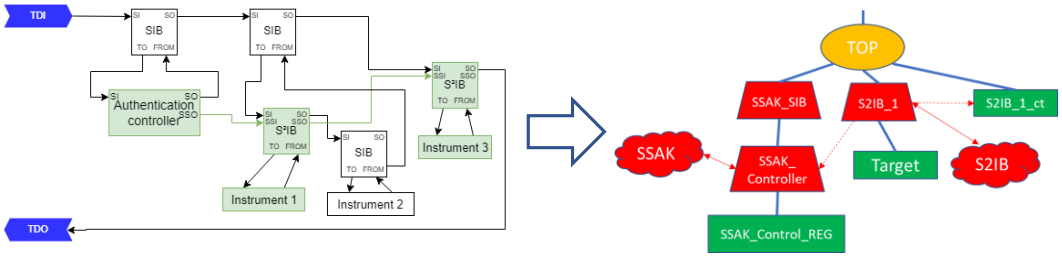


Figure 95 SSAK-Secured Scan Chain and its Abstraction

The first Path Selector is the "Secure SIB"(S2IB), on the right-hand side of the Circuit Model. It is a simple modification of a traditional SIB: instead of just changing the value of the controlling register (SSIB_1_ct, not shown in the scan chain diagram), it also asks for the SSAK-Controller to be "Selected". This second Path Selector "SSAK" implements the Authentication proper: its "Select" method implements the challenge through a set of 1687 operations:

- First, it reads the challenge from the" SSAK_Control_Reg";
- It computes the response based in its credentials
- It writes the computed response back in" SSAK_Control_Reg";
- It reads from "SSAK_Control_Reg" the status: authentication successful or not

Only if the last step is successful the Linker is considered as Open, otherwise an error is generated. This is completely automated, with no modification needed to MAST's core and it is transparent from the User's point of view, who just needs to provide his/her credentials as part of the 1687 Description: it is the red string in Figure 96.

```
SIB SSAK_SIB POST HIGH
  (
   LINKER SSAK_Controller SSAK SSAK_CONTROL_REG   1 "0x72c4358f5a8a07af3d0f7d560a872a2b
13"
     (
      REGISTER SSAK_CONTROL_REG 128 Bypass: "0x00000000000000000000000000000000"     )
     )
 REGISTER S2IB_1_ctrl 1 Hold_value Bypass: "0b0"
 LINKER S2IB_1 S2IB SSAK_Controller,S2IB_1_ctrl  1 "1"
   (   REGISTER Target 12 Bypass: "0xABC"    )
```

Figure 96 SIT Description of the SSAK example

This is a perfect case for justifying the instruction of SIT: describing such a setup in ICL would have been quite complicated, and would have required adding several custom rules, ending up in any case with a non-compliant solution.

Scan Encryption was actually extremely simple: the role of the Stream Cypher is to Transform Plaintext to Cyphertext and the other way around, i.e. to handle the Domain Crossing between Secure and Non-Secure areas. It can therefore be directly expressed as an Interface Translator, as shown in Figure 97

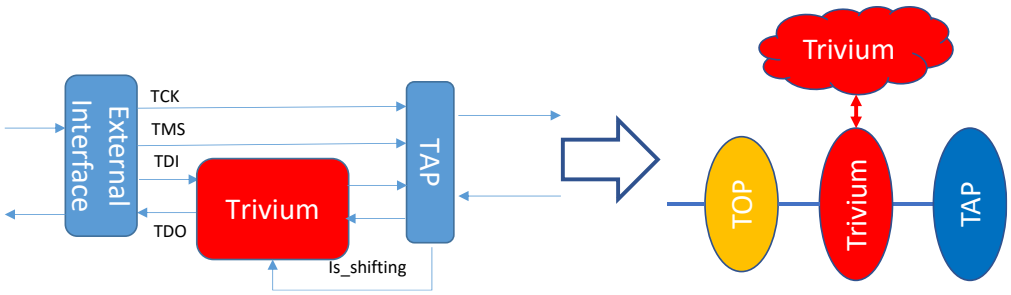


Figure 97 Trivium Stream Cypher and its Abstraction

The SIT description is extremely simple too:

```
TRANSLATOR top Simulation
 (
 TRANSLATOR Secure Trivium "0F62B5085BAE0154A7FA 288FF65DC42B92F960C7"
                                                  (
 …
```

Figure 98 SIT Description of the Trivium Example

The great novelty of our Abstraction is that neither Authentication nor Encryption are a special case, but they can be freely used inside a Design without neither modification to MAST nor User intervention. To demonstrate this "plug-and-play" capabilities, we devised the experimental setup of Figure 99 : we execute MAST against an RTL Circuit Under Test, simulated in Modelsim. The connection between MAST and the Simulator is assured through a Translator, as explained in Section 5.4.3. The RTL implementations of the SSAK Controller and the Trivium Stream Cypher come from the PhD work of Vincent Reynaud (TIMA) and Emanuele Valea (LIRMM) respectively, completely reused without any modification.

