

IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device

IEEE Computer Society

Sponsored by the
Test Technology Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 1687™-2014

IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device

Sponsor

Test Technology Standards Committee
of the
IEEE Computer Society

Approved 3 November 2014

IEEE-SA Standards Board

Abstract: A methodology for accessing instrumentation embedded within a semiconductor device, without defining the instruments or their features themselves, via the IEEE 1149.1™ test access port (TAP) and/or other signals, is described in this standard. The elements of the methodology include a hardware architecture for the on-chip network connecting the instruments to the chip pins, a hardware description language to describe this network, and a software language and protocol for communicating with the instruments via this network.

Keywords: access network, built-in self-test (BIST), boundary scan, debug, design for testability (DFT), embedded instruments, IEEE 1149.1™, IEEE 1687™, Instrument Connectivity Language (ICL), internal JTAG (IJTAG), Joint Test Action Group (JTAG), on-chip instrumentation, Procedural Description Language (PDL), test, Tool Command Language (Tcl)

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2014 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 5 December 2014. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-0-7381-9416-5 STD20033
Print: ISBN 978-0-7381-9417-2 STDPD20033

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (“IEEE-SA”) Standards Board. IEEE (“the Institute”) develops its standards through a consensus development process, approved by the American National Standards Institute (“ANSI”), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at <http://ieeexplore.ieee.org/xpl/standards.jsp> or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at <http://standards.ieee.org>.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this IEEE standard was completed, the IEEE 1687 Working Group had the following membership:

Kenneth Posse, *Chair*
Alfred Crouch, *Vice Chair*
Jeff Rearick, *Editor*
Michael Laisne, *Webmaster*

Paul Abelovski
Alan Bair
Bill Bruce
Krishna Chakravadhanula
C. J. Clark
Mike Coldewey
Jean-Francois Cote
Bruce Cowan
Adam Cron
Romi Datta
Stylianios Diamantidis
Jason Doege
Richard Dugan
Theodore Eaton
Heiko Ehrenberg
William Eklow
Brian Foutz
Pradipta Ghosh
Kevin Gorman
Suresh Goyal

J. J. Grealish
Scott Hartranft
Hong-Shin Jun
Rohit Kapur
Martin Keim
Bruno Latulippe
Andrew Levy
Guoqing Li
Ed Malloy
Harrison Miles, Jr.
Skip Meyers
Jay Nejedlo
Thai-Minh Nguyen
Hari Nookala
Rick Nygaard
Victor Orona
John Parham
Srinivas Patil
Michele Portolan

John Potter
Jeff Remmers
Paul Reuter
Mike Ricchetti
Ben Rice
Thomas Rinderknecht
Franciso Russi
John Seibold
Craig Stephan
Anthony Suto
Steven Terry
Brian Turmelle
William Tuthill
Brad Van Treuren
Hans Martin von Staudt
Hugh Wallace
Brian Wang
Mike Wiznerowicz
Christian Zoellin
Songlin Zuo

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Saman Adham
Gobinathan Athimolom
John Braden
Bill Brown
Gunnar Carlsson
Krishna Chakravadhanula
Keith Chow
C. J. Clark
Jean-Francois Cote
Adam Cron
Alfred Crouch
Russell Davis
Frans G. De Jong
Jason Doege
Sourav Dutta
Theodore Eaton
Heiko Ehrenberg
Peter Eijnden
William Eklow
Joshua Ferry
Kevin Gorman
Chris Gorringer
J. J. Grealish
Randall Groves
Peter Harrod
Kazumi Hatayama
Werner Hoelzl
Neil Glenn Jacobson

Hong-Shin Jun
Rohit Kapur
Martin Keim
Anzou Ken-Ichi
Michael Laisne
James Langlois
Roland R. Latvala
Philippe Lebourg
Adam Ley
Teresa Lopes
Greg Luri
Colin Maunder
Ian McIntosh
Reinhard Meier
Harrison Miles, Jr.
Jeffrey Moore
Richard Morren
Zainalabedin Navabi
Ion Neag
Jim O'Reilly
Kim Petersen
Ulrich Pohl
Irith Pomeranz
Michele Portolan
Kenneth Posse
John Potter
Jeff Rearick

Paul Reuter
Mike Ricchetti
Gordon Robinson
Andrzej Rucinski
Francisco Russi
Bartien Sayogo
John Seibold
Kapil Sood
Roger Sowada
Thomas Starai
Craig Stephan
Michael Stora
Walter Struppler
Anthony Suto
Efren Taboada
David Thompson
Brian Turmelle
William Tuthill
Louis Ungar
Srinivasa Vemuru
Tom Waayers
Hugh Wallace
Douglas Way
Oren Yuen
Janusz Zalewski
Daidi Zhong
Christian Zoellin
Songlin Zuo

When the IEEE-SA Standards Board approved this standard on 3 November 2014, it had the following membership:

John Kulick, *Chair*
Jon Walter Rosdahl, *Vice-chair*
Richard H. Hulett, *Past Chair*
Konstantinos Karachalios, *Secretary*

Peter Balma
Farooq Bari
Ted Burse
Clint Chaplain
Stephen Dukes
Jean-Phillippe Faure
Gary Hoffman

Michael Janezic
Jeffrey Katz
Joseph L. Koepfinger*
David J. Law
Hung Ling
Oleg Logvinov
T. W. Olsen
Glenn Parsons

Ron Peterson
Adrian Stephens
Peter Sutherland
Yatin Trivedi
Phil Winston
Don Wright
Yu Yuan

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Michelle Turner
IEEE-SA Content Publishing

Kathryn Bennett
IEEE-SA Technical Community Programs

Introduction

This introduction is not part of IEEE Std 1687™-2014, IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device.

The development of this standard stemmed from two independent (and unaware of each other) efforts at the 2004 International Test Conference to address issues surrounding the use of the IEEE 1149.1 test access port (TAP) for purposes well beyond boundary scan testing. These efforts were merged, and the Internal JTAG (IJTAG) Working Group was born. The scope of the effort was refined in the following months as more members joined the group to focus on the access to design for testability (DFT) features (more generally called *instruments*) inside devices. The development of the ideas comprising this standard can be traced by presentations at a series of conferences, workshops, and symposia, including ITC'05, BAST'06, ITSW'06, VTS'06, ETS'06, ITC'06, VTS'07, ETS'07, ITC'07, ETS'08, ITC'08, and IOLTS'09, ITC'11, ETS'12, ITC'12, and ITC'13.

Frequently asked questions

The IEEE 1687 web site will include an FAQ.^a

^aSee <http://grouper.ieee.org/groups/1687/>.

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
1.3 Background.....	2
1.4 Organization	2
1.5 Context	3
2. Normative references.....	4
3. Definitions, acronyms, and abbreviations	4
3.1 Definitions	4
3.2 Acronyms and abbreviations	10
4. Technology	11
4.1 Introduction	11
4.2 Serial access networks	11
4.3 On-chip instruments	18
5. Hardware architecture	19
5.1 Introduction to the IEEE 1687 network.....	19
5.2 Hierarchical IEEE 1687 networks	19
5.3 Controller.....	22
5.4 Instrument interface	22
5.5 Access network.....	23
5.6 Test data register.....	30
5.7 Local reset	34
5.8 Delivery and integration of instruments	38
5.9 Embedded TAP controller	39
5.10 Definitions of the structure of IEEE 1687 hardware architecture.....	44
5.11 Port functions of a module.....	45
5.12 Signals between and within IEEE 1687 modules	47
5.13 Components comprising a module	48
5.14 TAP finite state machine embedded in an IEEE 1687 module	50
5.15 Access network behavior	53
5.16 Plug-and-play interfaces	53
6. Instrument Connectivity Language (ICL)	56
6.1 ICL introduction	56
6.2 ICL overview	57
6.3 ICL lexical conventions and definitions	59
6.4 ICL primitive element keywords and statements.....	73
6.5 ICL informational statements	135
6.6 ICL connectivity	141
6.7 Inferring information in implicit ICL	141
6.8 Active values for control signals	142
7. Procedural Description Language (PDL): level-0	143
7.1 Purpose	143
7.2 PDL levels	143
7.3 Basic PDL concepts.....	144
7.4 Retargeting of PDL.....	147

7.5 PDL level-0 overview	149
7.6 PDL general rules	152
7.7 Generic PDL tokens.....	154
7.8 PDL numbers	155
7.9 PDL level-0 commands	157
8. Procedural Description Language: level-1 (Tcl)	182
8.1 Purpose	182
8.2 Tcl command extensions	182
8.3 PDL level-1 overview.....	183
8.4 PDL level-1 commands	183
8.5 PDL level-1 example	189
Annex A (informative) ICL grammar.....	190
Annex B (informative) PDL level-0 grammar.....	199
Annex C (informative) PDL level-1 grammar.....	203
Annex D (informative) PDL differences between IEEE Std 1687-2014 and IEEE Std 1149.1-2013	204
Annex E (informative) Examples	212
Annex F (informative) Design guidance	255
Annex G (informative) Bibliography	267

IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device

IMPORTANT NOTICE: *IEEE Standards documents are not intended to ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.*

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

1. Overview

1.1 Scope

This standard develops a methodology for access to embedded instrumentation, without defining the instruments or their features themselves, via the IEEE 1149.1™ test access port (TAP) and additional signals that may be required. The elements of the methodology include a description language for the characteristics of the features and for communication with the features, and requirements for interfacing to the features.

1.2 Purpose

IEEE Std 1149.1 specifies circuits to be embedded within a semiconductor device to support board test; namely, the TAP, TAP Controller, and a number of internal registers.¹ In practice the TAP and TAP Controller are being used for other functions well beyond boundary scan in an ad-hoc manner across the industry to access a wide variety of embedded instruments. The purpose of the IEEE 1687 initiative is to provide an extension to IEEE Std 1149.1 specifically aimed at using the TAP to manage the configuration, operation, and collection of data from this embedded instrumentation circuitry.

¹ Information on references can be found in Clause 2.

1.3 Background

There exists the widespread use of embedded instrumentation [such as built-in self-test (BIST) engines, complex I/O characterization and calibration, embedded timing instrumentation, etc.), each of which is accessed and managed by a variety of external controllers using a variety of mechanisms and protocols. Therefore, a need exists for standardization of these methods in order to facilitate an efficient and orderly process for the preparation of tests that access and control these embedded instruments. Given the widespread use of IEEE 1149.1 TAP and boundary-scan architecture as a gateway to many internal test, debug, and programming features, it is logical to build upon the foundation of the boundary scan standards (IEEE Std 1149.1-1990 [B2], IEEE Std 1149.1-2001, IEEE Std 1149.1-2013, IEEE Std 1149.4™-2010 [B3], IEEE Std 1149.6™-2010 [B4], IEEE Std 1532™-2002 [B5], and IEEE Std 1500™-2005) to support TAP-based access to internal device test features in a standardized manner.² IEEE Std 1149.1-2013 extended previous versions of that standard to include procedural access to reconfigurable test data registers (TDRs) accessible via the TAP. IEEE Std 1687 describes an instrument-centric approach to this problem that allows retargeting from instrument ports through a wide variety of on-chip network configurations (of which IEEE Std 1500 is one example) to a device interface (of which the TAP is one example). Like IEEE Std 1500, IEEE Std 1687 specifies how to build a conformant on-chip network while not forcing a specific device interface to the network. IEEE Std 1687 has taken a more descriptive than prescriptive approach for specifying the on-chip network, thus allowing many existing architectures to conform.

This standardization effort is intended to address the *access* to on-chip instrumentation, not the instruments themselves. The elements of standardized access include the following:

- a) The hardware architecture for connecting instruments to the TAP via a serial access network and/or to other interfaces via a serial or parallel access network,
- b) A hardware description language (HDL) that documents this access network between the instruments and device interfaces, and
- c) A procedure description language that documents the operation of the instruments.

Items that have been specifically included in the scope of the standard include the application of instrument-level procedures retargeted through various levels of hierarchy, the retargeting of intellectual property (IP)-level patterns to device-level TAP patterns, and the documentation of the connection of instruments to other (non-TAP) interfaces.

Items that have been excluded from the scope of the standard are the detailed off-chip protocols for instrument communication over non-TAP interfaces, synchronized interoperability between instruments, and the assurance of reusability of chip-level patterns at board or system levels.

The goal of IEEE Std 1687 is to facilitate the use and reuse of internal instrumentation by providing a standard yet flexible network architecture for accessing the instruments and standard descriptions of both the network and the operation of the instruments. Given the considerable number of devices containing internal instruments that are already available, this standard has made every reasonable attempt to provide support for the description of commonly used legacy implementations. The standard also supports forward-looking architectures that are scalable and uniform in their approach to accessing embedded instruments in a hierarchical manner.

1.4 Organization

This standard is organized as follows:

² The numbers in brackets correspond to those of the bibliography in Annex G.

Clause 1 provides background and context for this standard.

Clause 2 provides references necessary to understand this standard.

Clause 3 defines the terminology and nomenclature used in this standard.

Clause 4 is a tutorial that outlines the technologies addressed and utilized by this standard. This clause does not contain rules.

Clause 5 provides rules for the network architecture used by this standard.

Clause 6 provides rules for documenting the hardware architecture used by this standard.

Clause 7 provides rules for documenting the operation of the instruments used by this standard.

Clause 8 provides rules for manipulating the data produced by the instruments and for using Tcl as a programming language to manage instrument interactions.

Annex A (informative) specifies the language constructs used to describe the access network.

Annex B (informative) specifies the basic language constructs used to interact with the instruments.

Annex C (informative) specifies the advanced (Tcl) language constructs used to interact with the instruments.

Annex D (informative) identifies the subtle differences between the otherwise mostly common implementation of PDL.

Annex E (informative) presents many use cases showing applications of ICL and PDL to common circuits.

Annex F (informative) discusses practical issues for the implementation of commonly used circuits.

Annex G (informative) is a bibliography.

1.5 Context

The context in which this standard is applicable is illustrated in Figure 1, which shows a device under test (DUT) that includes a TAP interface, some pins and an I/O interface, and three (lightly shaded) access networks through which those interfaces are connected to four internal instruments. This standard defines the architecture of and a structured way to describe the hardware contents of the access network and the well-defined interfaces (darker shaded small rectangles) between the access networks and the instruments. This standard also describes the software content (e.g., test patterns and procedures) that is delivered over the access network to interact with the instruments.

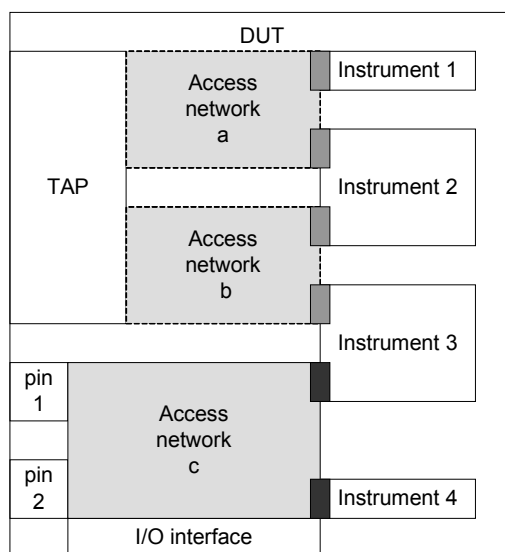


Figure 1—IEEE 1687 context

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 1149.1-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture.^{3, 4}

IEEE Std 1149.1-2013, IEEE Standard Test Access Port and Boundary-Scan Architecture.

IEEE Std 1500, IEEE Standard Testability Method for Embedded Core-Based Integrated Circuits.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.⁵

access network: *See:* **network.**

AccessLink: An **Instrument Connectivity Language (ICL)** keyword that is used to describe the details of the interface between the device pins and the **network**. The IEEE 1149.1 **test access port (TAP)** has a well-defined description built into **ICL**.

³ IEEE publications are available from The Institute of Electrical and Electronics Engineers (<http://standards.ieee.org/>).

⁴ The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc.

⁵ *IEEE Standards Dictionary Online* subscription is available at:
http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

active scan chain (or Active shift path): A set of serially connected shift register bits that advance by one bit per test clock cycle when enabled to shift. The network in the **module** (or modules) containing the segments of the scan chain is required to be selected in order to allow activity on the shift enable control signal.

