IEEE STANDARDS ASSOCIATION

IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks

IEEE Computer Society

Sponsored by the
Test Technology Standards Committee

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

IEEE Std 1149.6™-2015
(Revision of
IEEE Std 1149.6-2003)

IEEE Std 1149.6™-2015
(Revision of
IEEE Std 1149.6-2003)

IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks

Sponsor

Test Technology Standards Committee
of the
IEEE Computer Society

Approved 5 December 2015

IEEE-SA Standards Board

Abstract: IEEE Std 1149.1™ is augmented by this standard to improve the ability for testing differential and/or ac-coupled interconnections between integrated circuits on circuit boards and systems.

Keywords: ac-coupled signaling, boundary scan, circuit boards, differential signaling, IEEE 1149.6™, integrated circuits, interconnect test, printed circuit boards, test

This standard is dedicated to the memory of our friend and colleague, Carl Barnhart. This was Carl's standard from start to finish. Carl was our leader and mentor. We will dearly miss his passion, humor and vast knowledge.

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

PDF: ISBN 978-1-5044-0596-6 STD20753
Print: ISBN 978-1-5044-0597-3 STDPD20753

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 Advanced I/O Working Group Working Group had the following membership:

William Eklow, *Chair*
Carl Barnhart, *Vice Chair*
Kenneth P. Parker, *Editor*

John Braden
William Bruce
Christopher J. Clark
Heiko Ehrenberg
Peter Elias
Josh Ferry
Hongshin Jun
Siva Kumar Vijaya Kumar
Roland Latvala
Phillippe Lebourg
Adam W. Ley
Skip Meyers
Francisco Russi
Craig Stephan
Stephen Sunter
Anthony Suto
Brian Turmelle

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

Saman Adham
Bailarico Balangue Jr.
Carl Barnhart
John Braden
Susan Burgess
Juan Carreon
Keith Chow
Christopher J. Clark
Sourav Dutta
Heiko Ehrenberg
Peter Eijnden
William Eklow
Josh Ferry
James Grealish
Randall Groves
Peter Harrod
Werner Hoelzl
Noriyuki Ikeuchi
James Langlois
Roland Latvala
Philippe Lebourg
Adam W. Ley
Jeffrey Moore
Michael Newman
Nick S.A. Nikjoo
Kenneth Parker
Ulrich Pohl
Mike Ricchetti
Francisco Russi
Kapil Sood
Thomas Starai
Walter Struppler
Stephen Sunter
Anthony Suto
David Thompson
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 5 December 2015, it had the following membership:

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

Masayuki Ariyoshi
Ted Burse
Stephen Dukes
Jean-Philippe Faure
J. Travis Griffith
Gary Hoffman
Michael Janezic
Joseph L. Koepfinger*
David J. Law
Hung Ling
Andrew Myles
T. W. Olsen
Glenn Parsons
Ronald C. Petersen
Annette D. Reilly
Stephen J. Shellhammer
Adrian P. Stephens
Yatin Trivedi
Philip Winston
Don Wright
Yu Yuan
Daidi Zhong

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1149.6-2015, IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks.

History of the development of this standard

The development of this standard was started on 21 May 2001 by an ad hoc industry Working Group called by Agilent Technologies[1] and Cisco Systems. This group formulated this standard with the intention of handing it over to the IEEE for formal standardization when the underlying technology became understood.

The group adopted as its mission: To define, document, and promote a means for designing integrated circuits (ICs) that support robust boundary-scan testing of boards where signal pathways make use of differential signaling and/or ac-coupled technologies. This technology utilizes and is compatible with the existing IEEE Std 1149.1[2]. The goal is to upgrade the capabilities of IEEE Std 1149.1 to maintain the rapid and accurate detection and diagnosis of interconnection defects in boards and systems despite the fault-masking effects of differential signaling and the dc blocking effects of ac-coupled signaling.

The group first referred to itself as the "AC EXTEST" Working Group, but has since expanded its charter to consider topics now called "Advanced I/O."

Changes introduced by this revision

A summary of the changes includes the following:

- Deletion of Annex E content
- Changes driven by the 2013 update to IEEE Std 1149.1
- Added new level-detection behavior to the test receiver for ac-coupled channels
- Documentation of driver and test receiver analog parameters, including documenting programmability of those parameters
- Programmability of coupling capacitor shunts
- Documentation of non-compliance of certain pins to EXTEST_PULSE performance
- Addition of Procedure Description Language (PDL) routines for documenting the procedures to access programmable analog parameters
- IP Package support in Boundary-Scan Description Language (BSDL)
- New "AC" boundary register cell designs

This revision affirms what has been required from the previous version, and products that conform to the previous version are still compliant with this revision. There is one deletion: Annex E (informative) proposed an "Initialize" instruction. This topic has since been subsumed into IEEE Std 1149.1-2013.

Many of the changes found in this revision are in response to the major revision of the underlying IEEE 1149.1 standard that was released in 2013. This standard introduced new concepts, such as segmented data registers, the initialization data register, and register descriptions. These concepts, once adopted in new ICs,

[1] Now called Keysight Technologies.
[2] Information on references can be found in Clause 2.

will materially affect the design, and subsequent description in BSDL, for those devices that are to conform to this standard.

There are also changes that come from industry commentary and usage practices developed over the past decade. A known problem exists in the 2003 version of this standard for level-sensitive behavior when using the EXTEST instruction defined in IEEE Std 1149.1. Simply, in some cases it is impossible to ensure compatible drive and receive voltage levels. When a channel is guaranteed to be ac-coupled, and the only use for the EXTEST instruction is to detect a shorted capacitor, there is a new behavior defined that essentially performs a simple but robust continuity check. See option 2) of rule a) in 6.2.2.1.

A significant addition provided by this revision is the ability to provide for (and describe) variable, programmable parameters such as threshold and common mode voltages. This will assist the users of test generation and diagnosis tools in creating and debugging tests, where analog parameter mismatches could occur.

On-chip ac coupling capacitors can now have provisions to shunt them out of the input signal pathway under control of the test circuitry. Improvements are offered to the ability to test external (board level) capacitors for shorts and to compensate for mismatches in voltages between devices.

It is recognized in this revision that for some pins, it may not be possible for the EXTEST_PULSE instruction to transmit or receive data due to dynamic conditioning of the associated circuitry that is not fully completed by a single data pulse; that is, a longer set of transitions, such as that provided by EXTEST_TRAIN, may be required. Such pins that need this exception can now be identified in BSDL so that tools can make appropriate tradeoffs for them.

The 2013 revision of IEEE Std. 1149.1 also added a new PDL, which is used to describe how devices and their registers are used for various testing purposes. This revision requires the documentation of certain programmable features, when implemented in a device, in PDL.

The 2013 revision of IEEE Std. 1149.1 made significant additions to BSDL, which affect the BSDL representation of 1149.6 technology as described in Clause 7. Part of these changes are driven by the electronics industry's movement towards the usage of intellectual property packages, where portions of circuitry are bought and sold for inclusion in other IC designs. This means that, along with the circuitry description, any relevant test-oriented information about IEEE 1149.1 or IEEE 1149.6 implementations contained within such IP must also be provided in BSDL packages. Thus, significant changes for such support are found in Clause 7 — see 7.4 and 7.5, taking note of "Port Behavior" descriptions. New and updated examples for both BSDL and PDL appear in Clause 7.

Annex C contains information on ac boundary register cell designs. Two new cells [AC_40 and AC_41 (see C.9)] are documented there.

Updating BSDL to the new standard

Components compliant with the 2003 version of this standard will typically be compliant with this new version. However, the BSDL documentation must be updated. First, the BSDL will need to be updated to comply with the changes introduced in IEEE Std 1149.1-2013, including as a minimum:

- Changing all "Linkage" pin types to the new linkage and power pin types in the port declaration.
- Updating "Use" and component conformance statements to STD_1149_1_2013.
- Adding an <input spec> for all pins of type "input" in the Boundary-Scan Register description.
- Taking advantage of the optimal register structural descriptions (REGISTER_MNEMONICS, REGISTER_FIELDS, REGISTER_ASSEMBLY, etc.) and of the PDL language to document

programmable characteristics of the component that could not be documented in a standard form before. This would be particularly valuable when documenting any initialization built into the component and required for proper operation of the I/O and boundary tests.

Second, the BSDL will need to be updated to comply with the changes introduced by this standard:

— Updating AIO component conformance statement to STD_1149_6_2015.

— Adding to the AIO_Pin_Behavior attribute the new parameters for driver common-mode and peak-to-peak voltages, test receiver threshold and hysteresis voltages, bypass control of on-chip capacitors, and any ports that require the EXTEST_TRAIN instruction.

— Adding PDL procedures to document initialization of any programmable I/O parameters. This ties into the new initialization documentation introduced in IEEE Std 1149.1-2013.

A component compliant with IEEE Std 1149.1-2001 and IEEE Std 1149.6-2003 may have included proprietary capabilities for initializing I/O, including advanced I/O. If the capabilities are conformant to the new requirements of these standards, then they must be documented in order to be compliant. However, even if those proprietary capabilities are not conformant, if they can be made public by writing procedures in PDL using the reserved PDL procedure names where appropriate, doing so will improve the automation of test generation and execution.

Notice to readers

Those reading the PDF version of this document will notice many hyperlinks that allow the reader to jump to a referenced item. For example, a text reference to 4.1 will be a link to the subclause itself. Readers using this facility may return to their starting point by selecting the menu item View>Page Navigation>Previous View, or by adding a “Previous view” page navigation tool to the Toolbar in their PDF reader.

Also, the PDF version of this document contains bookmarks of major headings and page thumbnails, also helpful in navigating the document.

Contents

IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks

1. Overview

1.1 Scope

This standard defines extensions to IEEE Std 1149.1™ to standardize the boundary-scan structures and methods required to help ensure simple, robust, and minimally intrusive boundary-scan testing of advanced digital networks.[1] Such networks are not adequately addressed by existing standards, especially for those networks that are ac-coupled, differential, or both. Testing enabled by this standard will operate in parallel with IEEE Std 1149.1 testing of conventional digital networks and in conjunction with IEEE Std 1149.4™ testing of conventional analog networks. This standard also specifies software and Boundary-Scan Description Language (BSDL) extensions to IEEE Std 1149.1, which are required to support new I/O test structures.

1.2 Purpose

Existing boundary-scan test standards (IEEE Std 1149.1, IEEE Std 1149.4) do not fully address some of the increasingly common, newer digital network topologies, such as ac-coupled, differential interconnections on very high speed (1+ Gb/s) digital data paths. IEEE Std 1149.1 structures and methods are intended to test static (dc-coupled), single-ended networks. They are unable to test dynamic (ac-coupled) digital networks, since ac coupling blocks static signals. Differential networks are also inadequately tested by

[1] Information of references can be found in Clause 2.

IEEE Std 1149.1, which requires either the insertion of boundary cells between the differential driver or receiver and the chip pads (this often creates an unacceptable performance degradation), or insertion of single boundary cells before the differential driver and after the differential receiver (this reduces controllability and observability to the point that many board assembly defects cannot be detected). IEEE Std 1149.4 structures and methods are intended for testing analog networks, and in most cases are not able to test these newer digital networks as well. Specifically, IEEE Std 1149.4 provides the opportunity to inject dynamic (time-varying) or analog signals for test, but these structures intended for analog testing are often too intrusive (too high an impact on performance and pin count) for high speed chip designs, and require additional resources and test application time not otherwise required for testing digital circuits. Finally, very high-speed logic imposes new restrictions on test structures that were not considered in IEEE Std 1149.1.

1.3 Organization of the standard

Clause 1, Overview, provides an overview and context for this standard.

Clause 2, Normative references, provides references necessary to understand this standard.

Clause 3, Definitions and acronyms, defines terminology and acronyms used in this standard.

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

Clause 5, Instructions, provides rules for instructions used for testing.

Clause 6, Pin implementation specifications, provides rules for I/O pin implementation.

Clause 7, Conformance and documentation requirements, provides rules for conformance and documentation of devices designed to this standard.

Annex A,
(informative)
Applications and tools, shows how this standard is used in typical testing applications and how devices conforming to this standard can be verified before manufacture and tested in production.

Annex B,
(informative)
Noise rejection in edge-detecting mode, gives guidance for designing test receivers with noise rejection capabilities.

Annex C,
(informative)
Advanced I/O boundary-scan register cells, documents new boundary-scan register cells used by this standard.

Annex D,
(informative)
Test receiver design examples, shows how the input pins of some common logic families can be designed to conform to this standard.

Annex E,
(informative)
A proposed “INITIALIZE” instruction. This annex is omitted from this revision of this standard. The topic of test data initialization has been included in the revised IEEE Std 1149.1-2013.

Annex F,
(informative)
Bibliography

1.4 Context

Figure 1 shows a printed circuit board containing many types of devices. Of these, some could be compliant with IEEE Std 1149.1 for the support of testing activities. These devices contain boundary-scan testability circuitry, which allows them to participate in manufacturing tests that detect and diagnose faults such as open solder joints, shorts, and missing devices.

The additional testability elements added by this standard to these same integrated circuits (ICs) allow this interconnect testing, with enhanced coverage, to be conducted on differential signal pathways and/or where ac coupling (which blocks normal dc test signals) has been used on signal paths between ICs.

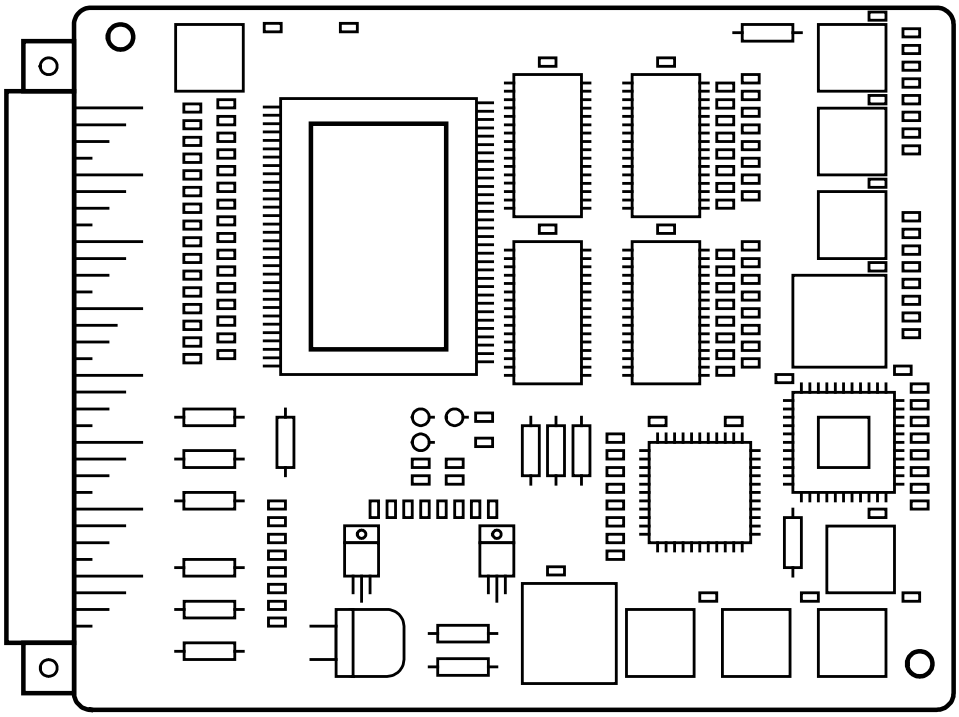


Figure 1—A printed circuit board containing a variety of components interconnected by printed wiring. Some ICs contain IEEE Std 1149.1 features that support boundary-scan interconnect testing

1.5 Objectives

The objective of this standard is to provide design guidance for testability circuitry added to an IC in addition to testability provisions specified by IEEE Std 1149.1, such that when an IC contains differential signaling and/or is ac-coupled with other ICs compliant to this standard, board and system level tests can be readily and accurately conducted with enhanced defect coverage.

Devices that adhere to this standard that are used in differential and/or ac-coupled signaling environments will realize significant savings in testing costs for boards and systems. Tools that are cognizant of the capabilities provided by this standard will be able to prepare, run, and interpret these tests in a highly automated fashion, with high diagnostic resolution.

This standard allows devices created by multiple vendors to operate together during testing despite the differing characteristics and parameters of the IC processes used to fabricate the devices. This standard also provides design guidance to board and system designers that will enhance the performance of the testability features of their products. This in turn will reduce system and production costs.

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™, IEEE Standard Test Access Port and Boundary-Scan Architecture.[2,3]

IEEE Std 1149.4™, IEEE Standard for a Mixed-Signal Test Bus.

3. Definitions and acronyms

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. [4]

3.1 Definitions

ac coupling: The use of series capacitance in a signal path. This coupling will block dc voltages on the drive side of the path from appearing on the receive side. Only the ac component of the driven signal will pass through the coupling, with the effect of high-pass filtering imposed on the original signal. *Contrast:* **dc coupling**.

ac pins: Advanced I/O pins that require a time-varying (ac) signal to permit testing of their interconnections. This includes all differential signal pins and differential or single-ended signal pins that are expected by design to support ac coupling. *Contrast:* **dc pins**.

NOTE—AC coupling could also be accomplished with transformers, which, as with capacitive coupling, form a high-pass filtered transmission structure. While the principles used and rules defined in this standard apply to transformer coupling, this coupling technology is less often used and is thus omitted to simplify discussion.[5]

[2] The IEEE standards or products referred to in Clause 2 are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.
[3]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[4]*IEEE Standards Dictionary Online* subscription is available at: http://ieeexplore.ieee.org/xpls/dictionary.jsp
[5] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

ac test mode: A test mode that enables boundary-scan testing between ac pins that are ac-coupled or dc-coupled. Testing dc-coupled ac pins in ac test mode could enable testing that cannot be supported in dc test mode due to voltage level incompatibilities. *Contrast:* **dc test mode**.

ac test signal: A signal generated by the ac test mode that is used to modulate static test data into a time-varying signal that can pass through ac coupling. A test receiver and detector are used to recover the static test data value from within the time-varying signal.

ac testing: Testing ac pins using an ac test mode.

advanced I/O: Input/output (I/O) circuits that are designed to convey digital information, the interconnections of which cannot, either by design or by common usage, be adequately tested with static digital signals such as those provided for in IEEE Std 1149.1. For example, such an I/O pin could be self-referenced (i.e., to its own average voltage), or referenced to another I/O pin, rather than to a fixed voltage.

bias: A high-impedance (relative to line and termination impedance, typically >1000 Ω) voltage source often used on the input of a mission receiver to cause it to output a deterministic state in the absence of an input signal, and/or to select the common-mode voltage seen by a differential receiver in ac-coupled signal paths.

bias network: A network of impedances, usually higher valued than termination impedances and usually located in or near the receiver, used to establish a common-mode or reference voltage.

boundary-scan testing: Testing of interconnections between IC pins as supported by IEEE Std 1149.1 and this standard. This testing technology looks for manufacturing defects along signal paths, which include open solder joints, broken bond wires, shorted signal traces, damaged drivers and receivers, etc. Tests are performed on many paths in parallel.

channel: A signal path or set of signal paths that transmits a single data stream from a source to a destination. *See:* **differential signaling**; **single-ended signaling**.

characteristic impedance: The ratio of the complex voltage and complex current of a signal traveling forward on a conductive path. A signal path is often terminated with an impedance that matches the characteristic impedance of the path. This makes the path appear to be infinitely long and prevents signal degradation due to reflections that occur at unterminated ends of the path. *See:* **termination**.

common-mode noise: A noise signal added equally to both signal paths in a differential signal channel. Common-mode noise will affect or completely disrupt a single-ended measurement of a signal on one leg of a differential receiver, yet this differential receiver will accurately recover the signal within the noise.

common-mode range: The range of common-mode voltage that a differential receiver is capable of receiving while maintaining reliable signal recovery. A differential signal with common-mode voltage within this range will be received correctly. Outside this range the receiver could fail to recover the data signal.

common-mode voltage: The offset from ground of the mean of the maximum and minimum voltages that appear on a pair of differential signals. A differential driver will, by its operational characteristics, define a common-mode voltage. A differential receiver will properly receive data over a range of common-mode voltages, but will likely have an optimal common-mode voltage where its performance is best. When the optimal common-mode voltage of a receiver is significantly different than that of its associated driver, ac coupling with appropriate bias can be used to match the two components of a differential channel.

NOTE—Common-mode voltage is discussed extensively in Clause 4 and 6.2.

comparator: An amplifier with two inputs labeled positive and negative, typically with very high input impedance. The amplifier usually has very high gain and produces an output signal that is the amplified difference of the positive and negative input signals. For all but the smallest differences, the output will be Vmax or Vmin, which are the most positive and most negative voltages the amplifier can produce on its output. A comparator can be used as a differential receiver. A comparator can be used to determine if an input signal is logically above or below a reference voltage.

current signaling: A signal encoded by the amplitude and direction of current flow. In a differential pair, a current signal is positive when current flows from positive to negative legs, and negative in the reverse direction. The voltage that appears on these same legs does not carry information. *Contrast:* **voltage signaling**.

dc coupling: The use of simple wires or small series resistances in a signal path. *Contrast:* **ac coupling**.

dc pins: Single-ended pins that are dc-coupled. DC pins only need be equipped with test resources defined by IEEE Std 1149.1. Single-ended pins that are ac-coupled are not normally testable as a signal path, but might be testable as logically independent pins if there are enough test resources on each pin. *Contrast:* **ac pins**.

dc test mode: A test mode that enables traditional boundary-scan testing (as defined by IEEE Std 1149.1) between dc pins. *Contrast:* **ac test mode**.

defect: A defect is an unacceptable deviation from a norm For example, an open solder joint. Because it is unacceptable, some remedial action is needed. *See:* **fault**; **manufacturing process defect**.

deprecated: Used in this standard for possible configurations or modes of operation that might not operate reliably and should be either avoided or treated with special care. *Synonyms:* **disapproved**, **belittled**, **discouraged**, **disparaged**.)

derived voltage reference: A voltage reference derived from: a) other references, such as a resistive divider between power and ground, or b) a resistive divider between two differential signals that recovers the common-mode voltage of the signals.

differential driver: A driver that accepts a single data stream and drives it onto two independent signal paths where one signal is the inverse of the other. The two signals are centered at the common-mode voltage.

differential receiver: A receiver that recovers a single data stream encoded differentially on two signal paths. It effectively subtracts the signal on its negative leg from that on its positive leg. This eliminates common-mode noise appearing on both legs.

differential signaling: The use of two independent signal paths in a channel to carry a single data stream where one path carries an inverted copy of the signal that appears on the other path. The original data signal can be reconstructed by taking the difference of the two signals. There is no reliance on a reference voltage for determining this signal. This has the property of eliminating common-mode noise in the transmitted signal. *Contrast:* **single-ended signaling**.

encoding protocol: A stream of data bits encoded into a new (typically longer) data stream that has characteristics favorable for its transmission on a channel. The encoded stream might have added redundancy to support error correction. The encoded stream could have extra bits added to deliberately increase the number of transitions that appear in the stream, effectively raising its apparent frequency and facilitating data transmission that encodes clocking information into the stream.

fault: A fault is a physical manifestation of a defect. For example, an open solder joint (a defect) on the input to an IC could cause one or more of its outputs to produce incorrect data (a physical manifestation). In many cases, a fault and its causal defect are not co-located.

float: The input to a receiver that is connected to an undriven signal, or a high-impedance connection to a receiver input, is said to float.

frequency: The number (f) of transition pairs that occur on a signal path in a period of time, expressed in Hertz (cycles per second). With respect to ac coupling, a frequency is high when the period ($1/f$) is small compared to the time constant of the coupling. A frequency is low when the period is large compared to the time constant of the coupling. The frequency appearing on a signal path could vary greatly over time as a function of the data being transmitted and the data encoding protocol.

high-pass filter: An electrical network that passes higher frequencies and attenuates lower frequencies (dc current is blocked).

