

IEEE Standard Testability Method for Embedded Core-based Integrated Circuits

IEEE Computer Society

Developed by the
Test Technology Standards Committee

IEEE Std 1500™-2022
(Revision of IEEE Std 1500-2005)



STANDARDS

IEEE Standard Testability Method for Embedded Core-based Integrated Circuits

Developed by the

Test Technology Standards Committee
of the
IEEE Computer Society

Approved 16 June 2022

IEEE SA Standards Board

THIS IS A REDLINE DOCUMENT PRODUCED BY TECHSTREET, AN ASME COMPANY.

This document is intended to provide users with an indication of changes from one edition to the next. It includes a full-text version of the new document, plus an indication of changes from the previous version.

Redlines are designed to save time and improve efficiencies by using the latest software technology to find and highlight document changes. More professionals are using valuable new technologies like redlines, to help improve outcomes in a fast paced global business world.

Because it may not be technically possible to capture all changes accurately, it is recommended that users consult previous editions as appropriate. In all cases, only the current base version of this publication is to be considered the official document.

This Redline is produced solely by Techstreet. IEEE has not prepared this Redline and is not responsible for its accuracy.

Redline Processing Notes:

1. ~~Red Text~~ - Red strikethrough text denotes deletions.
2. Blue Text - Blue underlined text denotes modifications and additions.

Abstract: A mechanism for the test of core designs within a system on chip (SoC) is defined. This mechanism is a hardware architecture and the core test language (CTL) is leveraged to facilitate communication between core designers and core integrators.

Keywords: core test, embedded core test, IEEE 1500, IP test, test reuse

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

Copyright © 2022 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 12 October 2022. 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 978-1-5044-8866-2 STD25531
Print: ISBN 978-1-5044-8867-9 STDPD25531

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/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 Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

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

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA 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 makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. 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: THE NEED TO PROCURE 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 is 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, nor 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 the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group. Statements made by volunteers may not represent the formal position of their employer(s) or affiliation(s).

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, 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 subcommittees of the IEEE SA Board of Governors 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 evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#).¹ An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.²

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 constitute 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.

¹ Available at: <https://development.standards.ieee.org/myproject-web/public/view.html#landing>.

² Available at: <https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html>.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US 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, neither IEEE nor its licensors waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, 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; <https://www.copyright.com/>. 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 10 years. When a document is more than 10 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 [IEEE Xplore](#) or [contact IEEE](#).³ For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#).⁴ Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

³ Available at: <https://ieeexplore.ieee.org/browse/standards/collection/ieee>.

⁴ Available at: <https://standards.ieee.org/standard/index.html>.

Patents

IEEE Standards are developed in compliance with the [IEEE SA Patent Policy](#).⁵

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 <https://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.

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users 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.

⁵ Available at: <https://standards.ieee.org/about/sasb/patcom/materials.html>.

Participants

At the time this IEEE standard was completed, the Embedded Core Test Working Group had the following membership:

Mike Ricchetti, *Chair*
Grady Giles, *Vice Chair*
Paul Reuter, *Editor*
Ben Niewenhuis, *Secretary*

Saman Adham
Roberto Benavides
Tapan Chakraborty
Adam Cron
Heiko Ehrenberg

Saurabh Gupta
Alan Hales
Ian Harrison
Adam Ley

Teresa McLaurin
Sankaran Menon
Thomas Rinderknecht
Tom Waayers
Jae Wu

Previous members of the Embedded Core Test Working Group are:

Karim Arabi
Luis Basto
Sudipta Bhawmik
Wu-Tung Cheng
Vivek Chickermane
Mike Collins

Francisco da Silva
Bulent Dervisoglu
Jason Doege
Rohit Kapur
Brion Keller

Eric Jan Marinissen
Mike Mateja
Fidel Muradali
Paul Soong
Lee Whetsel
Yervant Zorian

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

Abhishek Alladi
Roberto Benavides
Bill Brown
Tapan Chakraborty
Jean-Francois Cote
Adam Cron
Heiko Ehrenberg
Grady Giles
Saurabh Gupta
Ian Harrison
Peter Harrod

Kazumi Hatayama
Werner Hoelzl
Osamu Karatsu
Michael Laisne
Jim Lewis
Adam Ley
Ting Li
Ian McIntosh
Teresa McLaurin
Benoit Nadeau-Dostie

Ben Niewenhuis
Priyadarsan Patra
Paul Reuter
Mike Ricchetti
Thomas Rinderknecht
Stephen Schwarm
Roger Sowada
Walter Struppler
John Vergis
Tom Waayers
Oren Yuen

When the IEEE SA Standards Board approved this standard on 30 June 2022, it had the following membership:

David J. Law, *Chair*
Ted Burse, *Vice Chair*
Gary Hoffman, *Past Chair*
Konstantinos Karachalios, *Secretary*

Edward A. Addy
Ramy Ahmed Fathy
J. Travis Griffith
Guido R. Hiertz
Yousef Kimiagar
Joseph L. Koepfinger*
Thomas Koshy
John D. Kulick

Johnny Daozhuang Lin
Kevin Lu
Daleep C. Mohla
Andrew Myles
Damir Novosel
Annette D. Reilly
Robby Robson
Jon Walter Rosdahl

Mark Siira
Dorothy V. Stanley
Lei Wang
F. Keith Waters
Karl Weber
Sha Wei
Philip B. Winston
Daidi Zhong

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1500-2022, IEEE Standard Testability Method for Embedded Core-based Integrated Circuits.

IEEE Std 1500 is a scalable standard architecture for enabling test reuse and integration of embedded cores. It has serial and parallel test access mechanisms (TAMs) and a rich set of instructions suitable for testing cores, system on chip (SoC) interconnects, and circuitry. In addition, IEEE Std 1500 defines features that enable core isolation and protection. IEEE Std 1500 can reduce test costs and improve test quality by facilitating improvements in design automation, design-for-test (DFT) access, and DFT methodology.

Core test language (CTL) is the standard language for describing IEEE 1500 wrappers and hardware. The test patterns associated with cores may also be provided by CTL, or other IEEE Test Technology Standards Committee (TTSC) (or successor) sponsored standard test data languages. CTL is defined in IEEE Std 1450.6™.⁶

Objective of the IEEE 1500 effort

The Embedded Core Test Working Group was approved with the charter to develop a standard test access method for embedded cores in integrated circuits (ICs). That method is independent of the underlying functionality of the IC or its individual embedded cores. The access method supports the necessary testability requirements for detection and diagnosis of such ICs, while allowing for ease of interoperability of cores that originate from different sources. This method is usable for all classes of digital cores including hierarchical ones (15.2 discusses hierarchical core-wrapper configurations).

History and achievements of this standard

Development of IEEE Std 1500 began in 1997 with the original Working Group approval. During this period, the Embedded Core Test Working Group considered all aspects of core-based test and ensured that it would support a broad spectrum of users. It provided flexibility with both serial and parallel TAMs, definition of a core wrapper, and a set of both required and optional instructions. The standard was first published in 2005, and then reaffirmed in 2011.

Since the initial publication, and subsequent reaffirmation, IEEE Std 1500 has become widely adopted throughout the industry, with full support from IP designers and core providers, SoC designers, and electronic design automation (EDA) tool providers. IEEE Std 1500 fulfilled its scope and purpose of standardizing the reuse of nonmergeable cores, successful integration of cores into an SoC, and, through design automation, facilitating the transfer of test information and test data between core providers and core users. As the number and complexity of embedded cores used within an SoC grew, and technology advanced, the need for major revisions to IEEE Std 1500 became necessary. In 2019 a new working group was approved to revise the standard.

Changes introduced by this revision

While this is a major revision of the standard, the scope and purpose of the standard have not changed since it was last reaffirmed in 2011. The revisions in this version of the standard are for the most part optional. They are intended to provide clarifications or relaxations to the previous version's rules, and to optionally extend the capabilities of this standard to provide test access for cores and SoCs that have grown in size, number, and complexity. It is believed that these revisions were needed to support improved industry adoption and compliance. It is expected that cores that are compliant with the previous version of the standard remain compliant with this version.

⁶ Information on references can be found in Clause 2.