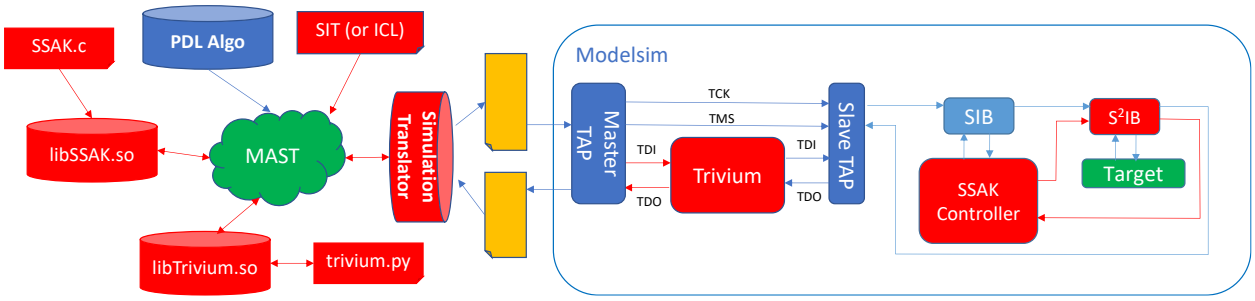


Figure 99 Experimental Setup for Secure Scan Chain Access

Thanks to MAST's interactive capabilities, we can perform both Stream Encryption and Authentication on the fly, simply providing the Translation Libraries. The Testbench is configurable: the assembly of SSAK and Trivium can be changed at will, and MAST can support any setup just by providing it the right SIT description. From the User's point of view everything is transparent: the PDL Algorithm is executed in the same way against its Target register, regardless of the security features in the middle.

This experiment also proves MAST capability of leveraging existing libraries: in the over-mentioned PhD works, the software parts were done using C for SSAK and Python for Trivium. We were able to directly reuse the original code by simply using standard API interfaces to C++, such as for instance [PYBIND].

This experimental setup is the first, and to this day unique, example of Security solutions fully integrated in the Test Flow in a true plug-and-play fashion.

5.4.5 Analog Interfaces

As stated in Section 2, several efforts are being deployed to extend 1687 to support the testing of Analog and Mixed Signal circuits. While these systems are not part of our Abstraction and are therefore not fully supported, features such as Interface portability and Interactive execution can be useful in that context. To prove this, we took the example of Figure 100: it is a Power Management IC (PMIC), a Small-D/Big-A design from Renesas. It is a system proving both an Analog Test Bus and a Digital Core, and it can be controlled through either an I2C controller (SDL/SCL), a GPIO Pin or an ATE interface (ATB).

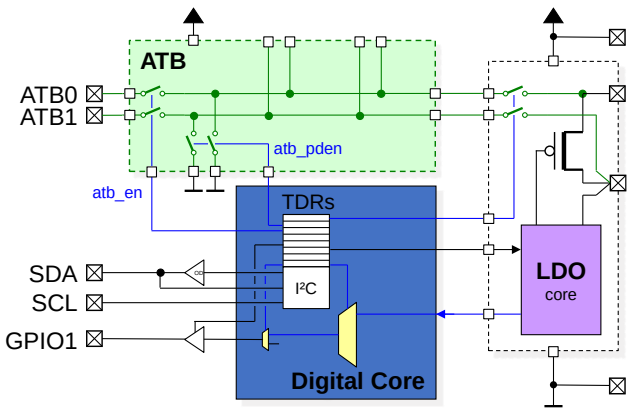
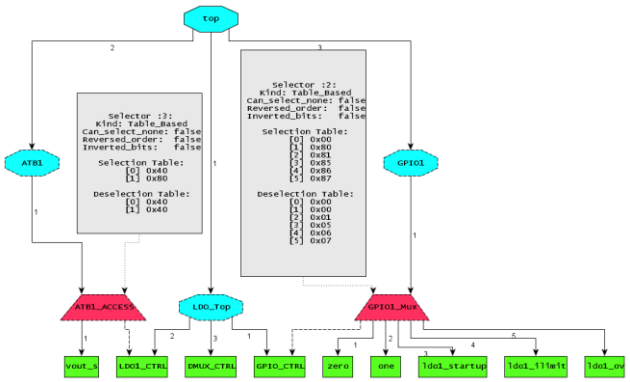


Figure 100 Example of a P1687.2 system, from [VSTA20]

In the paper, the authors showed how it was possible to define PDL commands inside the PMIC, and retarget them (manually) to obtain commands at the Interface levels. It is one of the first examples of the operation of such a Mixed Signal system being described in an EDA-friendly way thanks to DSLs, even though the software part was still unavailable.

We decided to apply our Abstraction to the system, obtaining the setup of Figure 101 : the three Interfaces can be modeled through their Protocol, while the internal Registers and Selection muxes can be directly described.




```
TRANSLATOR top MPSG
  PDL LDO_PDL
 (
  ACCESS_INTERFACE LDO_Top NOT_IN_PATH I2C "0x00, 0x10, 0x20, 0xF2"
  (
    REGISTER GPIO_CTRL 8 Bypass: "0x00"
    REGISTER LDO1_CTRL 8 Bypass: "0x00"
    REGISTER DMUX_CTRL 8 Bypass: "0x00"
  )
  ACCESS_INTERFACE ATB1  NOT_IN_PATH AnalogPort "ATB0"
  (
   LINKER ATB1_ACCESS Table_Based LDO1_CTRL 1 "0b01000000, 0b10000000,
0b01000000, 0b01000000"
       (
       REGISTER vout_s 8 Bypass: "0x00"
       )
  )
  ACCESS_INTERFACE GPIO1  NOT_IN_PATH DataPort "GPIO1"
  (
   LINKER GPIO1_Mux Table_Based GPIO_CTRL 5 "0b00000000, 0b10000000,
0b10000001, 0b10000101, 0b10000110, 0b10000111,              0b00000000,
0b00000000, 0b00000001, 0b00000101, 0b00000110, 0b00000111 "
       (
       REGISTER zero 1 Bypass: "0b0"
       REGISTER one  1 Bypass: "0b1"
       REGISTER ldo1_startup  1 Bypass: "0b0"
       REGISTER ldo1_ilimit  1 Bypass: "0b0"
       REGISTER ldo1_ov  1 Bypass: "0b0"
       )
  )

 )
```