HP_Mult: High-Pass Multiplier, a multiplier used to derive the minimum high-pass coupling time constant.

NOTE—See 6.2.3.1 and 6.2.3.3.

HPLP_Ratio: High-Pass-Low-Pass Ratio, a multiplier equal to the ratio of the HP time constant to the LP time constant, and used to derive the minimum ratio of high-pass coupling time constant (T_{HP}) to edge-detection filter time constant (T_{LP}).

NOTE—See 6.2.3.1 and 6.2.3.2.

hysteresis: From magnetics: lagging in the values of resulting magnetization in a magnetic material (such as iron) subjected to a changing magnetizing force. In this standard, hysteresis refers to the memory of an input state to an amplifier or buffer after that state is removed but before a different input state is applied. Typically, there is a hysteresis threshold that defines the difference between “no input” and “input.” As applied to electronics, a digital output circuit such as a comparator where the output switches to one output state when the input is above one level and switches to the opposite output state when the input is below a lower level, and the output does not switch at any intermediate level. For example, a buffer produces a high output when a voltage above 0.5 V is applied, produces a low output when a voltage below 0.3 V is applied, and does not change its output for voltages between 0.3 V and 0.5 V.

A hysteresis symbol in a buffer symbol is shown in Figure 2.

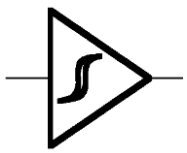


Figure 2—A hysteresis symbol in a buffer symbol

hysteretic: Adjective form of hysteresis, as in “hysteretic amplifier.”

hysteretic voltage: Given a specified threshold voltage where a logical interpretation of signal would change from 0 to 1 (or vice versa), a hysteretic voltage is a band of values on either side of the threshold where the logical interpretation will not change from its current interpretation. A signal that exceeds (in

either direction) the threshold voltage by more than the hysteretic voltage determines the current logical interpretation of the signal.

NOTE—Hysteretic voltage appears as V_{Hyst} in 4.7 and as V_{Hyst_Edge} and V_{Hyst_Level} in 6.2.1.

interconnect test: An IEEE Std 1149.1 boundary-scan test designed to detect and diagnose defects in the interconnection wiring between ICs. This standard extends the concept to include the testing of channels, where single-ended and differential signaling, and dc- or ac coupling could exist.

load termination: A termination placed at the far end (away from the driver) of a signal path used to match the characteristic impedance of the path. *Contrast:* **source termination**.

low-pass filter: An electrical network that passes lower frequencies, including dc levels, and attenuates higher frequencies.

LP_Mult: Low-Pass Multiplier, a multiplier used to derive the minimum edge-detection (usually a low-pass) filter time constant.

NOTE—See 6.2.3.1 and 6.2.3.3.

manufacturing process defect: A defect that is an unacceptable by-product of the manufacturing process. For board manufacture these include missing devices (ICs, resistors, capacitors, etc.), improperly mounted devices (e.g., rotated 180º), open solder joints, shorted solder joints, misaligned devices, and incorrect and dead devices.

mission logic: The circuitry inside an IC that performs its primary design function. *Contrast:* **test logic**.

mission mode: An operational mode in which a device performs its primary design function. *Contrast:* **test mode**.

negative leg: The signal path of a differential signal pair that has the opposite polarity as the original data signal.

null: The input state where the two inputs to a differential receiver that are supposed to be different (complementary) are instead receiving essentially the same value.

offset voltage: A constant dc voltage added to an ac signal.

operational modes: A device could function in several modes. For the purposes of this specification, two primary modes are considered. *See:* **mission mode**, **test mode**.

peak-to-peak voltage: The maximum voltage minus the minimum voltage appearing on a waveform. This is always a non-negative voltage.

NOTE—Peak-to-peak voltage appears as ΔV_{Min} in 6.2.1.

positive leg: The signal path of a differential signal pair that has the same polarity as the original data signal.

reference voltage: A low-impedance voltage source typically used to define a threshold for comparing signals. The low-impedance characteristic means it is resistant to conducting noise signals. *Contrast:* **bias**.

referenced termination: A termination for a differential channel where the two legs are both terminated to a reference voltage. This reference has an impedance that is low relative to the termination impedance, such that the two legs are independent. *Contrast:* **bias network**.

self-referenced comparison: The comparison of a signal with a delayed, averaged version of the same signal, used to detect signal transitions. This process does not need a static reference voltage to find a transition in a signal.

signal path: An electrical pathway formed by a simple conductor, a terminated pathway containing a series resistance, or an ac-coupled pathway containing a series capacitance that transmits a signal from a driver to a receiver.

signal reflection: A signal wave front traveling across a discontinuity in the characteristic impedance of the signal path will have a fraction of its energy reflected in the opposite direction on the path. The reflection will be of either the same or opposite polarity and will add into the waveforms appearing on the path, impacting their shape. *See also:* **transmission line**.

single-ended signaling: The use of a single signal path in a channel to carry a data signal. The signal is referenced to a static reference voltage. *Contrast:* **differential signaling**.

slew rate: Rate of change in either direction of voltage, measured in units of volts per second.

source termination: A termination placed near the source driver of a signal to satisfy dc current requirements of a driver and/or to match the characteristic impedance of a transmission line structure to reduce signal reflections. *Contrast:* **load termination**.

termination: An impedance usually near the end of a signal path used to satisfy the electrical matching requirements of the characteristic impedance of the signal path and reduce signal reflections. The impedance is typically low, often 100 Ω or less. *See also:* **characteristic impedance**, **load termination**, **referenced termination**, **source termination**, **unreferenced termination**.

test logic: Testability logic defined by this standard and IEEE Std 1149.1. *Contrast:* **mission logic**.

test mode: An operational mode of the device in response to the EXTEST or an ac testing instruction whereupon the pins are driving or receiving test data as controlled by the test logic of the device. The device pins have been disconnected from the internal mission logic. *Contrast:* **mission mode**.

test receiver: A circuit, which can operate in ac test mode and dc test mode, and examines an incoming signal. Its purpose is to extract test data from a signal that might have been altered by dc level shifting or ac coupling decay.

threshold voltage: A voltage level used to determine if a signal has a logical interpretation of 0 or 1; if the signal is below the threshold, it is interpreted as 0, and it is interpreted as 1 if the signal is above the threshold. For systems with hysteresis, the final interpreted state of the current voltage also depends on how the signal evolved in time and the value of the hysteretic voltage.

NOTE—Threshold voltage appears as $V_{Threshold}$, discussed in 4.8 and 6.2.2.

time constant: Typically, the product of resistance and capacitance of an RC network (e.g., a high-pass filter) measured in seconds. One time constant is the time for a capacitor to discharge 63% of its voltage through a resistor. In ac-coupled systems, the termination resistance combined with the coupling capacitor forms a high-pass filter.

NOTE—In discussions comparing time periods to time constants, a period is significantly longer (shorter) than a time constant if it is five (one-fifth) times the time constant value. For example, as shown in Figure 3 using the five-time-constant rule, a signal will decay to 0.7% of its original value. The one-fifth-time-constant rule means that 81.9% of a signal still remains.

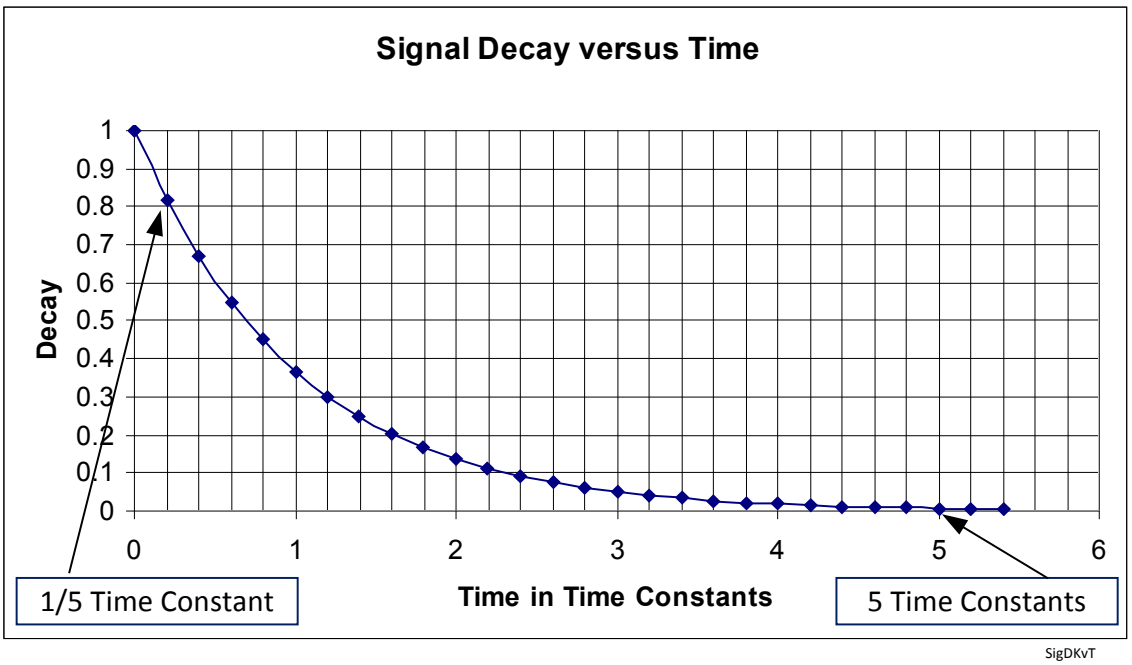


Figure 3—Signal decay versus time in units of time constants

transition: A voltage transition occurs when a signal traverses a specified voltage range in a specified time in either direction. *See:* **slew rate**.

transmission line: A signal path with a specific construction that produces a uniform, known characteristic impedance along its length. This minimizes degradation of a signal passing along this path that can result from impedance variations.

unreferenced termination: A termination for a differential channel where a termination impedance is connected between the two legs with no connection to a reference voltage.

voltage signaling: Signals are encoded by the voltage appearing on a wire compared to a reference voltage (single-ended) or the voltage difference appearing on a pair of wires (differential). *Contrast:* **current signaling**.

wrapper: In the context of this standard, a wrapper is composed of dedicated or shared test logic for a plurality of logic core terminals, providing each terminal with a control/observe capability via serially connected logic elements in test mode, and functional operation while in mission mode. Generally, a wrapper is a thin shell of circuitry around a core circuit that provides controllability and/or observability at the core inputs and outputs.

3.2 Acronyms

ASCI	application-specific integrated circuit
BSDL	Boundary-Scan Description Language

CML	current mode logic
CMOS	complementary metal-oxide semiconductor
HPLP_Ratio	High-Pass-Low-Pass Ratio
IC	integrated circuit
ICT	in-circuit test
IP	Intellectual Property
LP_Mult	Low-Pass Multiplier
LVDS	low-voltage differential signaling
LVPECL	low-voltage PECL (positive emitter-coupled logic)
PDL	Procedure Description Language (see IEEE Std 1149.1-2013, Annex C)
PLL	phase-locked loop
RC	resistor/capacitor
SERDES	serialize/deserialize
TAP	Test Access Port (from IEEE Std 1149.1)
TCK	test clock
Tcl	Tool Command Language (see IEEE Std 1149.1-2013, Annex C)
TDI	test data in
TDO	test data out
TDR	test data register
TMS	test mode select
TRST	test reset
VHDL	VHSIC High-Level Design Language
VHSIC	very high-speed integrated circuit

4. Technology

The presence of coupling capacitors in signal interconnections among chips, whether they are discrete devices mounted on a PC board or integrated inside an IC, prevents dc values from being driven between a driver and receiver. An ac boundary-scan methodology must therefore use a time-varying signal to pass through the ac coupling when in ac test mode.

Differential signaling is often used to increase signaling speeds and noise immunity, compared to single-ended signaling. Differential signaling, combined with termination schemes, can have significant defect-masking properties that reduce test effectiveness (see 4.3).

This clause contains tutorial discussions of ac coupling, differential signaling, and the effects of defects in such circuits needed to understand Advanced I/O testing technology. Rules based on these technologies are found in subsequent clauses.

4.1 Signal pin types

It is expected that a chip possessing pins requiring ac coupling will often also possess “normal” pins (i.e., intended to be dc-coupled). These dc pins would supply data and/or control to/from portions of the chip that do not require ac coupling. For test purposes, it is necessary that all pins be tested simultaneously with an EXTEST-like capability because that is how shorts (unwanted connectivity) between these pins are reliably detected and diagnosed. It is desirable for higher test throughput to test all pins simultaneously with an EXTEST-like capability as well.

NOTE—There are some issues related to simultaneously switching all outputs during the EXTEST instruction. These issues will be discussed in later clauses. The implementation of EXTEST on ac and dc pins will provide the needed shorts detection and minimize the noise due to simultaneous switching.

This standard will refer to dc and ac pins henceforth. DC pins are those for which IEEE Std 1149.1 provides adequate testing. AC pins are those Advanced I/O that require additional test resources in order to be adequately tested, including those pins intended for ac coupling or differential signaling.

AC pins are a principle target of this standard. IC designers implementing this standard are expected to identify such pins and add new test capabilities for them.

4.2 Signal coupling and coupling combinations

This subclause reviews a range of coupling options.

4.2.1 Single-ended dc

A basic, single-ended connection scheme is shown in Figure 4, along with the boundary-scan control and observation capability specified by IEEE Std 1149.1. This type of coupling has been quite common and is very testable using boundary-scan.

It is important to note that in a typical boundary-scan test, the time between launching a signal from a driver (at the falling edge of test clock (TCK) in the Update-DR or Update-IR TAP Controller state) and capturing that signal (at the rising edge of TCK in the Capture-DR TAP Controller state) is no less than 2.5 TCK cycles. Further, the time between successive launches on a driver is governed not only by the TCK rate, but by the amount of serial data shifting needed to load the next pattern data in the concatenated boundary-scan registers of the boundary-scan chain. Thus, the effective test data rate of a driver could be thousands of times lower than the TCK rate. For dc-coupled interconnect, this time is of no concern. For ac-coupled interconnect, the signal could easily decay partially or completely before it can be captured. If only partial decay occurs before capture, that decay will very likely be completed before the driver produces the next edge.

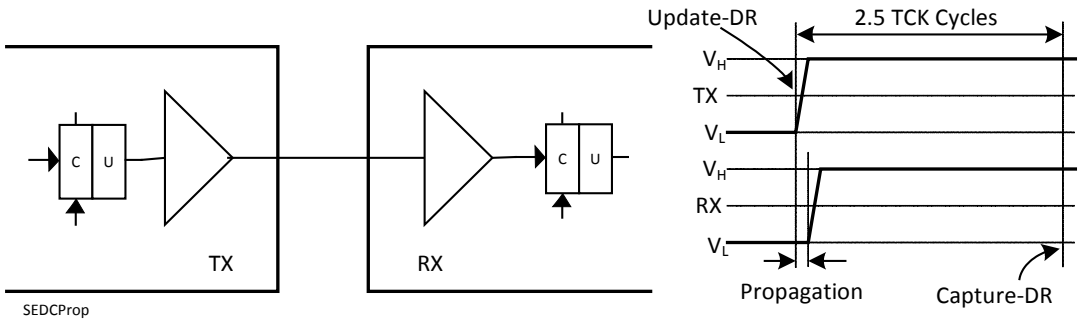


Figure 4— Basic single-ended signaling with boundary-scan control and observation

4.2.2 Single-ended ac

Figure 5 shows an ac-coupled single-ended connection (The termination resistor and voltage source shown could actually reside within the IC). The size of the capacitors used is determined by the mission requirements of the coupling and could vary widely across applications. While the devices may have been designed for dc coupling and actually contain boundary-scan resources, the ac coupling will block their operation. This interconnection configuration could only be tested at very high TCK frequencies or lower frequencies with very large coupling capacitors, and even then the results might be unreliable. TCK frequencies are likely to be significantly lower than the normal operating frequency of the channel being tested. Thus the data seen by the receiver could decay before it can be captured.

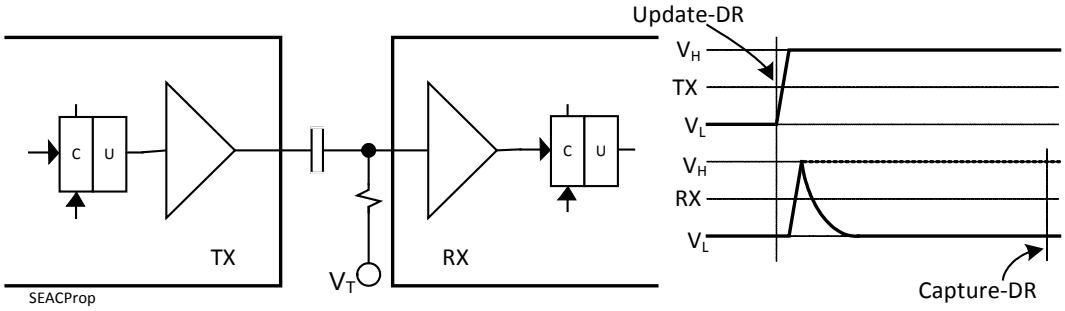


Figure 5— Basic single-ended signaling with ac coupling

In general, ac coupling can distort a signal transmitted across a channel depending on its frequency. For example, Figure 6 shows a channel transmitting a high and low frequency waveform across an ac channel and observed at the input of the receiver. The high frequency signal is relatively unaffected by the coupling. The low frequency signal is severely impacted. First, it decays to VT after a few time constants. Second, its peak-to-peak amplitude is double the input amplitude. A key item to note is that the transitions in the original signal are preserved, although their start and end points are offset compared to where they were in the high frequency case.

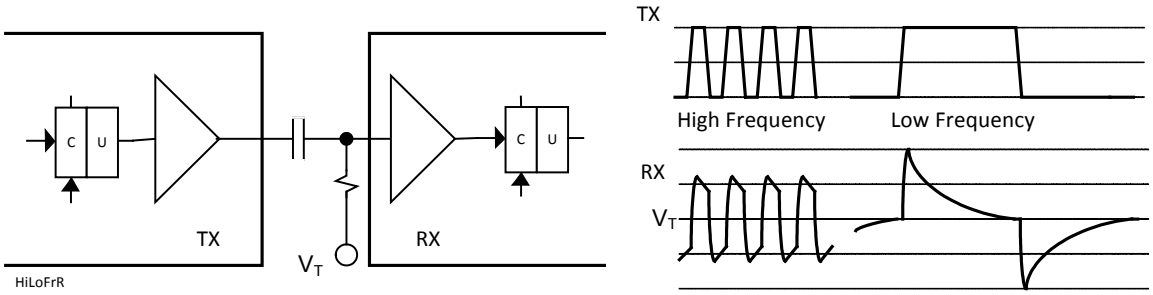


Figure 6— High- and low-frequency response of an ac-coupled channel

NOTE—The high- and low-frequencies in Figure 6 are relative to the ac coupling time constant. "High frequency" has a cycle time significantly less than the coupling time constant (< 1/6), and “Low frequency” has a cycle time significantly greater than the coupling time constant (> 6 times). For the remainder of this document, "High frequency" may be thought of as functional or mission mode behavior, while "Low frequency" may be thought of as test mode behavior clocked by TCK.

4.2.3 Differential dc

Figure 7 shows a basic differential dc-coupled signal path. The termination resistor might exist for impedance matching and/or source termination of the driver. The placement of boundary-scan resources is optional per IEEE Std 1149.1 in that they can be omitted altogether. The IEEE Std 1149.1 standard allows a designer to designate the differential signal path as “analog.” Then the digital-to-analog and analog-to-digital interfaces (optionally) can be provided with boundary-scan resources as shown in Figure 7. When this option is taken, it is then possible to test the analog signal path with boundary-scan algorithms. In this case, the signal path is viewed by the test logic as if it were single-ended, leading to diagnostic ambiguity and possible loss of test coverage (see 4.3).

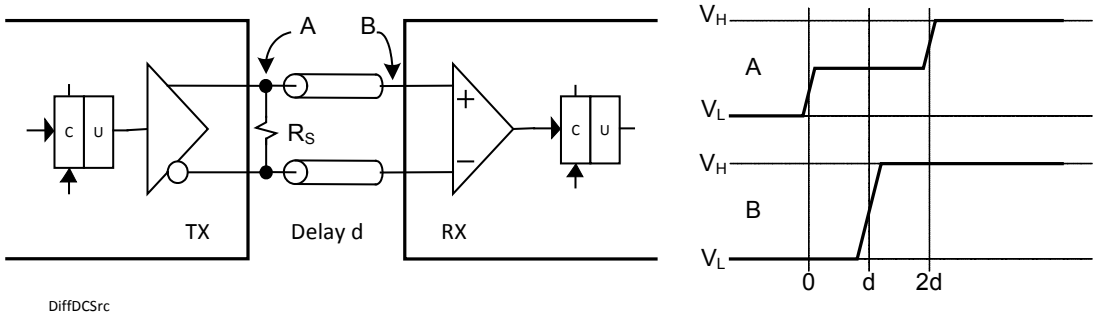


Figure 7—Basic differential signaling with dc coupling and source loading

The driver in Figure 7 could be producing a voltage signal, and the resistor is used to match the signal line impedance. Alternatively, the driver could be producing a current signal, where the direction of flow represents data, and the resistor is needed not only to match signal line impedance, but also to provide a dc current path to satisfy the driver’s requirements. This is called a source termination.

The driver in Figure 7 is looking into a termination R_S and transmission lines with characteristic impedance R_S as well. This forms a voltage divider which sends 1/2 the signal into the transmission lines. When the signal wave front reaches the receiver (after Delay d) its high-impedance does not match the characteristic impedance, which reflects the signal back down the lines toward the driver. This signal (1/2 the driver voltage) adds to the signal received so that the receiver perceives a full voltage swing. After the second transmission line Delay d, the reflected signal reaches the driver and brings the voltage seen there to

the full level. Thus a clean transition is seen at the receiver, but the signal seen at the driver is a two-step staircase. Since there is an impedance match at the driver, no new reflections occur.