The major changes, listed in the order in which they appear in this standard, are as follows:

In Clause 7:

- Permissions were added for data registers to be segmented, and for the scan path selected by the wrapper instruction register (WIR) to be reconfigured as a result of a data register Update operation. In 7.2 the rules that required data registers (DRs) and the selected scan path to be fixed length were removed, and new permissions were added to allow data register segments and the data register scan path to change length. A new Figure 2 was added to 7.2.2 showing DR segments accessed by inline segment insertion bits (SIBs).

In Clause 8:

- The optional Transfer terminal was removed from the wrapper serial port (WSP) terminals, and the Transfer operation was removed throughout the body of the draft. Transfer was not used by core providers and was not supported by EDA tools. It also is not compatible with the IEEE 1149.1™ test access port (TAP) controller. It was originally intended to support at-speed or transition delay fault testing, but other methods were adopted in the industry. As such, Transfer was removed from the standard. It is no longer listed as a WSP terminal and was removed from Figure 12 of the WSP.
- There is a new permission for the wrapper serial output (WSO) terminal to change on the rising edge of wrapper clock (WRCK) and the WSP may now include an optional wrapper serial output rising (WSOR) terminal. The WSOR terminal changes on the rising edge of WRCK and may be included when the mandatory WSO changes on the falling edge of WRCK. Refer to Clause 14 for WSP timing specifications with the new permissions.

In Clause 12:

- Rule 12.2.1(b) was updated to include the state of SIBs, in the case where the wrapper boundary registry (WBR) is segmented using SIBs.
- In 12.4, clarifications were made to the apply event and user-defined events.
- In 12.7, the cell naming conventions were updated for removal of transfer. The naming convention now specifies the capture position into the shift path of the WBR cell.
- In 12.8, Transfer was removed from the WBR cell examples, and some of the examples were updated with new WBR cells that do not require Transfer.
- There is an updated delay test example in 12.9. This was updated based on the removal of Transfer, using conventional delay test methods.

In Clause 14:

- The rules and permissions in 14.2 are updated to support the WSO and WSOR revisions.
- New timing diagrams for the wrapper serial output falling and rising (WSO[R]) revisions were added in 14.3. Figure 43 was added to show the permission for WSO rising edge and Figure 44 was added to show the permission for WSOR.
- A new 14.5 was added to show an example of a methodology to allow for higher shift frequencies when the CaptureWR, ShiftWR, and UpdateWR signals are timed as multi-cycle paths.

In Clause 15:

- There are new examples for hierarchical subcores, shown in Figure 51 through Figure 54.
- There is a new rule 15.2.2(a) requiring concurrent CaptureWR, ShiftWR, and UpdateWR operation of cores, when selected together in a scan path.

In Clause 16:

- Clause 16 was updated to clarify plug-and-play with respect to the WSO(R) revisions.

In Clause 17:

- Relaxed CTL compliance requirements (CTL required for hardware description, patterns may use other IEEE TTSC standards).
- Rules 17.2.1(a) and 17.2.1(b) were updated to specify only hardware-related test information be provided in CTL and test data and protocols specified in an IEEE TTSC (or successor) sponsored standard test data language.
- Rules 17.2.1(d) and 17.2.1(e) were updated to require specifying the maximum length of a scan chain, and the order of cells in a scan chain, when SIBs are present in the chain.
- A new permission 17.2.1(h) allows the use of other test data languages for test modes, test protocols, and test pattern data.
- In 17.4.1, rules are updated for use of other test data languages.

In Clause 20:

- Updated old CTL examples to comply with released IEEE Std 1450.6 (CTL). Added a new example showing minimal CTL usage.

In Annex A:

- The WBR bubble diagram definitions were updated in order to remove Transfer.
- Added new definitions for SIB bubble diagrams, and added a SIB example in Figure A.4.

In Annex B:

- Updates to the bubble diagram examples due to the removal of transfer.

In Annex C:

- Annex C was rewritten to include the relationship of IEEE Std 1500 to IEEE Std 1687™.
- The old example of interfacing an IEEE 1500 core to an IEEE 1149.1 TAP controller was replaced with a new example. The old example accessed the IEEE 1500 WIR together with the IR of the TAP controller.
- The new example integrates multiple cores within an SoC using IEEE Std 1149.1, IEEE Std 1500, and IEEE Std 1687. It describes a hierarchical serial access network that is accessed using the TAP controller and two TAMs. This new example accesses the IEEE 1500 WIRs using IEEE Std 1687 and the DR states of the TAP controller.

Two examples of hierarchically accessing the TAMs and cores from the IEEE 1149.1 TAP controller are given.

Contents

1. Overview	14
1.1 Background.....	14
1.2 Scope	15
1.3 Purpose	15
1.4 Word usage.....	15
1.5 Need.....	16
1.6 Access protocols	16
2. Normative references.....	16
3. Definitions, acronyms, and abbreviations	16
3.1 General	16
3.2 Definitions	17
3.3 Acronyms and abbreviations	22
4. Structure of this standard.....	24
4.1 Introduction	24
4.2 Specifications.....	24
4.3 Descriptions.....	25
4.4 Conventions	25
5. Introduction and motivations for two compliance levels.....	25
6. Overview of the IEEE 1500 scalable hardware architecture	26
6.1 Introduction	26
6.2 Wrapper serial port (WSP)	26
6.3 Wrapper parallel port (WPP).....	27
6.4 Wrapper instruction register (WIR).....	27
6.5 Wrapper bypass register (WBY).....	28
6.6 Wrapper boundary register (WBR).....	28
7. WIR instructions.....	28
7.1 Introduction	28
7.2 Response of the wrapper circuitry to instructions and data registers	29
7.3 Wrapper instruction rules and naming convention	31
7.4 WS_BYPASS instruction	33
7.5 WS_EXTEST instruction	34
7.6 WP_EXTEST instruction	36
7.7 Wx_EXTEST instruction.....	38
7.8 WS_SAFE instruction	39
7.9 WS_PRELOAD instruction.....	41
7.10 WP_PRELOAD instruction.....	43
7.11 WS_CLAMP instruction	44
7.12 WS_INTEST_RING instruction.....	46
7.13 WS_INTEST_SCAN instruction.....	48
7.14 Wx_INTEST instruction.....	50
8. Wrapper serial port (WSP)	52
8.1 Introduction	52
8.2 WSP terminals	52

9. Wrapper parallel port (WPP).....	54
9.1 WPP terminals	54
10. Wrapper instruction register (WIR).....	54
10.1 Introduction	54
10.2 WIR configuration and DR selection.....	55
10.3 WIR design.....	56
10.4 WIR operation	59
11. Wrapper bypass register (WBY)	61
11.1 WBY register configuration and selection.....	61
11.2 WBY design	62
11.3 WBY operation.....	63
12. Wrapper boundary register (WBR)	64
12.1 Introduction	64
12.2 WBR structure and operation	67
12.3 WBR cell structure and operation.....	68
12.4 WBR operation events.....	69
12.5 WBR operation modes.....	70
12.6 Parallel access to the WBR.....	72
12.7 WBR cell naming	75
12.8 WBR cell examples	77
12.9 IEEE 1500 WBR example	80
13. Wrapper states	90
13.1 Wrapper Disabled and Wrapper Enabled states.....	90
14. WSP timing diagrams.....	92
14.1 Introduction	92
14.2 Specifications.....	92
14.3 Description	93
14.4 Synchronous reset timing	99
14.5 Timing for fast pipelined shift	100
15. WSP configurations for IEEE 1500 system chips	102
15.1 Introduction	102
15.2 Connecting multiple WSPs.....	102
16. Plug-and-play (PnP)	107
16.1 Introduction	107
16.2 Background and definition	107
16.3 PnP aspects of standard instructions.....	109
16.4 PnP limitations on protocols.....	109
16.5 Non-PnP in IEEE Std 1500.....	109
17. Compliance definitions common to wrapped and unwrapped cores	110
17.1 Introduction	110
17.2 General rules.....	110
17.3 Per terminal rules.....	112
17.4 Test pattern information rules.....	114
18. Compliance definitions specific to unwrapped cores	117
18.1 General rules.....	117
18.2 Per terminal rules.....	118

18.3 Additional test information rules	119
19. Compliance definitions specific to wrapped cores	119
19.1 General rules	119
19.2 Per terminal rules	120
19.3 Wrapper protocol information rules	121
20. IEEE 1500 application	121
20.1 Introduction	121
20.2 CTL (IEEE 1450.6) overview	121
20.3 IEEE 1500 examples	123
Annex A (informative) Bubble diagram definition	152
Annex B (informative) Wrapper boundary register (WBR) cell examples	154
Annex C (informative) Relationship of IEEE Std 1500 to IEEE Std 1149.1 and IEEE Std 1687	162
C.1 Introduction	162
C.2 Example of integrating IEEE 1500 cores within an SoC	162

IEEE Standard Testability Method for Embedded Core-based Integrated Circuits

1. Overview

1.1 Background

IEEE Std 1500™ defines a scalable architecture for independent, modular test development and test application for embedded design blocks and also enables test of the external logic surrounding these cores. Modular testing is typically a requirement for embedded nonlogic blocks, such as memories, and for embedded pre-designed nonmergeable intellectual property (IP) cores. In addition, the IEEE 1500 architecture can also be used to partition large design blocks into smaller blocks of more manageable size and to facilitate test reuse for blocks that are reused from one system-on-chip (SoC) design to the next.