Alias: An **Instrument Connectivity Language (ICL)** keyword that assigns a name to a given group of ICL objects (**register** bits or **ports**), which are then allowed to be referred to collectively by that name.

black-box module: A **module** description in **Instrument Connectivity Language (ICL)** that has only ports, but no contents. These modules may be used to hide proprietary intellectual property (**IP**). Such modules are often used in conjunction with **iScan** commands in **Procedural Description Language (PDL)** and thus have a precisely defined length at any given time, which is associated with each **ScanInterface** of the black-box module.

NOTE—See 6.4.5 and 7.3.5 for details.

boundary-scan testing: Testing of board or backplane interconnections between device pins as supported by IEEE Std 1149.1, IEEE Std 1149.4, and IEEE Std 1149.6. This testing technology is intended to find manufacturing defects such as open solder joints, shorted signal traces, broken bond wires, damaged drivers and receivers, etc.

built-in self-test (BIST), built-in test (BIT): Test of a component of a system by itself; more specifically, by test circuitry added to the mission circuitry that causes the component to exercise itself and report the results with no or minimal interaction with external sources. BIST may be applied on embedded components of a chip (such as memories, analog blocks, or I/O interfaces), on entire chips, or on entire boards or systems.

capture-shift-update (CSU): The sequence of operations applied to a **test data register (TDR)** to capture new data from the data-input port, shift the data through the TDR, then update the newly shifted data into the update stage of the TDR.

care bit: A port or register that has a non-X value that must be present in order to satisfy an objective (such as enabling a **scan chain** or maintaining a previously written value).

client interface: The **port functions** that attach a **module** to a host interface in a network. A client interface consumes control signals from the host interface and exchanges data with it.

client module: The successor of a **module** in a **network** organized in a host-client fashion. A **host module's** client is closer to the **instrument**. A client module contains a **client interface** that connects to a **host interface**. *Contrast:* **host module**.

component: A hardware entity that is one of the primitive building blocks of the network, including storage elements, multiplexers, and logic.

controlling state: A representation in a software model of the set of logic values that have been written to **ports** and/or **storage elements** in the **network**.

NOTE—See description in 7.3.3.

core: A component of a **device** that may appear multiple times or may be reused in other devices. Also referred to as an IP (**intellectual property**) core or simply as IP.

data register: A set of one or more storage cells with a parallel input and a parallel output.

device: A collection of circuitry with a well-defined test boundary, generally including a **TAP** and physical **pins**.

device interface: The circuitry that connects the device **pins** to the instrument access **network**. A **TAP controller** is the most common example of a **device interface** for an instrument access network, but other interfaces (I²C, a parallel microprocessor bus, a PCI-express port, etc.) could also serve as device interfaces if correctly adapted to the instrument access network. The device interface itself is not considered part of the network.

embedded test access port (eTAP): A captive version of a TAP that acts as the interface to a portion of the network or a wrapped instrument rather than serving as the device-level **TAP**. An embedded **TAP** is typically accompanied by an embedded **TAP controller (eTAPC)**.

embedded test access port controller (eTAPC): A finite state machine largely corresponding to that specified in IEEE Std 1149.1, which responds to the **embedded TAP** signals used as the interface to a sub-network or instrument. To force synchronization, there is a slight restriction from the IEEE 1149.1 **TAP** controller state diagram with respect to the arc traversed when the **eTAPC** is being re-selected.

NOTE—See 5.14.4.

expected state: A representation in a software model of the set of logic values that have been read from ports and/or storage elements in the network.

NOTE—See description in 7.3.3.

fan-in: The set of primitive elements encountered by tracing backwards through the network from the inputs of a specific element.

global reset: A signal that places all **modules** into their specified reset states.

handoff module: A **module** that is intended to be exchanged between parties [**intellectual property (IP) providers** and **integrators**] or tools (e.g., simulators or rule checkers) for which all access network **ports** must be declared and must match. A common example: a wrapped instrument delivered by an IP provider must have all the ports of the **Instrument Connectivity Language (ICL)** module present in the associated **register transfer language (RTL)** module. Note that the RTL may have additional ports that are unrelated to the access network; these need not be declared in the ICL module.

hardware description language (HDL): Any of a number of languages that describe the structural components, connectivity, behavior, or combinations thereof, of a circuit, including Verilog®, VHDL, **Instrument Connectivity Language (ICL)**, etc.

NOTE—Verilog is a registered trademark of Cadence Design Systems, Inc.

host interface: The **port functions** that attach a **module** to a **client interface**. A host interface provides control signals to a client interface and exchanges data with it.

host module: The predecessor of a **module** in a **network** organized in a host-client fashion. A **client module**'s host is closer to the **device interface**. A Host Module contains a **host interface** to which a **client interface** connects. *Contrast:* **client module**.

IEEE 1687 network: *See:* **network**.

instrument: Any on-chip circuit for test, debug, diagnosis, monitoring, characterization, configuration, or functional use that can be accessed by, configured from, or communicated with by an on-chip network described in **Instrument Connectivity Language (ICL)** that connects to an external device interface. An instrument is typically in a **module** that has only a **client interface**.

instrument access network: *See:* **network**.

Instrument Connectivity Language (ICL): The name of the language (both syntax and semantics) used to describe the structure of the **instrument access network** that connects **instruments** to **device interfaces**.

instrument interface: The connection point between the **network** and the **ports** of an **instrument**. When used in the context of a network, this point is conventionally referred to as the *right-hand side* of the network and may be compliant with the description of a scan host interface or a **test access port (TAP)** host interface. When used in the context of an instrument, this point describes the ports of the instrument, which may be compliant with the description of a scan client interface or a **TAP** client interface.

NOTE—See 5.16.

integrator: The consumer of **intellectual property (IP)** to be joined with other components to produce a device (or possibly another IP). The integrator's role and interest in IEEE Std 1687 is to construct a network that connects the IP, and use that network to **retarget** lower-level **Procedural Description Language (PDL)** procedures to be applied at the higher-level device that he has produced.

intellectual property (IP): A component of a device, potentially supplied by a third party, that would be accompanied by an **Instrument Connectivity Language (ICL)** description and one or more **Procedural Description Language (PDL)** procedures. *See:* **core**.

iScan command: A **Procedural Description Language (PDL)** command that specifies the bit values to both extract from and assign to the scan flip-flops comprising a **scan chain** of a known length.

NOTE—See 7.3.5 and 7.9.12.

justify (as in “**justify** a value on a signal”): The process of finding a valid setting of controllable values such that the target signal acquires the desired value.

left-hand side (LHS) ports: **Module port functions** that connect to **host modules**, which include ResetPort, SelectPort, CaptureEnPort, ShiftEnPort, UpdateEnPort, ScanInPort, ScanOutPort, TCKPort, DataInPort, DataOutPort, and others. Note that ScanInPort, ScanOutPort, DataInPort, and DataOutPort functions may connect to **peer modules** or may be **right-hand side (RHS) host ports** that connect to other **client modules**.

NOTE—See 5.11.

local reset: A signal generated internal to the **network** based on desired conditions that may be used to place a **module** into its specified reset state. A local reset event is desirable when a localized action is preferred to a **global reset** event (which might have consequences that are too broad).

local reset assert (LRA): A scan register bit whose purpose is to control the activation of a local reset signal.

logic state: A representation in a software model of an intermediate set of logic values that have been written to **ports** and/or **storage elements** in the **network**. *Contrast:* **controlling state**.

NOTE—See description in 7.3.3.

logical hierarchy: An arrangement of **modules** with parent-child relationships (i.e., a parent module instantiates child modules, with there being a **top-level module** at the highest level of the hierarchy).

module (IEEE 1687 module): A component containing a test access mechanism that provides communication with on-device instrumentation, an instrument, an interface to the pins of a device, or other module instantiations.

network (access network, instrument access network, IEEE 1687 network): The circuitry that connects the device interfaces to the instrument interfaces. An example of a device interface currently used for IEEE Std 1687 is the **test access port (TAP)**. The network consists of one or more **modules**. The **network interface** is the external access point to the network, conventionally referred to as the *left-hand side* (LHS) of the network. An instrument interface is the other end point of the network and is conventionally referred to as the *right-hand side* (RHS) of the network. Intermediate modules, if included, may be arranged in a host-client fashion and may contain logical hierarchy.

network hierarchy: A configuration of **modules** with host-client relationships (i.e., where one module is a host relative to a **client module** on its right and a client relative to a **host module** on its left, using the convention that the **device interface** is on the far left and the **instrument interface** is on the far right).

network interface: The connection point between the instrument access **network** and an external **device interface** [such as the **test access port (TAP)**]. *See: AccessLink.*

PDL Command Stream: A series of **Procedural Description Language (PDL)** commands applied to an instrument or network or device. A PDL command stream may be contained within a file, or it may be generated on-the-fly by a **PDL** generator.

peer module: A **module** adjacent to a module in a **network** organized in a host-client fashion. A module and its peer both share the same **host module**.

pin: A physical connection point to a packaged **device**. Example: the solder bumps of a surface-mount device are pins.

plug-and-play: A descriptive term used to define a level of compliance of a **network interface**, indicating its type [either scan or **test access port (TAP)**] and its role (either host or client). The signals comprising a plug-and-play interface are sufficiently well specified such that corresponding interfaces (e.g., scan host and scan client) may be directly connected (i.e., “plugged”) together to join sections of a **network**.

NOTE—See 5.16.

port: A connection point (terminal) of a collection of circuitry in a **module** that is intended to be wired to other circuitry. Example: the D, Q, and CK signals of a D-flip-flop module are ports; the internal signal that connects the master latch to the slave latch of the D-flip-flop is not a port.

port function: A **module** port with a specific function relevant to the operation of an instrument, network, or interface. The terminology “port function” indicates a property associated with a **port** that describes its operational purpose (i.e., its function). For IEEE Std 1687, the set of port functions are grouped into **left-hand side** (LHS) **ports** including ResetPort, SelectPort, CaptureEnPort, ShiftEnPort, UpdateEnPort, ScanInPort, ScanOutPort, TCKPort, DataInPort, and DataOutPort, as well as **right-hand side** (RHS) **ports** including ToResetPort, ToSelectPort, ToCaptureEnPort, ToShiftEnPort, ToUpdateEnPort, ToClockPort, ToTMSPort, ToTRSTPort, ToTCKPort, and ToIRSelectPort. Note that ScanInPort and ScanOutPort may be either left-hand or right-hand side ports; the same is true for DataInPort and DataOutPort.

primary input/primary output: A directly accessible input pin or output pin of a device in a chip context, or a DataInPort or DataOutPort in an instance context.

primitive component: A member of the set of lowest-level structural elements from which access networks are constructed; the set includes ScanRegister, ScanMux, OneHotScanGroup, DataRegister, DataMux, OneHotDataGroup, ClockMux, and LogicSignal.

NOTE—See 6.4.1.

Procedural Description Language (PDL): The name of the language (syntax and semantics) used to describe the operation of the instruments.

raw instrument: An instrument with only DataIn, DataOut, or Clock port functions in its client interface.

register transfer language (RTL): A **hardware description language (HDL)** that represents the behavior of a circuit at a high level by describing the storage elements and logical operations between them. The elements of the circuit are often organized using a hierarchy of **modules** (which define sub-circuits) and the connections between instances of those modules.

register: One or more **storage elements** that share a name and whose bits are arranged as a vector range (e.g., MyReg[31:0]).

retarget: The process of mapping a set of operations in **Procedural Description Language (PDL)** at the ports and/or registers of a **module** through the **network** to a desired module at a higher level of the logical hierarchy and/or closer to a **device interface** in the network hierarchy. A retargeted PDL will perform the same operations on the original module, but may also perform additional operations to configure the network between the original module and the new **target module**.

retargeting tool (or retargeting software): A tool that implements the process of **retargeting**.

right-hand side (RHS) ports: **Module port functions** that connect to **client modules**, which include ToResetPort, ToSelectPort, ToCaptureEnPort, ToShiftEnPort, ToUpdateEnPort, ScanInPort, ScanOutPort, ToTCKPort, DataInPort, DataOutPort, and others. Note that ScanInPort and ScanOutPort functions may connect to **peer modules** or may be **left-hand side (LHS) client ports** that connect to a **host module**; the same is true for DataInPort and DataOutPort.

NOTE—See 5.11.

run-test/idle (RTI): A state in the **test access port (TAP)** controller state diagram in which the controller can loop while other activity is occurring (e.g. a test is running or the chip is idling).

scan chain segment boundary: A connection between two portions of a **scan chain**. The segment boundary may occur within an IEEE 1687 **module** between two scan registers, or at a module boundary. A scan chain segment boundary may be used as the location for a scan chain reconfiguration circuit (e.g., a **scan multiplexer** or a **segment insertion bit**) or a retiming circuit (e.g., a lockup latch).

scan chain: One or more flip-flops connected into a serial shift register, whose shifting activity is controlled by test control signals (e.g., ShiftEn and TCK). A scan chain is modeled in **Instrument Connectivity Language (ICL)** as one or more ScanRegister. *Syn:* **scan path**.

NOTE—See 6.4.8.

scan client interface: A specific type of **Client Interface** that is comprised of the signals defined in a **ScanInterface** with a single **scan chain**, which allows plug-and-play attachment to a **serial access network**.

ScanInterface: An **Instrument Connectivity Language (ICL)** keyword used to identify a well-defined set of related signals of a **serial access network** that are ports of a **module** and comprise a logical interface to one or more **scan chains**. A ScanInterface may include multiple scan chains (with unique Scanin/Scanout pairs), and a module may have multiple ScanInterfaces.

segment insertion bit (SIB): A single-bit register (including an update stage) on a scan chain that controls whether or not a bypassable segment of that scan chain is included in the active shift path.

serial access network (scan network): A **network** or sub-network with one or more serial data communication paths and no parallel data signals.

shift path multiplexer (or **scan multiplexer**): A circuit element that may alter the configuration of a scan chain by selecting which shift storage elements are in the scan chain and/or in which order they appear.

shift storage element (or **shift path storage element**): A register on a scan chain.

signal: A connection between two or more **port functions** of **modules**.

simulation model: The circuit model utilized by a circuit simulation tool to predict the responses to stimuli, often represented in **register transfer language (RTL)**.

storage element: A bi-stable circuit element, such as a latch or a flip-flop, which retains state until certain enabling and/or clocking conditions are met that cause it to take on a new state.

target module: The **module** to which a **retargeting** operation is destined; the **Procedural Description Language (PDL)** operations from an original module lower in the **logical hierarchy** or closer to the client side in a **network hierarchy** are mapped through the **network** to the target module. The target module is often the same as the **top-level module**.

test access port (TAP) controller (TAPC): The finite state machine specified in IEEE Std 1149.1 that responds to the **TAP** signals and drives some or all of the on-chip test logic.

test access port (TAP): The device interface pins defined in IEEE Std 1149.1: TDI, TMS, TCK, TDO, and optionally TRST*.

test data register (TDR): An IEEE 1149.1-compatible scan chain that can be connected between test data in (**TDI**) and test data out (**TDO**) when selected by an instruction in the **test access port (TAP) controller**.

test-logic-reset (TLR): A state in the **test access port (TAP) controller** state diagram that causes selected state elements to be loaded with their reset values.

top-level module: The **module** at the highest point of a **logical hierarchy**. A top-level module has only children and is not instantiated by any other modules.

variable-length scan chain: A scan chain whose connectivity (and thus length) can be altered as a function of the data within the scan chain, or from data contained in other scan chains or registers or pins.

wrapped instrument: An instrument whose parallel data ports have been connected to the data in and out ports of scan registers that comprise a scan wrapper. The interface to a wrapped instrument consists of the serial interface of the scan wrapper rather than the parallel interface of the original instrument.