In the case where the impedance R_S is moved to the receiver (to the right of the transmission lines), this is called load termination (and the resistor is renamed R_L) as shown in Figure 8. Now the driver is looking into the transmission lines with characteristic impedance R_L. The full waveform is transmitted (no divider is now present) and this edge propagates to the receiver. At the receiver, the termination resistor matches the line impedance, and thus, there is no reflection. The waveforms at both the driver and receiver are full transitions.

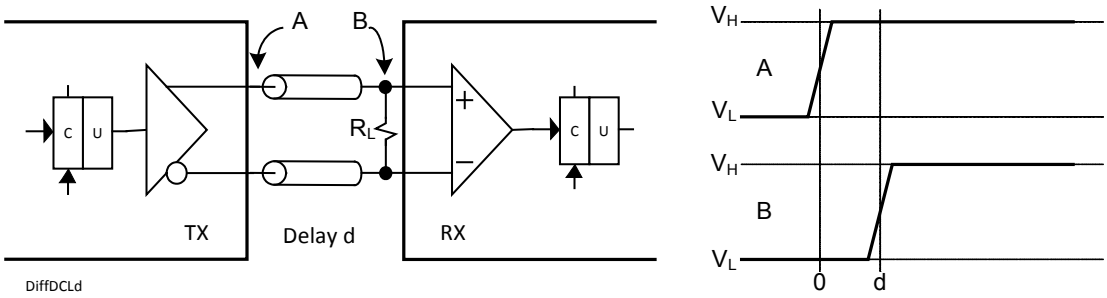


Figure 8—A dc-coupled driver and receiver with load termination

Figure 9 shows a driver/receiver pair that has been both source and load terminated. In this case, the voltage swing seen on both sides of the transmission line has been divided by two (note R_S equals R_L). This type of termination assures that in the case of an imperfect impedance match, the resulting reflections can be attenuated at both ends of the line.

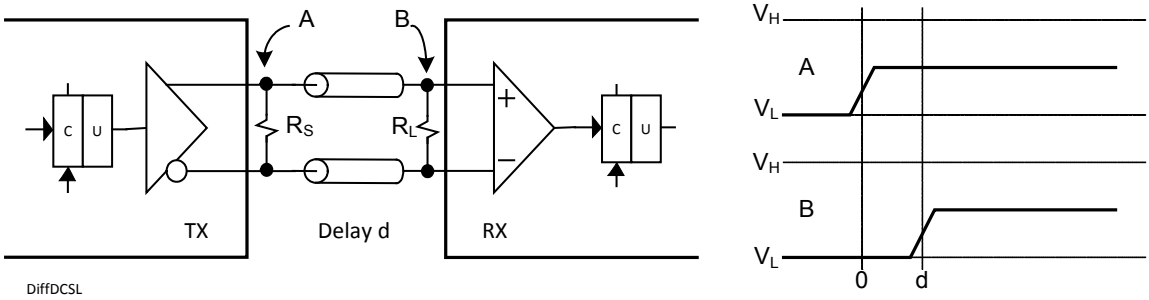


Figure 9—A dc-coupled driver and receiver with both source and load termination

4.2.4 Differential ac

Figure 10 shows ac coupling of a voltage driver and receiver whose voltage levels are assumed to be incompatible for dc coupling (the voltage levels used by the driver are too far removed from the acceptable common-mode range of the receiver). The bias network referenced to the common-mode voltage source V_{Bias} (the optimal common-mode point of the receiver) along with the dc blocking effect of the coupling capacitors forms a level shifter that allows this configuration to work properly. This enforced compatibility is a common reason why board designers use ac coupling. The size of the capacitors used is determined by the mission requirements of the coupling and could vary widely across applications.

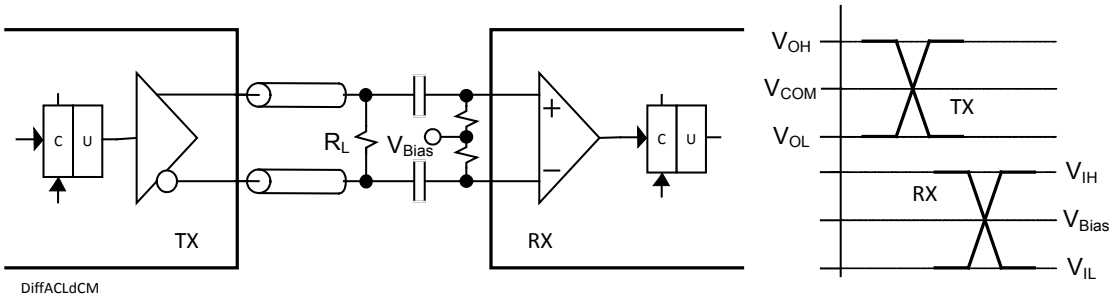


Figure 10—A basic differential ac signal path with load termination and receiver common-mode generation using a bias network

NOTE 1—The receiver waveforms in Figure 10 will decay to V_{Bias} if the driver frequency is low compared to the time constant of the coupling network. Since boundary-scan test data application rates can be low, the receiver might indeed see float (undriven) levels due to signal decay.

NOTE 2—It is assumed in Figure 10 that the distance between R_L and the receiver inputs is small, such that there are no significant transmission line effects beyond R_L.

Figure 11 shows a basic differential ac signal path with an unreferenced termination. The termination is used for impedance matching. The driver is a voltage driver and thus does not need a source termination to provide a current path.

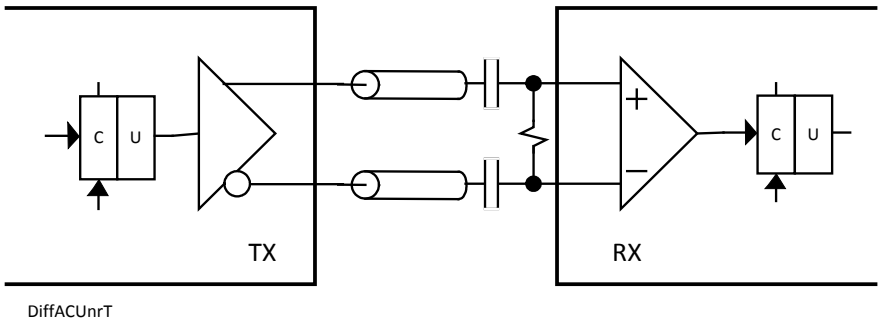


Figure 11—A basic differential ac signal path with unreferenced termination

Figure 12 shows a basic differential ac signal path with a current driver and source termination and also a referenced bias generator to select the common-mode voltage appropriate for the receiver. The source termination could also serve as an impedance match for the line. The bias network might use significantly higher value resistors as long as the line distance from the capacitors to the receiver is small. This will significantly increase the time constant of the coupling network.

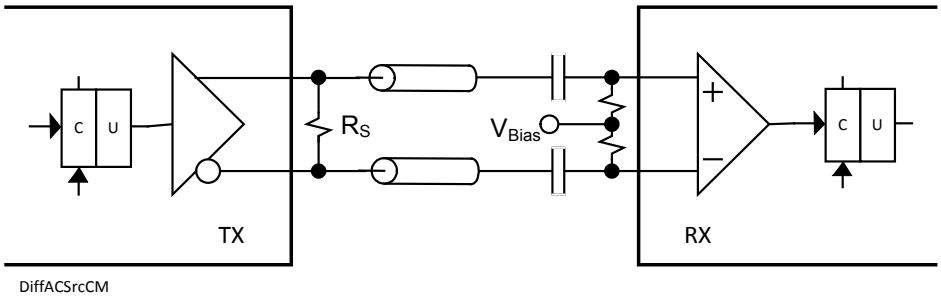


Figure 12—Basic differential ac signal path with source termination and bias provision

Finally, all the terminations, high-impedance bias networks, and even the coupling capacitors could ultimately be integrated into the receiver IC. Externally, the signal path appears to be dc-coupled but internally it is still ac-coupled, as shown in Figure 13. On-chip component defects will not need to be tested during board test. Thus only the interconnect defects (typically solder) will be relevant.

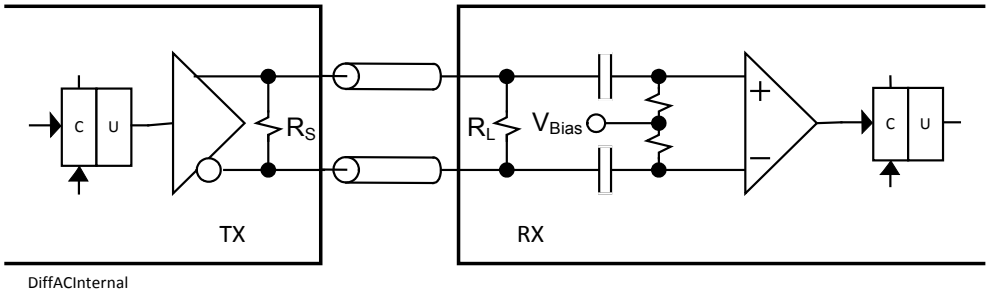


Figure 13—AC coupling, termination, and bias generation internal to the ICs

4.2.5 Intention: when ac capability is dc-coupled

The standard for ac testing proposed herein is intended to be implemented on ac pins of an IC. However, there is the possibility that a board designer might still choose to use dc coupling between devices that are dc compatible. Thus, a test developer could find a situation where ac testing is needed to test a dc-coupled signal path. This could occur when more than one ac-capable interface exists on an IC and one is ac-coupled while another is dc-coupled. The test developer would need to load an ac testing instruction into the device to test the ac-coupled interface. It is the intention of this standard that if dc coupling of ac-capable interface is possible and gives acceptable mission performance, then the ac test performance will also be acceptable.

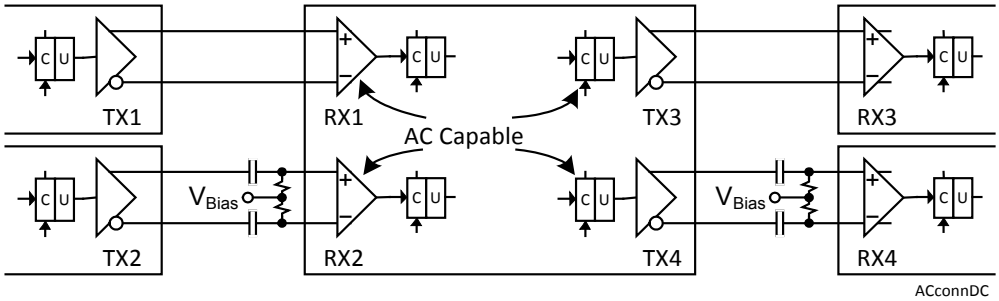


Figure 14—Combinations of ac-capable drivers and receivers connected to conventional drivers and receivers

For example, in Figure 14 a conventional IC (TX1, containing only EXTEST support) is dc-coupled to receiver RX1. An ac-capable driver TX2 is also connected to the RX2 receiver. To test all the signal paths simultaneously, the conventional device TX1 must be in EXTEST, while the other two use an ac testing instruction. If TX1, TX2, RX3, and RX4 are all ac-capable, then it is the intent of this standard that all of these configurations should be testable. If TX1, TX2, RX3, and RX4 are not ac-capable (only support EXTEST), then the configurations may or may not be testable. See a summary of capture behaviors for various coupling and testing scenarios given in 4.10.

4.3 The effects of defects

Defects are abnormalities in the structure of a circuit board (or similar assembly) that occur during manufacturing that must be found and corrected. This "manufacturing defect" model includes things like open solder joints, shorts, missing components, and dead devices. Not included in this model are performance-related issues, for example, the failure of a device to operate at its highest specified frequency at –40 ºC. This recognizes the traditional role of IEEE Std 1149.1 as a test standard for board level manufacturing defects.

The advent of ac coupling, especially in the differential signaling domain, threatens this role. boundary-scan testing is inherently a "DC" technology. Further, there is inherent redundancy in differential structures that can mask the presence of seemingly obvious defects. An example is shown in Figure 15.

In this case, the positive leg of the circuit is eliminated by a defect, for example, an open solder joint on the capacitor. Yet the receiver still receives the negative leg signal and compares it to the V_{ref} voltage (assuming that V_{ref} is a true voltage source with negligible impedance). The receiver will still produce the correct output, although its common-mode noise rejection capability is completely compromised. This might not be noticed until subsequent functional or performance testing is encountered. There, it could show up as an elevated bit error rate that would not provide very much diagnostic information. This simple example illustrates why it is important to monitor *both* legs of a differential pair independently, as covered in 4.6.4. In this case, the positive leg exhibits the "float" syndrome, or is simply said to float. In this case, the terminations, which might be internal in some devices, hold the leg at the reference voltage.

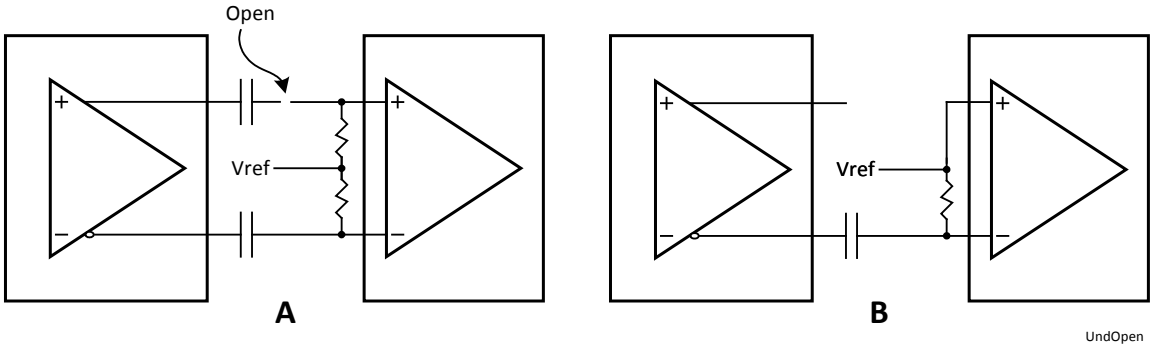


Figure 15—An ac-coupled differential path containing a defect (A) and an equivalent circuit (B)

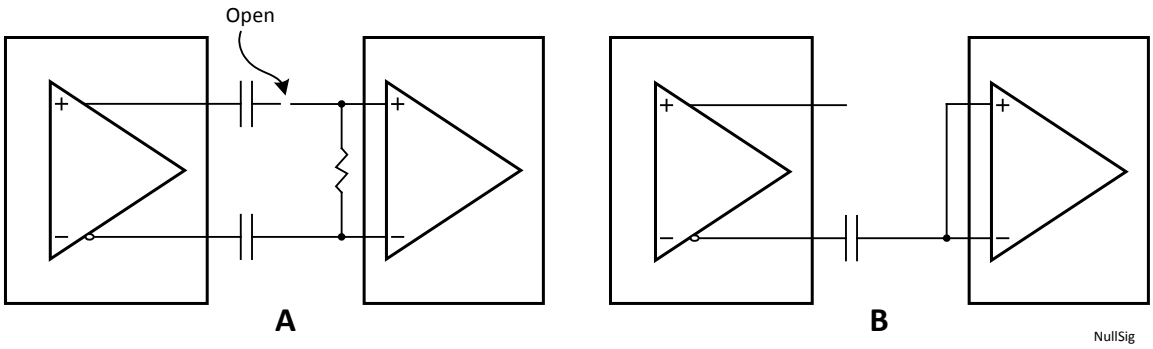


Figure 16—A null signal defect condition (A), where both legs see near-identical rather than complementary signals in the equivalent circuit (B)

The same defect, but with different termination, creates what is called the null syndrome, shown in Figure 16. As a result of unreferenced termination, the open defect allows the signal on the negative leg to appear on both legs of the receiver, in phase rather than 180º out of phase. Note the termination resistance and parasitic capacitances, etc., could make the signal on the two input pins slightly different, and the mission-mode receiver might respond unpredictably to these small differences.

One defect in particular could be troublesome to detect in ac-coupled structures: the shorted board-level capacitor (that is, a board-mounted capacitor between ICs). This defect restores dc coupling. This defect might go unnoticed particularly in differential signal paths, especially when the dc characteristics of the driver and receiver are reasonably similar. For this reason, it is important to support the standard EXTEST instruction because it can be used to test for shorted board-level capacitor defects. This can be done by supplying a stream of 0s and 1s to the driver side of the capacitor and showing that this stream does not show up on the receive side, i.e., it has been blocked by the capacitor. This assumes that the EXTEST instruction is changing the data at a slow enough rate to allow the signals to decay.

When a net is ac-coupled and could be used with unreferenced termination, as shown in Figure 16, it is important that the receiver provide some sort of bias voltage (usually high-impedance) to center the input signals at the optimal voltage offset for the mission receiver. Several defects produce the float syndrome described above, where the input just sits at this bias level. It is important to detect this condition for proper defect detection and diagnosis, meaning that for test we need to detect and capture any of three input states: a valid ‘1,’ a valid ‘0,’ and an invalid, or floating, input.

Finally, it is important to realize that for the defects that occur in high-speed circuits (where slew rates are often faster than signal path transmission delay), the transmission lines could alter the circuit’s faulty behavior as shown in Figure 17. Simulation of the circuit’s behavior may be necessary to understand the effects of defects. It is important to consider transmission lines as components of the simulated model when slew rates are elevated.

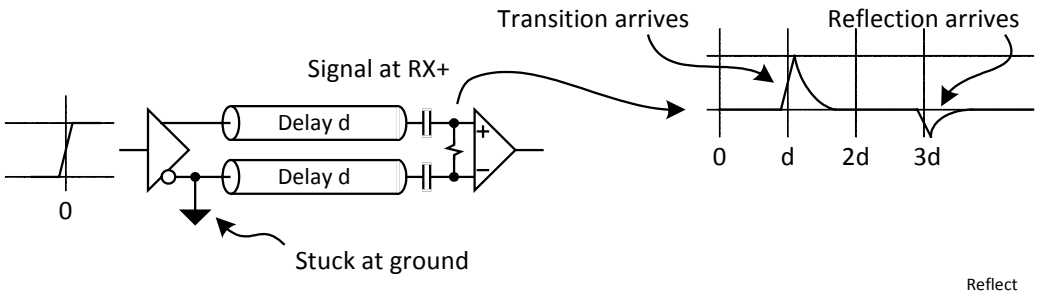


Figure 17—A defect that interacts with transmission lines to produce unusual effects

4.4 Defects targeted by the standard

This standard (as well as IEEE Std 1149.1) provides test support for detecting "manufacturing process defects" that are found on printed circuit boards coming out of the manufacturing process. These defects include missing devices (ICs, resistors, capacitors, etc.), improperly mounted devices (e.g., rotated 180º), open solder joints, shorted solder joints, and misaligned and dead devices. This standard focuses on those defects concentrated in ac-coupled and/or differential channels. As seen in 4.3, these defects could be difficult to detect or even be effectively masked from detection. Figure 18 and Table 1 show a representative set of defects considered by this standard. Note the model shows a single unreferenced termination, but the table gives results for other termination schemes.

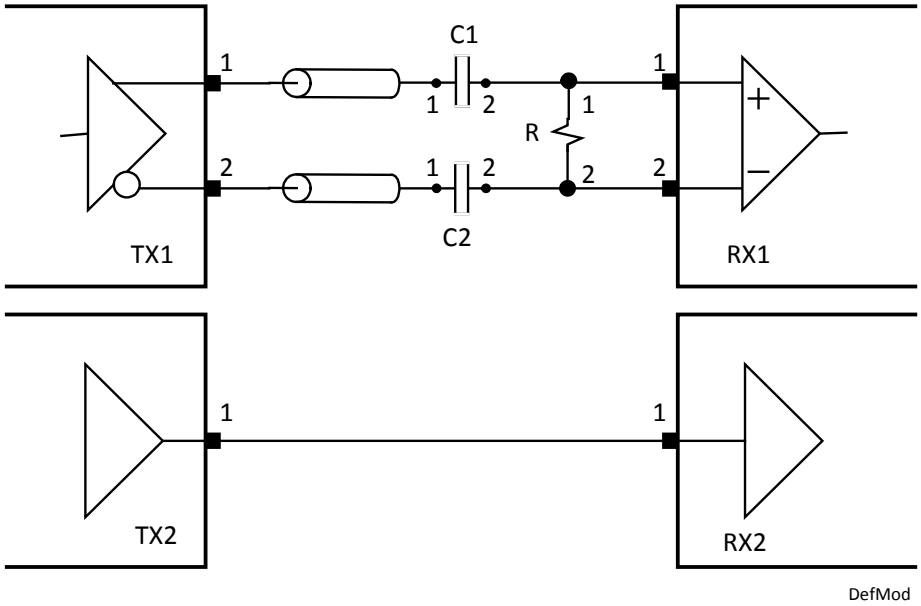


Figure 18—General ac-coupled channel and nearby dc channel used to illustrate defects

The least predictable defect syndromes occur when two signals driven by drivers of different technologies or protocols [for example, a differential LVDS driver and a "standard" single-ended complementary metal-oxide semiconductor (CMOS) driver] are shorted together. The two shorted signals tend to "average" as the drivers fight to control the combined net, attenuating the transition amplitude on the differential leg, though often not enough to prevent detection of the transitions. When the "other" net is shorted to one driver leg of a true differential driver (defect 12), both outputs of the driver will tend to track each other. As the short forces one output to a voltage different from its normal common-mode voltage, the other tends to follow. When the "other" signal transitions, this will appear to put large, same direction (common-mode) transitions on both legs of the driver. When the "other" net is shorted to one receiver leg of an ac-coupled channel (defect 13), it will control the center voltage of the received signal, possibly forcing it outside the common-mode range of the test receiver. The information that would allow detection and isolation of many defects is not only in the differential transitions, but also in the levels created by some defects. As a result, some defects could actually be easier to detect and diagnose using the IEEE Std 1149.1 EXTEST than using ac test instructions. For these and other reasons, it is necessary to perform a full interconnection shorts test of the board using EXTEST in all chips in addition to any ac test instructions, and base defect detection and diagnosis on the combined results.

The syndrome information presented in Table 1 is the result of hundreds of analog simulations on specific technologies and circuit designs. Different results can be expected for other technologies or implementations. Therefore, it is highly recommended that test receiver designs under consideration be simulated for both defect-free behavior and how they react to channel defects.