Figure 101 PMIC Abstraction and its SIT representation

As ICL lacks several elements to describe such a setup, as for instance bidirectional signals/ports, we described the System in SIT, with some minor liberties (ex: depicting “ports” as “registers”). The aim was not to “solve” the P1687.2 problem, but rather demonstrate our capability to tackle it. This was a success: MAST was able to perfectly replicate the manually-computed retargeted patterns, demonstrating the feasibility of exploiting its features also in this domain.

6 Short-to-Mid Term Perspectives

In this document, we presented the results of a research effort that spanned over more than 15 years. From the beginning, the aim was to identify the Directions with the biggest novelty and impact potential, and Develop the Abstractions and Solutions necessary to enable them. For these reasons, the Short-to-Mid Terms perspectives are focused on the usage of our Abstraction and the MAST tool as the enablers to explore new research directions, rather than on direct evolutions.

6.1 Silicon Lifetime Management

As mentioned in the State of the Art and in the summary of Kalpana Senthamarai Kannan's PhD, one of the big weaknesses of the most recent technological node is variability: one the one hand, physical parameters might have big differences at fabrication time, making calibration and testing extremely difficult, as highlighted in several places in this manuscript. However, the growing impact of aging is also being recognized as a key problem: systems will age differently depending on unpredictable parameters such as usage profiles, working environment, etc. This problem has left the pure theoretical speculations and is impacting the industry. The main EDA providers put their solutions at the forefront of their offers: for Synopsys the "Silicon Lifecycle Management" [SLM23], for Siemens EDA the "Tessent Silicon Lifecycle Solutions" [TSLS23]. Both solutions are focused on a "fleet management" approach, where data is collected in all stages of the life of a system, from production to deployment. A particular emphasis is put on the collection of lifetime data from embedded sensor and their centralization into cloud-based analytics.

However, all solutions are still centered in the Generation/Application duality. Embedded controllers can do little more than collect the data from the sensor, but cannot act on them. Moreover, embedded controllers have difficulty in handling DfT architectures such a 1687's dynamic topologies. Most of time, access to embedded instrument happens through pre-computed static sequences or through ad-hoc connections, with little or no hardware/software reuse from the Testing phase.

In this space, our Abstraction and MAST have a great role to play by pushing intelligence to the edge, i.e. to the systems themselves. For instance, a Machine Learning setup as the one developed during Kalpana'a PhD work is lightweight enough to be embedded. Thanks to MAST, we could use aging monitors to fine-tune the Aging model, and directly act on the DVFS commands with no external intervention. This was, in fact, the original target of the PhD. In coming years, we plan on pursuing in this direction by looking for collaborations in the field of Embedded Instrumentations, with a particular emphasis on RISC-V based systems and on the reuse of DfT infrastructure thanks to MAST's dynamic retargeting capabilities.

6.2 Security

In this document, we demonstrated how Security can become part of the standardized Test Flow, by integrating Scan Authentication and Scan Encryption into our new Abstraction. However, this is just the tip of the iceberg: Security must find its way in all steps of the Flow, from Factory Testing to Lifetime management. This can be done only by developing dedicated solutions that

can leverage the new integration, which are difficult to imagine at this moment in time. A possible direction could be, for instance to integrate Secret Key management into the Abstraction by modelling constructs like Physically Unclonable Functions, or to explore new partitioning of Cryptographic algorithms between the Hardware and Software parts of 1687-inspired solutions. We already started by proposing some extensions to the concept of Scan Encryption [J.2]. Last but not least, there are completely different solutions like Logic Locking that could be explored, or different threats models like Hardware Trojans or Side Channels. TIMA is well versed in such topics thanks to the work of Paolo Maistri, Regis Leveugle or Giorgio Di Natale, which we can leverage and expand.

6.3 Scaling up: FPGAs

Field Programmable Gate arrays have become so useful and commonly used that we tend to see them only as the valuable tool for prototyping they are, but rarely as a target in themselves. In fact, an FPGA is not so different from a 1687-based DfT target: as shown in Figure 102, they are modular systems, with a high number of resources (Slice/Instruments) connected through configurable routing (Switch Matrix/ScanMux), the whole accessed through a JTAG connection.

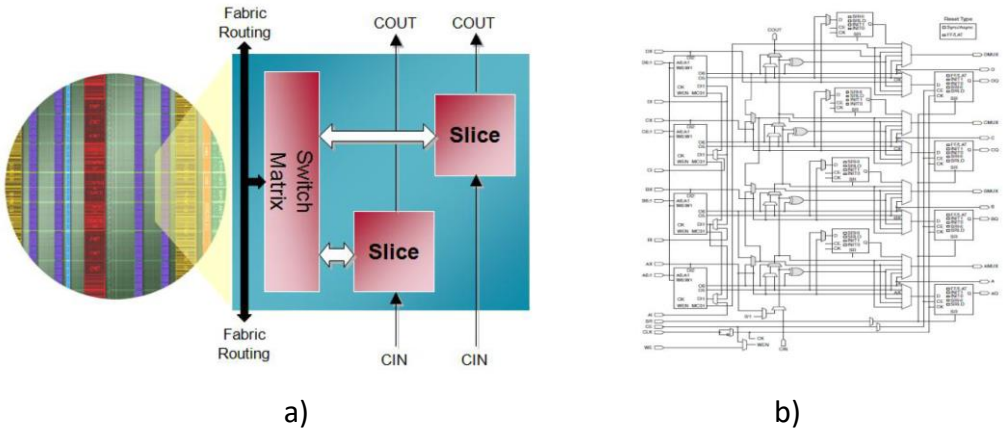


Figure 102 Xilinx/AMD 7-Series High-Level architecture a) and Slice details b)