The IEEE 1500 architecture comprises hardware requirements, through the definition of a standardized core wrapper, and uses a test-specific ~~languages~~ ~~language~~ to communicate information between core providers and core users. ~~This language is~~ The IEEE ~~1450~~ ~~P1450~~.6™ ~~1~~-core test language (CTL) is required for hardware-related test information pertinent to core integration. Test data and test protocols pertinent to core integration shall be provided in either CTL or another IEEE TTSC (or successor) sponsored standard test data language. Although IEEE Std 1500 limited itself to test aspects internal to nonmergeable cores, careful consideration was given to the interoperability of such cores, resulting in plug-and-play (PnP) requirement definitions. SoC-specific issues such as those related to the design of test access mechanisms (TAMs) are excluded from this standard and assumed to be addressed by the SoC designer.

¹~~Information on references can be found in Clause 2.~~

IEEE Std 1500 specifically focuses on defining test requirements for unidirectional non-tristate digital terminals, as these represent a minimum and mandatory set of requirements upon which the more complex bidirectional terminals are based. It is, therefore, implied that support for bidirectional or tristatable terminals is provided only to the extent that the individual unidirectional terminals, i.e., the bidirectional or tristatable terminal, are available for IEEE 1500 wrapper insertion. In addition, the hardware architecture defined in this standard anticipates a synchronous wrapper design methodology.

While IEEE Std 1500 does not discuss chip-level design, the architecture defined in this standard does not prevent interfacing with [other standards, namely standards based on IEEE Std 1149.1™ and IEEE Std 1687™](#)~~-based standards~~. An example of this interface is provided in Annex C for the reader's benefit.

All rules described in this standard apply to the case where the IEEE 1500 wrapper is enabled (the wrapper logic actively participates in the test of the core) except rules specific to the Wrapper Disabled state of the IEEE 1500 wrapper. In Wrapper Disabled state, the IEEE 1500 wrapper is disabled, allowing functional operation of the wrapped core. IEEE ~~P~~1450.6 constructs were added to this standard, where appropriate, to further guide the reader. ~~It is anticipated that~~ The reader [is directed to](#) ~~will~~ refer to these CTL constructs documented in IEEE [Std](#) ~~P~~1450.6. Additional discussion that complements the body of this standard are presented in annex clauses:

- Annex A contains the legend for IEEE 1500 wrapper cells.
- Annex B shows examples of IEEE 1500 wrapper cells.

Annex C [describes the relationship](#) ~~presents similarities~~ between IEEE Std 1500, [IEEE Std 1687](#), and IEEE Std 1149.1, and [also provides](#) ~~discusses~~ an example [SoC integration of several IEEE 1500 cores using these standards](#). ~~interface between IEEE Std 1500 and IEEE Std 1149.1.~~

1.2 Scope

IEEE Std 1500 ~~is has developed~~ a standard design-for-testability method for integrated circuits (ICs) containing embedded nonmergeable cores. This method is independent of the underlying functionality of the IC or its individual embedded cores. The method ~~supports creates~~ the necessary requirements for the test of such ICs, while allowing for ease of interoperability of cores that ~~might may~~ have originated from different sources. This method is usable for all classes of digital cores, including hierarchical cores.

1.3 Purpose

The aim of IEEE Std 1500 is to provide a consistent scalable solution to the test reuse challenges specific to the reuse of nonmergeable cores, while preserving the IP aspects that are often associated with these cores. This objective is achieved through provision of a core-centric methodology that enables successful integration of cores into SoCs.

1.4 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall equals is required to*).^{7, 8}

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should equals is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may equals is permitted to*).

⁷ The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

⁸ The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

1.5 Need

IEEE Std 1500 provides a bridge between core providers and core users and also facilitates the automation of test data transfer and reuse between these two entities via the use of the IEEE P1450.6 CTL. This automation relies on information requirements (the information model) placed on the core provider to help ensure that the core can be successfully integrated by the core user. The result is shorter time to market for core providers and core users.

The data transfer and reuse from the core provider to the core user are based on the premise that the core test data are left unchanged, while the test protocol is adapted from the IEEE 1500 hardware interface to the SoC.

1.6 Access protocols

Access protocols that reside above the IEEE 1500 core might need to be activated before accessing the core wrapper and core data registers. These protocols are outside the scope of this standard. An example of such protocols might be one that provides security access to the SoC.

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 1149.1TM, IEEE Standard Test Access Port and Boundary-Scan Architecture. [9, 10](#)²

IEEE [Std P1450.6TM](#), ~~IEEE, Draft Standard for~~ Standard Test Interface Language (STIL) for Digital Test Vector Data—Core Test Language (CTL), ~~<http://grouper.ieee.org/groups/ctl/>~~.

[IEEE Std 1687TM, IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device.](#)

3. Definitions, acronyms, and abbreviations

[3.1 General](#)

[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.](#) ¹¹

This clause lists some definitions of terms that have been used throughout this standard. In addition, a list of acronyms and abbreviations is also provided. The criteria for differentiating various terms are based on the following rules:

- a) General terminology in the scope of electrical engineering and/or test technology [that](#), ~~which~~ do not influence the definition of this standard itself. These terms are used in this standard without further explanation. It is assumed that the readership has sufficient background knowledge to understand these terms.

⁹ [The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.](#)

^{10 2} IEEE publications are available from The Institute of Electrical and Electronics Engineers (<https://standards.ieee.org/>), ~~445 Hoes Lane, Piscataway, NJ 08854, USA (<http://standards.ieee.org/>).~~

¹¹ [IEEE Standards Dictionary Online is available at: <http://dictionary.ieee.org>. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.](#)

~~2) Some of these terms may already be defined in The Authoritative Dictionary of IEEE Standards Terms. Therefore, someone looking for a particular definition could always consult The Authoritative Dictionary to verify the meaning of the term.~~

b) Terms that are specific to this standard.

- 1) These terms are defined in this clause.
- 2) General definitions are valid throughout this standard.
- 3) Local definitions are relevant only to a specific clause of this standard.

Effort has been made to achieve consistent usage of all terms (e.g., wrapper, TAM).

3.2 Definitions

access mechanisms: Mechanisms by which signals ~~are may be~~ propagated to and from a core, from either embedded circuitry or from the primary inputs and outputs of the system chip. There are two types of access mechanisms:

- a) Functional access: The mechanism for moving stimuli to and observing responses from a core or user-defined logic (UDL) during functional operation or normal mode.
- b) Test access: The mechanism for moving stimuli to and observing responses from a core or UDL during nonfunctional operation or test mode.

auxiliary clock (AUXCK): A functional clock that ~~can may~~ be used in conjunction with wrapper clock (WRCK) during core test for capturing, shifting, ~~and~~ updating, ~~and optionally transferring~~ test data in a wrapper.

bypass: As applied to core wrappers, an abbreviated sequential path connecting a wrapper serial input (WSI) to a wrapper serial output ~~falling and rising~~ (WSO ~~[R]~~).

CaptureWR: A wrapper terminal used to enable and control a Capture operation in the selected IEEE 1500 wrapper register (WR).

cell functional input (CFI): For input wrapper cells, the cell's input, which is connected to a wrapper functional input (WFI); for output wrapper cells, the cell's input, which is connected to a core output.

NOTE—See CFI pin in Figure [17](#) ~~16~~.

cell functional output (CFO): For input wrapper cells, the cell's output, which is connected to a core input; for output wrapper cells, the cell's output, which is connected to a wrapper functional output (WFO).