3.2 Acronyms and abbreviations

BIST	built-in self-test
BSDL	Boundary Scan Description Language
CSU	capture-shift-update
DR	data register
EDA	electronic design automation
eTAPC	embedded TAP controller
FSM	finite state machine
IC	integrated circuit

ICL	Instrument Connectivity Language
IP	intellectual property
IR	instruction register
JTAG	Joint Test Action Group
LHS	left-hand side
LRA	local reset assert
PDL	Procedure Description Language
PI/PO	primary input / primary output
PLL	phase-locked loop
RHS	right-hand side
RTI	run-test/idle
RTL	register transfer language
SerDes	serializer-deserializer
SIB	segment insertion bit
TAP	test access port
TCK	test clock
TDI	test data in
TDO	test data out
TDR	test data register
TLR	test-logic-reset
TMS	test mode select
TRST*	test reset (* indicates active low)

4. Technology

4.1 Introduction

Access to embedded instrumentation involves two key on-chip hardware technologies: access networks and embedded instruments. Some essential hardware background material is provided here and in Clause 5 to provide the framework upon which IEEE Std 1687 is built. The language [Instrument Connectivity Language (ICL)] used to describe the hardware is covered in Clause 6.

The networks that provide access to embedded instruments have taken many forms, ranging from serially accessed wrappers to direct device pin connections. The former category has a well-established technology base enabled by other standards, the essential components of which are described in the next subclause.

4.2 Serial access networks

The signaling protocol used for an IEEE 1687 serial access network is consistent with IEEE Std 1500, and a common interface to a serial access network of instruments is the IEEE 1149.1 TAP and its associated

TAP controller. Since a basic knowledge of these portions of IEEE Std 1149.1 and IEEE Std 1500 is useful for a complete understanding of the operation and control of an IEEE 1687 serial access network, the relevant principles from those standards are summarized here. Though these fundamental principles are not expected to change in future version of those standards, it should be noted that IEEE Std 1687 utilizes the features as documented in the versions in existence solely at the time of this writing.

4.2.1 IEEE 1149.1 test access port

The interface specified by IEEE Std 1149.1 is referred to as the *test access port* (TAP), and consists of the following device pins:

- TMS: Test Mode Select (input to control the TAP state machine)
- TDI: Test Data In (input to provide test control and data to the device)
- TDO: Test Data Out (output to observe test status and data from the device)
- TCK: Test Clock [input to clock the TAP state machine and test data registers (TDRs)]
- (optional) TRST*: Test Reset (active low) (input to asynchronously reset the TAP controller)

This interface is connected to the TAP controller.

4.2.2 IEEE 1149.1 test access port controller

The IEEE 1149.1 TAP controller consists of a state machine with 16 states whose principal purposes are to load an instruction into the TAP Instruction Register (IR) and access the associated Test Data Register (TDR) between Test Data In (TDI) and Test Data Out (TDO). The state diagram for the TAP controller is shown in Figure 2. The diagram is dominated by two symmetrical paths that specify the sequence of operations for a register (a TDR on the left, and the instruction register on the right). A TDR consists of one or more serially connected flip-flops that can optionally capture data from a circuit at cell inputs, can serially shift data through the chain of cells, and/or optionally provide updated data to the circuit via cell outputs.

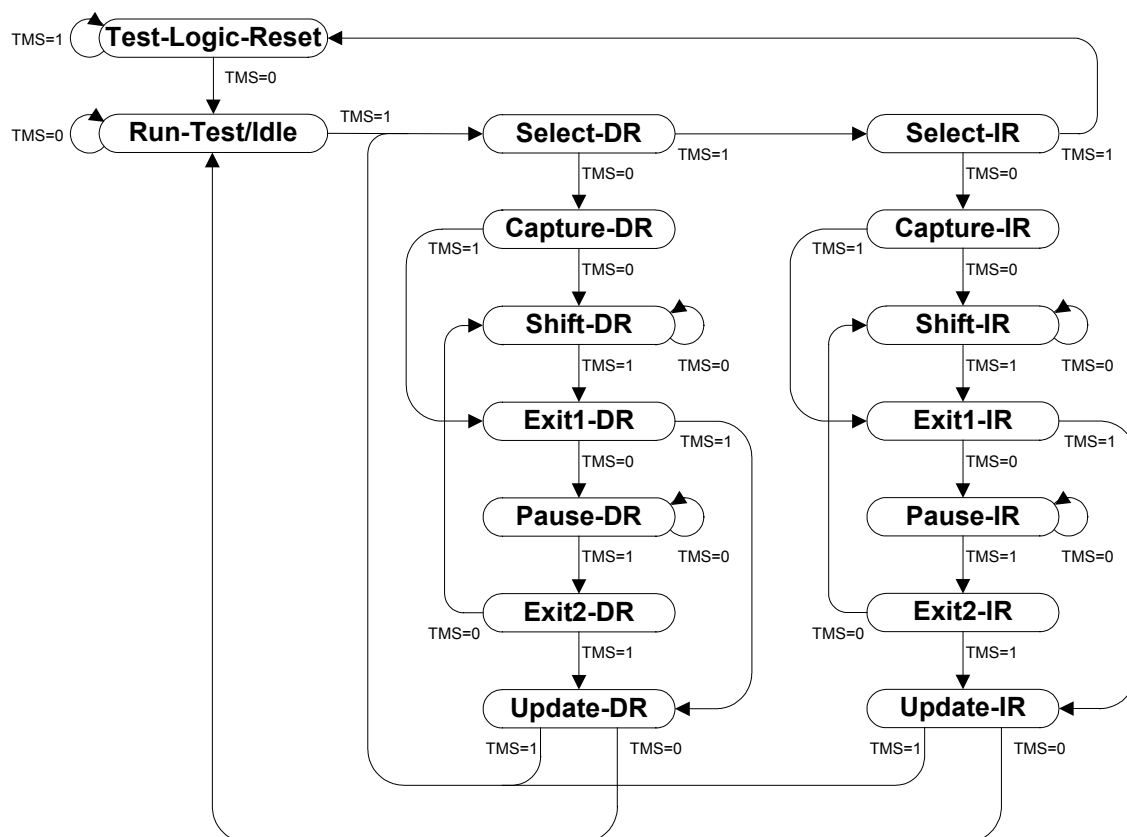


Figure 2—TAP Controller State Diagram

The capture, shift, and update operations for these registers occur during the corresponding states in the TAP state machine. Transitions between states are controlled by the value of the Test Mode Select (TMS) pin during the rising edge of Test Clock (TCK). By definition, a state is entered and exited on the rising edge of TCK; therefore, the first TCK edge seen in the new state is the falling edge. This is important in determining data transfers that may take place in a state. Figure 3 shows the relationship of the TAP states and the acceptable timing windows during which the signals that enable the capture, shift, and update operations are active. A circuit diagram of a single-bit TDR is shown for reference, but multiple shift cycles are included in the event-ordering diagram.

Two styles of TAP Controller designs exist with respect to clocking. The gated-clock style supplies separate Capture, Shift, and Update clocks by gating TCK with the Capture, Shift, and Update states of the TAP Controller. The enabled-clock style supplies TCK along with state signals indicating when the TAP Controller is in each of its states. While it is possible to connect an IEEE 1687 serial access network to either TAP clocking implementation, this document illustrates only the latter—a continuous TCK with separate state signals.

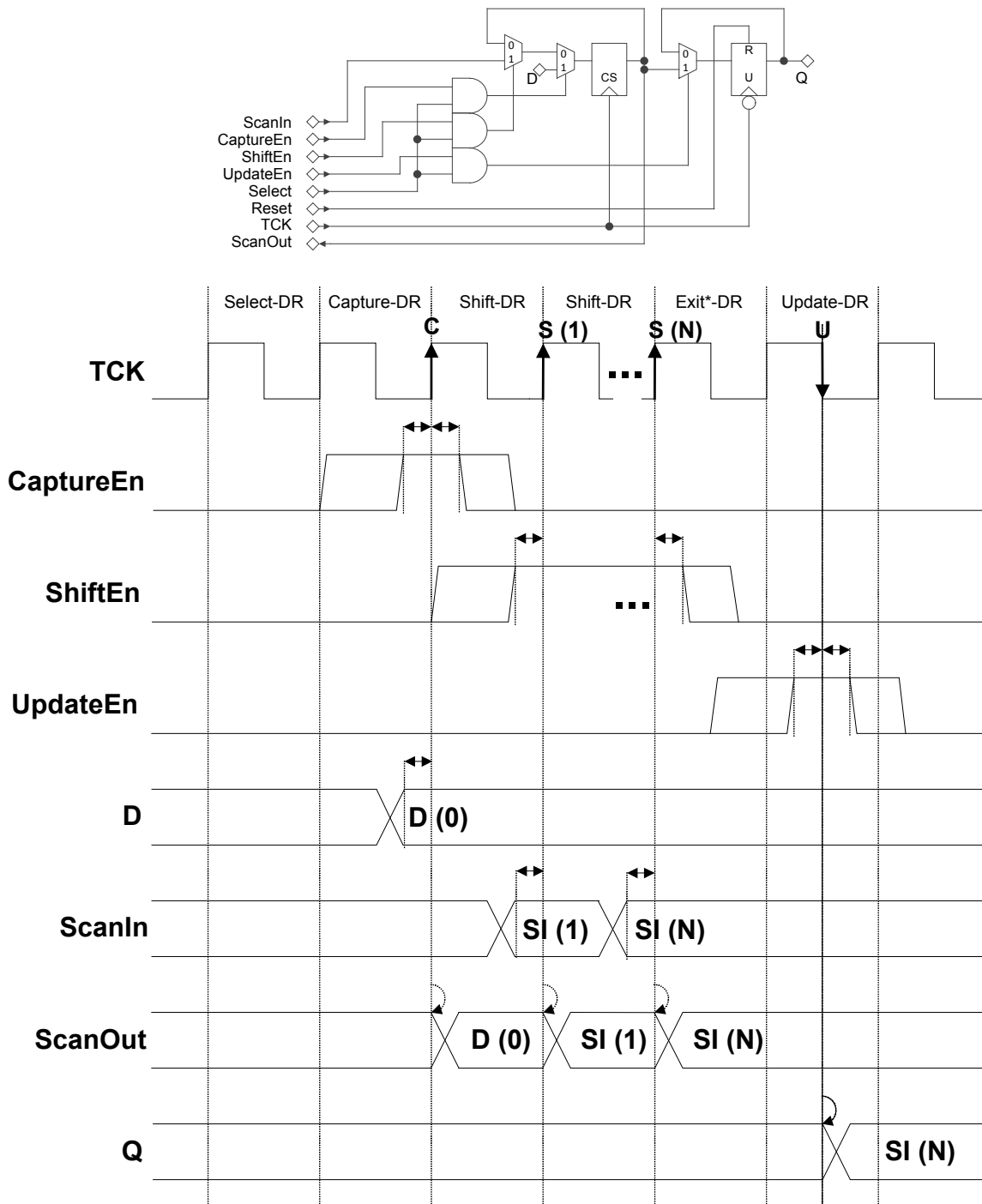


Figure 3—TAP state and enable signal timing

The three main register operations (capture, shift, and update) that are triggered during the corresponding TAP controller states are shown with arrows on the appropriate TCK edge. Space is included for multiple shift operations, with the first (1) and last (N) ScanInput (SI) data bits labeled. The corresponding ScanOut data stream begins with the data (D) captured into the flip-flop labeled “CS” and is followed by the shift data. Minimum setup time and hold time margins are shown with horizontal arrows before and after the appropriate clock edge, respectively. The clock edges after which output data may change are indicated with curved arrows.

4.2.2.1 Test data registers

A very common hardware mechanism through which information is exchanged to and from an instrument is the *test data register* (TDR). Each instruction loaded into the TAP Instruction register may place one, and only one, TDR between the TDI and TDO ports of the IEEE 1149.1 TAP Controller. The actions associated with the TDR that is selected by the current instruction are as follows:

- Data is captured into a TDR cell on the rising edge of TCK when the TAP is in the CAPTURE-DR state.
- Data is transferred from one TDR cell to the next in the serial chain on the rising edge of TCK when the TAP Controller is in the SHIFT-DR state.
- Data is transferred to the Update register of a cell on the falling edge of TCK when the TAP state machine is in the UPDATE-DR state. If a TDR cell is a “Status” cell (i.e., does not supply data to an instrument), then no Update register exists and nothing takes place on this falling edge.

One feature that deserves clarification is that for some single-cycle-long states, such as Capture-DR (and for Shift-DR in the case of a 1-bit long TDR), the edge on which the action occurs (capturing or shifting) is actually coincident with the edge that signifies the exit from the state (as shown by the up-arrows in Figure 3).

4.2.2.2 IEEE 1149.1 test access port and IEEE 1500 timing example

A simpler-to-understand version of the timing diagram in Figure 3, showing fixed timing on TCK edges rather than with the permissible setup and hold windows, is shown in Figure 4 and will be used as a reference throughout the remainder of this document. This timing is consistent with that of the Wrapper Serial Port (WSP) of IEEE Std 1500 and is also valid for IEEE Std 1687. One key difference from Figure 3 is that the TDR shown in the circuit diagram contains a retiming latch (also called a *lock-up latch*) on the ScanOut port. This is a requirement of the ScanOut port of an IEEE 1500-compliant wrapper. The only impact that the presence of this latch has is to delay the time at which the ScanOut data changes by one phase of TCK; logically, the circuits of Figure 3 and Figure 4 work the same way as seen from a TAP connected to them.

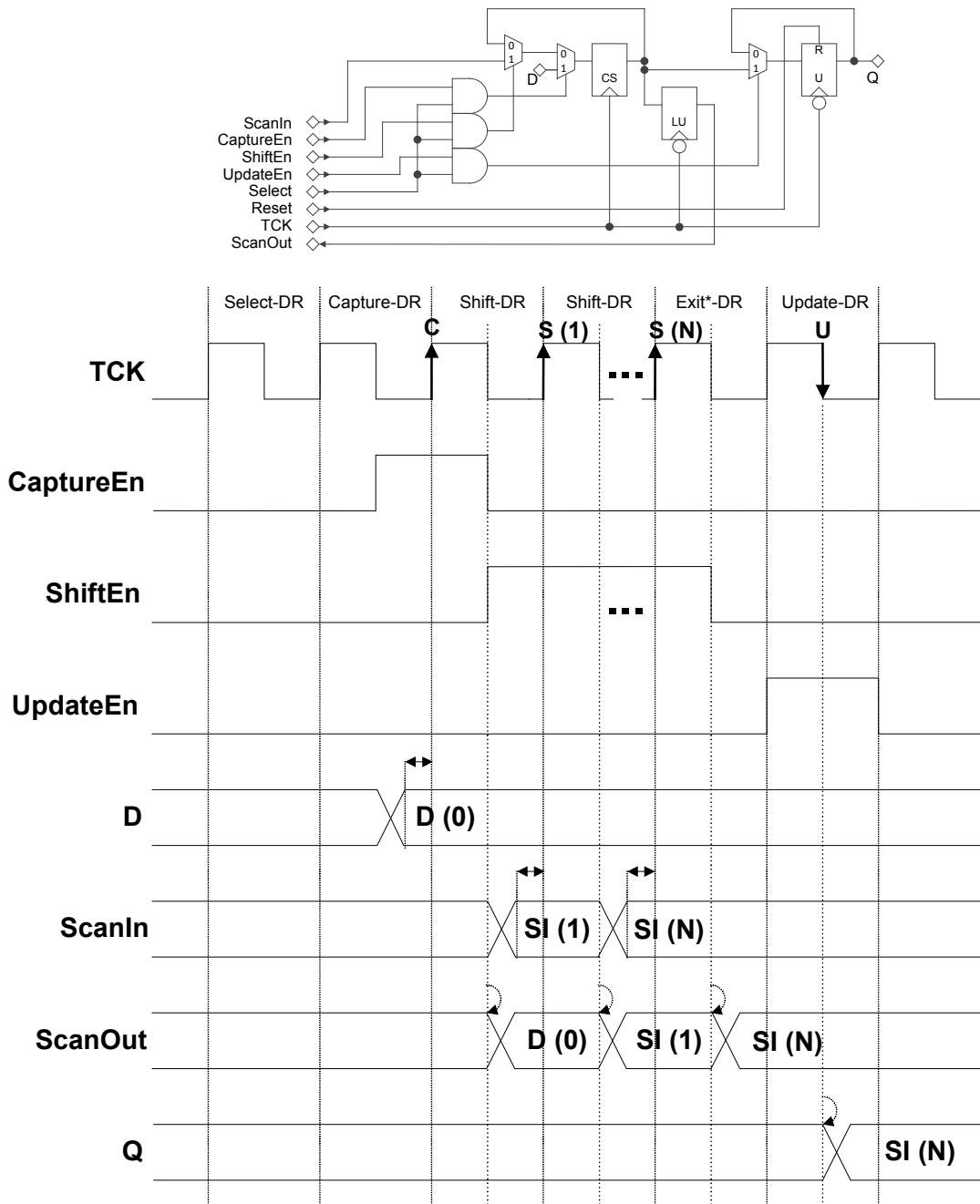


Figure 4—TAP state and enable signal timing, consistent with IEEE Std 1500

4.2.2.3 Reset mechanisms and timing

There are three distinct reset mechanisms in IEEE Std 1687 as follows:

- Asynchronous reset from the TRST* pin
- Synchronous reset that occurs when the TAP is in the test-logic-reset (TLR) state
- Synchronous local reset that may be produced by a TDR.