Table 1—Potential defects for the circuit in Figure 18

Defect ID	Defect site (Note 1)	Possible defect cause(s)	Typical receiver ac test syndromes (Note 2)
1	TX1 pin 1 open	Open solder joint, broken bond wire	Pin-1-follows-pin-2 null, possibly with detectable reflections on both pins (unreferenced), or pin 1 float (referenced)
2	C1 pin 2 open	Open solder joint, missing capacitor	Pin-1-follows-pin-2 null (unreferenced), or pin 1 float (referenced)
3	RX1 pin 1 open	Open solder joint, broken bond wire	Pin 1 float
4	TX1 pin 1 short to VDD	Pin-to-pin short, solder splash	Pin 1 float, possibly with detectable reflections if unreferenced
5	TXI pin 1 short to ground	Pin-to-pin short, solder splash	Pin 1 float, possibly with detectable reflections if unreferenced
6	TX1 pins 1, 2 shorted together	Pin-to-pin short, solder splash	Both pins float, transients at switching times suppressed by voltage and time hysteresis
7	C1 pins 1, 2 shorted together	Solder splash, internal short in C1	Pin 1 passes EXTEST, which would normally "fail" due to dc blocking of capacitor
8	C1 pin 1 short to C2 pin 2	Pin-to-pin short, solder splash	Pin-2-follows-pin-1 null, or both pins float (high frequency shorted legs, low frequency driven by driver pin 1)
9	RX1 pin 1 short to VDD	Pin-to-pin short, solder splash	Pin 1 float, stuck-at-1 detected by EXTEST. Pin 2 common-mode shifted and might also appear to be stuck-at-1 in EXTEST (unreferenced).
10	RX1 pin 1 short to ground	Pin-to-pin short, solder splash	Pin 1float, stuck-at-0 detected by EXTEST. Pin 2 common-mode shifted and might also appear to be stuck-at-0 in EXTEST (unreferenced).
11	RX1 pins 1, 2 shorted	Pin-to-pin short, solder splash	Both pins float (transients at switching times suppressed by voltage and time hysteresis)
12	TX1 pin 1 short to TX2 pin 1	Pin-to-pin short, solder splash	There is a large number of effects of device-to-device shorts. See text.
13	RX1 pin 1 short to RX2 pin 1	Pin-to-pin short, solder splash	There is a large number of effects of device-to-device shorts. See text.
14	R pin 1 open	Open solder joint, missing resistor	Much longer time constant on both pins (unreferenced) or just pin 1 (referenced), which might "pass" EXTEST.

NOTE 1—Defects that are equivalent by symmetry are omitted.
NOTE 2—Referenced termination (to low-impedance voltage sources) will isolate legs, while unreferenced termination will allow interactions between legs. This produces differences in faulty behavior.

4.5 Differential termination and testability

Extensive study of differential channels and the effects of defects within those channels have shown that the effects of defects are heavily influenced by the termination schemes used. There are three functions of terminations:

— To provide for proper dc current paths needed for proper driver functioning (usually source termination)

— To provide impedance matching of transmission lines

— In some ac-coupled cases, to set common-mode operating points for the receiver

A given termination could provide one or more of these functions. The following subclauses discuss termination options.

4.5.1 Unreferenced termination

Source termination is often needed for current signaling technologies where the direction of current flow is used to encode binary data. A typical example is a low-voltage differential signaling (LVDS) driver/receiver pair. Since all differential receivers operate by comparing voltages, a current signal is translated into a voltage signal for that comparison. Figure 19 shows an LVDS driver dc-coupled to a voltage receiver, where R_S provides both source termination and impedance matching. The direction of current flow represents data.

NOTE 1—It is assumed that R_S is placed very close to the receiver so that any transmission line effects are only present between the driver and the resistor.

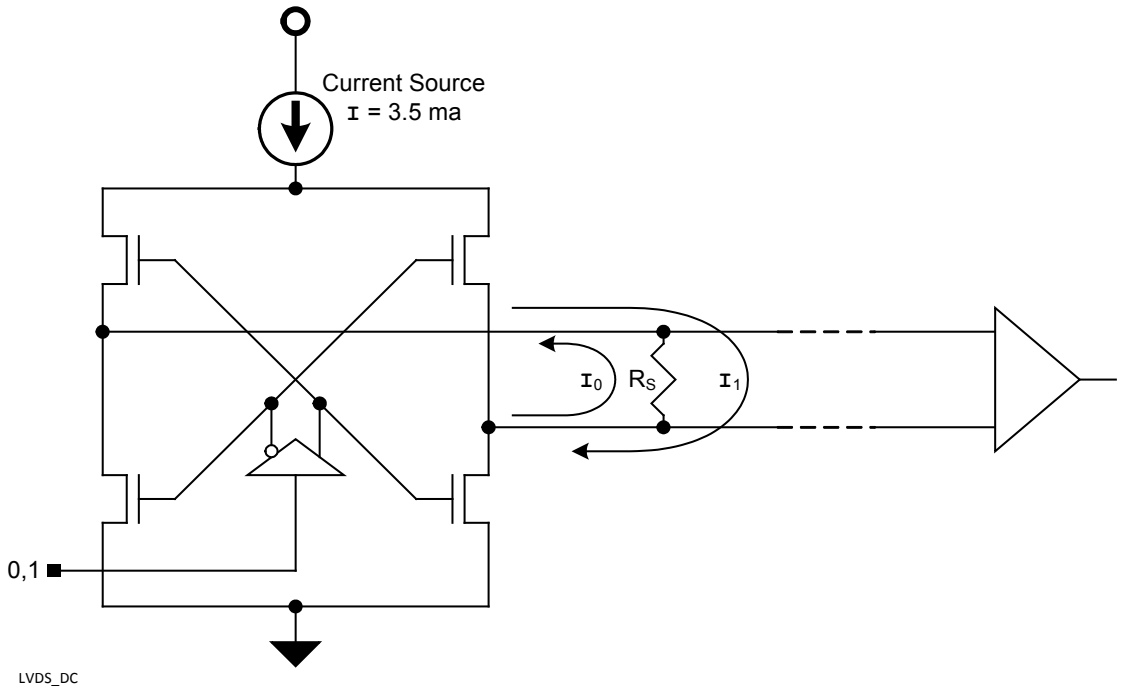


Figure 19—LVDS driver and receiver, dc-coupled

This form of termination is called unreferenced because there is a single resistor R_S rather than two resistors center tapped to a reference voltage (see Figure 21). In effect, the receiver is referenced to the common-mode voltage of the driver/resistor combination. This means the receiver's common-mode range must be compatible with the driver's output.

Figure 19 shows an unreferenced ac termination. Resistor R_S still provides source and line termination. Resistor R_L provides termination and a current path on the receive side, since the comparator input impedance is effectively infinity. Note R_L and C determine the time constant of this coupling.

NOTE 2—The receiver in Figure 19 will have its own biasing circuitry built-in to establish its common-mode operating point.

If the time between signal transitions is long, the voltage across R_L will decay to zero volts or a null input condition with results that are undefined. Note that some defects can also cause this to occur, such as a missing capacitor, C. In this case, both inputs get essentially the same signal, resulting in a very small differential signal, if any.

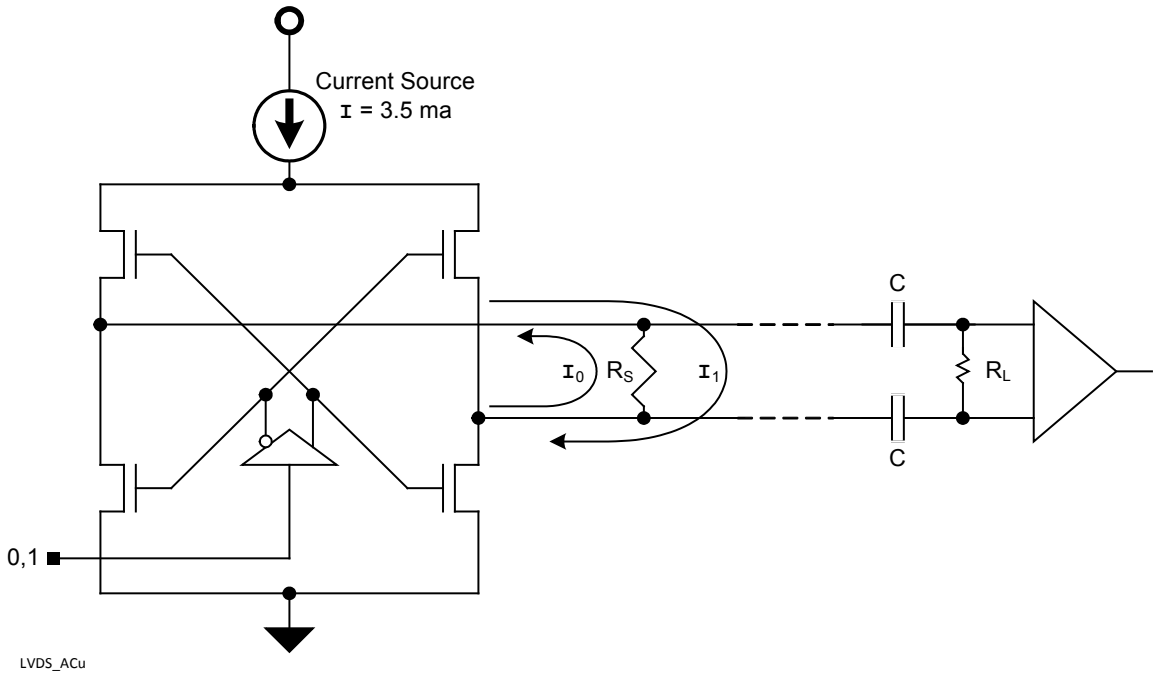


Figure 20—LVDS driver and receiver, ac-coupled with unreferenced termination on the receive side

4.5.2 Referenced termination

Figure 21 shows an ac-coupled, referenced termination. This type of termination is used to set the common-mode voltage of the receiver at its optimal value (here, V_{Bias}). Resistors, R_B, along with capacitance, C, determine the time constant of the coupling. The value of R_B might be larger, simply providing bias and a larger time constant, or could be smaller to provide bias, load impedance matching, and a smaller time constant. In the referenced termination case, if a defect such as a missing capacitor is present (this defect is shown in Figure 15) the leg with the missing capacitor will see V_{Bias} while the other leg will see a valid signal (the unreferenced case presented both legs with essentially the same signal, a null input state). Since test structures are to be added to both legs (see 4.6.4) they will respond in a deterministic way.

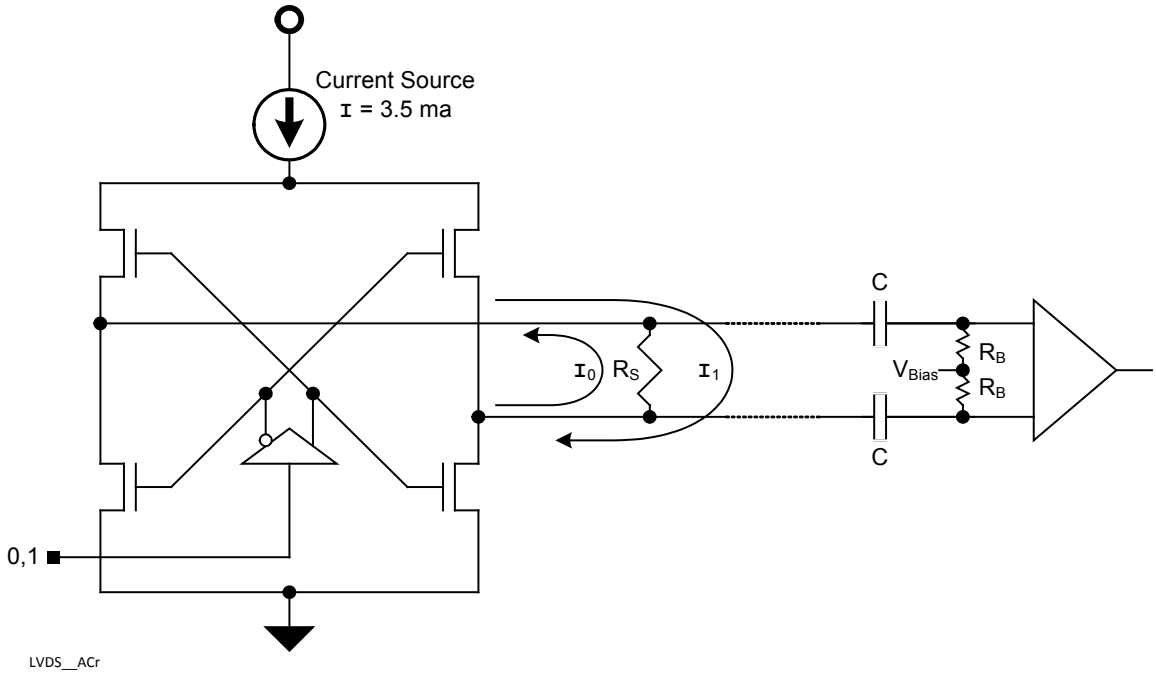


Figure 21 —LVDS driver and receiver, ac-coupled with referenced termination on the receive side

NOTE—It is assumed here that the V_{Bias} reference has low enough impedance to isolate the two legs. If this is not true, then the receiver could see signals more like the null condition.

4.6 Test signal implementation

In order to test the various combinations of single-ended and differential signal paths with variations on coupling, modifications have to be made to the drive and receive sides of the path.

4.6.1 Single-ended drive

Figure 22 shows a single-ended output stage modified in the familiar way given by IEEE Std 1149.1 for test purposes. One of two signals, the normal mission signal or test data is selected for transmission by a mode signal (this figure does not show the full detail of the boundary-scan register cells that supply test data). The test data is either the content of the boundary-scan register Update latch (U) when executing the (DC) EXTEST instruction, or an “ac Signal” when an ac testing instruction is active. The ac signal is a test waveform suited for transmission through ac coupling. The concept here is that the single-ended driver itself is not modified and no additional logic is inserted into the mission signal path except the multiplexer already required by IEEE Std 1149.1, which is to select the path between mission and test data. The multiplexer selection for ac mode is not inserted in the mission data path, but in the test data path.

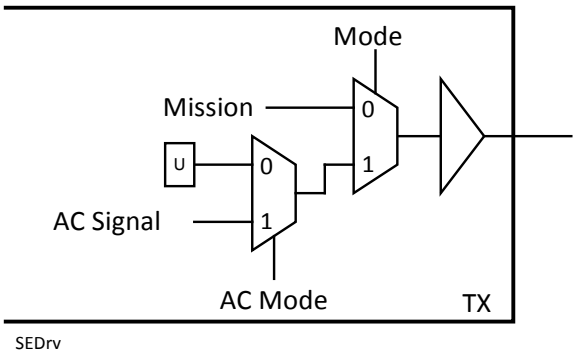


Figure 22—A single-ended driver path

4.6.2 Differential drive

There are two options for implementing a differential driver when incorporating test. The first is shown in Figure 23 where the selection between test and mission data is performed before the conversion to differential signaling. This means there will be only one data stream presented to the two differential pins and that the data will be transmitted in true differential form, using either current or voltage modes of the driver in question. Due to aggressive performance requirements in some higher-speed driver designs, it is expected that this option will often be chosen.

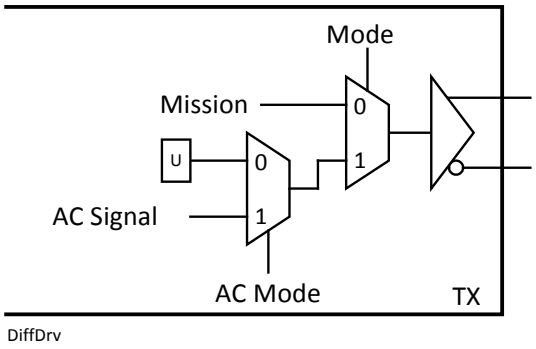


Figure 23—Full differential driver for both mission and test modes (more common choice)

A second option for implementing testability in a differential driver is shown in Figure 24. In this case, the mission mode signal path is differential, while in test mode the mission driver is disabled and two single-ended drivers with independent test data sources are enabled. Each controls a single side of the differential signal path. During test, the path is now a pair of independent single-ended signals. The two new single-ended test drivers must have similar drive characteristics as the mission driver to assure they are compatible with the loading and coupling that the mission driver would encounter.

NOTE 1—In describing this case in BSDL (see 7.2 and 7.3), the test mode of the driver signals are described, which is single ended. Thus, the signal pair is not described as differential (see the Grouped port identification, B.8.8 in IEEE Std 1149.1-2013).

NOTE 2—If such a device supports IEEE Std 1149.4, then the structure in Figure 24 could be implemented, but with drivers of insufficient drive capacity to drive the load impedance (this is because IEEE Std 1149.4 might implement the digital “driver” with relatively high-impedance switches). In this case, a hybrid of Figure 23 and Figure 24 may be implemented where ac testing operates using the model in Figure 23.

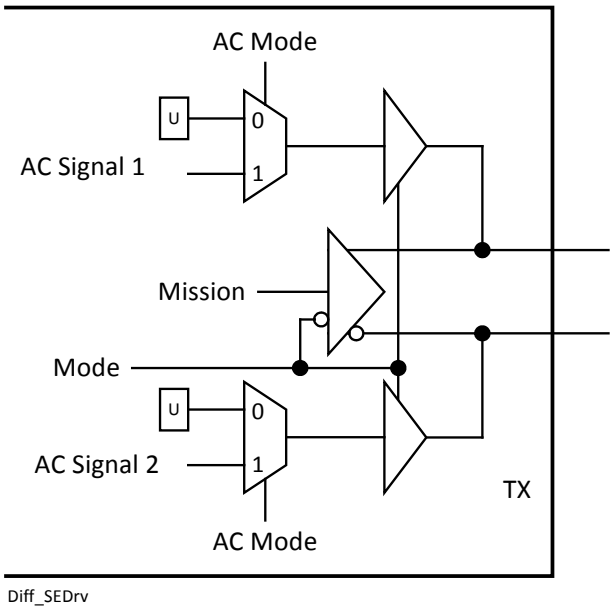


Figure 24—Differential mission/single-ended test mode driver (less common choice)

This option has desirable testability and diagnosibility features in that it removes some of the redundancy inherent in differential signaling, but it also reduces some of the noise immunity that differential signaling affords and could generate more noise during testing since the test signals on the two legs are no longer balanced and offsetting. There is additional cost in that the drive specifications (slew rate and amplitude) for the two added drivers must be substantially similar to those of the mission driver into the mission load. There could be unacceptable mission performance degradation with this approach that makes it the less commonly chosen option.

4.6.3 Single-ended test signal reception

Figure 25 shows two options for single-ended test signal reception, again familiar from IEEE Std 1149.1, but with a provision for detecting an ac test signal when an ac testing instruction is active. When an ac testing instruction is active, a specialized test receiver (see 4.7) detects transitions of the ac signal seen at the input and determines if this represents a logic '0' or '1.' When EXTEST is loaded, the input signal level is detected and sent to the output of the test receiver to the boundary-scan register cell. One option shown in Figure 25 supports INTEST (control and observe capability), and the other uses a simpler observe-only structure that will not support INTEST.

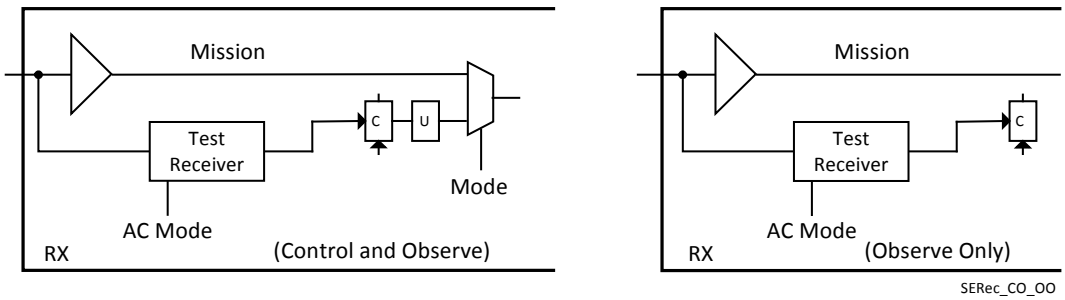


Figure 25—Single-ended signal reception, with and without support for the INTEST instruction

NOTE—The test receiver may be connected to the output of the mission input amplifier if it is a true amplifier (e.g., unity gain) rather than a threshold comparator. The test receiver is intended to process the actual waveform seen at the input pin, not an interpretation of the waveform.

A single-ended receiver has some form of reference used to distinguish a '0' from a '1' and this feature is used during (DC) EXTEST as well. However, during ac testing, the signal could decay to some intermediate value that cannot be reliably received by the mission receiver. The test receiver is therefore used to sense the ac test signal transitions.

4.6.4 Differential test signal reception

A differential receiver is modified for testability as shown in Figure 26. Preferably, the mission differential receiver path is modified to capture test data seen by the mission receiver in differential mode as required by IEEE Std 1149.1. This boundary cell on the mission receiver is optional, but recommended, in this standard. See 7.6.2 for an overview and 7.6.3 for how to code these cases in BSDL.

Each leg of the differential signal path has its own added test receiver. The purpose of each receiver is to monitor a leg of the signal path independently. A test receiver is required on each pin to provide adequate defect coverage and diagnosis capability.

NOTE—No variations in the test receiver are needed for INTEST support since this is an observe-only monitor.

Again, when an ac testing instruction is active, the specialized test receivers (see 4.7) detect transitions of the ac signals seen at the inputs and determine if these transitions represent a logic '0' or '1.' When EXTEST is active, the input signal level is compared to a fixed reference designed into the test receiver, or programmed in an initialization sequence.

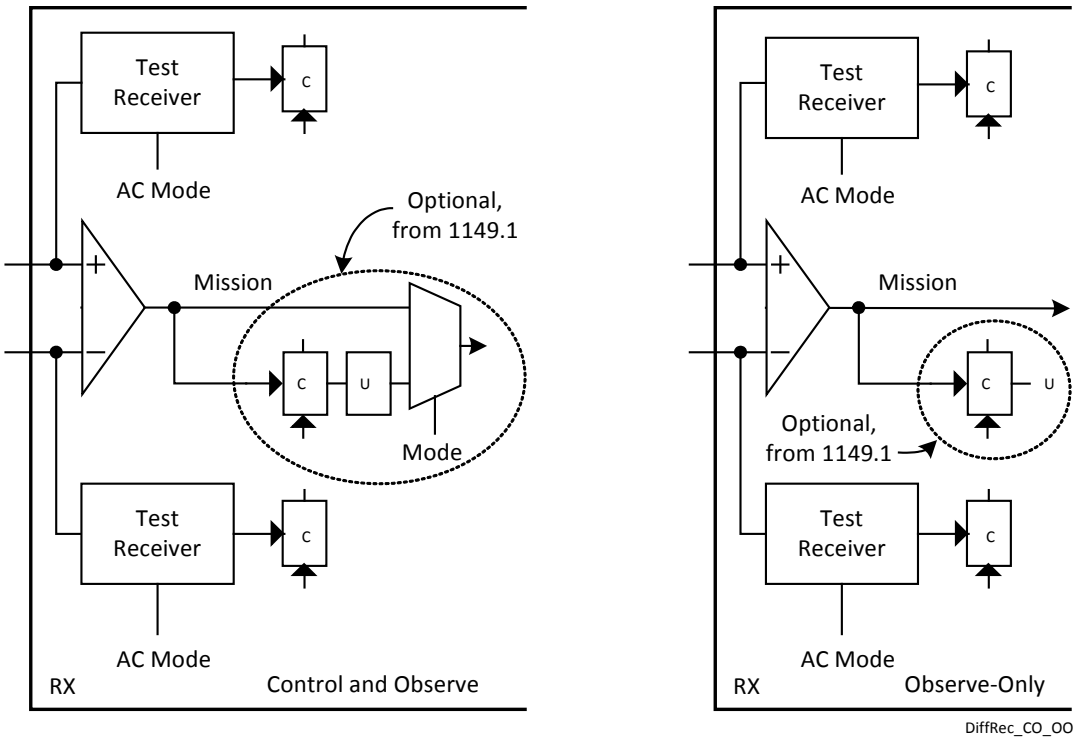


Figure 26—Differential signal reception, with and without support for the INTEST instruction

4.6.5 Coupling compatibility and EXTEST

When a test receiver is dc-coupled to a driver, either by design or due to a shorted board-level capacitor, and the EXTEST instruction is active, then one of two effects will be seen in the defect-free case:

— If the driver voltage levels bracket the threshold value(s) of the test receiver (i.e., one is above and the other below), the driven data will be captured, or

— If the driver levels do not bracket the threshold value(s) of the test receiver, then a static (stuck-at) value or only one polarity of the data will be observed by the test receiver.