The Bitstream that is used to configure an FPGA is exactly the same as a Test Vector: a bit-by-bit assignment of internal resources, without any knowledge of the role of each bit; computed by an external EDA Tool (in Xilinx's flow, it is Vivado). This means that even if recent FPGA provide sophisticated dynamic reconfiguration capabilities, everything needs to be computed offline and bitstreams are simply pushed to the FGPA.

It is possible to image representing an FPGA fabric as a 1687 network described in ICL, with PDL routines used to deliver configuration bits. The routing information would be distributed through both ICL and PDL, so that the retargeting result would be the configuration bitstream. In such a setup, an application of our Abstraction and MAST would allow to convey routing information to the final bitstream, allowing true dynamic reconfiguration depending on both the user requirements and the FPGA status (ex: rerouting to avoid faulty locations).

However, such a research direction would demand a lot of inside information on both the FPGA architectures and the Routing algorithms. This could be done either through Open-Source solutions or looking for partnerships with FPGA providers.

6.4 Mixed Signal Testing

In Section 5.4.5, we demonstrated how both our Abstraction and MAST can be useful in the domain of Mixed-Signal Testing. However, we did not go further than a feasibility proof: as it stands, it is neither directly usable nor scientifically relevant. However, the subject is extremely interesting: the Abstraction, User Needs and Best Practices of Digital and Analog Testing are extremely different, and most often than not contradictory. Where the former is based on a set of quantitative models and abstractions, the latter rather relies on a deep knowledge and qualitative "know-how". A real solution to the problem must start by a deep analysis of the two worlds, identifying not only the common points but also the biggest differences and critical points. Only then it will be possible to understand the problem and propose new automation strategies that will be really adapted to the field and be able to provide both added value and scientific innovation.

It extremely difficult to find someone having the right mix of competences to tackle this issue. For this reason, I started collaborating with Emmanuel Simeu, Professor at the Université Grenoble Alpes and member of TIMA, expert in Analog and Mixed signal testing. With our complementary competences and skill sets we jointly advise the PhD of Jules Quentin Kouamo, started in November 2023 and focused on this topic.

7 Conclusions and Long-Term Directions

"If a tree were to fall on an island where there were no human beings, would there be any sound?"[1]. It is a classical philosophical question, whose typical (but far from unique) answer is: Does a sound really matters if no one is there to appreciate it? The same question can be applied to our field of interest: if no one knows about a new exciting technology, will it really make a difference?

On the one hand, Academic Researchers have the privilege of looking at the "bigger picture", looking for new ways of solving open problems. But looking too far, one might lose sight of what happens close to home: this sort of "research presbyopia" can prevent excellent results to be truly exploited, simply because they solve issues that are too specific or too far-fetched.

On the other hand, Industrial Researchers have the privilege of working on real-world problems that can solve real needs and have wide applications. But while getting caught in the pressure of obtaining fast and efficient solutions for the problem at hand one can easily lose sight of the bigger picture. The risk of this "research myopia" is to get stuck in an endless loop of fixes and patches, where the accumulation of small solutions does not generate any real value because it does not have any real direction.

The "sweet spot" lays in the middle: Industry can be an invaluable source of information and propose exciting challenges, but only if Research can keep looking far ahead at the bigger picture. This balance is not easy to find and maintain, but once achieved it can start a virtuous cycle and produce unique results.

In this manuscript, we showed how the Testing flows is getting richer and more complex. Designers need to take these aspects in account as early as possible, all the while requiring new and sophisticated features. The now mainstream "Design for Test" is often declined in "Design for X", with X ranging from Manufacturing to Reliability and pretty much everything in between depending not only on the Designer's specifications but also on the final User's needs and constraints. These domains have traditionally been hardware driven: companies would implement their own devices and architectures, and then use in-house tool or custom software fixes to insert them into the final design. This has now changed: solutions like IEEE 1687 have a rather small hardware component and are mostly software-based, requiring dedicated Software suites of important size. In the meantime, the pressure for lowering cost still kept increasing: few companies, even the biggest ones, can still afford to support in-house development. Moreover, the shift of DfX towards the early phases of Design puts a serious stress also on both custom Tool

[1] The Chautauquan, June 1883, Volume 3, Issue 9, p. 543

and commercial EDA solutions, as they need to remain compatible downstream in the heterogenous Design and Manufacturing flow.

From this melting pot of diverse and somewhat contradictive requirements, two clear tendencies have been emerging:

- The shift from in-house software, too expensive and difficult to maintain, to commercial EDA tools;
- The development of Standards to maximize reuse and assure inter-vendor portability.