The timing diagrams in Figure 5 and Figure 6 illustrate the second and third reset mechanisms. The first (asynchronous) mechanism is omitted from the timing diagrams to avoid unnecessary clutter: no matter when TRST* is asserted, it causes an immediate reset.

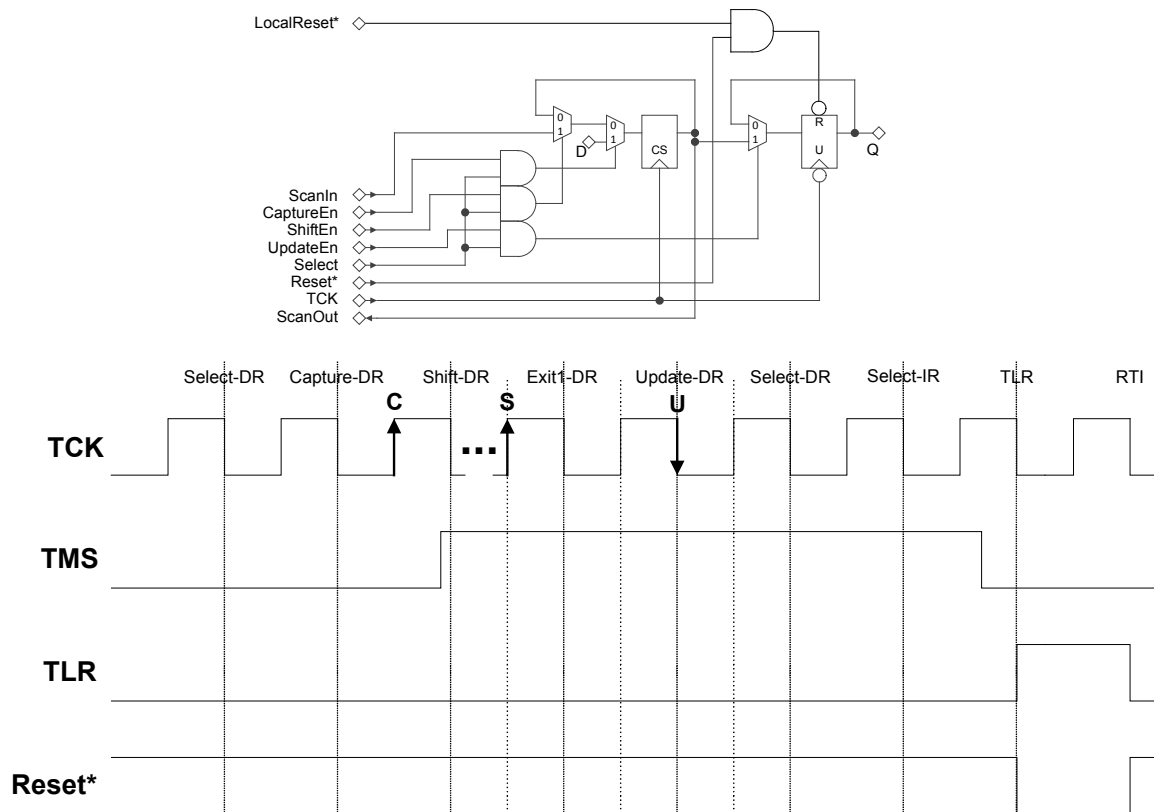


Figure 5—Test-Logic-Reset timing

The duration of the reset event shown in Figure 5 is dependent upon the TMS input; the figure shows a single-cycle spent in the test-logic-reset (TLR) state with an immediate transition to the run-test/idle (RTI) state based on TMS being set to 0.

When a local reset is used (it is optional), the timing of the events is as shown in Figure 6. The salient feature is that the local reset is a self-clearing signal: it returns to its de-asserted value prior to the next time the TDR generating it executes a capture operation.

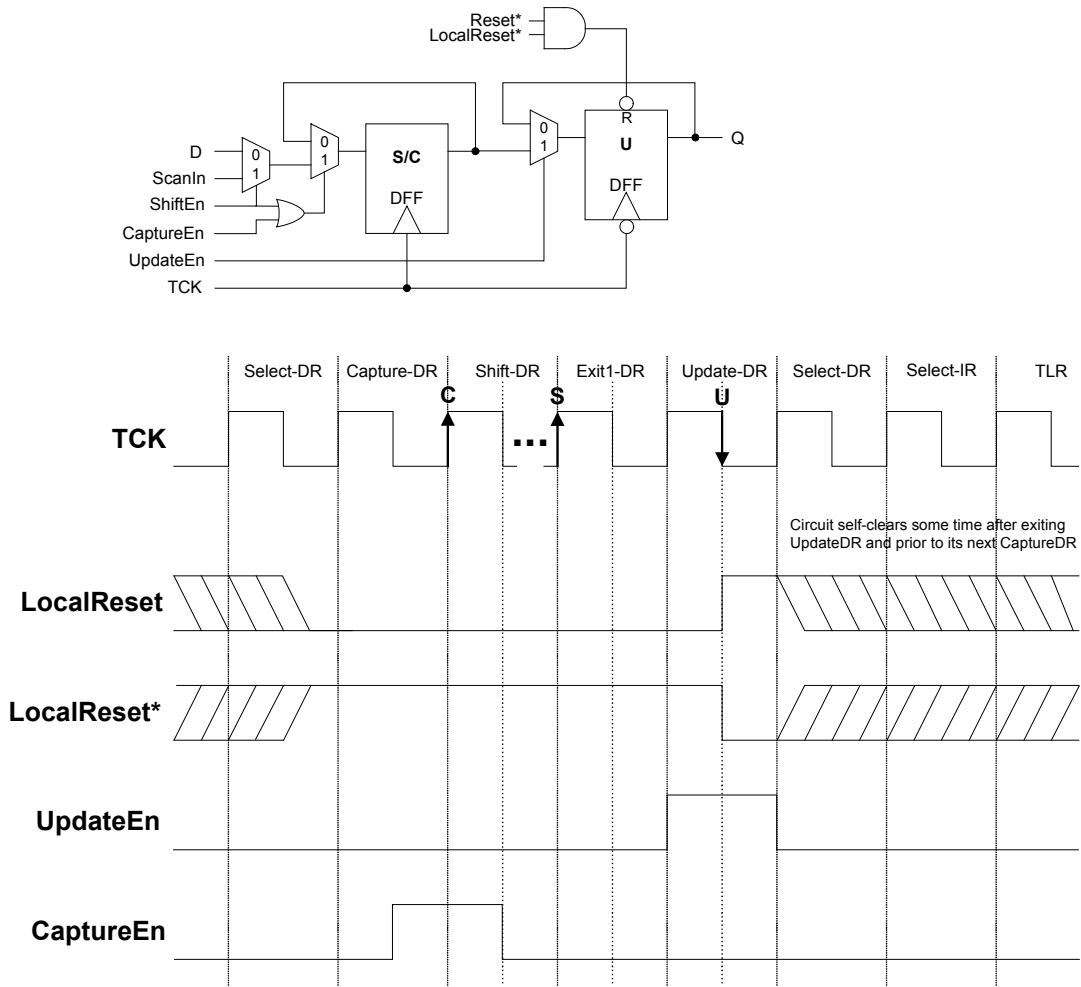


Figure 6—Local Reset Timing

4.3 On-chip instruments

Instruments are circuits embedded into a device intended for a specific control or data-collection purpose. Some examples of embedded instruments are as follows:

- Phase locked loop circuits: generate clocks at a specified frequency
- Clock control blocks: gate and control system clocks
- Memory BIST controllers: perform self-test of arrays of storage elements
- Logic BIST controllers: generate stimulus for and monitor response from logic
- Process monitoring and speed binning circuits: measure the performance of the fabricated device
- EFUSE blocks: program configurable elements of the device by selectively blowing electrical fuses
- I/O bit error rate counters: measure the rate at which transmission errors are received
- I/O eye diagram mapping engines: adjust voltage and timing parameters to measure signal integrity

These instruments interface to the device environment via their I/O ports. Those ports necessary for the operation, control, and data management for these instruments may be connected to an instrument access network via an instrument interface designed for that purpose. While the architecture of the instruments is not dictated, it is assumed that the instrument interface is static and event-driven (meaning that an instrument initiates actions when its inputs are changed and holds its resulting outputs until told to respond to new stimulus when its inputs change again). This document defines the requirements for the design of such instrument interfaces, including two types of “plug-and-play” standard interfaces to which instruments may be joined for compatible connection to the network. This document also defines a mechanism for describing the interconnections of the network, as well as a mechanism for procedurally controlling the network (i.e., sending data and commands to the individual instruments and/or receiving data from the individual instruments).

5. Hardware architecture

5.1 Introduction to the IEEE 1687 network

The basis of the hardware architecture is the instrument access network (simply “network” henceforth), which connects one or more device interfaces to one or more instruments. A device is said to contain one network (described in ICL); that network may contain multiple (possibly disjoint) portions (or sub-networks). A compliant network consists of at least one module, which may include logical hierarchy (i.e., the top-level module may have child instances, and these instances may themselves be parents of other children, and so on).

IEEE Std 1687 facilitates access from the chip boundary to instruments embedded within the device. Achieving this access requires at least three portions of an access architecture: (1) the device interface; (2) the instrument interface; and (3) the access network that resides between the device interface and the instrument interface. IEEE Std 1687 covers the mechanisms to build, describe, and operate those three architectural elements. IEEE Std 1687 can be connected to many types of controllers and interfaces, but pays particular attention to the very popular serial access networks used in IEEE Std 1149.1 and IEEE Std 1500. The signals available to operate the IEEE 1687 serial access network are shown in Table 1.

Table 1—Signals available to operate a serial access network

Signal name	Function
ScanIn	Serial data input
ScanOut	Serial data output
CaptureEn	Enables the shift storage elements to acquire data from their parallel inputs
ShiftEn	Enables the serial scan chain to shift
UpdateEn	Enables the update storage elements to copy data from the shift storage elements and present it on their parallel outputs
Select	Activates the interface
Reset	Reset the update storage elements to the reset values (if specified)
TCK	Test clock to which the capture, shift, and update operations are synchronized

5.2 Hierarchical IEEE 1687 networks

The notion of hierarchy is important for IEEE 1687 networks and has two dimensions: network hierarchy (in the host-client sense) and logical hierarchy (in the parent-child sense). The components of an IEEE 1687 network are described in one or more modules. An example of a single-module network with endpoints included is shown in Figure 7.

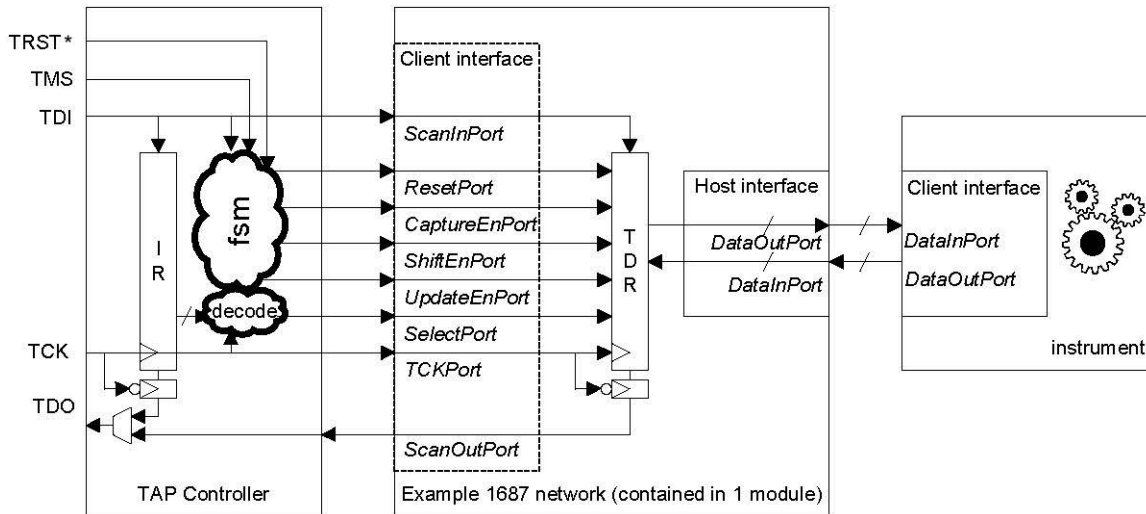


Figure 7—Single-module network architecture example

The convention used throughout this document is to represent the device interface (e.g., the TAP) to the left of the network and the instrument(s) to the right. Accordingly, the network has two symbolic interfaces: a client interface on the left side that interacts with the controller, and a host interface on the right side that interacts with the instrument(s). Likewise, each module within the network has a symbolic left-hand side (LHS, facing its host module, ultimately the controller), and a symbolic right-hand side (RHS, facing its client module, ultimately the instrument). The LHS interface of a module is called its *client interface*; the RHS interface of a module is called its *host interface*.

Two important concepts apply to host and client interfaces in IEEE Std 1687. First, instruments at the lowest level of integration usable in IEEE Std 1687 have only client interfaces, and those client interfaces consist of only “data” signals (though the actual function of those signals may be control or status or actual data). Second, some interfaces that contain specific sets of signals are given the label “plug and play” and called out as entities that establish specific types of compliance (see 5.16). These two concepts come together for an entity called a “wrapped instrument with a plug-and-play interface,” which is a common method to deliver an instrument from an IP provider to an integrator.

Modules may be organized in a host-client fashion. A portion of a multiple-module network is shown in Figure 8, with its host and client interfaces highlighted. The internal contents of each of the modules (labelled “reconfiguration logic” in Figure 6) represent the components that comprise a portion of the network; these will be described in subsequent subclauses in Clause 5 and Clause 6.

From the perspective of logical hierarchy, all the modules in Figure 8 are siblings. It is also common to have logical hierarchy within an IEEE 1687 network, as shown in Figure 9. The boxes labeled A through I represent modules that contain elements of the IEEE 1687 network such as registers and muxes and wires (not shown).

The use of logical hierarchy in the network description supports IP reuse, partitioning, and, of course, modularity (which is widely used in modern HDLs as well as software languages).