NOTE—See CFO pin in Figure [17](#) ~~16~~.

cell test input (CTI): A wrapper boundary register (WBR) cell's test data input.

cell test output (CTO): A wrapper boundary register (WBR) cell's test data output.

~~3.1.9 control: The process of applying test pattern stimuli.~~

core: Predesigned circuit block that can be tested as an individual unit.

core data register (CDR): Optional data register that belongs to a core being wrapped.

core input: An input terminal of an unwrapped core.

core integrator: An entity that incorporates one or more cores into [a an](#) system on chip (SoC).

core isolation: A test mode feature preventing core-to-core or core-to-UDL (i.e., user-defined logic) interaction.

core output: An output terminal of an unwrapped core.

core provider: An entity that designs cores that can be reused in other designs.

core test: A test methodology that is applied to an embedded core.

core test language (CTL): A standard language for core suppliers to provide test data that can be used to test a core once it is integrated into a system on chip (SoC). The language presents a format to describe test and support data so that the core can be effectively integrated, reused, and tested.

NOTE—See IEEE [Std P1450.6](#) reference documentation.

dedicated shift path: A shift path comprising storage elements that do not participate in functional operation.

dedicated wrapper (cell): A wrapper style that does not share hardware with core functionality. This style allows certain test operations to occur concurrently and transparently during functional operation. This definition could apply to individual cells.

event: An uninterrupted, predefined sequence of one or more steps that apply to a data register, e.g., wrapper boundary registry (WBR), as a whole. Within the interval of the predefined sequence there can be a particular instant that characterizes the event when the nominal action of the event occurs. This instant, if it exists for a particular event, is referred to as the *characteristic instant*. Predefined events are Shift, Update, Capture, and Apply.

external safe state: A configuration of safe state in which the outputs of a core are in a state that prevents them from interfering with a block of logic outside the core. *See also:* **internal safe state; safe state.**

firm core: A predesigned block of functional logic, such as a macro, megacell, or memory, that has a process technology-dependent netlist representation and so is ~~may be~~ amenable to some modification.

hard core: A predesigned block of functional logic, such as a macro, megacell, or memory, that has a physical implementation that cannot be modified.

hybrid instruction: A wrapper instruction that has mixed use of wrapper serial port (WSP) and wrapper parallel port (WPP) terminals.

input cell: A wrapper boundary register (WBR) cell that is provided on a core input.

internal safe state: A configuration of safe state whereby a core is protected from the impact of a test outside the core. *See also:* **external safe state**; **safe state**.

interoperability: *See:* **plug-and-play (PnP)**.

inward facing (IF) mode: The test mode where core inputs are controlled by the wrapper boundary register (WBR) and core outputs are observed by the WBR.

mergeable core: With respect to testability, a core that can be integrated with other cores and user-defined logic (UDL) into a system on chip (SoC) so that a uniform design-for-test (DFT) methodology can be applied across the entire system. A typical mergeable core is provided using a register transfer level (RTL) or gate-level description.

merged core: With respect to testability, a core that is integrated with other cores and user-defined logic (UDL) into a system on chip (SoC) so that a uniform design-for-test (DFT) methodology could be applied across the entire system.

nonmergeable core: With respect to testability, a core that cannot be integrated to apply a uniform design-for-test (DFT) methodology to the entire system on chip (SoC). A typical non-mergeable core comes with a physical design implementation that does not accommodate modification of the test methodology. A nonmergeable core might ~~may~~ be represented as a black ~~block~~-box design, making standard automatic test pattern generation (ATPG) impossible on such a core.

nonmerged core: With respect to testability, a core that has not been integrated with other cores and user-defined logic (UDL) into a system on chip (SoC) so that a uniform design-for-test (DFT) methodology could be applied across the entire system.

normal mode: The mode in which the wrapper boundary register (WBR) does not interfere with the functional operation of a wrapped core.

~~**3.1.34 observation:** The process of monitoring pattern response.~~

output cell: A wrapper boundary register (WBR) cell that is provided for a core output.

outward facing (OF) mode: The test mode where wrapper functional outputs (WFOs) are controlled by the wrapper boundary register (WBR) and wrapper functional inputs (WFIs) are observed by the WBR.

parallel instruction: A wrapper instruction that uses wrapper parallel port (WPP) terminals and also configures the wrapper bypass register ([WBY](#) ~~WBR~~) between wrapper serial input (WSI) and wrapper serial output [falling and rising](#) (WSO [\[R\]](#)).

pattern set: A collection of test vectors intended for manufacturing test. In the context of core test language (CTL), a pattern set is a collection of pattern constructs and their associated macros and procedures brought together with PatternBurst and PatternExecs.

plug-and-play (PnP): A minimum level of [required](#) interoperability between various core wrappers in a system on chip (SoC). [See Clause 16 for detailed requirements.](#)

safe data: Data that satisfy safe state configuration requirements. These data are user-defined.

safe state: A property whereby a test of one block of logic is prevented from interfering with or damaging another block of logic. *See also:* **external safe state**; **internal safe state**.

[segment:](#) [A set of contiguous \(in terms of the scan path\) data register bits obeying the rules for wrapper data registers and from which one or more data registers can be assembled.](#)

[segment insertion bit \(SIB\):](#) [A scan cell with at least shift and update storage elements and a scan-multiplexor that is used to include or exclude a data register segment from the active scan path. SIBs can be in-line with the data register segments, or on a separate path in order to remotely control the segments.](#)

SelectWIR: The IEEE 1500 wrapper terminal that determines the selection of a wrapper register (WR). A value of 1 represents selection of the wrapper instruction register (WIR), and a value of 0 represents selection of a wrapper data register (WDR).

serial instruction: A wrapper instruction that exclusively uses wrapper serial port (WSP) terminals.

serial scan chain: The scan chain configuration inside a wrapped core where an internal scan chain is concatenated with the wrapper boundary register (WBR) chain for the purpose of running the WS_INTEST_SCAN instruction.

shared wrapper (cell): The wrapper style that shares logic between the test and functional modes of operation. Shared cells typically include registered core inputs and outputs that can be used in test mode to control and observe core test data. This style prevents simultaneous functional and test operation uses of the shared register.

ShiftWR: The wrapper terminal used to enable and control a Shift operation in the selected IEEE 1500 wrapper register (WR).

silent shift path: A wrapper boundary register (WBR) shift path comprising a dedicated shift path and implemented to support a Shift operation that keeps wrapper functional output (WFO) terminals static.

soft core: A predesigned block of functional logic such as a macro, megacell, or memory with a register transfer level (RTL) representation. Soft cores are inherently process technology independent.

standard test interface language (STIL): The language in IEEE Std 1450™ for representing digital test vector data. The core test language (CTL) is an extension of STIL to support a standard way of representing [test architecture and optionally the](#) test data for a core.

system on chip (SoC): An entire system integrated on a single chip. It ~~can~~ ~~may~~ include one or more cores with user-defined logic (UDL) integrated by the core user or system integrator.

[terminal:](#) [A connection point from the core or wrapper to the logic outside of the core or wrapper.](#)

test access mechanism (TAM): A feature of a system-on-chip (SoC) design that enables the delivery of test data to and from cores or core wrappers.

test access mechanism (TAM) harness: Wrapper boundary register (WBR) logic that enables the coupling of a TAM to cell test inputs (CTIs) and cell test outputs (CTOs).

test input (TI): The serial test data input of a wrapper boundary register (WBR).

NOTE—See TI in Figure [17 16](#).

test mode: A [state configuration](#) whereby a block of logic is made ready for test.

test output (TO): The serial test data output of a wrapper boundary register (WBR).