Unfortunately, behind highly-flying commercial announcements, EDA companies are actually playing the safe card and only proposing incremental improvements in legacy flows and are not really embracing this "paradigm shift" [REA12]. The only true way forward is through a deep evolution of the whole Test Flow and its relationships with the Design and Verification ones. Such changes, as shown in Section 3, 4 and 5, are the perfect example of the virtuous research cycle we try to achieve: a deep Analysis of a real problem, followed by a rigorous Abstraction and an efficient Implementation. This last step is pivotal: only by having a working platform we are able to both demonstrate the benefits of the new abstractions and leverage them to tackle new problems. The actors of the world of Testing are extremely conservative: new technologies are accepted and deployed only if they provide significant and concrete advantages. The results presented in Section 5 do demonstrate the full portability of Functional Routines from Simulation to ATE and in-field Embedded Testing, but their real value is in their usage. In fact, these approaches and tools are powerful "enablers": by leveraging them, we can tackle "impossible" problems that traditional tools cannot handle because of their intrinsic limitations. For instance, in Section 5.4.4 we showed how the new Test Flow is able to efficiently include Security into the standard flow, and in Section 5.4.5 how Analog Test can be part of it too. Both these results have been deemed "impossible" for a long time.

Future research will build upon the basis we have developed and will follow mainly four strategical axes: **Automation, Integration, Extension and Proximity**

Automation is key: the results presented in Section 5 demanded a significant effort in terms of analysis, abstraction and development, in a period spanning more than a decade. It was essential to develop the new Test Flow, and to demonstrate it is both effective and useful. However, to obtain real traction it is not feasible to ask the same type of effort for new users, being them academic or industrial. Automation is therefore needed to guarantee ease of use. This is not simply a short-term development but rather a whole rethinking of the different steps and their usage: information must be collected, processed and results assembled in a completely automated way. A new abstraction must be analyzed, specified and implemented. Only in this way it will be able to sustain the burden of extreme scaling. In this category fall the evolutions presented in Section 6, which in their turn will be the stepping stones for longer-terms goals.

Integration is the other side of the same medal: the whole field of Design and Test relies on a set of shared Abstraction levels and of the Flow that connects them. Incremental innovations usually happen outside of the traditional flows and require significant effort to be implemented. It is the difference between a promising Proof-of-Concept and a real-implementation: the so-called Valley of Death [ELL22] that lays between TRLs 4 and 6. The only way to reach Automation is to interface with existing Tool suites: this is where Standards come into place. By influencing their scope and development process, we are able to prepare the road to integration: standards are the best way to interact with third-party proprietary tools, and guarantee that your solutions will be actually usable. Solutions such as MAST ally research excellence with compatibility with the IEEE 1687 and IEEE P1687.1 standards to propose innovative solutions to real-world problems. These results will be the base of future discussions with industrial partners for both technology transfers and new use cases.

The domain of Electronics systems is in constant **Evolution**. Short-term solutions might answer today's challenges, but swiftly become obsolete. This is especially true for Standards: certifying the status-quo might solve the immediate issue, but it will not pass the trial of time. Only active interaction with the Standard bodies and Working Groups can guarantee that the solutions will remain future-proof by continuously challenging and pushing their boundaries with new applications and use cases. Evolution also constantly shifts the edges between Physical and Functional elements: the push toward System-on-Chips resulted in integrating into the same chip IP traditionally on different physical supports. Solutions like Chiplets are blurring even more the picture, spreading functionalities over different, but not fully independent, supports. All these continuous changes open new, exciting opportunities.

This leads to the last axis, the one with the biggest research potential: **Proximity**. The current Flows lack the capability of exchanging information between Abstractions, leading to dead-ends due to the impossibility of leveraging the capabilities of neighboring domains. A particularly interesting example is indeed the hand-off between the Design and the Test flows. Design ends at the Validation/Verification step, where an in-depth analysis is done to prove that the final Circuit Design truly follows the high-level Specifications. It is probably the step where there is the deepest understanding of the System. However, all this is lost when passing to the Test step: Structural test only needs the circuit itself, and makes no assumptions on it. As explained in several parts of this Document, this choice is made both because it allows a quantitative measurement of the circuit's "correctness", but also because the Testing Execution phase is traditionally extremely limited. Thanks to our Abstraction this is now not true anymore: Testing can also benefit from a powerful and flexible execution environment. In the following years, we will explore ways to export the knowledge of Verification/Validation and leverage it during the Testing, to obtain new and exciting possibilities.

8 Bibliographie

[1149.1] "IEEE Standard for Test Access Port and Boundary-Scan Architecture," in IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001), vol., no., pp.1-444, 13 May 2013, doi: 10.1109/IEEESTD.2013.6515989, first published in 1990, revised in 2001 and 2013

[1450] "IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data," in IEEE Std 1450-1999 , vol., no., pp.1-140, 1 Sept. 1999, doi: 10.1109/IEEESTD.1999.90563.

[1450.6] "IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data-Core Test Language (CTL)," in IEEE Std 1450.6-2005, vol., no., pp.1-120, 5 April 2006, doi: 10.1109/IEEESTD.2006.8328064.

[1500] "IEEE Standard Testability Method for Embedded Core-based Integrated Circuits," in IEEE Std 1500-2022 (Revision of IEEE Std 1500-2005) , vol., no., pp.1-168, 12 Oct. 2022, doi: 10.1109/IEEESTD.2022.9916221.

[1532] "IEEE Standard for In-System Configuration of Programmable Devices," in IEEE Std 1532-2002 (Revision of IEEE Std 1532-2001), vol., no., pp.0_1-141, 2003, doi: 10.1109/IEEESTD.2003.94229.