Thus, interconnects between dc-coupled differential ICs might or might not pass data through the test receivers when performing EXTEST (this is a reason to use an ac testing instruction to test ac pins even when the ac pins are dc-coupled). It is probable (at least for single-ended signals) that the fact they are dc-coupled in the mission design indicates they will pass EXTEST data. Test generation tools should examine the logic family information of the two ICs to determine if this will happen.

When performing EXTEST on a dc-coupled differential channel, the output of the mission receiver can be used to detect only about half the possible board faults listed in 4.4. To detect all of the faults, test receivers are needed on each leg of the channel. On the other hand, if the test receivers are unable to detect the input correctly due to a mismatch of the driver common-mode and test receiver threshold voltage levels, then observing the output of the mission receiver is needed to have any coverage of the channel.

Since the signals on a dc-coupled differential pair could contain a significant and variable common-mode offset as well as the test signal, the design of the ac test receiver must account for the fact that there is no fixed reference available to discriminate a logic ‘0’ from a ‘1.’ A general discussion of the test receiver is given in 4.7, 4.8, and 4.9, and rules for its implementation are given in 6.2.

4.7 Test receiver support for ac testing instructions

The principle purpose of the test receiver seen in Figure 25 and Figure 26 is to extract a test signal even when the received signal contains an unknown offset. Because of the offset, a simple comparison of the test signal to a static reference might not reliably extract the test signal. Using the opposite leg of the pair as a reference (as is done by the mission receiver) will often mask important defects. A solution is to look for information contained in the *transitions* of the signal. These will be independent of the offset seen in Figure 27. Valid transitions have a defined voltage swing, ΔV , and transition time, Δt .

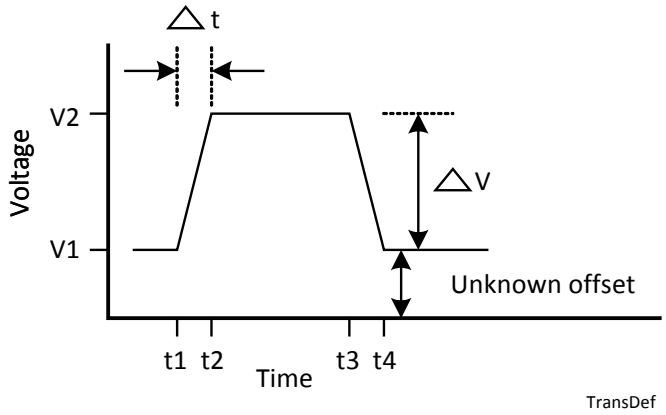


Figure 27—A signal with unknown voltage offset and the signal transitions it contains

One way to find the transitions in a signal with an unknown offset is to compare the signal with a delayed version of itself, that is, to use its recent history as a reference. This is illustrated in Figure 28. The original signal, a delayed version, and the output of the hysteretic comparator are shown. The output is a faithful reconstruction of the original waveform, delayed by the time it takes for the input waveform to pass the hysteresis threshold. The output waveform has been converted to standard logic levels, that is, the unknown offset is removed.

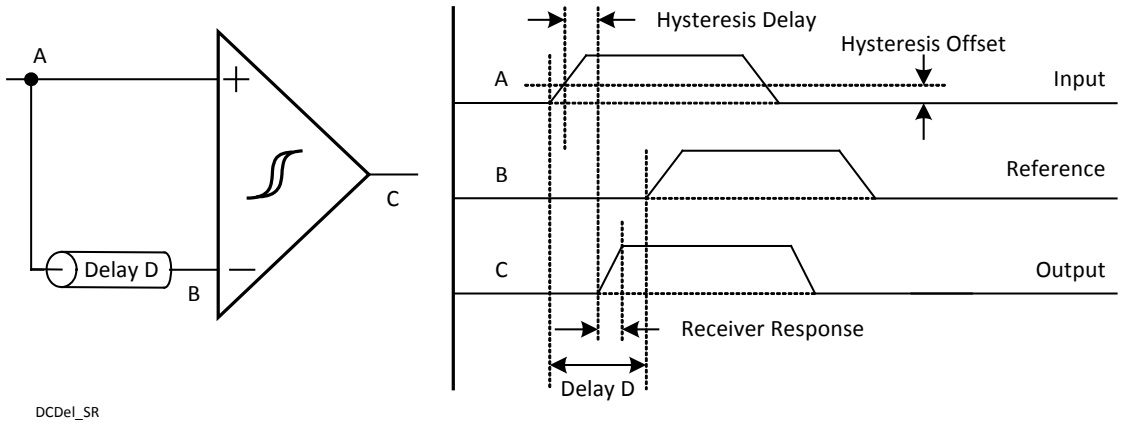


Figure 28—Delayed self-referenced reconstruction of a dc-coupled input waveform with unknown voltage offset

It will be important that the delay D is longer than the transition times to be sensed. The use of hysteresis (the hysteresis voltage) can eliminate unwanted response to small signal noise (runt pulses). Additional filtering in the design of the comparator (the hysteresis delay) can eliminate response to larger signal noise of insufficient duration (noise spikes).

The waveforms that are applied to the test receiver might or might not be ac-coupled to the driver. When ac-coupled, they might or might not decay significantly, depending on the coupling time constant. For the signals shown in Figure 28, the test receiver is either dc-coupled or ac-coupled with a very long time constant. If ac-coupled signals with periods that are long with respect to the coupling time constant are applied to this simple circuit, the decaying signals will cause the comparator to reset itself early (after Delay D) in response to the delayed reference edges, and the reconstructed waveform will differ from the original as in Figure 29, where the reconstructed waveform has been shortened.

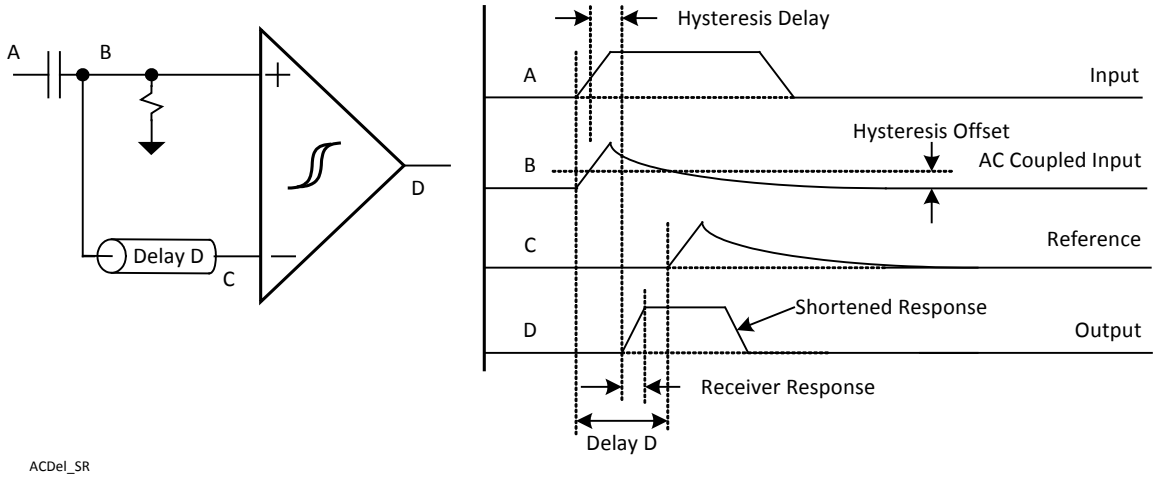


Figure 29—An ac-coupled waveform with short coupling time constant

A low-pass filtered delay solves this problem as shown in Figure 30, a concept called "self-referencing." The simple low-pass filter with a filtering time constant of $R_F \times C_F$ is used to hold the reference input to the test receiver at a constant value that is largely unaffected by short term events such as the high-pass filtered edges seen when ac coupling (low time constant) is used. If high time constant ac coupling or dc coupling is used, then the low-pass filter adjusts the reference point again to represent near term history of the input signal. The hysteresis voltage and hysteresis delay are used to control response to noise.

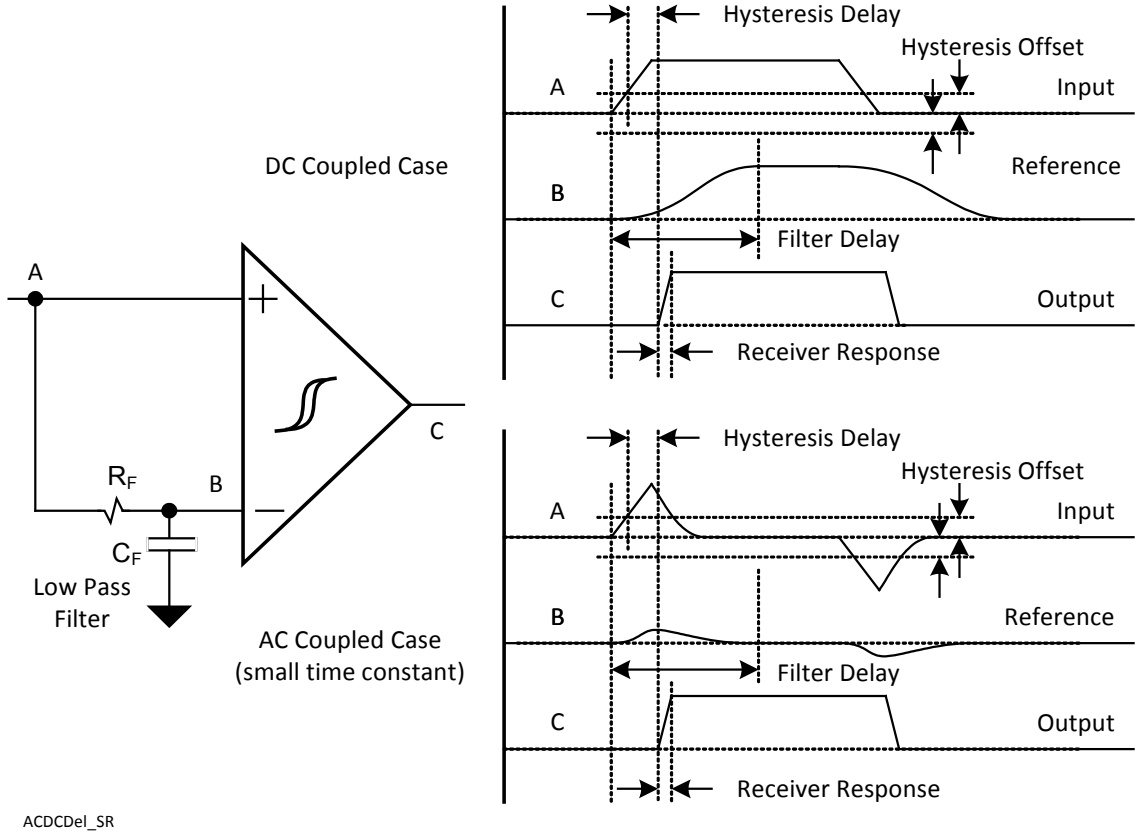


Figure 30—Delayed and filtered self-referenced waveform reconstruction of both dc- and ac-coupled waveforms by a test receiver

To summarize the concepts shown in Figure 30, the test receiver reconstructs an original waveform driven from either a single-ended driver or one leg of a differential driver that is either ac or dc-coupled, and is insensitive to dc offsets that could exist in the driven waveform. It does this by responding to the edges of the original waveform that are still present despite dc- or ac coupling.

Figure 31 shows an ac-coupled, differential signal channel including the driver (U1), board, and receiver (U2). In addition to the mission receiver, there are test receivers on each receiver pin. This model supports the ac testing instructions, and sample waveforms are shown at critical nodes.

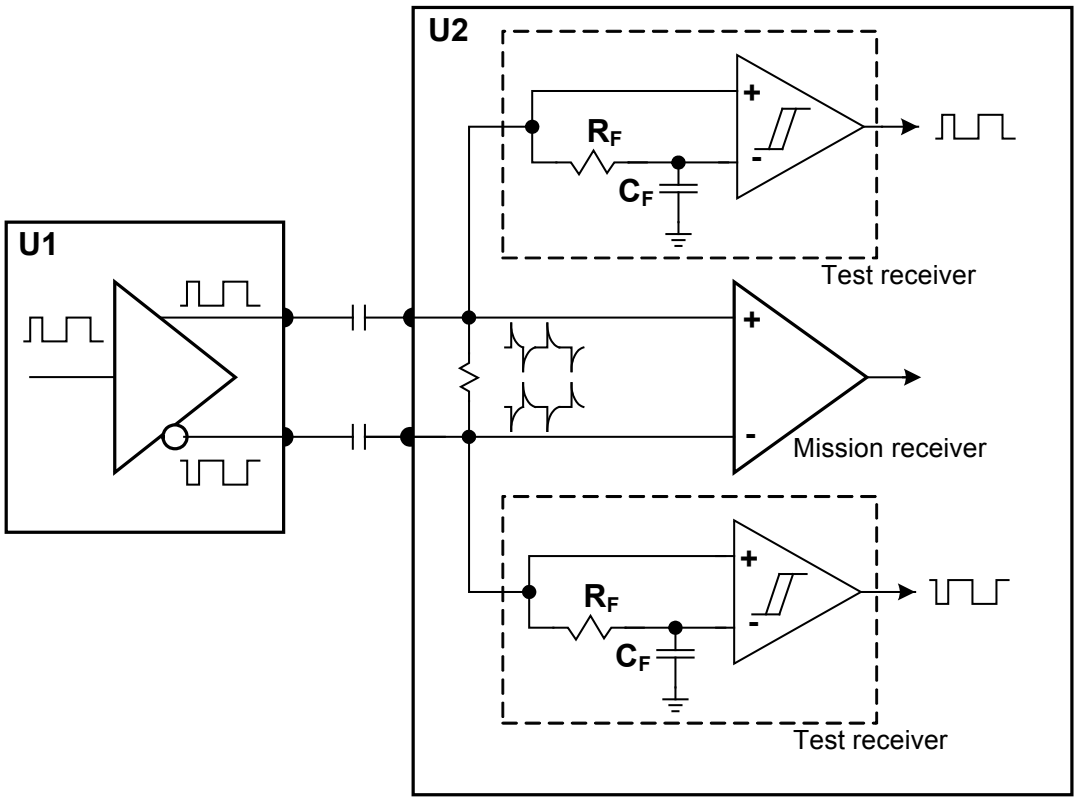


Figure 31—Differential driver, ac coupling, and test receiver model

Figure 32 shows an analog simulation model of the self-referenced design shown in Figure 30. The hysteretic comparator of Figure 30 is expanded into two simple comparators: one to sense rising edges and the other to sense falling edges, the outputs of which can be easily monitored during simulations; two V_{Hyst} voltage sources, to allow explicitly setting the hysteresis voltage; and a D-type flip-flop memory element (not implemented in the simulations), to emulate the hysteretic memory and hold the reconstructed signal. The internal bandwidth, or slew rate limits of the simple comparators, are set in simulation to determine the hysteresis delay. The comparator outputs set or clear the hysteretic memory flip-flop upon receipt of rising and falling edges, respectively, reconstructing the original waveform. The values of R_F and C_F are chosen to provide a delay longer than the expected transition times being sensed. Typical values of R_F and C_F might be 5 kΩ and 5 pF, which could be integrated into an IC. The input waveform at point A might be high-pass filtered by ac coupling, or it could be a normal dc-coupled digital waveform (with an unknown offset). In either case, the edges are used to reconstruct the original digital waveform. This simulation model will be used throughout this standard when discussing the test receiver, with the understanding that the actual implementation will typically be a single analog hysteretic comparator circuit.

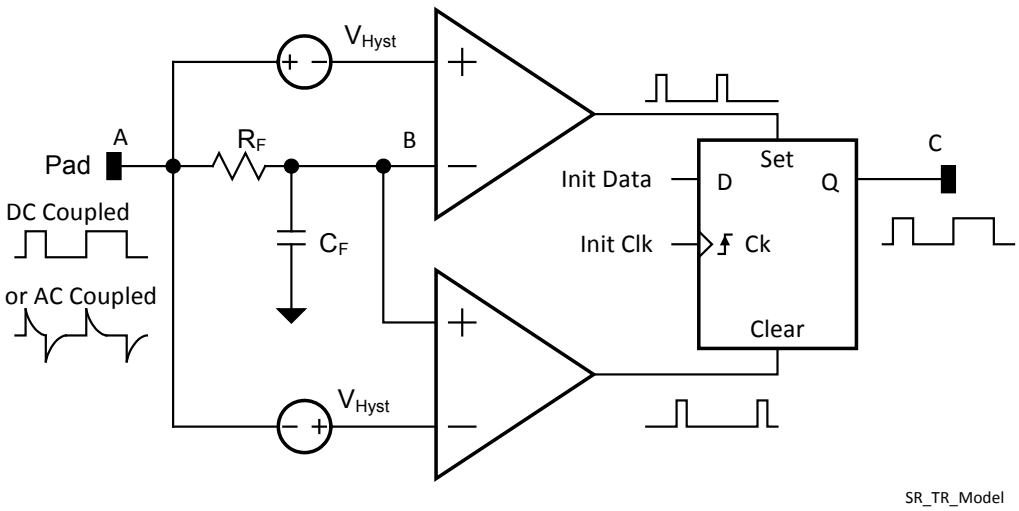


Figure 32—A simple self-referenced test receiver model of the concept in Figure 30

NOTE—The memory inherent in a hysteretic comparator would normally be modeled here as a simple Set/Clear asynchronous flip-flop. However, as will be discussed below, and detailed in Clause 6, it is necessary to initialize this memory at certain times during the test, so this hysteretic memory will be modeled throughout this standard as a D-type flip-flop with asynchronous set and clear. It could also be modeled in other ways, including as a level-sensitive (transparent) latch with asynchronous set and clear, and with appropriate changes to the clock signal timings.

4.8 Test receiver support for the (DC) EXTEST instruction

The test receiver behavior for ac testing has been described. However, it also is important to support the standard (DC) EXTEST instruction. This amounts to "turning off" the edge-detecting capability of the test receiver and having it respond only to levels. This is further complicated by the problem that there could be common-mode offsets added to the signals. This necessitates choosing a reference voltage that can be used for single-ended comparison.

This standard provides two possibilities for the selection of that reference voltage. The first establishes a threshold voltage intended to detect the difference between a logic '1' and '0,' and fully supports IEEE Std 1149.1, and the second establishes a threshold voltage at the limit of the test receiver input voltage range and only provides a continuity test for detection of shorted board-level capacitors.

When attempting to define a threshold voltage ($V_{Threshold}$) that can detect logic levels, one choice is to use the internal bias voltage used to set the common-mode voltage point of the mission receiver (the "sweet spot"). This voltage will work well as both a bias voltage for the input pin and as a threshold voltage for the test receivers when ac coupling is used, as shown in Figure 33. However, if the receiver IC is dc-coupled to the driver, possibly caused by a shorted coupling capacitor, then the test receiver might or might not receive data depending on whether the high and low values driven by the driver are above the threshold voltage plus the hysteresis voltage and below the threshold voltage minus the hysteresis voltage, respectively. If data is not received, the test receiver could perceive a constant logic value, either '1' or '0.'

Alternatively, the threshold voltage for the test receivers could be made programmable during test mode and set by the requirements of the actual board design during an initialization process prior to board test. This would significantly improve the ability of a test receiver to work with multiple different driver designs. What is important is that the threshold voltage be determined independently of the observed voltages on the input legs of the differential channel since otherwise a board defect could cause the threshold voltage to be set to a completely inoperable level.

Note that since the test receiver is not self-referenced in this configuration, a high-impedance voltage V_{Bias} in Figure 33, which has the same value as $V_{Threshold}$, must be connected to the pin. When the input pin is undriven, the voltage on the comparator inputs is within the hysteretic dead zone, and both comparator outputs are off.

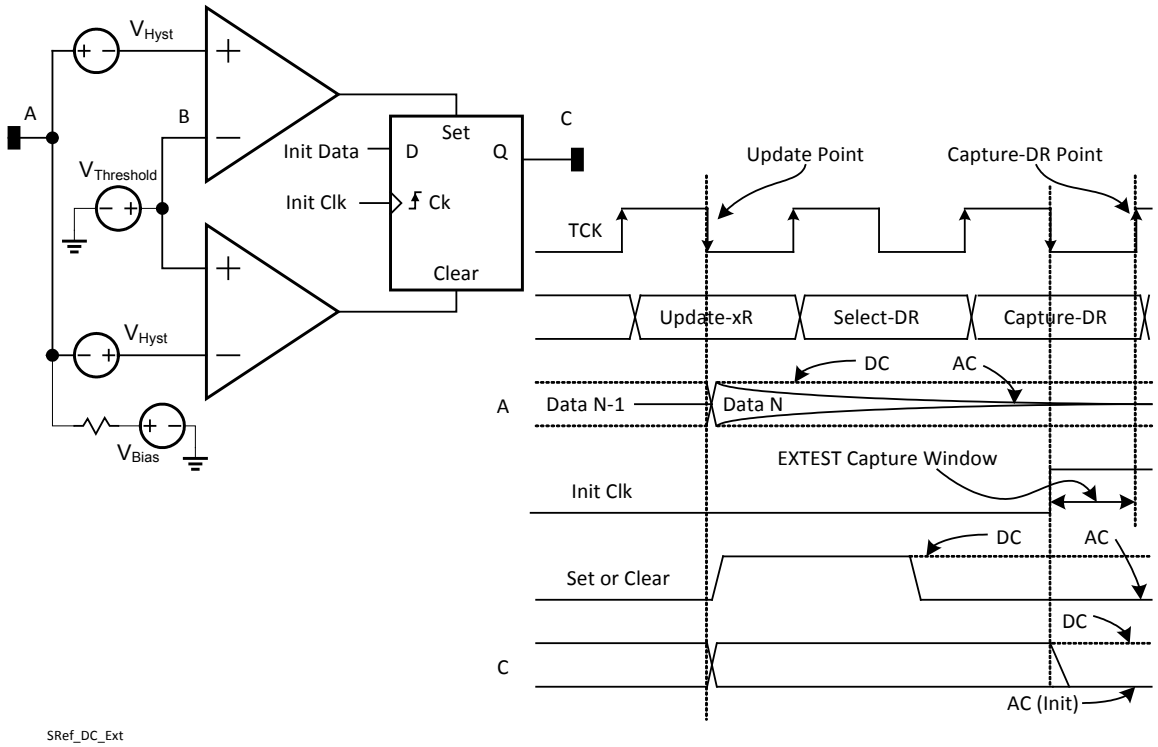


Figure 33—Static reference used to support (DC) EXTEST.