NOTE—See TO in Figure [17 16](#).

test protocol: A sequence of control operations required for a test. A test protocol comprises functions and/or sequences. Functions [can may](#) consist of other functions and/or sequences, while sequences comprise a series of logic 0 and 1 values applied to specified terminals. At the lowest level, a test protocol is just a series of logic 0 and 1 applied to specified test control terminals. The protocol [would will](#) typically also contain symbolic references to the test data that are to be applied to or observed at a specified test data or system data port. For example, a scan protocol might involve the repetition of the following operations:

- a) Apply a logic level to assert an internal scan chain SCAN_ENABLE control signal.
- b) Apply a sequence of n clock pulses to a clock port while applying data to the SCAN_DATA_IN of the scan chain(s). Observe data at the SCAN_DATA_OUT of the scan chain(s) as the data are clocked through the scan chains.
- c) De-assert SCAN_ENABLE and clock the scan chains one or more times while controlling primary inputs and observing primary outputs.
- d) Repeat steps ~~(a)~~, ~~(b)~~, and ~~(c)~~ until the application of the scan chain patterns is complete.

test reuse: The ability to apply a predetermined test pattern associated with a core after this core has been integrated into a system on chip (SoC). Test reuse is a consequence of design reuse and often requires the adaptation of a test protocol to reflect the core's new environment within an SoC.

[Test Technology Standards Committee \(TTSC\): Organization within IEEE that promotes standards in the test industry.](#)

~~3.1.58 transferDR: The IEEE 1500 wrapper terminal provided to enable and control the Transfer operation for the wrapper boundary register (WBR).~~

update register: The register used to prevent the outputs of a shift register from propagating to other circuitry during Shift operation. After shifting is complete, the content of the associated shift register is parallel-loaded (updated) into the update register.

UpdateWR: The wrapper terminal used to enable and control an Update operation in the selected IEEE 1500 wrapper register (WR).

user-defined logic (UDL): Logic added by the system chip integrator (i.e., not a reused design), as interface circuitry or part of the feature set that differentiates the system-on-chip (SoC) product.

wrapper: Circuitry added around an embedded core to facilitate test reuse and to interface between a test access mechanism (TAM) and the embedded core.

wrapper boundary register (WBR): The portion of a wrapper comprising wrapper cells. *See: wrapper cell.*

wrapper bypass register (WBY): The IEEE 1500 register providing an abbreviated data register between wrapper serial input (WSI) and wrapper serial output (WSO[R]).

wrapper cell: The wrapper element associated with at least one ~~a single~~ core terminal.

wrapper clock (WRCK): The IEEE 1500 clock.

wrapper configuration: An arrangement of interconnected wrappers in a system on chip (SoC).

wrapper data register (WDR): The IEEE 1500 register (e.g., wrapper boundary register [WBR], wrapper bypass register [WBY]) used to perform IEEE 1500 operations.

wrapper functional input (WFI): The wrapper input terminal corresponding to the functional core input of a wrapped core.

wrapper functional output (WFO): The wrapper output terminal corresponding to the functional core output of a wrapped core.

wrapper input: An input terminal to a wrapped core. In the case where there is a shared wrapper cell, the term *wrapper input* takes precedence over the term *core input*.

wrapper instruction register (WIR): The IEEE 1500 register used to serially load and store IEEE 1500 instructions.

wrapper output: An output terminal to a wrapped core. In the case where there is a shared wrapper cell, the term *wrapper output* takes precedence over the term *core output*.

wrapper parallel port (WPP): A user-defined set of wrapper terminals corresponding to a test access mechanism (TAM) and providing parallel access to the wrapper boundary register (WBR) and/or an embedded core.

wrapper register (WR): A wrapper instruction register (WIR) or wrapper data register (WDR).

wrapper reset (WRSTN): The IEEE 1500 active low reset.

wrapper serial input (WSI): The wrapper serial data input.

wrapper serial output (WSO): The wrapper serial data output.

wrapper serial output rising (WSOR): An optional second wrapper serial data output. When implemented, the WSOR terminal outputs the same value as WSO, but changing on the rising edge of WRCK rather than the falling edge of WRCK.

wrapper serial output falling and rising (WSO[R]): WSO(R) is used in this document when it applies to both the WSO and optional WSOR terminals.

wrapper serial port (WSP): Standard wrapper terminals providing serial access to the IEEE 1500 wrapper. The WSP comprises the following signals—~~comprise the WSP~~: wrapper reset (WRSTN), wrapper clock (WRCK), SelectWIR, ~~TransferDR~~, ShiftWR, CaptureWR, UpdateWR, wrapper serial input (WSI), and wrapper serial output (WSO[R]). Additionally, user-defined auxiliary clock(s) (AUXCKs) can ~~may~~ be part of the WSP.

Wx : Shorthand nomenclature to refer to multiple instructions that instructions that have the same suffix. For example, Wx PRELOAD refers to WS PRELOAD, WP PRELOAD, etc.

3.3 Acronyms and abbreviations

ASIC application-specific integrated circuit

ATPG automatic test pattern generation

AUXCK auxiliary clock

CDR core data register

CFI cell functional input

CFO cell functional output

CSU CaptureWR, ShiftWR, UpdateWR

CTI cell test input

CTL core test language

CTO cell test output

DFT design-for-test

DR data register

EDA electronic design automation

FPGA field-programmable gate array

IC integrated circuit

IF inward facing

I/O input/output

IP intellectual property

[IR instruction register](#)

OF outward facing

PnP plug-and-play

RF radio frequency

RTL register transfer level

[SIB segment insertion bit](#)

SoC system on chip

STIL standard test interface language

TAM test access mechanism

TAP test access port

TCK IEEE 1149.1 test clock

TI test input

TO test output

[TTSC Test Technology Standards Committee](#)

UDL user-defined logic

WBR wrapper boundary register

WBY wrapper bypass register

WDR wrapper data register

WFI wrapper functional input

WFO wrapper functional output

~~WGL waveform generation language~~

WIR wrapper instruction register

WPC wrapper parallel control

WPI wrapper parallel input

WPO wrapper parallel output

WPP wrapper parallel port

WR wrapper register

WRCK wrapper clock

WRSTN wrapper reset w.r.

t with respect to

WSC wrapper serial control

WSI wrapper serial input

WSO wrapper serial output

WSOR wrapper serial output rising

WSO(R) wrapper serial output falling and/or rising

WSP wrapper serial port

4. Structure of this standard

4.1 Introduction

The three mandatory hardware blocks along with the standard instructions and the mandatory serial port are described in separate clauses of this standard:

- Standard instructions: Clause 7
- Wrapper serial port (WSP): Clause 8 and Clause 14
- Wrapper instruction register (WIR): Clause 10
- Wrapper bypass register (WBY): Clause 11
- Wrapper boundary register (WBR): Clause 12

The above hardware clauses are complemented with information model and compliance discussion split across the following clauses:

- Compliance definitions common to wrapped and unwrapped cores: Clause 17
- Compliance definitions specific to unwrapped cores: Clause 18
- Compliance definitions specific to wrapped cores: Clause 19

4.2 Specifications

Subclauses entitled “Specifications” contain the rules, recommendations, and permissions that define this standard:

- a) *Rules* specify the mandatory aspects of this standard. Subclauses that are rules contain the word *shall*.
- b) *Recommendations* indicate preferred practice for designs that seek to conform to this standard. Subclauses that are recommendations contain the word *should*.

- c) *Permissions* show how optional features may be introduced into a design that seeks to conform to this standard. These features ~~will~~ extend the application of the test circuitry defined by this standard. Subclauses that are permissions contain the word *may*.

It is assumed throughout this standard that rules take precedence over recommendations and recommendations take precedence over permissions.

4.3 Descriptions

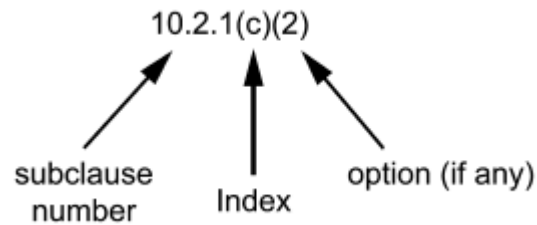
Material not contained in subclauses entitled “Specification” is descriptive material that illustrates the need for the features being specified or their application. This material includes schematics that illustrate a possible implementation of the specifications in this standard. Annex clauses to this standard contain alternative implementation examples. The descriptive material also discusses rationale for inclusion of certain features.

The descriptive material contained in this standard is for illustrative purposes only and does not define a preferred implementation. Most examples provided are intentionally somewhat ambiguous to not display bias ~~toward~~ ~~towards~~ a particular implementation style. Where discrepancies between examples and specifications ~~may~~ occur, the specifications ~~always~~ take precedence. Readers should exercise caution when using these examples to ~~facilitate~~ ~~ensure~~ full compliance in their specific applications. In particular, it is emphasized that the examples are designed to effectively communicate the meaning of this standard. As always, a particular implementation ~~might~~ ~~may~~ not operate properly with respect to timing and other parametric characteristics.