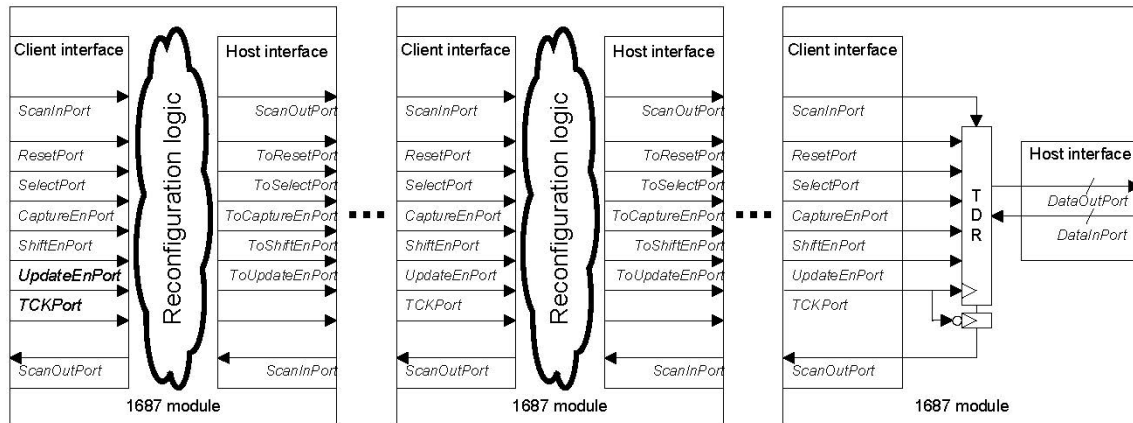


Figure 8—Multiple-module instrument access network

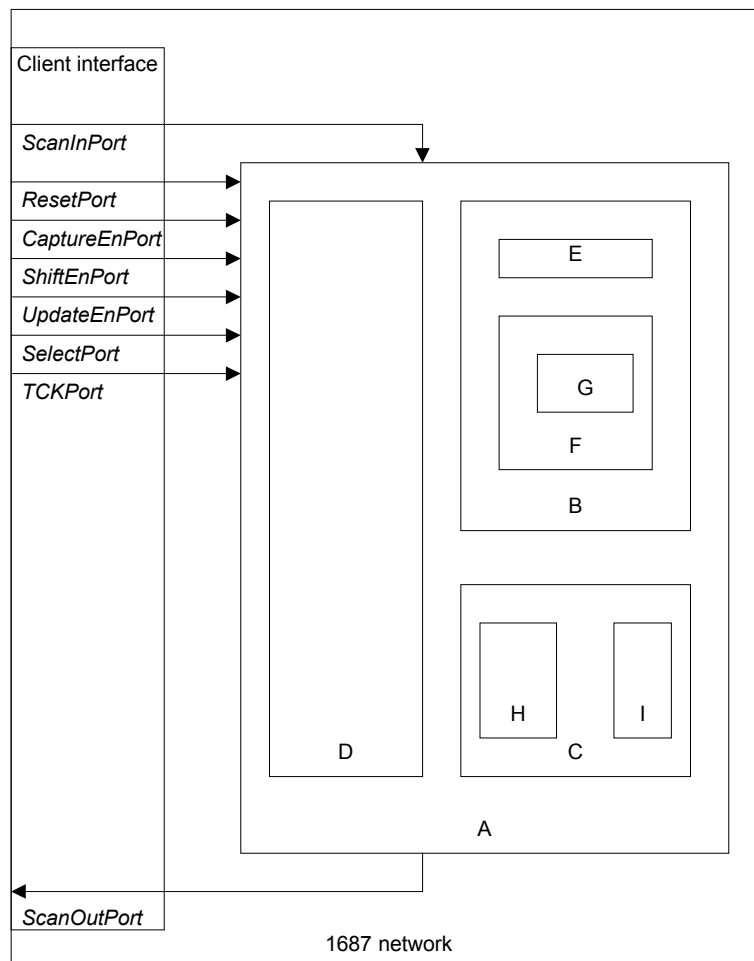


Figure 9—IEEE 1687 network with logical hierarchy

5.3 Controller

A controller generates the operation signals for the access network and may be as simple as direct connection of the operation signals of the access network to the chip's pin boundary, or may be as complex as a controller containing state machines and selection logic (such as the IEEE 1149.1 TAP and TAP Controller) or even a complete microcontroller or microprocessor.

IEEE Std 1687 defines and illustrates how to connect the access network to an IEEE 1149.1 Joint Test Action Group (JTAG) controller as in Figure 10, but also enables other controller types. The signals and protocols generated by the controller (independent of its type) are illustrated throughout Clause 4 and Clause 5 and are directly compatible with IEEE 1149.1.

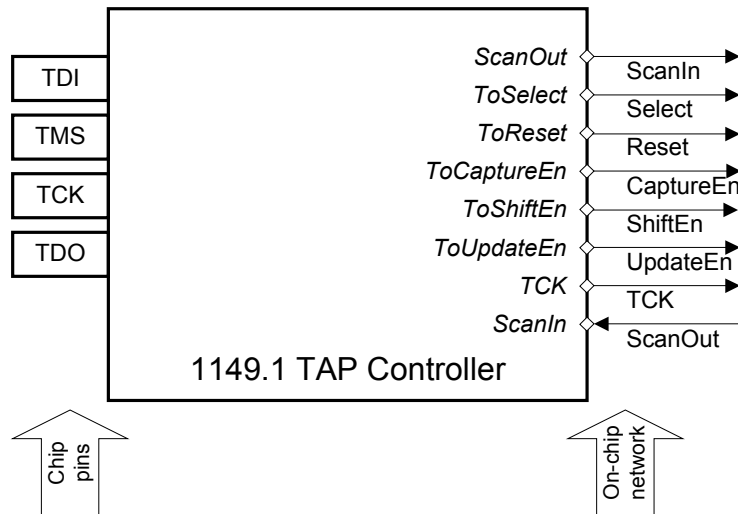


Figure 10—Example controller and controller signals to operate an access network

The IEEE 1149.1 Controller operates by converting clock (TCK) and control (TMS) inputs and a serial data (TDI) input at the chip's pin boundary into signals that can be used to select, access, and operate embedded registers and logic inside the chip. Data from inside the chip can be observed outside of the chip through a serial data (TDO) output pin. In general, an instruction encoding installed in the IEEE 1149.1's Instruction Register selects a TDR to be in the serial scan chain and provides all of the signals necessary to configure and operate the TDR. For the case of IEEE Std 1687, an instruction, known as the AccessLink Instruction, selects a portion of the network to be the active scan chain connected between TDI and TDO; and the Controller provides operation signals that sequence in accordance with the IEEE 1149.1 finite state-machine (FSM) for the selected scan chain. More than one AccessLink Instruction may exist to select independent portions of the network, and more than one AccessLink statement may exist to associate different controllers with the network.

5.4 Instrument interface

The instrument interface for IEEE 1687 purposes is defined as the signals on the instrument that are required to operate and control the instrument. For example, Figure 11 shows a Memory BIST instrument and its target, a memory array – and note that the memory array has a selection option of connecting to the Memory BIST instrument or to functional operation signals. There are many signals shown in the diagram of the instrument system, but for this example, only the Reset, Start, Stop, Done, and Fail signals are required to configure, operate and observe the instrument. The other chip-level signals, the functional data and address bus, the functional read/write, the functional clock; and the internal “instrument”-to-

“instrument’s target” signals, the BIST_R/W, BIST_En, BIST_Data, and BIST Address – do not need to be “accessed by an IEEE 1687 instrument access network” in order to operate the instrument.

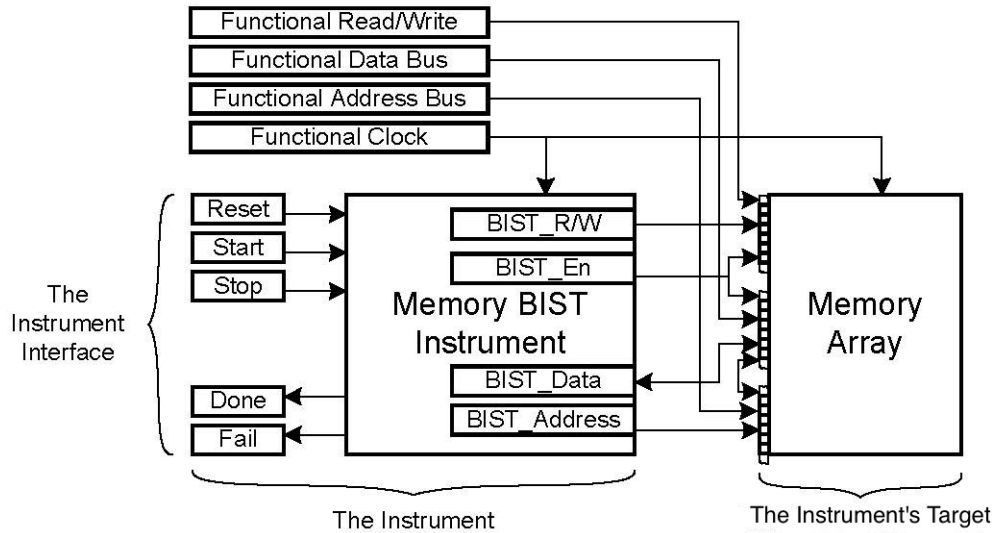


Figure 11—Instrument and target of instrument

In many cases, these other non-accessed signals are meant to remain hidden and may be considered proprietary. In addition, the internal structure of the instrument itself may be considered proprietary. When an instrument is delivered from a provider in a standalone manner (often called a *raw instrument*), it is delivered with documentation that identifies all of the signals and their manner of connection (which may include, among other items, a Verilog RTL model, a CTL file, timing/voltage/power specifications, etc.) to the chip integrator. After integration into a device, the detailed information about the instrument and its connections may be considered proprietary. Therefore, for IEEE Std 1687, only the limited subset of signals required to configure, operate, and observe the instrument needs to be separately and specifically identified and documented in ICL. The signals on a raw instrument (an instrument with no IEEE 1687 network interface) are represented by generic input signals known as DataIn and generic output signals known as DataOut.

5.5 Access network

Given the defined endpoints of the controller on one side and the instrument interface on the other, the access network is defined as everything in between. This network consists of architectural elements including scan chains, registers, muxes, control signals, etc. The network may be represented by many possible configurations to be of optimal value for any given implementation. The IEEE 1687 access network is not meant to be an overly restrictive architecture that limits the possible connections to instruments. It is meant to solve the problems of access optimization, instrument scheduling, and implementation in accordance with engineering budgets. Different network configurations allow for active scan chain length management, flexible access to instruments, and minimization of the registers needed or routes involved with the operation signals. The following paragraphs describe common network configurations: static, locally reconfigurable, remotely reconfigurable, and configurable via segment insertion or path selection.

Figure 12 shows the concept of a single instrument IEEE 1687 network where the controller is on the left, the Instrument Interfacing Test Data Register (TDR) is in the middle, and the Instrument is on the right. The data path from the controller to the TDR is a serial scan chain that, for this example, operates in accordance with the IEEE 1149.1 FSM in the TAP Controller. The data is delivered to the TDR on ScanIn signals that ultimately source from the IEEE 1149.1 TAP’s TDI, and the data is returned to the TAP

Controller through ScanOut signals that ultimately connect to the TAP's TDO. By convention in this document, scan data is presented with the least significant bit (LSB) as the rightmost bit that is physically closest to the output scan port. Therefore, scan data always shifts visually from left-to-right. The TDR is a set of IEEE 1149.1 Scan Cells that contain both a shift-side, and a ripple-free update side. A "shift-in and update" writes to the instrument's parallel DataIn signals, and a "capture and shift-out" reads the instrument's parallel DataOut signals. This instrument access network is selected by the installation of an instruction in the IEEE 1149.1 Instruction Register (IR), which is documented by the AccessLink statement in ICL.

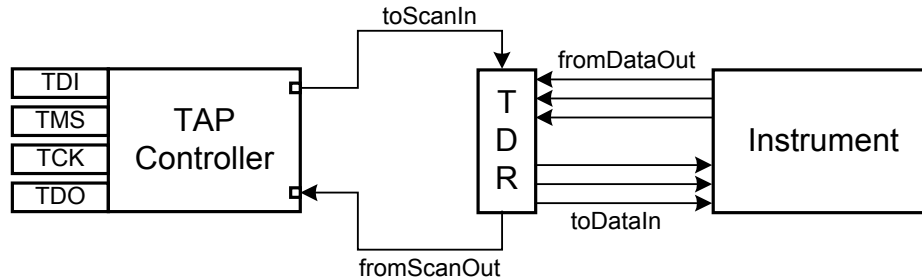


Figure 12—Conceptual example of a single instrument IEEE 1687 network

Figure 13 shows another conceptual IEEE 1687 network configuration that includes two instruments accessible by the serial scan chain from the controller. The TDR shown has two register fields, each of which communicates with a different instrument (A and B). Since both register fields are on the same serial scan chain they may be viewed as one contiguous TDR, which is consistent with the definition of a TDR as being the entire collection of the scan flops connected between TDI and TDO. However, the point of view typically taken by the IEEE 1687 architecture separates TDRs into their component register fields (conventionally referred to as ScanRegisters or simply registers) as a more natural method to interface with each instrument independently.

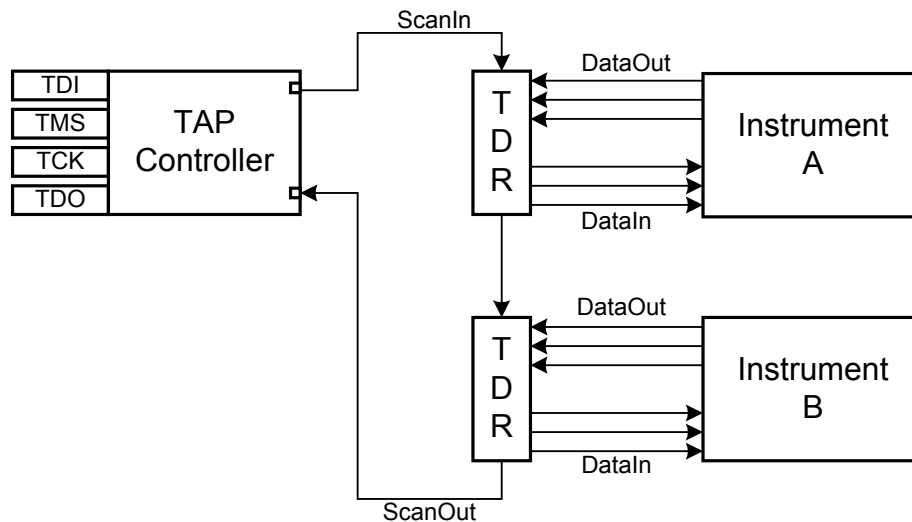


Figure 13—Conceptual multiple instrument single scan chain IEEE 1687 network

5.5.1 Reconfigurable scan chains

It is expected that some instruments may be provided as IP units from different suppliers. The IP may be delivered wrapped and complete with the TDR that contains the scan chain and interface signals to and from the instrument, or the IP may be delivered raw with only its DataIn and DataOut ports. Either an automation tool or a human integrator will incorporate the different IP into an overall chip-level scan chain.

In some cases, multiple complex networks containing tens or hundreds of instruments may need to be incorporated into the overall chip-level IEEE 1687 network. In order to avoid having extremely long scan-paths resulting from the daisy-chaining of the interface TDRs of all the instruments, other connection options are also supported.

One such connection option is to allow registers to be added to and subtracted from the active scan chain without requiring tens or hundreds of instruction encodings in the IEEE 1149.1 TAP Controller. Not only is this inefficient, but the TAP Controller may not exist at the time the instruments and network are created. In this case, the IP provider may need to create an optimized network architecture with a flexible configuration control mechanism. There are multiple methods that can be used to accomplish this.

Figure 14 shows a conceptual IEEE 1687 network configuration that supports multiple instruments and includes an element known as a Segment-Insertion-Bit (SIB) to allow the overall scan chain to be of variable length. In this example, when asserted, the SIB causes the scan chain to expand to include (and allow hierarchical access to) the register that provides the instrument interface to Instrument C. When de-asserted, the SIB will return to its closed configuration, which bypasses this register and instead provides a shortcut (as short as one bit) to this segment of the scan chain. In this de-asserted mode, the SIB can be thought of as being a single-bit bypass register for a wrapped instrument (e.g., the combination of the bottom-most TDR with the instrument labeled C in Figure 14) or even an entire network of instruments. For illustrative purposes, the convention of the SIB opening the associated segment when asserted will be used in these examples, but a SIB could be built with the opposite behavior.