In Figure 33 comparators could set or reset the hysteresis flip-flop after the falling edge and before the rising edge of TCK in the Capture-DR state. If the driver and receiver are dc-coupled, then the level driven will be seen at the receiver and force the comparator output to the driven value, assuming that the threshold voltage is properly placed at the midpoint of the driver swing.

If the receiver IC is ac-coupled to the driver, then Figure 33 shows the transition signal decaying away to less than the bias voltage plus or minus the hysteresis voltage prior to the *Capture-DR* TAP Controller state. The initial value loaded into the hysteresis at the falling edge of TCK in the *Capture-DR* TAP Controller state will then be captured. When testing for a shorted board-level capacitor the desired effect is that a shorted capacitor will pass data from the driver in EXTEST, but a good capacitor will not. This effect depends on the time constant of the ac coupling relative to the time between *Update-DR* and *Capture-DR* TAP Controller states. When testing for a shorted board-level capacitor, the test software must wait for enough time to pass for the signal to decay before entering *Capture-DR*, either by stopping TCK or by spending additional TCK cycles in the *Run-Test/Idle* TAP Controller state.

On the other hand, if testing the (deprecated) case where the driver pins are dc-only pins, yet are ac-coupled to ac pins (see 4.10), then the test for opens on the net will need to keep the time between the *Update-DR* and *Capture-DR* TAP Controller states short enough that the signal will not have decayed below the hysteresis thresholds. This might also require a large coupling time constant.

In situations where the common-mode voltage of the driver is completely unknown, the channel is required to be ac-coupled, and EXTEST is only used to detect a shorted board-level capacitor. Then the V_{Bias} and

$V_{Threshold}$ shown in Figure 33 could be forced to a level at either limit of the input voltage range of the comparators. In this case, the test receiver will not detect the value being driven by the driver, but only whether or not there is a dc connection between the driver and receiver (See Figure 45 and Figure 46 in 6.2.2.4 for specifics).

When the common-mode voltage of the driver and the threshold voltage of the receiver are known and documented for the level-sensitive test mode, then the test software can make rough predictions of the behavior of the board circuit under test. Obviously, test environment noise and process variation need to be taken into account when the predicted behavior has very small margins. When either the driver common-mode voltage or the receiver threshold, or both, are programmable this makes it much easier for the test software to set up a robust test with good margins. Again, when the net being tested is required to be ac-coupled, then this standard allows a simple continuity check for a shorted coupling capacitor instead of the level-detection of the transmitted data value normally required.

4.9 A general test receiver for dc and ac testing instructions

A test receiver simulation model and a circuit implementation that support both the ac and (DC) EXTEST instructions are required. This could be accomplished simply by taking the two structures already shown and selecting between them with multiplexers. However, it is possible to merge their behaviors into a more efficient structure as shown in Figure 34 (Parts A and B), for both the simulation model (Part A) and a possible implementation (Part B). In this structure an analog multiplexer selects between (DC) EXTEST support and ac testing support.

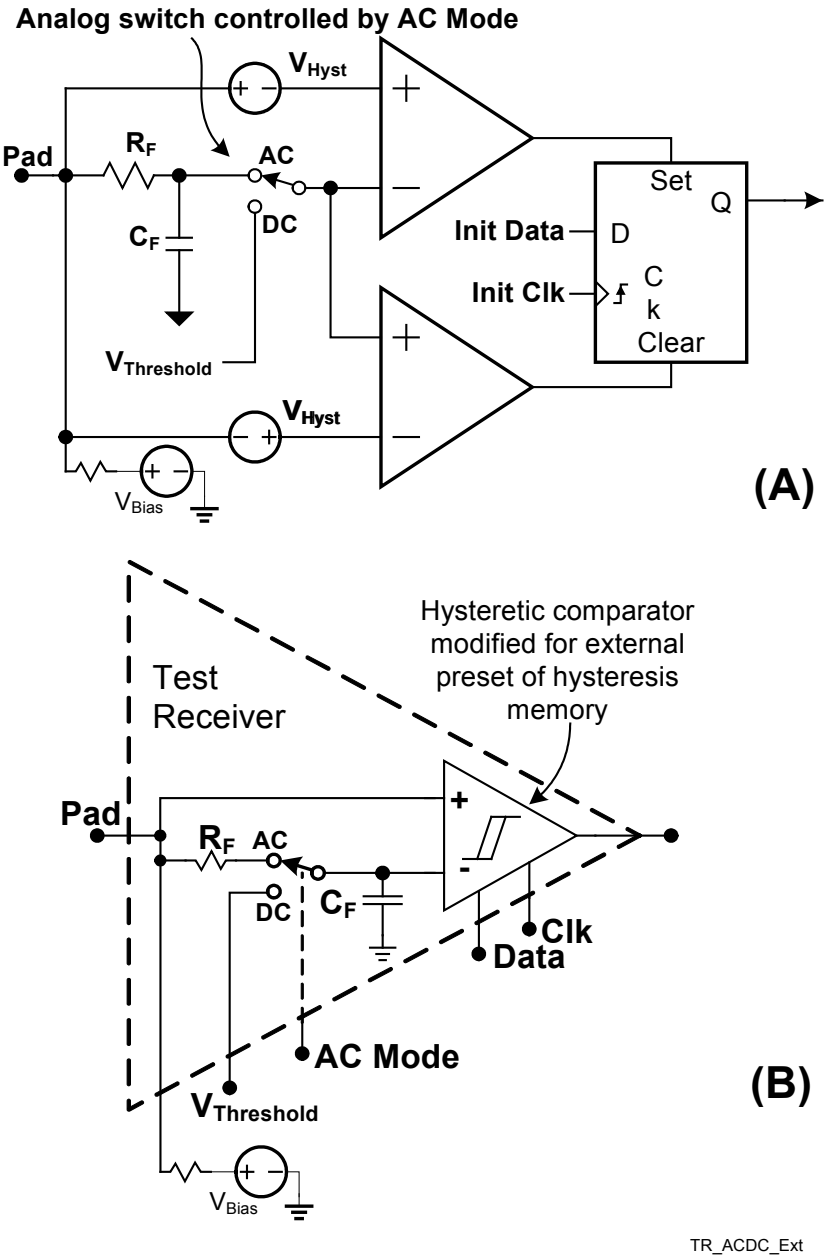


Figure 34—A test receiver model that supports both ac and dc EXTEST

Implicit in the above discussions is the fact that the test receiver, whether in ac or dc test mode, actually detects three mutually exclusive input states: a valid logic '1,' a valid logic '0,' and an invalid float state where the input stays within the range of the threshold voltage plus and minus the hysteretic voltage. As the boundary-scan register cell can only capture a binary value, we initialize the hysteretic memory to a known, or default, value (see Figure 33) that will not change if the input is floating. This initialization defines the start of the sampling window, and also allows the test software to differentiate between driven data and the stuck-at-1, stuck-at-0, and float syndromes through appropriate choices of the data being driven and the initialization data for the test receiver (a rising edge of TCK in the *Capture-DR* TAP Controller state always defines the end of the sampling window). An alternate means of initializing the hysteretic memory and preserving the boundary-scan register cell value is modeled in 6.2.5 and Figure 58 in the discussion of integration of the test receiver with the boundary-scan register cell.

4.10 Boundary-scan capture data versus configuration

The content of Table 2 shows what boundary-scan data will be captured by a test receiver for various combinations of coupling, dc compatibility, and whether a given IC is executing EXTEST or an ac testing instruction

Note in Table 2 that both rows with the driving IC performing a dc test coupled to a receiving IC performing an ac test are deprecated.

In the case where the driver and receiver are dc-coupled, then provisioning the mission receiver with a boundary cell allows the channel to be tested with IEEE Std 1149.1 EXTEST (dc tests) in both ICs. This structure maintains at least the IEEE Std 1149.1 EXTEST behavior and fault detection for differential channels.

The case where the driver and receiver are ac-coupled but performing EXTEST, testing for continuity through the coupling becomes problematic, and this case is also deprecated.

Board designers should avoid these situations. See the discussion following Figure 46 (in 6.2.3.3) and A.3.4.4.1 for further discussion and guidance.

Any time a receiver with only IEEE Std 1149.1 capabilities is ac-coupled to its driver, the ability to test that net becomes dependent on the coupling time constant, TAP navigation, and requires a minimum TCK during the period of the test (*Update-DR* through *Capture-DR* TAP Controller states). As this is highly dependent on tester capabilities, and including these nets in the test of the board as a whole could create conflicting requirements with other net configurations, board designers should avoid this situation. See A.3.4.4.2 for further discussion and guidance.

Table 2—Boundary-scan capture results for various combinations of driver-to-receiver coupling, test instruction, and driver-to-receiver dc compatibility

Driver to receiver coupling	Channel test performed by pin of:		Capture result when dc levels (drive/receive) are:		Comments
	Driving IC	Receiving IC	Compatible	Incompatible	
DC	DC test (Note 1)		Data (Note 3)	Stuck-at 0/1 (Note 4)	Typical test for dc coupling per IEEE Std 1149.1
	AC test (Note 2)		Data		Preferred test of differential channels
	DC test	AC test	Data when driver toggles at *Update-DR,* or Default when driver does not toggle (Note 5)		Static drive with AC receive deprecated: 1. Test data might not transition for multiple test cycles, preventing receiver from detecting data, 2. Driver only transitions at *Update-DR* (noisiest point in test).
	AC test	DC test	Data	Stuck-at 0/1	ac action compatible with dc receiver.
AC	DC test (for typical TCK frequencies)		Default		Used to test shorted board-level capacitors if receiver is ac-capable. Will transmit data (a failure) or will retain a default value (pass).
	AC test		Data		Preferred test for all ac-coupled channels.
	DC test	AC test	Data or default, depending on the relationship between *Update-DR* to *Capture-DR* time vs. net time constant(s), when driver toggles at *Update-DR*, or Default when driver does not toggle.		Static drive with AC receive deprecated: 1. Test data might not transition for multiple test cycles, preventing receiver from detecting data, 2. Driver only transitions at *Update-DR* (noisiest point in test).
	AC test	DC test	Default		Could be used to test shorted board-level capacitors if receiver is ac-test capable, but dc-to-dc test preferred. Deprecated otherwise.

NOTE 1—In Table 2, “dc test” implies that the IC has the EXTEST instruction active, or the channel pins are dc Pins, or the TAP Controller state sequence does not transit the *Run-Test/Idle* TAP Controller state.

NOTE 2—“ac test” implies that the IC has either the EXTEST_PULSE or EXTEST_TRAIN instruction (see 5.2) active, that the channel pins are ac pins, and that the state sequence transits the *Run-Test/Idle* TAP Controller state.

NOTE 3—“Data” indicates boundary-scan data is successfully transmitted.

NOTE 4—"Stuck-at 0 or 1" indicates the incompatible levels will be seen as either a '1' or '0' depending on where the test receiver threshold is set. An attached differential mission receiver, by contrast, does not see stuck-at behavior due to its common-mode range, and the recommended boundary cell on the mission receiver will successfully capture the transmitted data.

NOTE 5—"Default" indicates the response is the same that a floating test receiver would produce.

4.11 Noise sources and sensitivities

It has been pointed out that one advantage of differential signaling is better noise immunity for some types of noise, primarily common-mode noise. Because this standard looks at signals from a single-ended perspective, some of this noise immunity could be diminished or lost completely. Therefore it is important to be aware of the various sources of noise, their extent, magnitude, and duration. This standard addresses the effects of noise through the use of hysteresis (voltage and time) in the input test receiver, and by causing drivers to change states (differentially) at times other than the known-to-be-noisy *Update-IR* and *Update-DR* TAP Controller states. Designers should carefully evaluate all noise sources and their effects before determining hysteresis parameters. This standard assumes that "Best Design Practices" have been used to reduce noise effects wherever possible, especially on that noise which is not common-mode. This standard does not address noise issues that will impact mission operation. The standard will only address those sources of noise where noise immunity could have been diminished by single-ended testing of the receiver.

There are several sources of noise in the board test environment. The most prevalent sources of noise are ground bounce and coupling effects. Ground bounce consideration will be focused on on-chip switching of internal signals and on on-chip simultaneous switching of input/output signals, but will not consider ground bounce issues that are caused by inadequate grounding and bypassing of the board. Signal integrity issues are not covered since they would be considered a part of Best Design Practices.

NOTE—The term "ground bounce" is intended to include distortions that occur in any power supply rail including ground. Ground bounce attacks the assumption that there are stable voltage sources when in reality, the power and ground rails used by devices and boards are subject to transient inductive and resistive voltage offsets that can disrupt circuit behavior.

4.11.1 Noise effects generated by on-chip switching of internal signals

The best way to reduce noise would be to disable all mission mode circuitry on-chip. If devices have on-chip oscillators or phase-locked loops (PLLs) along with functional (mission) circuitry, which is not disabled during testing, or a clock input which has not been disabled during testing, there could be some "ground bounce" internal to the device that could affect both driver and receiver operation. This noise, synchronous with the on-chip oscillator, would be minimized by good power distribution layout and good on-chip decoupling, and this would be considered a Best Design Practice. For example, the magnitude of this noise effect in a recent CMOS technology can be modeled as a triangular current pulse on the power supply rails with duration of 0.5 nsec (see Figure 35). The amplitude of the pulse will increase or decrease depending upon on-chip decoupling. The voltage domain response will be determined by the specific design implementation (RLC) of the on-chip power distribution system, but it will typically be a damped sine wave with peak amplitude of about 5% of the supply voltage for best practices, though up to 15% has been observed. Proper care in separating I/O power distribution from internal circuitry power distribution is also important so that on-chip generated noise does not severely impact test receiver performance.

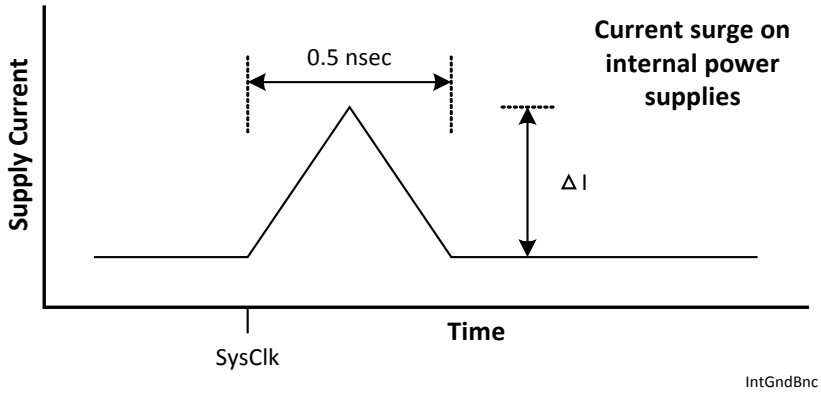


Figure 35—Internal supply current surge in CMOS, synchronized to an internal clock edge, a source of noise in the power distribution system

The noise pulse can be filtered by appropriate hysteresis settings in test receivers. In general, device designers should assure that on-chip logic, which is not part of the test logic, be forced to a quiescent state during boundary-scan testing. Internal oscillators and PLLs should be disabled when possible, or their output distribution disabled.

4.11.2 Noise effects that are generated by switching of I/O signals

Noise effects that are caused by simultaneous switching of component outputs are well understood by the design community. In general, simultaneous switching of single-ended outputs can create significant current surges in either or both I/O voltage rails. This results in significant voltage excursions that can affect operation of circuits on-chip as well as signal integrity across the board. Differential drivers contribute very little to the current surge since they are balanced, but can be affected by resulting voltage excursions, as can differential or single-ended receivers. Simultaneous switching could be significantly worse during boundary-scan testing since all drivers are updated with a single clock. The chip and board designers must address these issues for both mission and test operation of the chip. Therefore, much of the noise generated by simultaneous switching of outputs should be addressed as Best Design Practices.

Noise effects that are generated by simultaneous output switching could be greater in amplitude and duration than noise due to on-chip switchin, which was discussed in 4.11.1. Noise due to simultaneous switching of outputs can be simulated as shown in Figure 36. The current surge induced could last several nanoseconds (ns) and the resulting voltage sag (reduction in I/O voltage, Vdd-Vss) could range from 500 mV to 1000 mV, depending on the application and I/O voltage levels.

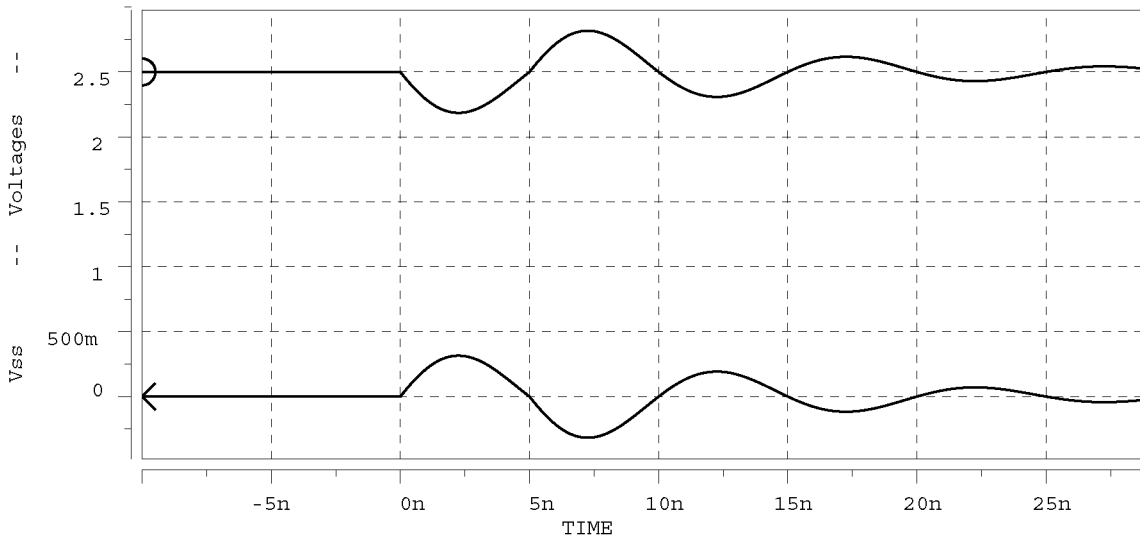


Figure 36—Example of a signal model used for simulation of supply voltage (Vss, Vdd) bounce within an IC due to simultaneous I/O switching

The traces in Figure 36 show typical voltage fluctuations due to the almost simultaneous switching of many single-ended outputs for a .18 micron CMOS chip, assuming an equal number of outputs switch from '0' to '1' and from '1' to '0.' The current surge peaks about 5 ns after the falling edge of TCK in the *Update-DR* TAP Controller state and returns to quiescent after about another 5 ns. These traces show sag of about 750 mV total and extended ringing that will depend on the RLC characteristics of the supply distribution network.

The testability circuitry provided by this standard minimizes the effects of simultaneous switching of outputs based on the way that signals are driven from the output boundary cells and captured at the receiver. Outputs under control of ac testing instructions (ac output pins) will switch additional times while the dc pins are not switching in order to avoid the noise generated by the dc pins. In addition, ac input pins, for both ac and dc testing instructions, will be sampled in a way that will minimize the window in which the noise effect can impact the test results. It is also assumed that switchin differential signals will not have a significant noise effect due to the current balancing that is inherent to differential signaling. Noise effects can still be minimized by judicious selection of hysteresis parameter values and well-designed board-level power distribution and bypassing. See 6.2.3 for discussion of selection of hysteresis parameters.

4.11.3 Noise effects generated by signal coupling

Inductive coupling from a signal (S) to a differential pair (D1 and D2) is shown Figure 37. This coupling effect might or might not be considered common-mode based on the layout of the printed wiring. If the distance of S from the differential pair is significantly greater than the distance, d, of the two traces of the differential pair, the effect will be nearly equal (i.e., common-mode) on both traces.

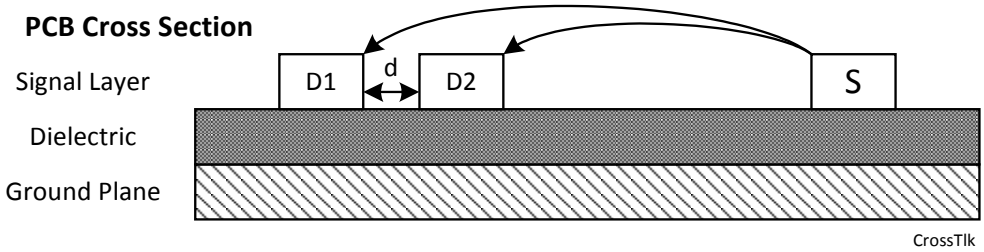


Figure 37—Crosstalk from a single aggressor signal (S) to two differential victim signals (D1 and D2)

This type of behavior could be considered as Best Design Practice, since the benefit of differential signaling will be lost if the coupling effect is greater on one of the two traces. As before, this standard will not address layout issues that cause coupling effects that also impact mission performance.

The magnitude of the coupling effect is related to the transition time of the signal, the distance from neighboring traces, and the trace "coupling length" for which this "coupling distance" is maintained. Coupling effects could be several hundred millivolts and could last for several nonoseconds based on the distance that the traces stay adjacent (coupling length).

Again, because ac-coupled and dc-coupled signals are switched and sampled at different times during the test, the coupling effect can be minimized to some extent. Since software exists that can model the effect of coupling, the test engineer should proactively work with the board designer to help ensure that coupling effects can be minimized (both magnitude and duration) on differential pairs that use test receivers. All coupling effects should fall within the hysteresis windows (meaning the test receiver will filter them out). The test engineer should be able to provide the design engineer with information pertaining to when signals switch and how they are sampled to determine possible problems areas that might need to be changed in the board layout

4.11.4 General notes to design and test engineers

It is important that the chip designer, board designer, and test engineer work proactively to minimize the effect of noise on the boundary-scan test. Chip designers should work with both board designers and test engineers to determine the optimal hysteresis settings based on the environment that the chip is expected to operate in for both mission and test modes. The test engineer must now work with the board designer to minimize noise effects in areas where these effects will impact test results and should work with the chip designers to optimize noise immunity for any components built to the specification herein. How to determine the appropriate hysteresis values to obtain acceptable noise immunity will be covered later in this standard (see 6.2.3). It is important that the board designer, chip designer, and test engineer all understand how to determine these values and their effect in order to minimize any potential impacts to testing.