4.4 Conventions

The following structural conventions are used in this standard:

- The rules, recommendations, and permissions in each specifications subclause are contained in a single alphabetically indexed list. References to each rule, recommendation, or permission are shown in the form:



- When the reference and the referring text belong to the same subclause, only the index is indicated (i.e., the subclause number is omitted).

5. Introduction and motivations ~~for~~ **of** two compliance levels

IEEE Std 1500 recognizes the heterogeneous aspect of the embedded core market, i.e., a diversity driven by the need to cover a variety of design functions implemented in digital logic, analog logic, memory, radio frequency (RF), field-programmable gate arrays (FPGAs), or combinations of the above. Cores come in many different “flavors” (e.g., hard, firm, soft) and are being used and/or sold within companies as well as between companies. Within its focus on nonmerged digital logic and memory cores, IEEE Std 1500 intends to support the above diversity and associated business models. This requirement poses a need for flexibility that was translated into the definition of the following two levels of compliance to this standard:

- **IEEE 1500 unwrapped compliance:** This compliance level refers to a core that does not have a (complete) IEEE 1500 wrapper, but does have a IEEE [1450.6](#) ~~1500~~ CTL description on the basis of which the core could be made IEEE 1500 wrapped compliant, either manually or by using dedicated tools. The CTL [test data](#) ~~program~~ describes the core test knowledge at the bare core terminals.

- **IEEE 1500 wrapped compliance:** This compliance level refers to a core that incorporates an IEEE 1500 wrapper function and comes with [necessary test data described using a combination of IEEE 1450.6 CTL and other IEEE Test Technology Standards Committee \(TTSC\) \(or successor\) sponsored standard test data language](#). All hardware-related test information pertinent to core integration, which includes describing the wrapper and the wrapper's external terminals, shall be described in CTL, while test pattern data and test protocols pertinent to core integration can be described in CTL or another IEEE TTSC (or successor) sponsored standard test data language. ~~an IEEE 1500 CTL program. The CTL program describes the core test knowledge, including how to operate the wrapper, at the wrapper's external terminals.~~

The motivation behind the two different levels of compliance is a need to match the core-provider/core-user business model and provide the flexibility that is required in testing core-based system chips. The ultimate compliance goal should be to make a core compliant to the IEEE 1500 wrapped level. Nevertheless, the two levels of compliance provide the option to become IEEE 1500 wrapped directly or via the intermediate step of being IEEE 1500 unwrapped.

The first alternative, i.e., direct generation of [an](#) ~~a~~ IEEE 1500 wrapped core, provides the possibility to integrate the wrapper functionality with the core itself and hence [reduce](#) ~~minimize~~ the performance and area impact of the wrapper. The second alternative is to first create an IEEE 1500 unwrapped core, which is then, in a separate step, turned into an IEEE 1500 wrapped core. This allows the SoC integrator to take advantage of the scalability of the standardized wrapper and instantiate the wrapper with particular parameter values, which [takes](#) ~~take~~ into account certain aspects of the system chip environment in which this particular core version is used.

6. Overview of the IEEE 1500 scalable hardware architecture

[6.1 Introduction](#)

The IEEE 1500 core wrapper comprises the following:

- Serial interface terminals forming the WSP

- [An optional](#) ~~A~~ user-defined set of wrapper terminals forming the wrapper parallel port (WPP) and providing parallel access to the wrapper
- A WIR
- A WBY
- A WBR

Figure 1 illustrates the standard components of the IEEE 1500 wrapper.

6.2 Wrapper serial port (WSP)

The WSP terminals serve as the primary interface to the IEEE 1500 wrapper. This set of serial terminals could be sourced from chip-level pins, ~~or from~~ an embedded controller such as an IEEE 1149.1TM-based controller, [or a serial TAM](#) as described in Annex C. The WSP is used to load and unload instructions and data into and out of the IEEE 1500 registers. In addition to the wrapper serial input (WSI) and wrapper serial output (WSO [and optional WSOR](#)) terminals shown in Figure 1, the WSP contains wrapper serial control (WSC) terminals used to control the operation of all IEEE 1500 registers.

The WSP is described with further details in Clause 8.

6.3 Wrapper parallel port (WPP)

The WPP is a user-defined set of wrapper terminals providing a parallel interface to the IEEE 1500 wrapper. These terminals are used when the wrapper is configured into parallel mode. The WPP terminals consist of the wrapper parallel input (WPI) terminal(s), wrapper parallel output (WPO) terminal(s), and wrapper parallel control (WPC) terminals. The set of WPC terminals may include elements of the set of WSC terminals ([i.e., wrapper clock \[WRCK\], auxiliary clock \[AUXCK\], wrapper reset \[WRSTN\]](#)). The WPP is described with further details in Clause 9.

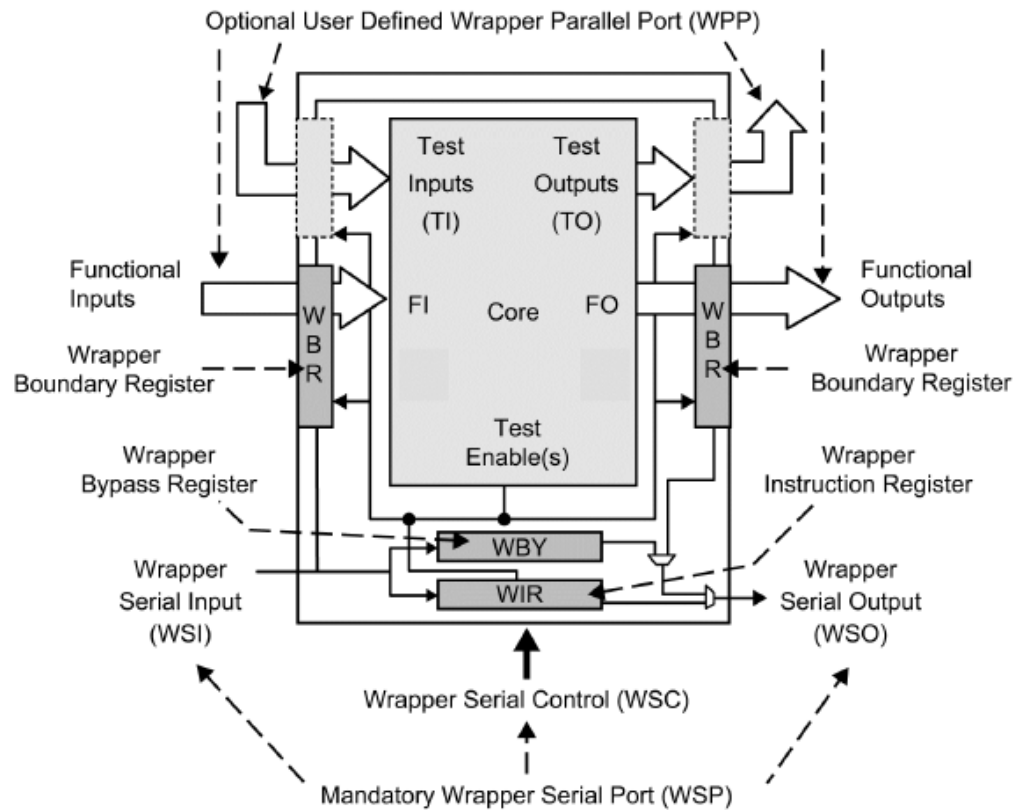


Figure 1 — Standard IEEE 1500 wrapper components

6.4 Wrapper instruction register (WIR)

The WIR enables all IEEE 1500 wrapper operations. This register is loaded via the WSP with instructions that select an IEEE 1500 data [scan path](#) register. The WIR can optionally be interfaced to the core for establishing test mode or functional operation. The WIR is described with further details in Clause 10.

6.5 Wrapper bypass register (WBY)

The WBY provides a bypass path for the WSI-WSO(R) terminals of the WSP. The WBY is the default data register between WSI and WSO(R) and should be selected by the current wrapper instruction when no other data register is selected. The WBY is intended to provide a minimum length scan path through the wrapper, so that when several IEEE 1500 wrappers are serially chained together in a system on chip (SoC), the wrappers that do not require a data register to be accessed can be bypassed with a short scan path through their WSI--WSO(R) terminals. The WBY is described with further details in Clause 11.