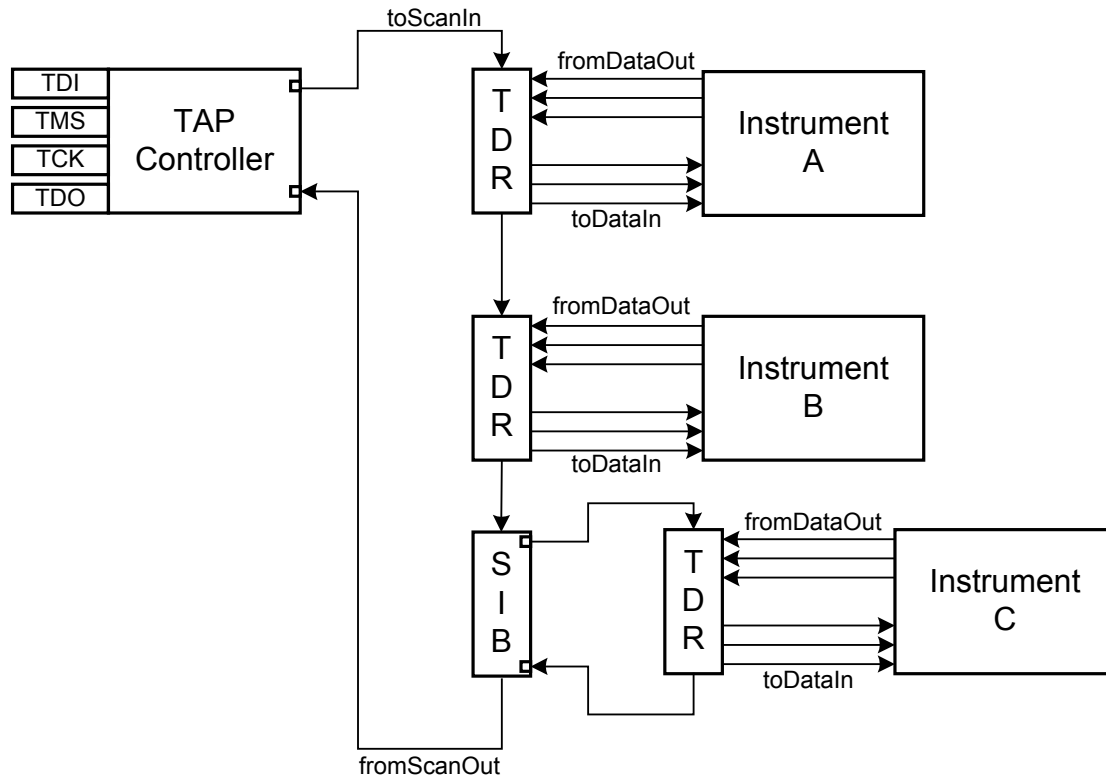


Figure 14—Conceptual multiple instrument variable-length scan chain IEEE 1687 network

An expanded view of an example SIB configuration is shown in Figure 15. This shows how a shift-update cell can be coupled in-line with and adjacent to a multiplexer to allow the selection of an alternate scan chain. The normal scan chain when the SIB is closed is from the TDI of the TAP to the shift-bit of the SIB and from the shift-bit of the SIB to Instrument A's TDR and then back to the TDO. The update-bit of the

SIB defaults to the de-assert value (e.g., Logic-0), which propagates through the ToSelect output signal from the SIB to disable the operation control signals of the TDRs attached to instruments B and C. If an assert value is shifted into the SIB's shift-bit and an update operation is conducted, then the ToSelect signal enables the Instrument B and Instrument C TDR control signals. When the IEEE 1149.1 state machine next enters the Capture and then Shift states, the Instrument B and C TDRs will operate and they will take the position of being in front of the SIB's shift-bit in the active scan chain. Note that placing the multiplexer in front of the shift register in the SIB configuration is a best practice in that it prevents the formation of long combinational paths with deep hierarchy, which could happen if the multiplexer were placed after the shift-bit. Also note that placing a Delay Bit after the Update Bit for the generation of the ToSelect signal is a best practice to prevent a race condition where the SIB receives its update and (prematurely) deselects the downstream TDRs before they have a chance to update. Please consult Annex F for detailed explanations and design guidance for implementation of SIBs.

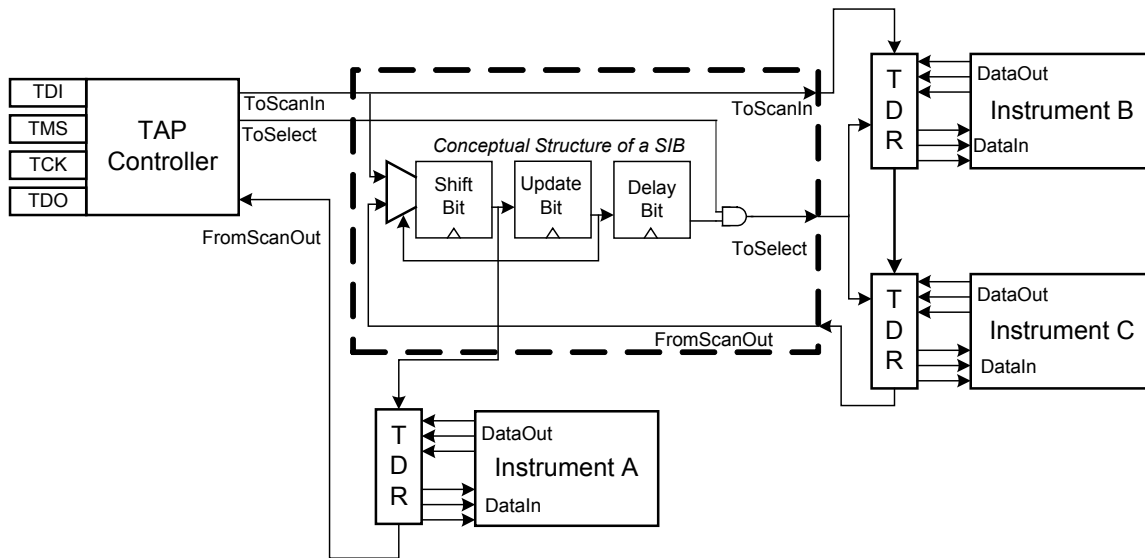


Figure 15—SIB configuration accessing an alternate scan chain

The in-line SIB contains both the scan multiplexer and the register bit connected to the control input of that mux, and this register bit is included in the same scan chain as the segment that it inserts (i.e., it is “in-line” with the segment). Figure 16 presents an alternative approach, where the SIB is exchanged with a mux only. The control input to the mux is connected to a TDR that is not part of the same scan chain as the elements it is selecting. This is referred to as remotely controlled scan mux architecture. The source of the control bit can be either another TDR (shown at the top of the figure where TDR0 produces Select1A to place TDR1A into scan chain 1) or a decoded instruction from the TAP (shown at the bottom of the figure where Select2A places TDR2A into scan chain 2).

Both the locally controlled (SIB) and remotely controlled scan mux configurations illustrate the concept of “opening” a network. The SIB configuration only adds a scan segment after the appropriate scan mux control value has been shifted in and then updated. For example, in the IEEE 1149.1 FSM, the sequence from SelectDR-to-UpdateDR (known as a ScanDR) would be used. In that context, an initial ScanDR is required to open a SIB, then subsequent ScanDRs can operate the newly opened scan chain. Opening nested SIBs is a sequential process: a SIB nested deep in the network cannot be operated until all the previous SIBs gating its access have been opened. The act of opening or closing a SIB is known as a “configuration operation” and is referred to as “reconfiguring the network.”

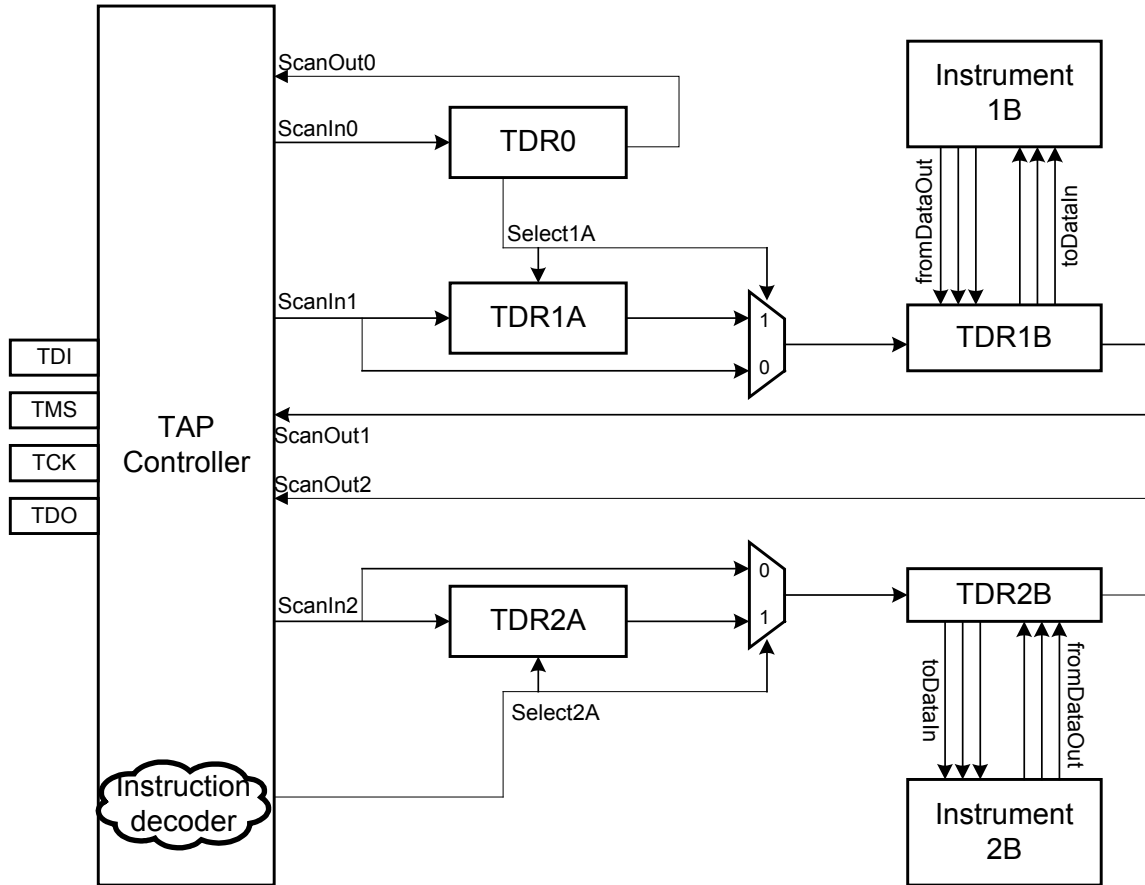


Figure 16—Remotely controlled scan muxes: two variants

Figure 17 shows an example of an IEEE 1687 network that has two levels of hierarchy—there are two SIBs in level one that make a 2-bit scan chain from the TAP Controller’s TDI to TDO. When the first SIB from TDI is opened, then the TDR Interface to instrument A is accessible as well as the SIB that provides access to instrument B. This SIB is in level two and can be opened when another ScanDR is conducted after the level-one SIB is opened. Closing a SIB can also be done by conducting a configuration operation, and placing the de-assert value in the SIB after an update, or it can be done by asserting a reset operation. The SIBs will revert to the closed configuration when a reset occurs. Note that when a SIB closes, but not in conjunction with a reset, it de-asserts the ToSelect signal from the network segment to which it provides access. This in turn “freezes” that segment of the scan chain to its current value (making the segment persistent). To change the value in the segment of the scan chain requires either a reset or opening the SIB and changing the value through a capture or scan-in operation followed by an update.

Another connection option is to swap one segment of an active scan chain for another. Each scan chain segment may be of the same length or they may be of different lengths; there is simply a trade-off of access time versus access length. This concept can be expanded to allow many mutually exclusive selectable scan chains to exist in parallel with, for example, one or more instruments contained within the active scan chain at any given time.

Figure 18 shows a conceptual example of an IEEE 1687 network configuration that supports a split, mutually-exclusive, selectable scan chain. An in-line ScanMux control bit (S1) may be used to select the TDR associated with either Instrument A or Instrument B; note that the default configuration is required to be defined for when the network is first accessed or after a reset operation occurs.

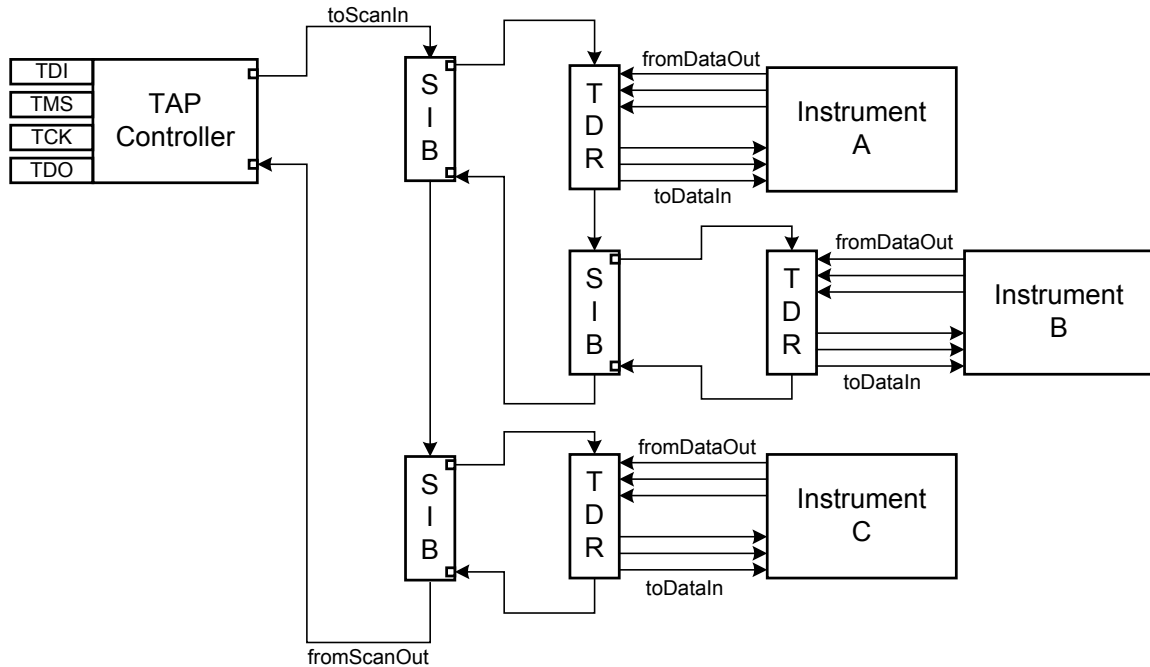


Figure 17—Multiple SIB configuration with hierarchical depth

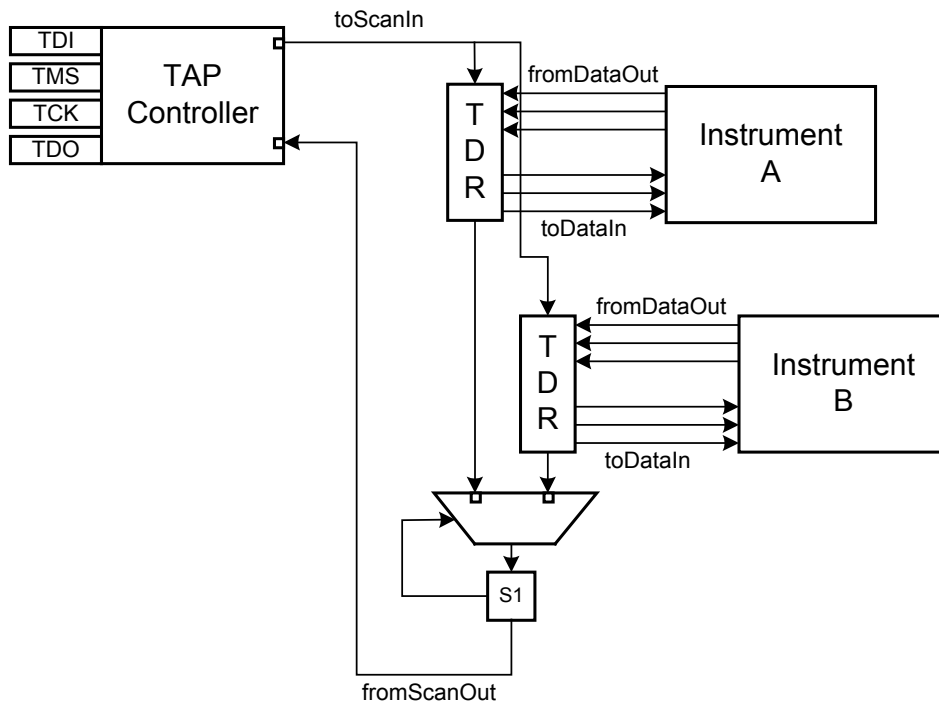


Figure 18—Conceptual multiple instrument split scan chain IEEE 1687 network

Figure 19 shows a combination of both techniques that allows instrument C to be included with either instrument A or instrument B depending on the state of the update-bit in the ScanMux control bit S1 and in the SIB configuration.