Finally, as is the case with most "structural" test techniques [in-circuit test (ICT) and boundary-scan], the board should be "disabled" as much as possible during test. This would include controlling and/or disabling any functional clocks and any non-boundary-scan logic that could be actively transitioning during boundary-scan testing of the board.

5. Instructions

5.1 IEEE Std 1149.1 instructions

All instructions provided by IEEE Std 1149.1 perform as specified in that standard for all dc pins. For ac pins, all IEEE Std 1149.1 instructions perform as specified with the exception of an inversion between the boundary register cell and the negative leg of a differential driver (see 6.2.2).

5.1.1 Rules

All instructions specified by IEEE Std 1149.1 shall perform as specified in that standard, and for any such instruction that controls or observes pins, all ac and dc pins shall also perform as specified by that standard, with one exception: for any output or bidirectional pair of ac pins controlled by a differential driver, there shall be a logical inversion between the boundary-scan register data cell and one of the ac pins.

NOTE—In addition, the ac pins perform per rules specified herein.

5.1.2 Description

IEEE Std 1149.1 allows a single data register cell to control a differential driver but treats this situation as a digital-to-analog boundary, where no rules are given to govern the behavior of the analog portion. IEEE Std 1149.1 does maintain that there be no signal inversion between data register cells and digital pins. Subclause 5.1.1 clarifies that a single data register cell providing data to a differential driver that controls a pair of ac pins will have no inversion with one pin and signal inversion with the other. Thus a differential receiver connected to these two pins (and provisioned with independent test receivers on both legs, per 6.2) will see data and inverted data from the driver's boundary-scan register data cell.

IEEE Std 1149.1 is the foundation for this standard. The (DC) EXTEST instruction enables level-detecting behavior (see 6.2.2) on signal paths containing ac pins. This instruction is useful for detecting shorted board-level capacitors on ac-coupled paths, and certain other possible defects (defects 7, 9, 10, 12, 13 and 14 listed in Table 1).

5.2 AC testing instructions

This standard mandates the addition of two new instructions. The first is EXTEST_PULSE (see 5.3), and the second is EXTEST_TRAIN (see 5.4). These instructions are similar in that they cause the outputs of drivers connected to ac pins to change state at least two times after entering the *Run-Test/Idle* TAP Controller state, and differ in the details of how those transitions are generated.

Throughout this subclause, and in several others, these instructions are described as generating and processing an "ac test signal." This signal never appears at an I/O pin where it can be directly observed. It is a tool for visualizing the requirements of this standard and does not need to be implemented as documented herein. However, its actions upon other signals that are observable at I/O pins are required. The ac test signal essentially modulates test data so that it will propagate through ac-coupled channels, for devices that contain ac pins.

5.2.1 Recommendation

AC tests should use the EXTEST_PULSE instruction unless there is a specific requirement for the EXTEST_TRAIN instruction.

5.2.2 Description

A test receiver (see 6.2) is required to detect transitions rather than levels whenever an ac-test instruction is active and to detect levels otherwise. When testing with IEEE Std 1149.1, there might not be a transition after each data scan operation. That is, for a given boundary-scan register cell, the test data in the new pattern will often match the data in the old pattern. Further, all transitions occur essentially at the same time, potentially creating a lot of switching noise in the board test environment. This standard therefore provides two instructions that generate additional transitions on ac driver pins, controlled by entry to and exit from the *Run-Test/Idle* TAP Controller state. These additional transitions help ensure that the test receiver will have transitions to detect and also move the final transition relatively late in the test period, after the switching noise has dissipated. The transition direction is always relative to the value in the driver boundary register cell, so that the test receiver can recover the test data value.

Any ac-testable channel designed in accordance with the rules of this standard will exhibit dynamic behavior. Most often, this will simply be either the high-pass filter time constant of the ac coupling, or the low-pass or high-pass filter time constant of the edge-detecting circuitry of the test receiver (see 6.2.3), or both. It is the intent of this standard that the test receivers and mission drivers otherwise exhibit static behavior. That is, the driver and the hysteretic comparator of the test receiver would not add additional dynamic (time and input history dependent) behavior to the channel. However, we cannot ignore the possibility of dynamic logic or parasitic time constants.

The two new ac-test instructions provided by this standard differ primarily in the number and timing of transitions to provide flexibility in dealing with the specific dynamic behavior of the channels being tested. There could be other intentional or unintentional dynamic behavior that could affect the channel. This will often take the form of a node within the driver or test receiver, which drifts in voltage (accumulates charge) away from its operating point when held to a static input value. This would be normal in dynamic logic, and could also happen due to parasitic elements. If a single transition is sufficient to restore the normal operating point, then there would be no problem since even EXTEST_PULSE produces two transitions, and the response to the last transition is captured. In some cases, it could take many transitions to restore the normal operating point, and the EXTEST_TRAIN instruction would have to be used. Note that this assumes that the time required to drift away from the acceptable operating point range is many times the high-pass or low-pass time constants designed into the channel, and dictates the time between transitions for both instructions. In other words, the operating point must remain stable for long enough to allow the high and low pass filters to decay in order to achieve the behavior of the input signal that is assumed in ac-mode operation. Otherwise, it might not be possible to comply with this standard.

The EXTEST_PULSE instruction generates two driver transitions in addition to any transition at *Update-DR* and allows a tester to vary the time between them depending on how many TCK cycles the TAP is left in the *Run-Test/Idle* TAP Controller state. This time is intended to allow both the high-pass coupling filter and edge-detection filter to settle to a dc steady-state value.

The EXTEST_TRAIN instruction provides multiple additional transitions, the number of transitions depending on how long the TAP is left in the *Run-Test/Idle* TAP Controller state. If the TAP Controller is kept in the *Run-Test/Idle* state for a single TCK cycle, then both EXTEST_PULSE and EXTEST_TRAIN will generate identical output waveforms from the drivers when the driver Update cell has the same value. If the TAP Controller is kept in the *Run-Test/Idle* state for a number, n, of TCK cycles, then EXTEST_PULSE will generate two transitions with a time between the transitions of n TCK cycles, and EXTEST_TRAIN will generate 'n' or 'n+1' transitions, with a single TCK cycle between subsequent transitions (see Figure 38).

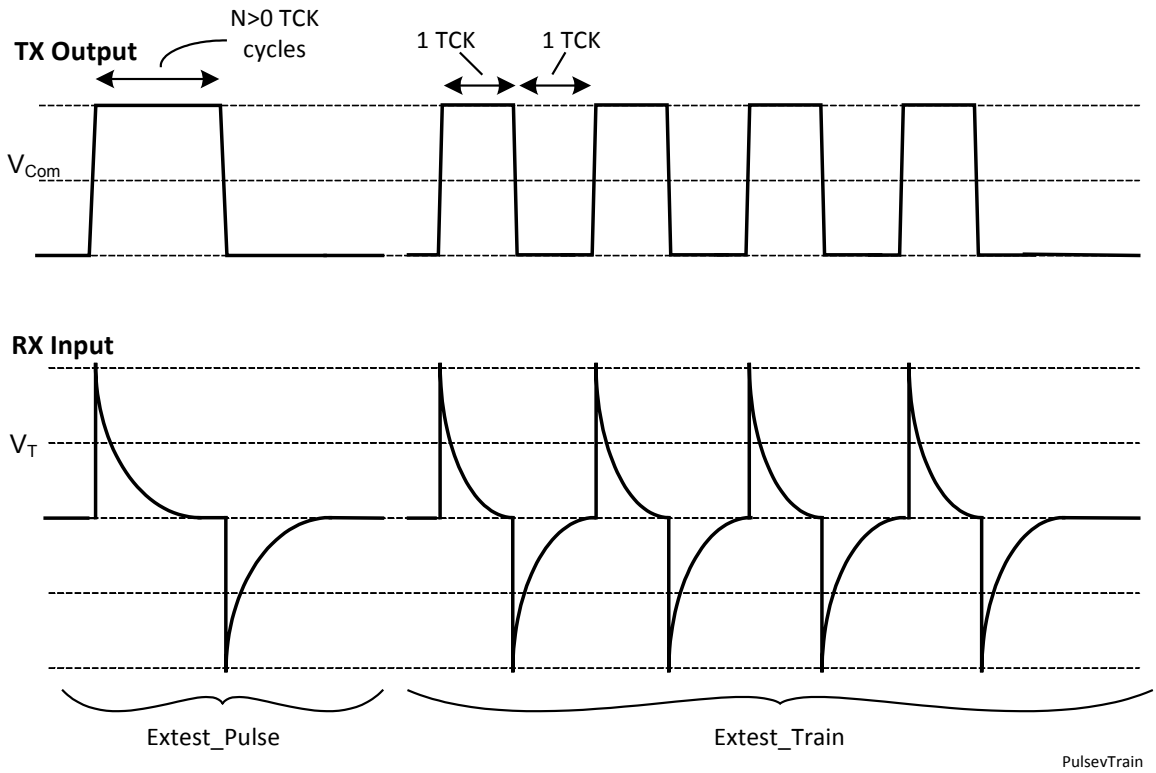


Figure 38— Comparison of EXTEST_PULSE and EXTEST_TRAIN performance.

If the minimum time between transitions (T_{Test}) required by rules j) and k) in 6.2.3.1 is greater than the TCK cycle time required by other tests, then the TCK cycle time would have to be extended during the *Run-Test/Idle* TAP Controller state for the EXTEST_TRAIN instruction. Some testers may have difficulty with such manipulation of TCK, and slowing TCK for the entire test is usually undesirable. Hence, the EXTEST_PULSE instruction is normally preferred.

EXTEST_TRAIN is intended to allow a transient condition, other than that of the high-pass coupling filter or edge-detection filter, to settle to the desired operating point, an ac steady-state value. EXTEST_PULSE assumes that any such transient behavior is negligible. Such transient behavior could be caused by using dynamic logic in the drivers or test receivers, or by secondary and even unintended parasitic dynamic effects. Industry experience has found that a few boards will experience false fails using EXTEST_PULSE, but will pass using EXTEST_TRAIN. While the result captured in the test receiver boundary-scan cell is determined by the last transition generated by the driver, which will be identical for both ac instructions, all transient conditions significantly affecting the board channel, especially the transition shape and size, must have died away before that transition and captured value can be considered valid.

For example, if the test receiver is designed for a channel that is guaranteed to be ac-coupled [see rule a) in 6.2.3.1], then the EXTEST_PULSE instruction would normally be used and the duration in the *Run-Test/Idle* TAP Controller state should be at least three times the high-pass coupling time constant [see rule k) in 6.2.3.1]. This allows the first additional transition to decay away to the dc steady-state value for the channel, and helps ensure that the full amplitude of the final transition is added to or subtracted from that steady-state value. This establishes a known initial condition for the final transition and permits reliable specification of the detection hysteresis of the test receiver. As this situation is explicitly anticipated and described in the rules, the EXTEST_PULSE instruction is recommended for use unless there is a specific reason to use the EXTEST_TRAIN instruction.

The EXTEST_TRAIN instruction also facilitates setting up a test that is based on a frequency appearing on pins. For example, in some test technologies, signals on devices are detected with capacitively coupled

sense plates positioned over the device (or portions of a device). These signals are AC, rather than single edges or a single pulse. The EXTEST_TRAIN instruction provides the opportunity to produce a $F_{TCK} / 2$ signal on pin drivers compatible with this standard. Other ad hoc tests using frequency can also be supported. Without the EXTEST_TRAIN instruction, it is often impractical to attempt to produce a frequency with the EXTEST instruction, since each fully driven cycle requires two full shift/update cycles with a BReg that could be excessively long, effectively dividing the TCK frequency by a large number. Thus, only relatively low frequencies could be attained.

NOTE— Note, IEEE Std 1149.8.1™ [B1][6], introduced in 2012, also allows the production of a frequency on drivers, but with the added capability to produce a frequency of F_{TCK} / n, where n is an integer between 2 and some upper bound integer N that allows the division of the maximum supported TCK frequency down to as low as 8 kHz. The other capability that IEEE Std 1149.8.1 has beyond what the EXTEST_TRAIN instruction provides is the ability to individually select whatever subset of pins will be subject to toggling, whereas this standard requires all (enabled) ac drivers to toggle when EXTEST_PULSE or EXTEST_TRAIN is in effect. IEEE 1149.8.1 is designed to be compatible with this standard, so that one device could support both standards.

5.3 The EXTEST_PULSE instruction

This standard specifies a new test mode instruction, EXTEST_PULSE, which governs new capabilities defined for ac pins (see 4.1). Whenever the EXTEST_PULSE instruction is active, all dc pins will perform as if the IEEE Std 1149.1 EXTEST instruction is active.

5.3.1 Rules

The following are the EXTEST_PULSE instruction rules:

a) An EXTEST_PULSE instruction shall be provided for components that possess ac pins.

b) The EXTEST_PULSE instruction shall become active at the falling edge of TCK in the Update-IR TAP Controller state.

   NOTE— By "active" it is meant that (enabled) ac drive pins respond to the content of the boundary-scan register as specified below, and ac receive pins behave as specified in 6.2.3.

c) The EXTEST_PULSE instruction shall select only the boundary-scan register to be connected for serial access between test data in (TDI) and test data out (TDO) in the *Shift-DR* TAP Controller state.

   NOTE— This is the same register, in length, organization, and construction, as that targeted by the EXTEST instruction.

d) DC pins shall perform exactly as specified for the EXTEST instruction by IEEE Std 1149.1 whenever the EXTEST_PULSE instruction is active.

e) When a device contains an ac pin with a driver, the driver is in an active (enabled) state, ac behavior is selected, and the EXTEST_PULSE instruction is active, and the output signal on that ac pin shall be controlled as follows (all conditions shall be met):
   1) The output signal shall be forced to the state matching the value in the associated boundary-scan register data cell for its driver (true and inverted values for a differential pair), at the falling edge of TCK in the *Update-IR* and *Update-DR* TAP Controller states.
   2) If ac behavior is selected for the pin, then the output signal shall transition to the opposite of that state (an inverted state) on the first falling edge of TCK that occurs after entering the Run-Test/Idle TAP Controller state.

[6] The numbers in brackets correspond to those of the bibliography in Annex F.

3) If ac behavior is selected for the pin, the output signal shall transition back to the original state (a noninverted state) on the first falling edge of TCK after leaving the Run-Test/Idle TAP Controller state.

4) The output signal shall not change state at any other time when the EXTEST_PULSE instruction is active.

NOTE 1— From IEEE Std 1149.1, a driver could have an optional control cell in the boundary-scan register that controls whether it is enabled to drive a valid state or produces an undriven state.

NOTE 2— A driver could have an optional control cell in the boundary-scan register that controls whether ac or dc behavior is selected (see 6.5). AC behavior is assumed if no cell is provided.

NOTE 3— This instruction is specifically exempted from IEEE Std 1149.1-2013, rule e) in 9.3.1, which would preclude the transition following the exit of *Run-Test/Idle* TAP Controller state.

f) When a device contains an ac pin with a driver, the driver is in an active (enabled) state, dc behavior is selected, and the EXTEST_PULSE instruction is active, the output signal on that ac pin shall be controlled exactly as specified for the EXTEST instruction by IEEE Std 1149.1.

g) When a device contains an ac pin with a driver, the driver is in an inactive (disabled) state, ac or dc behavior is selected, and the EXTEST_PULSE instruction is active, the output signal on that ac pin shall remain in an undriven state.

5.3.2 Permissions

The following are the EXTEST_PULSE instruction permissions:

a) The device designer may specify a minimum width requirement for pulses produced by the EXTEST_PULSE instruction that exceeds the minimum width necessitated by rules provided by this standard.

NOTE—See rules j) and k) in 6.2.3.1 for governance on minimum pulse widths.

b) The binary value(s) for the EXTEST_PULSE instruction may be selected by the device designer.

5.3.3 Description

The EXTEST_PULSE instruction implements new test behaviors for ac pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for dc pins.

The EXTEST_PULSE instruction enables edge-detecting behavior (see 6.2.3) on signal paths containing ac pins, where test receivers reconstruct the original waveform created by a driver even when signals decay due to ac coupling.

One possible mechanism for achieving the required behavior is with an internal signal called the ac test signal. The ac test signal will be Exclusive-ORed with the signal from the associated boundary-scan register data cell, and causes data produced by drivers to be inverted on the first falling edge of TCK after entering the *Run-Test/Idle* TAP Controller state and to be restored on the first falling edge of TCK after leaving this state, as shown in Figure 39. This generates a pulse of inverted data on a driver that is as wide as the time spent in the *Run-Test/Idle* TAP Controller state. If the *Run-Test/Idle* TAP Controller state is not entered, then the output behavior of the ac pins is not distinguishable from that of the (DC) EXTEST instruction.

Permission a) in 5.3.2 allows a designer to specify a minimum pulse width produced by the EXTEST_PULSE instruction that is longer than the minimum already governed by rules j) and k) in 6.2.3.1 for determining T_{Test}. A minimum pulse width allows the test receiver inputs to essentially decay to a static dc level. A designer could have other design parameters that require more time than this minimum.

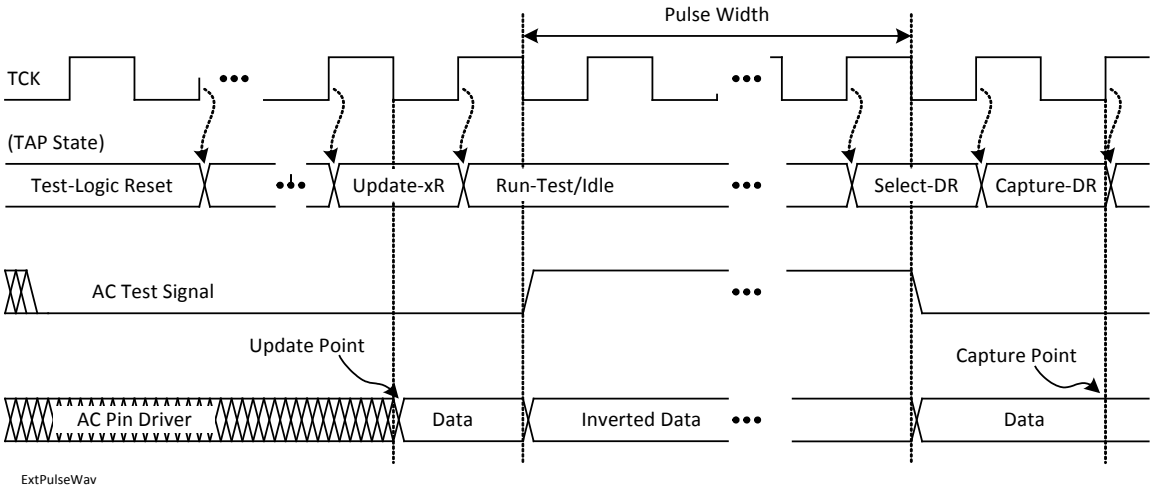


Figure 39— Behavior of (active high) ac test signal when EXTEST_PULSE is effective

5.4 The EXTEST_TRAIN instruction

This standard specifies a new test mode instruction, EXTEST_TRAIN, which governs new capabilities defined for ac pins (see 4.1). All dc pins will perform as if the IEEE Std 1149.1 EXTEST instruction is active whenever the EXTEST_TRAIN instruction is active.

5.4.1 Rules

The EXTEST_TRAIN instruction rules are as follows:

a) An EXTEST_TRAIN instruction shall be provided for components that possess ac pins.

b) The EXTEST_TRAIN instruction shall become active at the falling edge of TCK in the *Update-IR* TAP Controller state.

NOTE— By "active" it is meant that (enabled) ac drive pins respond to the content of the boundary-scan register as specified below, and ac receive pins behave as specified in 6.2.3.

c) The EXTEST_TRAIN instruction shall select only the boundary-scan register to be connected for serial access between TDI and TDO in the *Shift-DR* TAP Controller state.

NOTE— This is the same register, in length, organization, and construction, as that targeted by the EXTEST instruction.

d) DC pins shall perform exactly as specified for the EXTEST instruction by IEEE Std 1149.1 when-ever the EXTEST_TRAIN instruction is active.

e) When a device contains an ac pin with a driver, the driver is in an active (enabled) state, ac behavior is selected, and the EXTEST_TRAIN instruction is active, the output signal on that ac pin shall be controlled as follows (all conditions shall be met):

1) The output signal shall be forced to the state matching the value (a noninverted state) in the associated boundary-scan register data cell for its driver (true and inverted values for a differential pair), at the falling edge of TCK in the *Update-IR* and *Update-DR* TAP Controller states.

2) If ac behavior is selected for the pin, then the output signal shall transition to the opposite of that state (an inverted state) on the first falling edge of TCK that occurs after entering the *Run-Test/Idle* TAP Controller state.

3) If ac behavior is selected for the pin, the output signal shall invert its state on subsequent falling edges of TCK while still in the *Run-Test/Idle* TAP Controller state.

4) If ac behavior is selected for the pin, the output signal shall transition back to the original state (a noninverted state) on the first falling edge of TCK after leaving the *Run-Test/Idle* TAP Controller state.

5) The output signal shall not change state at any other time when the EXTEST_TRAIN instruction is active.

NOTE 1— From IEEE Std 1149.1, a driver could have an optional control cell in the boundary-scan register that controls whether it is enabled to drive a valid state or produces an undriven state.

NOTE 2— A driver could have an optional control cell in the boundary-scan register that controls whether ac or dc behavior is selected (see 6.5). AC behavior is assumed if no cell is provided.

NOTE 3— This instruction is specifically exempted from IEEE Std 1149.1-2013, rule e) in 9.3.1, which would preclude the possible transition following the exit of *Run-Test/Idle* TAP Controller state.

f) When a device contains an ac pin with a driver, the driver is in an active (enabled) state, dc behavior is selected, and the EXTEST_TRAIN instruction is active, the output signal on that ac pin shall be controlled exactly as specified for the EXTEST instruction by IEEE Std 1149.1.

g) When a device contains an ac pin with a driver, the driver is in an inactive (disabled) state, ac or dc behavior is selected, and the EXTEST_TRAIN instruction is active, the output signal on that ac pin shall remain in an undriven state.

5.4.2 Permissions

The EXTEST_TRAIN instruction permissions include the following:

a) The device designer may specify a minimum number of pulses that shall be produced by a device performing the EXTEST_TRAIN instruction while the TAP Controller is in the *Run-Test/Idle* state.

NOTE—No maximum number is to be specified. This permission governs the minimum number of TCK cycles that should be provided by a tool while in the *Run-Test/Idle* TAP Controller state.

b) The device designer may specify a maximum period of time within which the minimum number of pulses specified by permission a) above shall occur, provided the time between transitions meets the rules provided by this standard.