6.6 Wrapper boundary register (WBR)

The WBR is the data register through which test data stimuli are applied and pattern responses are captured. This register allows internal testing of the core, as well as testing of external connectivity to other cores and SoC integration circuitry, in response to an instruction loaded into the WIR. The WBR is described with further details in Clause 12.

7. Wrapper ~~WIR~~ instructions

7.1 Introduction

The WIR allows instructions to be serially entered into the wrapper circuitry via the WSI and the WSC. This clause defines the minimum set of instructions that shall be supplied and the operations that occur in response to those instructions. Optional instructions and the resulting operation of the wrapper circuitry are also defined, together with the requirements for extensions to the instruction set defined in this standard. The tests applied using IEEE 1500 instructions ~~can are to be completely~~ specified with CTL or a combination of CTL and another IEEE TTSC (or successor) sponsored standard test data language. The CTL identifies attributes of the wrapped core such as the following:

- Location and number of core wrapper input and output terminals interfacing with external circuitry;

- Protocol used to input and output the wrapper scan data,
- Condition or state of the core input and output terminals during the external circuitry test, excluding the test input (TI) and test output (TO) terminals and the terminals interfacing with the external circuitry ~~and~~
- Configuration of the WBR~~.~~

The IEEE 1500 wrapper has various modes of operation. There are modes for functional (nontest) operation, inward facing (IF) test operation, and outward facing (OF) test operation. Different test modes also determine whether the serial test data mechanism (WSI–WSO[\[R\]](#)) or the parallel test data mechanism (WPI–WPO), if present, is being utilized.

Instructions loaded into the WIR, together with the IEEE 1500 wrapper signals, determine the mode of operation of the wrapper and possibly the core itself. [In addition, the selected data register path can be determined by the contents and operation of the data register itself, or other data registers.](#)

There is a minimum set of instructions and corresponding operations that shall be supplied. Optional instructions and their corresponding behavior are also defined, together with the requirements for extension of the instruction set. All instructions that establish test modes that utilize the parallel port WPI and WPO are optional, as the presence of this port is optional. Furthermore, IEEE Std 1500 also allows for user-defined instructions.

IEEE Std 1500 has a set of instructions that are defined to use only the serial interface (WSP) and a corresponding set of instructions that are defined for the parallel interface. IEEE Std 1500 [shall](#) ~~must~~ allow accessibility to test the core. There is one main core test instruction—Wx_INTEST (user-specified core-test instruction)—that is flexible enough to allow any core test to execute. IEEE Std 1500 does not define the test for the core, and this very flexible instruction was specified so that it could be defined differently for each core test. There are two other instructions that are mandatory: an instruction for functional mode (WS_BYPASS) and an instruction for external test mode (WS_EXTEST). WS_BYPASS puts the wrapper into the bypass configuration and allows access to all functional terminals of the core. WS_EXTEST is the serial EXTEST configuration of the wrapper. Even if there is a WP_EXTEST mode (for parallel access), there [should](#) ~~must~~ still be a WS_EXTEST instruction capability.

7.2 Response of the wrapper circuitry to instructions and data registers

7.2.1 Specifications

Rules

- a) Each instruction shall select the wrapper register(s) (WRs) that operate while the instruction is active.
- b) WRs and WR segments that are not selected by the active instruction shall be controlled so that operation of selected WRs, selected WR segments, and ~~they do not interfere with the~~ operation of the core circuitry can continue unhindered ~~or the selected WRs~~.
- c) Each instruction shall cause a WR path to be selected to shift data between WSI and WSO(R) ~~with the proper WSC signals~~.
- d) During parallel and serial instructions, the proper WSC signals shall be used to shift data between WSI and WSO(R).
- e) The data register path selected between WSI and WSO shall ~~be of fixed length for each instruction, and the length shall~~ remain constant for the duration of the Shift operation while the instruction is active.
- f) If a data register path configuration is controlled by an inline segment insertion bit (SIB); the update of the segment insertion select signal shall not change until at least one-half clock cycle after the data register Update.
- g) The WBY shall be selected to shift data between WSI and WSO(R) while parallel instructions are active.
- h) Data that are loaded per an instruction into the WBR and to be used by a subsequent instruction shall ~~must~~ be held until the subsequent instruction becomes active.

Permissions

- i) The WR path ~~{of rule 7.2.1(e)}~~ between WSI and WSO(R) may comprise one or more serially connected wrapper data registers (WDRs), ~~or~~ core data registers (CDRs), or data register segments.
- j) As the result of a data register Update operation, data register path configurations may change.
- k) A user-defined wrapper instruction may enable an alternate mechanism for access to WDRs and/or CDRs.

7.2.2 Description

The instructions loaded into the WIR are interpreted in order to achieve two key functions: configure the WDRs and control the mode of the core. The active instruction identifies the wrapper or CDRs that ~~may~~ operate while the instruction is active. Several wrappers ~~wrapper~~ or CDRs can ~~may~~ be set into test modes simultaneously. Also, an instruction identifies the WR path that is used to shift data between WSI and WSO(R) during scan operation. The selected data register paths can also be controlled by data registers, e.g., SIBs. A particular instruction, or data register contents, can ~~may~~ result in one or more wrapper or CDRs being serially connected between WSI and WSO(R) or WPI and WPO. Further, the active instruction can ~~may~~ select one or more other registers, separate from the register(s) between WSI and WSO(R) or WPI and WPO, to perform other test functions. An unselected ~~Nonselected~~ wrapper or CDRs should be controlled so that they do ~~will~~ not interfere with the operation of the core circuitry or with the operation of a WR path. Rule 7.2.1(~~h~~ ~~f~~) allows for interdependent instructions such as WS_PRELOAD and WS_CLAMP to properly establish and use setup data.

During all instructions, there shall always be at least one register between WSI and WSO. During parallel and serial instructions, the WSC shall control the shift path between WSI and WSO(R). Hybrid instructions do not require that the WSC be used to control the shift path between WSI and WSO(R).

Permission 7.2.1(i) and 7.2.1(j) allows data registers (i.e., WDRs and CDRs) to be segmented and/or accessed hierarchically. This may be done by adding SIBs or other scan control bits that, when updated, cause the selected data register path configuration to change. Note that in order to access such data register segments, a sequence of multiple instructions, multiple shift, and update operations might be required.

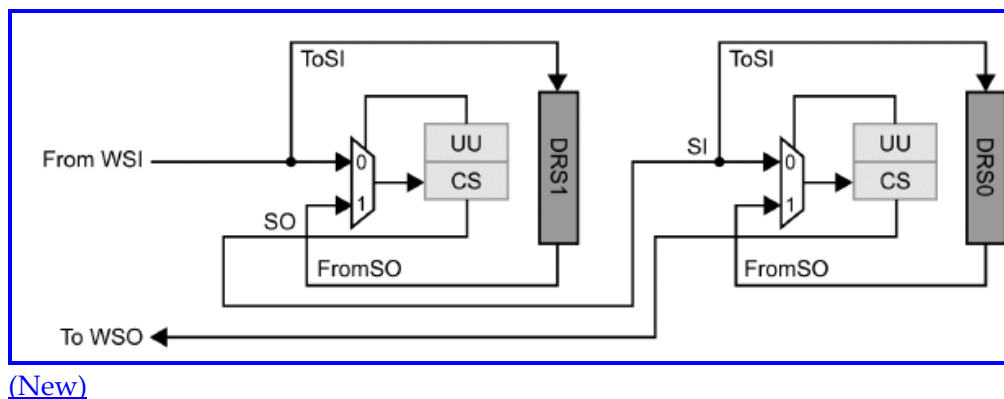


Figure 2 – DR segments accessed by inline SIBs

Figure 2 shows an example of a segmented data register that is accessed using two inline SIBs. Each of the two data register segments, DRS1 and DRS0, are either selected to be in the WR path, or bypassed through their respective SIB, depending on the Update value of the SIB.

Rule 7.2.1(e) requires that the selected data register path remain constant during the Shift operation, the path configuration need not change until the Update operation, per permission 7.2.1(j). Rule 7.2.1(f) requires that when using inline SIBs, the segment select signal shall not change until at least one-half clock cycle after the data register Update. This is required to prevent the inline SIB updating the select too early and causing the controlled data register segment to deselect before it updates its data register. Note that the two SIBs are in the data register scan path, even when both data register segments are unselected. See Figure 53 for an example of an inline SIB example. Also reference IEEE Std 1687 for other aspects and examples of SIB designs.