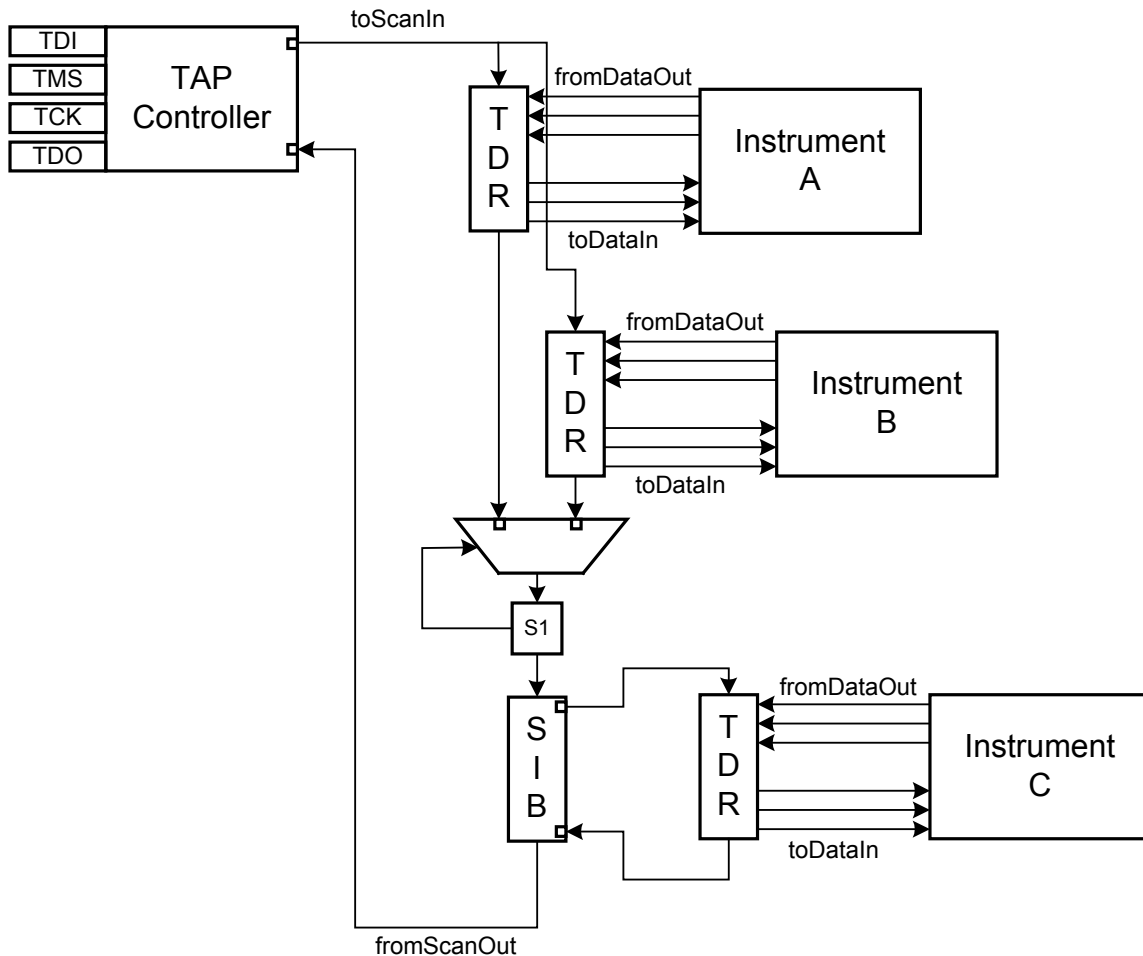


**Figure 19—Conceptual multiple instrument swap and add
IEEE 1687 scan chain network**

5.5.2 Broadcast scan

Scan chains need not be limited to point-to-point serial connections. The concept of broadcast scan, where a point in the scan chain fans out to drive multiple downstream scan segments in parallel, is also supported in IEEE Std 1687. This configuration allows a reduction in the time spent loading scan segments intended to receive identical data. Figure 20 shows an example of a scan broadcast configuration. The structure involves a group of instances, each of which has a uniquely identified scan interface, which share a common scan data input and can be selected simultaneously. That common scan input happens to come from a ScanRegister (marked “BC” in the figure) whose update stage controls the scan multiplexers, but that ScanRegister could just as easily have been at the end of the scan chain as the beginning, or even been on a completely different scan chain.

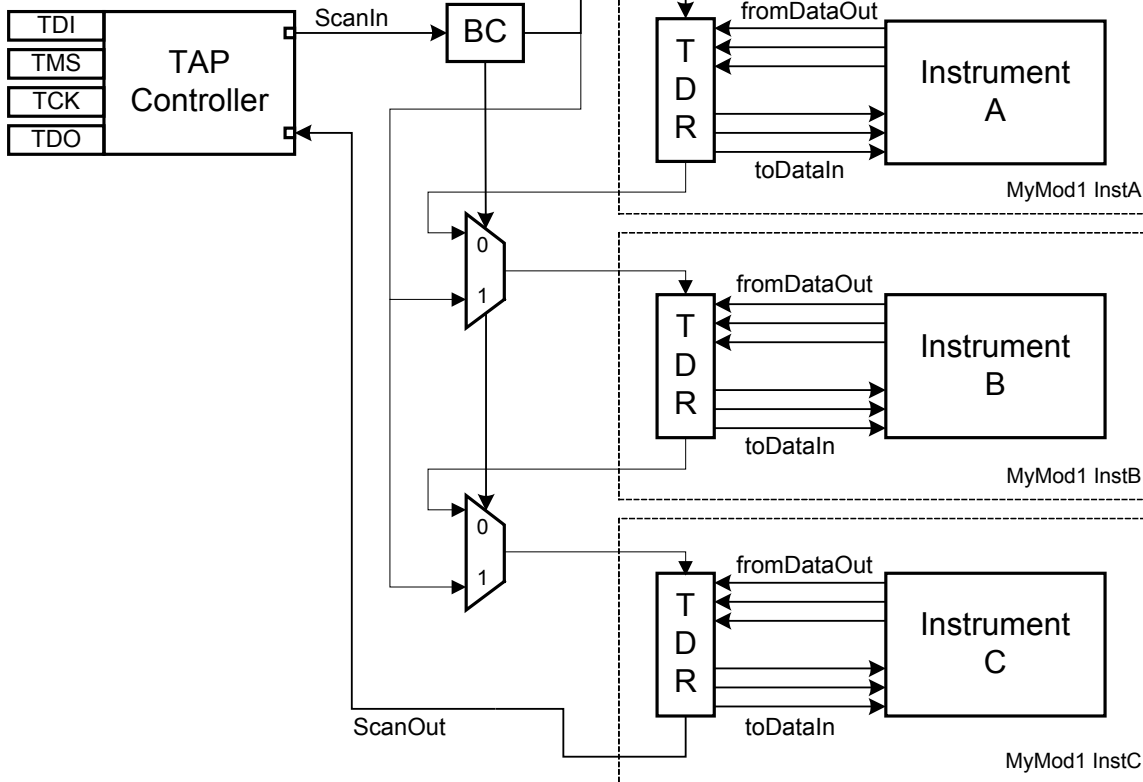


Figure 20—Broadcast scan example

It is important to note that the notion of scan broadcast applies only to data being written to (i.e., serially loaded into) scan chains. While broadcast mode is active, the data being read from (i.e., serially unloaded from) a scan broadcast group comes from one and only one of the instances in the broadcast group (Instrument C in the example in Figure 20, since it happens to be the one connected to be on the active scan chain when the broadcast control bit BC is active). When broadcast mode is not active, each of the instances is accessible for read operations (in one long chain of Instrument A, Instrument B, Instrument C in Figure 20). Note that the scan broadcast configuration in this example has some unusual features (e.g., there is no capability to read each instrument individually). A more common broadcast scan configuration is shown in Annex E.

Each of these IEEE 1687 network configurations presented allow different engineering trade-offs to be made and different scheduling options to be taken. These examples illustrate just some of the IEEE 1687 network configurations that are allowable IEEE 1687 networks.

5.6 Test data register

The IEEE 1149.1 Test Controller (TAP Controller) is just one possible controller that may operate an IEEE 1687 network. If used, it naturally generates the signals required to control, configure, operate, and observe the TDRs as originally shown in Figure 10. The TDRs provide elements that conduct both the network control and configuration through ScanMux elements and the control and observation of instruments through the instrument interface element. These TDRs are very similar to some of the defined IEEE 1149.1 cells but are a much simpler subset in that they only need to conduct the Read, Write, or Select operations.

An example of a conceptual 3-bit Write-Only TDR, with only the data-path elements and data signals shown to indicate how the serial data flows through the system and is applied to the instrument, is represented in Figure 21. Note that the scan chain numbering is generally from zero and is indexed with zero being at the ScanOut end and incrementing 1 integer per bit until the ScanIn is reached. The ScanIn signal shown feeding Shift-Bit-2 is ultimately sourced from the TDI signal from the chip's TAP when an appropriate IEEE 1687 network instruction is installed and asserted in the IEEE 1149.1 Instruction Register (IR). In a conventional IEEE 1149.1 operation, the scan data flows from the ScanIn toward the ScanOut by one bit on each rising-edge of TCK when the IEEE 1149.1 State Machine is in the ShiftDR state. The scan operation creates data ripple on the output of the Shift-Cells as the data moves through the serial scan register; however, the Update-Cell prevents the rippling scan data from being seen on the parallel toDataIn signals. The data in the Shift-Cells is transferred to the Update-Cells on the falling-edge of TCK when the IEEE 1149.1 State Machine is in the UpdateDR state. When this data transfer occurs, the toDataIn signals reflect the logic values contained within the respective Update-Cells. The sequence of a shift-in operation followed by an update operation constitutes a write operation if the update bits are connected to an instrument interface, and constitutes a select operation if the update bits are connected to other network elements such as multiplexers or gating functions

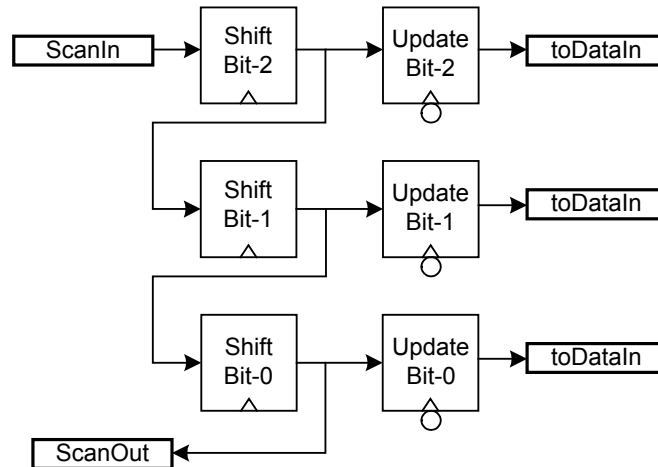


Figure 21—Example conceptual diagram of a Write-Only TDR

A conceptual example of a 3-bit Read-Only TDR is shown in Figure 22. Only the scan data path elements and instrument data signals shown to indicate how the serial data flows through the circuit and is captured from the instrument. The Read-Only TDR is comprised only of Capture-Shift bits with no Update bits. The Read-Only TDR conducts a read operation in two stages. In the first stage, instrument data that is present on fromDataOut signals is captured on the rising-edge of TCK when the state machine is leaving the Capture state. In the second stage, the captured data is shifted out for observation. The ScanIn signal shown is ultimately sourced from the TDI signal from the TAP when an appropriate IEEE 1687 network instruction is installed in the IEEE 1149.1 Instruction Register (IR). The capture operation occurs before the shift operation in the IEEE 1149.1 state machine. In a conventional IEEE 1149.1 TDR scan operation, the scan data flows from the toScanIn toward the fromScanOut by one bit on each rising-edge of TCK while the IEEE 1149.1 State Machine is in the ShiftDR state.

A conceptual example of a 3-bit Read-Write TDR is shown in Figure 23. Only the scan data path elements and instrument data signals shown to indicate how the serial data flows through the circuit and is captured from or applied to the instrument. The Read-Write TDR conducts a read by capturing instrument data presented on fromDataOut signals and then shifting the captured data out—this same shift operation can bring in new data to be written as a simultaneous shift-in. Figure 23 also shows that the concept that Write-Only bits, Read-Only bits, and Read-Write bits may be mixed in the TDR.

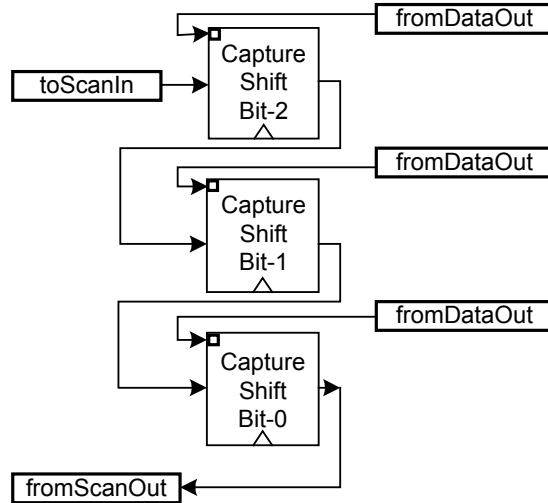


Figure 22—Example conceptual diagram of a Read-Only TDR

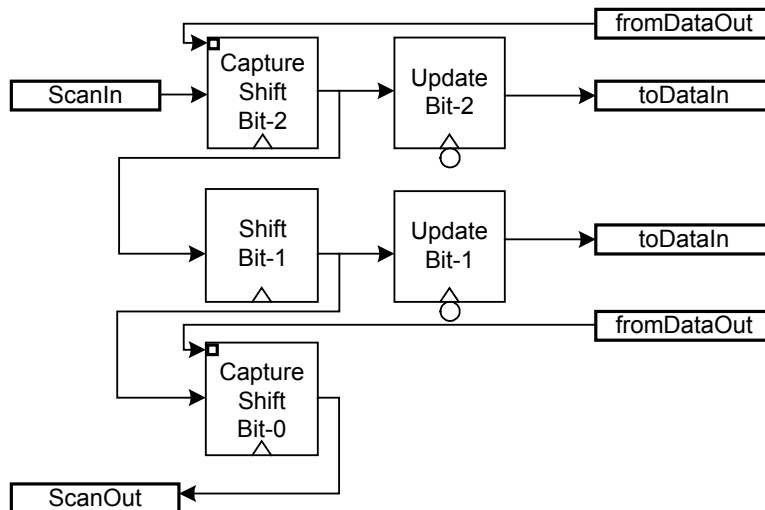


Figure 23—Example diagram of a Read-Write TDR

An instrument interface does not need to be the exclusive controller of an instrument. Functional signals may also access and operate instruments. A TDR that supports a functional connection is shown in Figure 24 where either bit-2 can drive a data input to the instrument or a functional signal (“Func_In”) can drive the data input. In IEEE 1149.1 terms, there would be an instruction encoding in the instruction register that selects whether a TDR addresses the instrument or allows functional signals to pass through (e.g., the boundary scan register). An IEEE 1687 network allows in-line data-multiplexer select bits so the data passing through the scan chain may configure the very same scan chain. This is one of the main differences between older versions of IEEE Std 1149.1 (-1990, a-1993, b-1994, -2001) and both IEEE Std 1687 and the new IEEE Std 1149.1-2013. The older versions of IEEE Std 1149.1 always required traversal of the IR side of the TAP state machine to reconfigure the active scan chain.