[1687] "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," in IEEE Std 1687-2014, vol., no., pp.1-283, 5 Dec. 2014, doi: 10.1109/IEEESTD.2014.6974961.

[3GPP] 3GPP Homepage, https://www.3gpp.org/ , regularly updated

[7SER] "7-Series Architecture Overview", https://xilinx.eetrend.com/files-eetrend-xilinx/forum/201509/9204-20390-7_series_architecture_overview.pdf, 2013, retrieved 12/2023

[AGRA84] D. Agrawal, S.K. Jain, and D.M. Singer, "Automation in Design for Testability", Proc. IEEE Custom Integrated Circuits Conf, 1984, pp.159-163.

[ARMv7], "ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition" , https://developer.arm.com/documentation/ddi0406/latest/

[BAR17] Mark Barnes, "Alexa, are you listening?", MWR Info Security, https://labs.mwrinfosecurity.com/blog/alexa-are-you-listening/, Aug 1st 2017.

[BAST19] BASTION Benchmarks Homepage, https://gitlab.com/IJTAG/benchmarks , last updated in 2019

[BOOST] "Boost C++ Libraries", https://www.boost.org/

[BSCAN2] "BSCAN2 – Multiple Scan Port Linker" Reference Design RD1002, Lattice Semiconductor, March 2014

[CAN18] R. Cantoro et al. F. G. Zadegan, M. Palena, P. Pasini, E. Larsson, and M. Sonza Reorda, "Test of reconfigurable modules in scan networks," IEEE Transactions on Computers, 2018.

[CHA20] Chafkin, Max; King, Ian (October 4, 2017). "Apple and Qualcomm's Billion-Dollar War Over an $18 Part". Bloomberg.com. Archived from the original on December 4, 2020. Retrieved October 4, 2017.

[DAM19] A. Damljanovic, A. Jutman, G. Squillero, and A. Tsertov, "Post-silicon validation of ieee 1687 reconfigurable scan networks," in 24th IEEE European Test Symposium (ETS) IEEE, 2019.

[DASIL19] M. Da Silva et_al, "Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, doi: 10.1109/TCAD.2018.2818722.

[DEPA94] "Design Patterns: Elements of Reusable Object-Oriented Software", Gamma, Helm, Johnson and Vlissides, Addison-Wesley, 1994, ISBN 0-201-63361-2

[DWO13] J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz, and M. Thornton, "Don't forget to lock your SIB: hiding instruments using P1687," in Proc. IEEE Int. Test Conf. (ITC), Sep. 2013.

[EHR09] Ehrenberg, Heiko; WENZEL, Thomas. Combining Boundary Scan and JTAG Emulation for advanced structural test and diagnostics. White Paper, GOEPEL electronics, 2009.

[ELF95] Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2 (May 1995)

[ELL22] Ellwood P. et al, "Crossing the valley of death: Five underlying innovation processes", Technovation, Volume 109, 2022, ISSN 0166-4972, https://doi.org/10.1016/j.technovation.2020.102162 .

[FTDI] Future Technology Devices International Homepage, https://ftdichip.com/, updated on a regular basis

[FNMT] Homepage of the Fédération des Micro et Nanotechnologies, https://fmnt.fr/

[GAIS] FrontGrade Gaisler Homepage, https://gaisler.com/index.php/

[GIRK92] Girkar, Milind, and Constantine D. Polychronopoulos. "Extracting task-level parallelism." ACM Transactions on Programming Languages and Systems (TOPLAS) 17.4 (1995): 600-634.

[GOOR90] A.J. Goor and C.A. Verruijt, "An overview of deterministic functional RAM chip testing", ACM Computing Surveys, vol. 22, no. 1, March 1990.

[HERY98] Herity, Dominic. "C++ in embedded systems: Myth and reality." Embedded Systems Programming 11.2 (1998): 48-71.

[HSDL] "Hierarchical Scan Description Language", http://www.asset-intertech.com/support/hsdl.html, 1992

[I2C14] "I2C-bus specification and user manual", http://www.nxp.com/ documents/ user_manual/ UM10204.pdf, April 2014

[IEEE] Institute of Electrical and Electronics Engineers Homepage, https://ieee.org/, regularly Updated

[IEEESA] IEEE Standard Association's Homepage, https://standards.ieee.org/, regularly Updated

[iNEMI09] 2009 iNEMI Roadmap, https://www.inemi.org/pr040109

[KAP99] G. Kaplan, "Industrial electronics [technology 1999 analysis and forecast]," in IEEE Spectrum, vol. 36, no. 1, pp. 68-72, Jan. 1999, doi: 10.1109/6.738329.

[KAPU08] R. Kapur, S. Mitra and T. W. Williams, "Historical Perspective on Scan Compression," in IEEE Design & Test of Computers, vol. 25, no. 2, pp. 114-120, March-April 2008, doi: 10.1109/MDT.2008.40.

[KELL90] B.L.Keller and T.J. Snethen, "Built-In Self-Test Support in the IBM Engineering Design System," IBM Journal of Research and Development,Vol. 34(2/3),pp. 406-415,1990

[LIHN06] Lihn H., Reusable, Low-cost, and Flexible Multidrop System JTAG Architecture."ITC'06. IEEE International. IEEE, 2006. p. 1-10.

[MAR93] E. Martin, O. Sentieys, H. Dubois and J. L. Philippe, "GAUT: An architectural synthesis tool for dedicated signal processors," Proceedings of EURO-DAC 93 and EURO-VHDL 93- European Design Automation Conference, Hamburg, Germany, 1993, pp. 14-19, doi: 10.1109/EURDAC.1993.410610.