[In Figure 2, CS is a single storage element for the shift and capture events, and UU is two storage elements, for the update event and the delay of the segment select. Figure 53 also uses a second update storage element to delay the data register path select. The second update storage element can be clocked with the subsequent rising or falling edge of the WRCK or AUXCK clock, following the data register update from the assertion of the WSP UpdateWR signal. It is also possible to satisfy rule 7.2.1\(f\) by adding at least one-half clock cycle delay instead of a second storage element.](#)

The standard instructions are defined in Table 1. Only the WSP is utilized during the serial instructions.

Table 1 —Instruction list

Instruction	Mandate	Description
WS_BYPASS	Required	Allows normal (functional) mode and puts the wrapper into bypass mode.
WS_EXTEST	Required	Allows external test using a single chain configuration in the WBR.
WP_EXTEST	Optional	Allows external test using a multiple scan chain configuration in the WBR.
Wx_EXTEST	Optional	A user-specified external test instruction.
WS_SAFE	Optional	Puts the core into a quiet mode and outputs a predefined static (safe) state from all output ports. It also puts the WBR into bypass mode.
WS_CLAMP	Optional	Outputs a programmable static (safe) state from all output ports. It also puts the wrapper into bypass mode. This is presumed to be preceded by either a Wx_PRELOAD instruction or a Wx_EXTEST instruction.
WS_PRELOAD	Conditionally required ^a	Loads data into the single silent shift path of the WBR.
WP_PRELOAD	Optional	Loads data into the multiple silent shift paths of the WBR.
WS_INTEST_RING	Optional ^b	Allows internal testing using a single chain configuration in the WBR.
WS_INTEST_SCAN	Optional ^b	Allows internal testing by concatenation of the wrapper chain with a single internal chain.
Wx_INTEST	Required	A user-specified core test instruction.

[\(New\)](#)

Instruction	Mandate	Description
WS_BYPASS	Required	Allows normal (functional) mode and puts the wrapper into bypass mode.
WS_EXTEST	Required	Allows external test using a single chain configuration in the WBR.
WP_EXTEST	Optional	Allows external test using a multiple scan chain configuration in the WBR.
Wx_EXTEST	Optional	A user-specified external test instruction.
WS_SAFE	Optional	Puts the core into a quiet mode and outputs a predefined static (safe) state from all output ports. It also puts the WBR into bypass mode.
WS_CLAMP	Optional	Outputs a programmable static (safe) state from all output ports. It also puts the wrapper into bypass mode. Preceded by a Wx_PRELOAD instruction.
WS_PRELOAD	Conditionally required ^a	Loads data into the single silent shift path of the WBR.
WP_PRELOAD	Optional	Loads data into the multiple silent shift paths of the WBR.
WS_INTEST_RING	Optional ^b	Allows internal testing using a single chain configuration in the WBR.

(Deleted)

Table 1—Instruction list (continued)

Instruction	Mandate	Description
WS_INTEST_SCAN	Optional ^b	Allows internal testing by concatenation of the wrapper chain with a single internal chain.
Wx_INTEST	Required	A user-specified core test instruction.

(Deleted)

^a WS_PRELOAD is required only if there is a silent shift path in the WBR.

^b This instruction is also considered a Wx_INTEST instruction, just more completely (specifically) defined. This instruction can be used as the required Wx_INTEST instruction.

7.3 Wrapper instruction rules and naming convention

7.3.1 Introduction

There is a naming convention for the instructions.

W<Parallel/Serial/Hybrid> _<Mode>_<Configuration> (e.g., WS_INTEST_SCAN)

- **W:** Prefaces all standard IEEE 1500 instructions.
- **Parallel/Serial/Hybrid:** An S denotes a serial mode instruction. A P denotes a parallel mode instruction. An H denotes a hybrid instruction in which both the serial and parallel data ports are utilized. Note that a standard parallel instruction (e.g., WP_EXTEST) ~~shall~~ **must** have the WBY between WSI and WSO(R) but remain distinct from a hybrid instruction. An instruction by which any register other than WBY is configured between WSI and WSO(R) and that also uses the parallel port is classified as a hybrid instruction, except for the WP_EXTEXT instruction [see permission 7.6.2(j)].
- **Mode:** A shortened description of the instruction mode, such as BYPASS, PRELOAD, etc.
- **Configuration:** A shortened description of the configuration selected by a particular instruction. For instance, during the serial instructions WS_INTEST_SCAN and WS_INTEST_RING, SCAN denotes that internal scan chains are included in the single scan chain between WSI and WSO(R). RING indicates that only the wrapper chain is between the WSI and WSO(R).

7.3.2 Specifications

Rules

- a) All serial instructions (e.g., WS_ ~~INTEST~~ ~~EXTEST~~) shall use the WSP solely, except for WP_EXTEST [see permission 7.6.2(j)].
- b) All parallel instructions (e.g., WP_EXTEST) shall use the WPP solely.
- c) At least one ~~INTEST~~ instruction with mode INTEST shall be provided ~~is required~~.
- d) Each instruction name in Table 1 shall be used for the standard instruction associated with it.
- e) No user instruction shall utilize the standard instruction names.
- f) When Wx is stated in this standard, the x shall be replaced with P, S, or H.
- g) The WS_EXTEST instruction shall be provided ~~is required~~.
- h) The WS_BYPASS instruction shall be provided ~~is required~~ and shall be used during any operation where the wrapper is disabled.
- i) All standard instructions shall have a unique opcode.
- j) Standard instruction names, ~~such as WP_EXTEST~~, defined in this standard, such as WP_EXTEST, shall not be reused as names for instructions with functionality other than the functionality described in this standard. The standard instruction names are reserved.

Recommendations

- k) User and private instructions should follow the same naming convention as the standard instructions.

Permissions

- l) User-defined instructions may share opcodes with standard instructions.

- m) Standard instruction names may be used as a prefix for other instructions defined by the user, such as WP_EXTEST_BIST.

7.3.3 Description

Rules 7.3.2 + (a) and 7.3.2 + (b) are meant to specify usage of the WSP and WPP for serial and parallel instructions. Additional instruction-specific description is provided in 7.4 through 7.14.

Where permission 7.3.2 + (l) is exercised, the corresponding user-defined instruction is expected to be identical or a superset of the standard instruction that has the same opcode as this user-defined instruction.

7.4 WS_BYPASS instruction

7.4.1 Introduction

The mandatory WS_BYPASS instruction enables the functional configuration of the wrapper. WS_BYPASS is selected when no test operation of that core is required and allows only the WBY to be selected. The WBY provides a minimum-length serial path between the wrapper's WSI and the WSO(R). This allows more rapid movement of test data to and from other core wrappers, provided the wrappers are connected serially.

7.4.2 Specifications

Rules

- a) Each wrapper shall provide a WS_BYPASS instruction.
- b) The WS_BYPASS instruction shall select the WBY to be connected for serial access between the WSI and WSO(R) of the wrapper.
- c) While the WS_BYPASS instruction is ~~active selected~~, the operation of the wrapper circuitry shall have no effect on the operation of the core circuitry.

- d) While the WS_BYPASS instruction is [active](#) ~~selected~~, all wrapper boundary cells that can operate in either system or test modes shall perform their system function.
- e) While the WS_BYPASS instruction is [active](#) ~~selected~~, the WSC shall be utilized for test control.

Permissions

- f) The binary code for the WS_BYPASS instruction may be selected by the wrapper designer.

7.4.3 Description

During testing of a particular core or cluster of cores on an SoC, it [might](#) ~~may~~ be inconvenient to drive data through the entire length of a WBR. Instead, the WBR is replaced by the WBY. This allows data to transfer through the wrappers of the cores that are not being tested, more directly to and from the wrappers of the cores that are being tested, provided these wrappers are serially connected. The bypassed cores will have the ability to continue to run in system mode. The WS_BYPASS instruction could be used for this purpose. This instruction is the active instruction after wrapper reset (WRSTN) is asserted. Figure [3](#) ~~2~~ shows an example of the configuration of the wrapper during the WS_BYPASS instruction. [In the figure, the WSP path goes from WSI, through Bypass to WSO, and is controlled by WSC.](#)