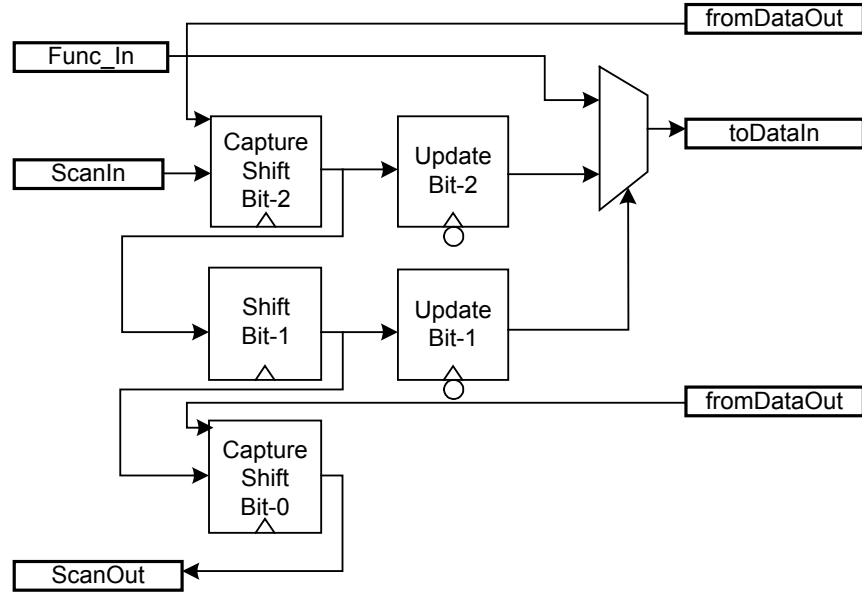


Figure 24—Example 3-bit Read-Write TDR that allows direct functional access

The TDR can be made of Read-Only bits, Write-Only bits, and Read-Write bits. TDRs supporting such bits are required to be able to conduct the capture, shift, and update operations when directed to do so from either the operation of the raw control signal or the TAP state-machine. The ability to support these functions is facilitated by the delivery to each TDR bit of both data, such as the ScanIn and ScanOut discussed earlier, and the control signals, as shown in Figure 25. The control signals line up with the various required operations: the ability to shift data from ScanIn to ScanOut is enabled by a shift-enable control signal (ShiftEn); the ability to capture data is enabled by a capture-enable control signal (CaptureEn); the ability to present and apply data is enabled by an update-enable control signal (UpdateEn); and the ability to reset the apply portion of the bit is enabled by a reset control signal (and that same signal can be used to optionally reset the shift-capture bit as well as the mandatory reset of the update bit). The actions are all synchronized to the rising or falling edge of the test clock.

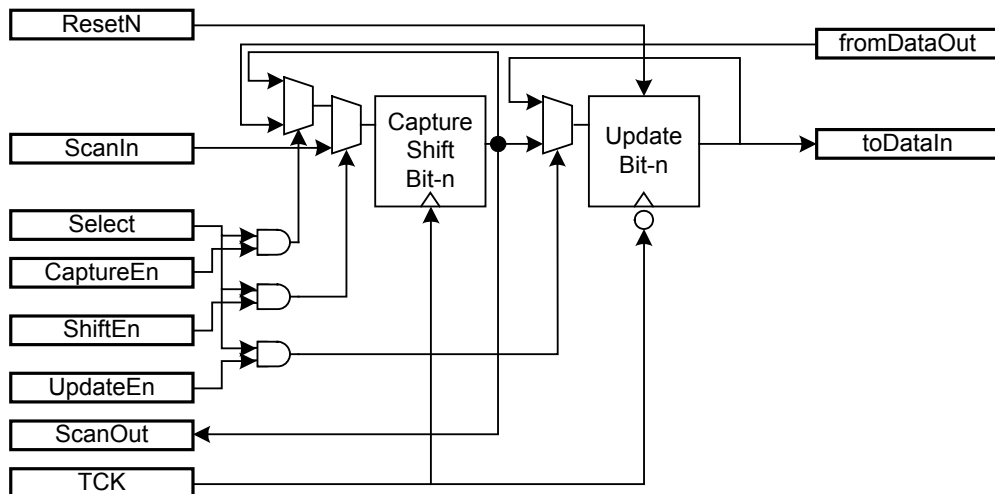


Figure 25—Example Read-Write TDR bit with control signals shown

One of the main requirements of operation of a TDR is that the data output bits of the TDR may change only when the TDR is part of the active scan chain—when not actively selected, but not in reset, the Update stage of the TDR should be “frozen” (resulting in no changes). The Capture/Shift stage may also freeze when not selected, but is not required to do so—the requirement is only for the Update stage. This requirement is implemented by the use of a selection signal (Select) that when asserted allows the control signals of the TDR to become active. There are many different methods to ensuring that the TDR is enabled when the Select signal is asserted and also ensuring the TDR is frozen when the Select signal is de-asserted. One typical implementation is to gate the control signals for shift, capture, and update (effectively gating the data operations, as shown by the AND gates in Figure 25), or to gate the TCK clock signal. In addition, the gating and distribution of the clock and control signals may be placed at a very high level such as the source of the selection signal (a WIR instruction, an IR instruction, a SIB, or a scan register driving a ScanMux control bit), or the gating may be applied local to the TDR or TDR bits where the control signals or clock are distributed as a tree across the chip.

5.7 Local reset

Many devices include a feature that performs what is known as a Global Reset of the test infrastructure. For example, in many IEEE 1149.1-controlled devices, assertion of the IEEE 1149.1 TRST* signal or entry into the TLR state of the TAP state machine generates a reset, which places a known state in the update-stage of some set of user-chosen TDR bits, optionally places a known state in the shift-side of some set of user-chosen TDR bits, and connects the IDCODE Register in between TDI and TDO. This type of reset is used to place a known state into the IEEE 1149.1 TAP Controller and a portion (perhaps the entirety) of the attached register architecture and is thus referred to as a Global Reset. The two different reset mechanisms (TRST* or TLR) may have different effects on the TDRs, depending on how the associated signals are connected to the reset ports of the TDRs.

A Global Reset may not be optimal when using the IEEE 1149.1 TAP Controller for reset control of an IEEE 1687 network. An IEEE 1687 network may be made of several different sub-networks that may be delivered by different instrument and IP providers. Each sub-network may optionally be placed behind an IEEE 1687 SIB construct so it may be accessed and treated independently. For example, multiple cores may be delivered with IEEE 1500 wrapper serial ports (WSPs) and each of these IEEE 1500 WSPs may be accessed by a unique SIB on a scan chain in a serial access network, as illustrated in Figure 26. To further facilitate independence and portability, each IEEE 1500 WSP's SelectWIR signal may be generated by an in-line ScanMux control bit (a shift-update TDR bit), labeled SWIR in Figure 26.

In this context, the impact of applying a Global Reset can be disruptive. Specifically, a Global Reset event from the TAP would close the SIBs that access the IEEE 1500 units and would force all IEEE 1500 wrapper architectures to a known reset state, regardless of whatever active state was in operation (assuming that reset forces the SIB configuration to a closed state and that the IEEE 1500 implementation defaults to the bypass instruction). In other words, a Global Reset may be useful for avoiding a scan operation that installs a default value into a particular (resettable) register, but it comes at the cost of also placing the associated default states into all the other (resettable) network registers.

A better and more useful configuration is to allow the IEEE 1687 network to include the ability to reset portions of the active or inactive scan chain without resorting to the Global Reset. This type of function is known as a Local Reset. The following discussion describes the use of circuitry to generate a local reset, but it is important to note that other types of circuits could be designed to suppress a global reset (see 6.4.6.13).

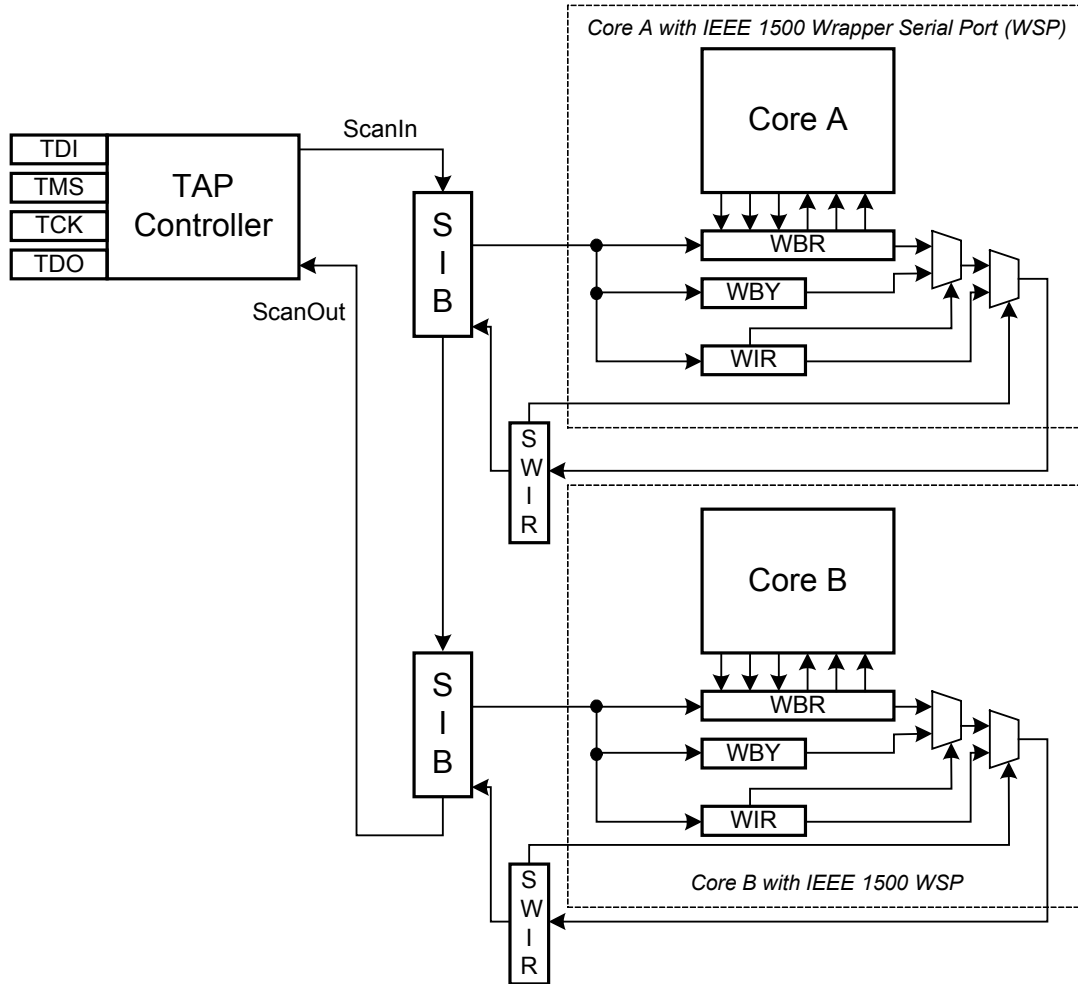


Figure 26—Example 2-SIB architecture with inline ScanMux control bit for SWIR

For example, the two-SIB architecture shown in Figure 27 may include another pair of control bits (Scan Registers labeled “LRA” for “Local Reset Assert” downstream from each of the SIBs) that, when asserted, would create the Local Reset signal via logic (labeled “LR gen” near each LRA bit). The Local Reset assertion would occur when an assert value is in the shift stage of the LRA register and the IEEE 1687 control signals generate an update operation. Since the update-side of a TDR-Bit normally holds state from one update assertion to the next update assertion (the next pass through the DR side of the IEEE 1149.1 FSM, for example), it is stipulated that the assertion of a local reset is required to not prevent the capture or shift operations of any scan registers with which it is associated. This stipulation enables the local reset to be de-asserted by shifting in new data and performing another update operation. It is common for reset to affect only the update stage of a scan register, so this stipulation is usually trivially satisfied. However, if the shift stage of a scan register is also affected by reset, there is a potential for the active scan chain to be broken by being held in reset (since both capture and shift operations occur between any two update events). To prevent this, a reset affecting a shift element is required to resolve itself to a de-assert state before the next shift event. There are several physical implementations, using several different possible methods that allow this capability, and since any implementation is valid, this type of shift-stage reset can simply be specified as “self-clearing.” One method of self-clearing, for example, is to permanently place the update stage of the local reset register in its reset state and selectively block the reset during the update operation; if the local reset register is written with the Local Reset de-assertion value during update, then nothing changes. However, if the local reset register is written with the Local Reset assertion value during update, then a Local Reset pulse will be created and this pulse can then place the target subset of IEEE 1687 network into a known defined state.

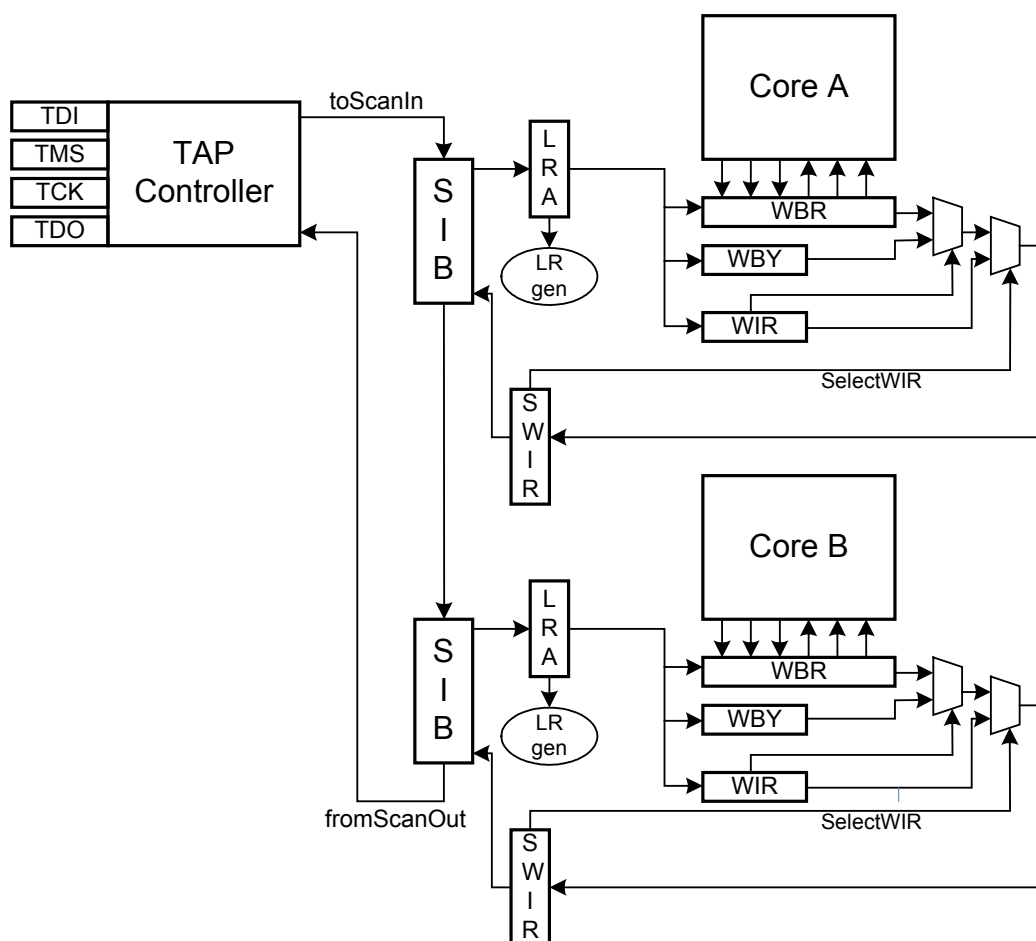


Figure 27—Local reset signals from inline control bits

Note that all IEEE 1687 network elements react to their module's Global Reset when it asserts, and those with a defined Local Reset also react when the Local Reset asserts; therefore, the reset input to the individual elements that make up a TDR that are also targeted by a Local Reset is the OR (assuming active-high signaling) of the module's Global Reset and the Local Reset (Figure 28).

In a complex IEEE 1687 network, one or more Local Resets may be generated to control different sections of the IEEE 1687 network.

Figure 29 shows an in-line control bit that can create the Local Reset for the previous bit in the scan chain. The Local Reset is generated on the falling edge of TCK as the IEEE 1149.1 FSM passes through the UpdateDR state. This local reset propagates to the target bit immediately, but arrives to the control bit's update stage through a delay operation. The only criterion for the delay is that its effect is complete prior to the next capture event. This feedback of the local reset signal results in a self-clearing pulse.