[MCLA12] T. McLaurin, F. Frederick and R. Slobodnik, "The DFT challenges and solutions for the ARM® Cortex™-A15 Microprocessor," 2012 IEEE International Test Conference, Anaheim, CA, USA, 2012, pp. 1-9, doi: 10.1109/TEST.2012.6401534.

[MERA19] Marc Merandat, Vincent Reynaud, Emanuele Valea, Jerome Quevremont, Nicolas Valette, Paolo Maistri, Regis Leveugle, Marie-Lise Flottes, Sophie Dupuis, Bruno Rouzeyre, Giorgio Di Natale, "A Comprehensive Approach to a Trusted Test Infrastructure," in International Verification and Security Workshop, Rhodes 2019.

[MIL94] L. Milor, A Sangiovanni-Vincentelli, "Minimizing Production Test Time to Detect Faults in Analog Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol 13, No. 6, June 1994.

[OOC] "Object Oriented C Programming", https://staff.washington.edu/gmobus/Academics/TCES202/Moodle/OO-ProgrammingInC.html, retrieved on December 2023

[OPENOCD] Open On-Chip Debugger Homepage, https://openocd.org/

[P1687.1] IEEE P1687.1 WG website, https://ieee-sa.imeetcentral.com/16871/, updated on a regular basis

[PYBIND] https://pybind11.readthedocs.io/en/stable/advanced/pycpp/index.html

[RAFA15] B. Rafal, K. Michael A and H.-J. Wunderlich, "Fine-Grained Access Management in Reconfigurable Scan Networks," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, pp. 934-947, 2015.

[PARHI91] K.K. Parhi, D.G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding", IEEE Transactions on Computers, Volume: 40 , Issue: 2 , Feb 1991

[PE15] Andersson, Henrik (2015-04-23). "application/vnd.microsoft.portable-executable". IANA. Retrieved 2017-03-26.

[REA05] J. Rearick et al., "IJTAG (internal JTAG): a step toward a DFT standard," IEEE International Conference on Test, 2005., Austin, TX, USA, 2005, pp. 8 pp.-815, doi: 10.1109/TEST.2005.1584044.

[REA12] Jeff Rearick, "DFT and Testing vs. Inflection Points and Paradigm Shift", "Keynote address," 2012 IEEE International Test Conference, 2012, pp. 8-10, doi: 10.1109/TEST.2012.6401519.

[ROS10] K. Rosenfeld and R. Karri, "Attacks and Defenses for JTAG," in IEEE Design & Test of Computers, Jan.-Feb. 2010, doi: 10.1109/MDT.2010.9.

[SAR17] P. Sarson and J. Rearick, "Use models for extending IEEE 1687 to analog test," 2017 IEEE International Test Conference (ITC), Fort Worth, TX, USA, 2017, pp. 1-8, doi: 10.1109/TEST.2017.8242068.

[SEDA] Siemens EDA Homepage, https://eda.sw.siemens.com/, regularly updated

[SEDA-IN] Siemens Silicon Insight, https://eda.sw.siemens.com/en-US/ic/tessent/test/siliconinsight/

[SKSU13] Sk Subidh Ali, Samah Saeed, Ozgur Sinanoglu, Ramesh Karri. New Scan-Based Attack Using Only the Test Mode and an Input Corruption Countermeasure. 21th IFIP/IEEE

International Conference on Very Large Scale Integration - System on a Chip (VLSI-SoC), Oct 2013, Istanbul, Turkey. pp.48-68, ff10.1007/978-3-319-23799-2_3ff.

[SLM23] "Silicon Lifecycle Management", https://www.synopsys.com/solutions/silicon-lifecycle-management.html, retrieved on 12/08/2023

[SPARCV8], "The SPARC Architecture Manual Version 8", https://www.gaisler.com/doc/sparcv8.pdf

[STAPL] Altera Jam STAPL Software, https://www.altera.com/support/software/download/programming/jam/jam-index.jsp

[STARSY] Synopsys's STAR Hierarchical Subsystem (SHS) homepage, https://www.synopsys.com/implementation-and-signoff/test-automation/designware-shs.html , updated on a regular basis

[SUN09]S. Sunter, "EDA for Analog DFT? – Designers Must Get on the Bus," Panel 2.3, IEEE International Test Conference, 2009.

[SVF99] "Serial Vector Format Specification", ASSET InterTech Inc. Revision E, 8 March 1999

[SYNO] Synopsys Homepage, https://www.synopsys.com/, regularly updated

[TANE15] Tanenbaum, A. S., Bos H., "Modern Operating Systems (4 ed.)". Pearson Education, 2015, Inc. ISBN 978-013359162-0

[THIE19] Thiemann et al , "On Integrating Lightweight Encryption in Reconfigurable Scan Networks," , Proc European Test Symp. (ETS 2019)

[TLR-EU] "Technology readiness levels (TRL); Extract from Part 19 - Commission Decision C(2014)4995", https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf 2014. Retrieved 11 November 2019

[TRL-NASA] "Technology Readiness Level Definitions" https://www.nasa.gov/wp-content/uploads/2017/12/458490main_trl_definitions.pdf . Retrieved 6 September 2019

[TSE16] A. Tšertov et al., "A suite of IEEE 1687 benchmark networks," 2016 IEEE International Test Conference (ITC), Fort Worth, TX, USA, 2016, pp. 1-10, doi: 10.1109/TEST.2016.7805840.

[TSLS23] "Tessent Silicon Lifecycle Solutions", https://eda.sw.siemens.com/en-US/ic/tessent/, retrieved on 12/08/2023

[VAL19] E.Valea et al., "Stream vs block ciphers for scan encryption", Microelectronics Journal,2019,doi:10.1016/j.mejo.2019.02.019.

[VSTA20] "Industrial Application of IJTAG Standards to the Test of Big-A/little-d Devices – plus Updates to the Latest State of IEEE P1687.2" , H. M. von Staudt, M. Benhebibi,J. Rearick, M. Laisne, Distinguished ITC 2020 Paper INTERNATIONAL TEST CONFERENCE ASIA

[VTB03] B. G. Van Treuren and J. M. Miranda, "Embedded boundary scan," in IEEE Design & Test of Computers, vol. 20, no. 2, pp. 20-25, March-April 2003, doi: 10.1109/MDT.2003.1188258.

[VTB05] B. G. Van Treuren, B. E. Peterson and J. M. Miranda, "JTAG-based vector and chain management for system test," IEEE International Conference on Test, 2005., Austin, TX, USA, 2005, pp. 10 pp.-787, doi: 10.1109/TEST.2005.1584041.

[WEST81] D. Westcott, "The Self Assist Test Approach to Embedded Arrays," Proc. Intl Test Conf, 1981, pp.203-207.

[ZC702] Zynq-7000 All Programmable SoC ZC702 Evaluation Kit Quick Start Guide https://www.xilinx.com/content/dam/xilinx/support/documents/boards_and_kits/zc702_zvik/xtp310-zc702-quickstart.pdf

9 Glossary

ATE: Automated Test Equipment, i.e. the machine responsible for applying test patterns to a target SUT

ATPG: Automated Test Pattern Generation.

BIST: Built-In Self Test. An instrument embedded inside a component to test it independently from external data.

BSDL : Boundary Scan Description Language, the normative language for JTAG

CSU: Capture-Shift-Update, the base date exchange operation of JTAG

DfT: Design for Test. Features added to a design to boost its testability

DSL: Domain Specific Language. A language developed to

DUT: Design Under Test

HAL: Hardware Abstraction Layer, the part of an Operating System responsible for interaction with the Hardware

HDL: Hardware Description Language. A computer language used to describe an electronic component

ICL: Instrument Connectivity Language. A DSL used to describe IEEE 1687 hardware topologies

IJTAG: Internal (or Instrument) JTAG, a denomination of the IEEE 1687 Standard

IP: Intellectual Property. In Electronics design, it usually refers to a sub-system described in an HDL that can be reused in a bigger design

IP-Based Design: a design paradigm where a system is seen as a composition of independely-developed IPs

JTAG: Joint Test Action Group, a denomination of the IEEE 1149.1 Standard

Pattern: A set of input vectors and the expected outputs for a fault-free element.

PCB: Printed Circuit Board

PDL: Procedural Description Language, used in IJTAG to describe Test Intent

PUK: Physical Unclonable Key. An hardware component providing an unique key based on the physical variability of a given circuit.

RSN: Reconfigurable Scan Network. A scan network whose topology can dynamically change

RTL: Register Transfer Level. The abstraction level used to model digital circuits

Solver: in IJTAG, it is the software responsible for topology resolution

Retargeter: the software responsible for translation device-level pattern to top-level

Segment: In IJTAG, a TDR subset which is selectable through a ScanMux

SIB: Segment Insertion Bit, the reference dynamic topology element for IEEE 1687

SUT: System Under Test

TAP: Test Access Port

TCL: Tool Command Language, a scripting language used for interacting with EDA tools.

TDR: Test Data Register, a register selected by the DR branch of the JTAG TAP FSM

Résumé

La complexité des systèmes électroniques actuels, les volumes de production et la qualité imposée par des applications critiques telles que l'automotive mettent les approches de test structurel traditionnelles sous forte pression. Pour surmonter ce problème, le domaine des tests connaît une évolution profonde, dominée par des nouvelles techniques de « Design for Test » qui poussent l'automatisation au cœur même des systèmes et par des nouveaux standards comme IEEE 1687. Mais ces nouveautés génèrent elles-mêmes des nouvelles contraintes, telle que la sécurité ou le besoin de supporter des systèmes mixtes analogiques/ numériques. Dans cette soutenance, nous passerons en revue 15 années de recherche dans le domaine des tests automatisés afin de spécifier, développer et mettre en œuvre de nouveaux flux logiciels capables de dépasser les limitations existantes et de permettre la nouvelle (r)évolution.

Mots-clés : Test, Standards, Flot de Conception Aidée par Ordinateur (CAO), IEEE 1687, IJTAG, Système sur puce

Abstract

The complexity of today's electronic systems, the production volumes and the quality imposed by critical applications such as the automobile are putting traditional structural-based testing approaches under great pressure. To overcome this problem, the testing field is going through a profound evolution, dominated by new "Design for Test" techniques that push automation to the very heart of systems and new standards like IEEE 1687 and its derivations. But these same novelties generate new constraints such as security, or the need to handle mixed analog/digital systems. In this defense, we will review 15 years of Research in the field of Automated Testing in order to specify, develop and implement new Software Flows able to over legacy limitations and enable the new (r)evolution.

Keywords: Test, Standards, Electronics Design Automation (EDA) Flow, IEEE 1687, IJTAG, SoC