NOTE 1— See rule e) in 6.2.2.1 and rules j) and k) in 6.2.3.1 for determining T_{Test}.

NOTE 2— For example, 10 pulses are required and should occur in no more than 5 ms. This implies 20 TCK cycles within 5 ms, or an average TCK frequency of at least 4 kHz.

c) The binary value(s) for the EXTEST_TRAIN instruction may be selected by the device designer.

5.4.3 Recommendations

a) Device designers should avoid exercising permission b) in 5.4.2 whenever possible.

b) If use of permission b) in 5.4.2 cannot be avoided, then the required maximum time requirement should be made as large as possible.

5.4.4 Description

The EXTEST_TRAIN instruction implements new test behaviors for ac pins and simultaneously behaves identically to IEEE Std 1149.1 EXTEST for dc pins.

The EXTEST_TRAIN instruction enables edge-detecting behavior (see 6.2.3) on signal paths containing ac pins, where test receivers reconstruct the original waveform created by a driver even when signals decay due to ac coupling.

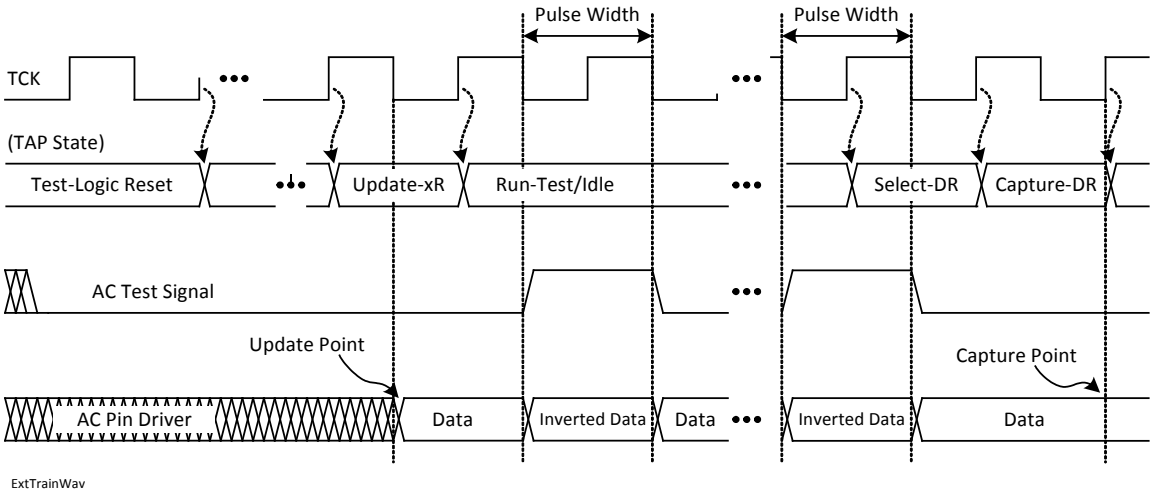


Figure 40 — Behavior of (active high) ac test signal when EXTEST_TRAIN is effective

NOTE 1—Depending on when the *Run-Test/Idle* TAP Controller state is exited (on an even or odd count of TCK cycles), there could be an extra half cycle of time where the ac test signal is held low (on odd counts). This increases the window of noise vulnerability by 1/2 TCK cycle.

NOTE 2—If, as allowed by IEEE Std 1149.1, TCK is halted, a pulse or time between pulses could be stretched.

The EXTEST_TRAIN instruction, by action of the ac test signal, causes data produced by drivers to be inverted on the first falling edge of TCK after entering the *Run-Test/Idle* TAP Controller state, and to be subsequently toggled on each falling edge of TCK while remaining in this state as shown in Figure 40. This generates multiple pulses on a driver, each pulse cycle being two cycles of TCK wide. The first falling edge of TCK after leaving the *Run-Test/Idle* TAP Controller state will restore driver data if it is not already matching the value in the Update flip-flop. If the *Run-Test/Idle* TAP Controller state is exited after only one cycle of TCK, then the EXTEST_TRAIN instruction is not distinguishable from the EXTEST_PULSE instruction. If the *Run-Test/Idle* TAP Controller state is not entered, then the ac output pin behavior while the EXTEST_TRAIN instruction is active is not distinguishable from that of the (DC) EXTEST instruction.

The derivation of the output signal inversions from the falling edge of TCK when generating a pulse train helps ensure that the duty cycle of TCK does not affect the "squareness" of the pulse train. However, interruptions in TCK that are allowed by IEEE Std 1149.1 will have the effect of stretching a pulse or duration between pulses. Typical TCK frequencies used in test equipment could vary from 100s of kilohertz to 10s of megahertz. The TCK rate used to drive a chain of devices is also limited by the slowest TCK rate of any member of that chain, as documented in BSDL. This standard is intended for use in many device types, including devices where system clock rates are several decades higher than these typical TCK rates. The device designer needs to remember this basic range on TCK rates and how it could be quite different from the mission performance range.

Permission a) in 5.4.2 allows a designer some flexibility in driver implementation if the driver has dynamic behavior that must be conditioned by some minimum number of pulses before a downstream receiver actually samples its output. If needed, permission b) in 5.4.2 allows the designer to further state a time frame within which these pulses should occur. However, this begins to approximate a "minimum TCK" requirement that should be observed by testing tools, which might prevent a given tool from being able to perform the EXTEST_TRAIN instruction. Recommendation a) in 5.4.3 advises designers not to add this restriction whenever possible, so as to avoid implementing an instruction that cannot be practically used. Recommendation b) in 5.4.3 advises the use of a large maximum time period when such time is needed.

5.5 ac test signal generation

A possible implementation for generation of an ac test signal is shown in Figure 41, for both the EXTEST_TRAIN and EXTEST_PULSE instructions. This circuit would probably be located near the TAP Controller since it uses decoded information (e.g., RTI State and Pulse/Train) from the TAP Controller to generate the global ac test signal.

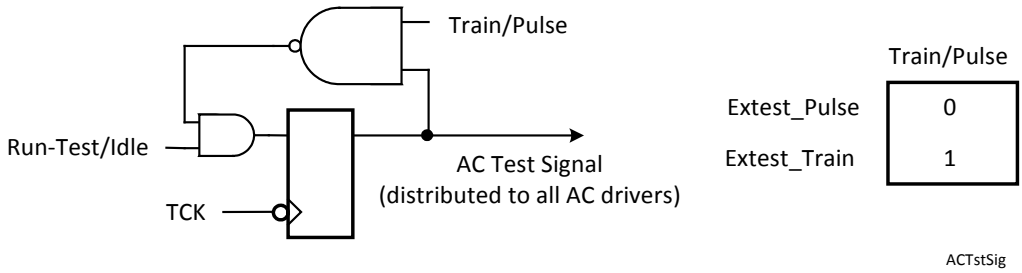


Figure 41 — Possible circuitry to generate a global ac test signal

6. Pin implementation specifications

6.1 Pin identification

The device designer is expected to survey the complete set of pins of the device and categorize them as the following:

— Power/ground and other analog reference pins

— TAP signal pins (per IEEE Std 1149.1-2013, Clause 4, The Test Access Port)

— Compliance dc pins: I/O signal pins specifically addressed by IEEE Std 1149.1 enable pins (per IEEE Std 1149.1-2013, 4.8, Subordination of this standard within a higher level test strategy)

— AC pins: I/O signal pins that are differential or are expected by design to support ac coupling to other signal pins, or other advanced designs that require the test structures of this standard

— DC pins: I/O signal pins specifically addressed by IEEE Std 1149.1 or other Standards in the IEEE 1149 family

Thus a designer can partition pins into those governed by this standard (i.e., ac pins), and those governed by IEEE Std 1149.1 or other Standards in the IEEE 1149 family.

6.1.1 Rules

Pin implementation rules include the following

a) The device designer shall enumerate a possibly empty set of device signal pins that require the additional test capabilities defined in this standard and shall designate these pins “ac pins.”

NOTE—At this release of this standard, there would be no reason to implement this standard if the set of ac-coupled pins is empty. The “possibly empty” modifier anticipates future additions to this standard.

b) The device designer shall categorize all other device pins per the rules in IEEE Std 1149.1 or other Standards in the IEEE 1149 family.

NOTE—Those I/O signal pins identified per IEEE Std 1149.1 are herein collectively called "dc pins.”

6.1.2 Description

The device designer effectively partitions the I/O signals into those addressed only by IEEE 1149.1 (or other standards in the 1149 family) and those (ac pins) addressed by this standard.

6.2 Input test receivers

All ac pins (see 4.1) that receive data into an IC (with the exception of self-monitoring outputs) are equipped with test receivers. Single-ended ac pins will have one test receiver and differential channels will have one test receiver per leg. The IC designer will design the mission performance of the input to accept some range of voltage changes and slew rates. The rules given below assume that an input will respond to a minimum input voltage change and that the slew rate of input changes will be at or above some minimum. These minimums are defined by the performance requirements of the mission of the IC and are depicted in Figure 42 for both ac- and dc-coupled signals.

When an ac testing instruction is active, it is the purpose of the test receiver to reconstruct the test waveform driven by the upstream driver when either ac- or dc coupling is used. It does this by reacting to the edges and not the levels of the input waveform. When (DC) EXTEST is in effect, the test receiver behaves as a level detector.

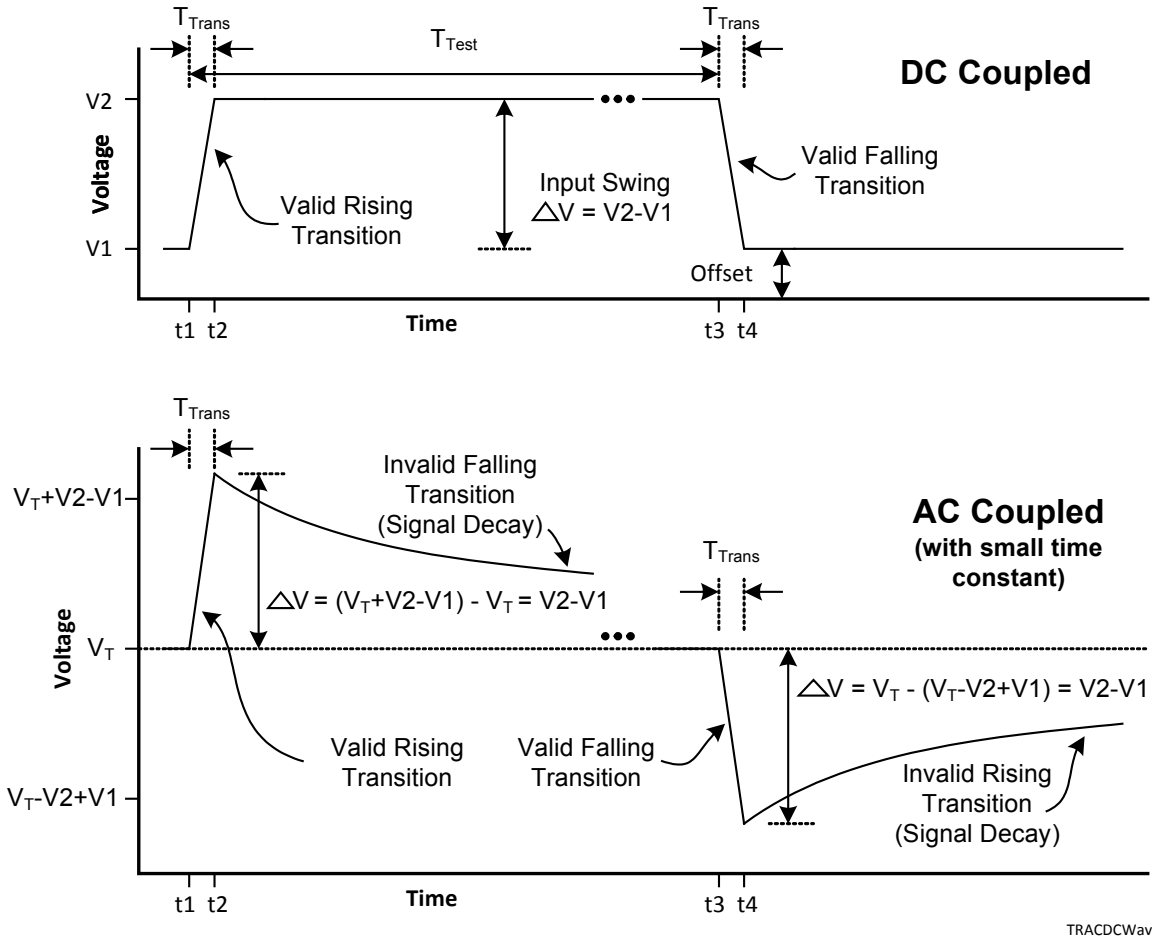


Figure 42—Definitions of voltage changes and transition times for waveforms presented to the test receiver using either dc- or ac coupling

The value of T_{Test} is the minimum time for the input signal to decay. It is important for the signal to decay almost completely (at least three dominant time constants for the net) after a signal transition that can be caused by boundary-scan testing and applies to both dc and ac-coupled input signals. This value is dependent on test-related parameters such as TCK frequency for the device (for EXTEST_TRAIN) or the time spent in *Run-Test/Idle* (for EXTEST_PULSE), or the time between falling TCK edges in *Update_xR* and *Capture_DR* (for EXTEST).

NOTE— IEEE Std 1149.1 EXTEST does not take attenuation or transition time into account. The test transitions are very far apart (at best, once per scan and update of the boundary-scan register). Receiver values are captured no sooner than 2.5 TCK cycles after a possible transition.

The *minimum input swing,* termed ΔV_{Min}, is the smallest change in voltage during an active transition (ΔV in Figure 42) that is expected to occur during tests specified by this standard. Typically, this will be the ΔV_{Min} of a driver of the same protocol and technology reduced by some factor to account for attenuation expected at the maximum test frequency $(1/(2T_{Test}))$ in Figure 42. This attenuation is generally significantly less than the attenuation expected at maximum functional channel frequency. The transition time, T_{Trans} in Figure 42, is the maximum amount of time this minimum voltage swing is expected to take. Signal transitions that take significantly longer than T_{Trans} are not considered valid. For example, in Figure 42, the passive, undriven, falling decay on a capacitively coupled signal that follows an actively

driven rising transition is not a valid falling transition. The rules that follow will define a test receiver that can differentiate a valid signal transition from a decay transition.

In the following rules, a designer is required to identify parameters seen at the test receiver inputs such as minimum and maximum voltage swing (ΔV_{Min}, ΔV_{Max}) and maximum transition time (T_{Trans}), but with some latitude.

The preferred way to determine these parameters is to start with the transmission protocol specification. For example, the LVDS Standard (IEEE Std 1596.3 or ANSI/TIA/EIA-644) specifies that the voltage swing at the driver should be 454 mV maximum and 247 mV minimum. It also states the voltage swing at the receiver should be 100 mV minimum, but this is at maximum data rate, as shown in Figure 43. The standard does not state an expected attenuation at slower test data rates e.g., 5 MHz to 50 MHz, but conservative engineering judgment might suggest that the minimum voltage swing at the receiver, for that frequency, would be no less than, for example, 80% of the minimum swing at the driver, or about 200 mV in this example. The maximum swing of the driver (454 mV) could then be used as ΔV_{Max} and the attenuated minimum swing of the driver (200 mV) used as ΔV_{Min}.

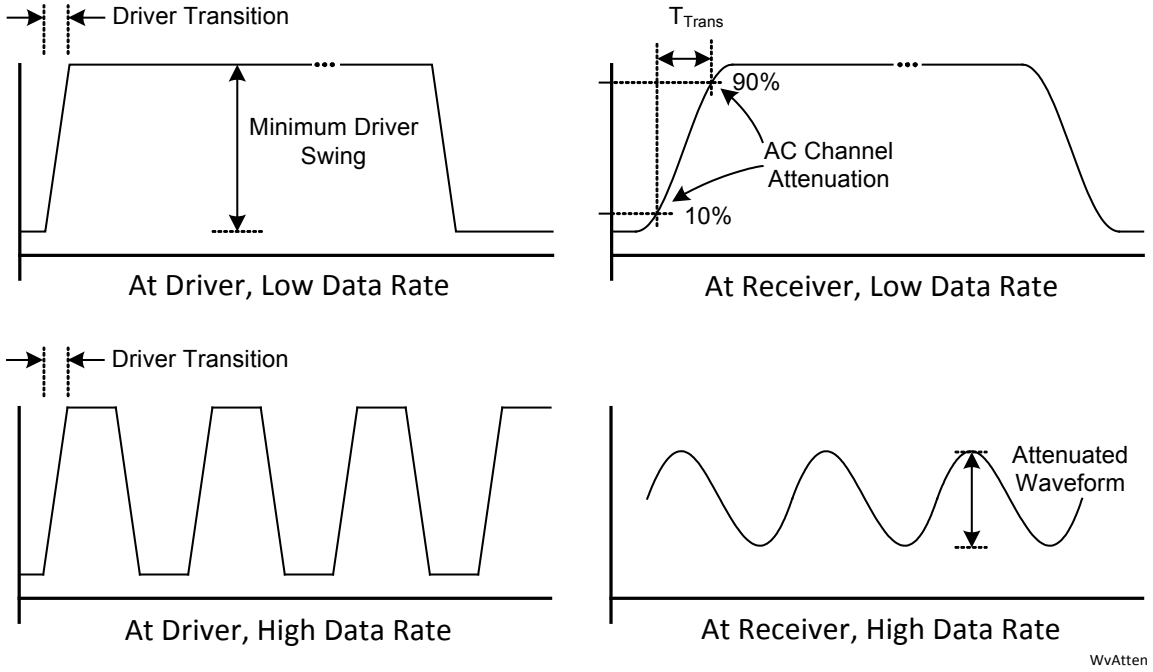


Figure 43 — Driver waveform and the attenuated waveform seen at the receiver. The attenuation is a function of driver data rate.

The LVDS standard further specifies a maximum driver transition time of 1.5 ns for data rates greater than 200 Mbps, or 30% of the bit-width otherwise. The standard documented here provides that the transition time of the driver is the same in test mode as in functional operation, so one should be able to use this same value for test. However, the transition time at the receiver will be longer than the transition time at the driver due to high frequency attenuation, even in low data-rate operation. So at the receiver, a better estimate for T_{Trans} would be 50% of the minimum bit time (25% of the maximum channel frequency cycle time) since longer transition times than that would cause severe amplitude attenuation at the maximum data rate and probably compromise the reliability of the channel. To continue the example, assume an LVDS driver and receiver designed for 200 Mbps maximum bit rate, then the bit time would be 5 ns. The LVDS standard dictates 30% of the bit time, or 1.5 ns, for the driver transition time, and for this standard one could use 50% of the bit time, or 2.5 ns, for the receiver transition time called T_{Trans}.

When there is no other guidance, a possible way to determine these parameters is to consider the actual characteristics of a driver implemented in the same technology for the same channel protocol. Over acceptable manufacturing process variation, temperature variation, etc., this driver will produce a maximum and a minimum voltage swing into a specified range of loads. This voltage swing will normally be larger than needed to assure the receiver reliably recovers data, to allow for high frequency attenuation in the channel between the driver and receiver (see Figure 43). As described above for the LVDS example, a conservative attenuation factor for the maximum expected test frequency can be determined using appropriate engineering tools and judgment, and applied to the minimum voltage swing to determine an appropriate minimum voltage swing during test at the test receiver (the maximum voltage swing is not usually adjusted to allow for very tight coupling with negligible attenuation).

In addition, both driver and receiver will be designed for a specific maximum channel frequency or bit width. Again, a good estimate for T_{Trans} at the receiver, in a typical channel design, would be 50% of the bit width or 25% of the cycle time at maximum frequency.

If the designer is expecting the receiver to be connected, ac- or dc coupling, to a driver of a different transmission protocol, or implemented in a different technology, then analysis of the board channel design will have to be performed to verify that the channel will attain the desired data rate. By extending this analysis, using the techniques described above, the designer can determine the required parameters.

These types of considerations allow the designer to define the minimum and maximum voltage swings and transition time required by the following rules. This “test design point” information could be supplied as part of the data sheet for the final device.

The offset voltage shown in Figure 42 could vary depending on different circuit configurations. For example, an IC with an input pin possessing a test receiver might be ac- or dc-coupled to a driver intentionally by a board designer. Depending on which is chosen, there could be different offsets observed at the test receiver. When a defect occurs, the coupling could be affected. For example, a shorted coupling capacitor could create dc coupling to a receiver that was intended to be ac-coupled.

6.2.1 General properties of ac input pins and test receivers

6.2.1.1 Rules

The ac input pins and test receivers rules include the following:

a) All ac pins (see 4.1) that receive input data, whether they are single-ended or one leg of a differential pair, shall have exactly one test receiver function monitoring the mission pin signal.

NOTE 1— This rule applies to input and bidirectional pins (see ac pin definition in 4.1).

NOTE 2— The test receiver is connected to a signal equivalent to the corresponding mission receiver input. This could be after a coupling capacitor or linear buffer and is not necessarily the actual input pad. For simplicity of discussion, however, the term “pin” is often used.

b) With respect to the required or expected mission behavior of the input signal at an ac input pin, the device designer shall identify a *minimum valid signal transition* in terms of its *minimum input swing* (ΔV_{Min}) and maximum transition time (T_{Trans}) over which this minimum input swing could occur.

NOTE 1— In the event that the pin is one-half of a differential pair, the determination of ΔV_{Min} is with respect to a fixed reference (e.g., ground) and not with respect to the other member of the differential pair.

NOTE 2— If the transition times of rising and falling edges are different, then the longer of the two is defined as the maximum.

NOTE 3— See permission c) in 6.2.1.2 for the case where there is no maximum transition time specification for the mission receiver. See also the discussion in 6.2 on determining T_{Trans} .

NOTE 4— All parameters identified (from design specifications, analog simulations, etc.) are documented per rule 7.2.1.1 e) in either BSDL or in a printed datasheet.

c) With respect to the required or expected mission behavior of the input signal at an ac input pin, the device designer shall identify a *maximum valid signal transition* in terms of its *maximum input swing* (ΔV_{Max}).

NOTE—If the pin is bidirectional, then the pin driver could determine ΔV_{Max}, but some external, ac attenuated driver would be used to determine T_{Trans} and ΔV_{Min} .

d) The test receiver shall be implemented to operate in two modes of operation:

1) A level-detecting mode selected by the (DC) EXTEST instruction (see 6.2.2), and

2) an edge-detecting mode selected by an ac testing instruction (see specifications in 6.2.3).

e) The test receiver shall be implemented with hysteresis voltage offsets for the two operational modes called:

1) V_{Hyst_Level} in the level-detecting mode (see 6.2.2), and

2) V_{Hyst_Edge} in the edge-detecting mode (see 6.2.3).

f) The output (hysteretic memory) of a test receiver shall be cleared of any prior state at a time determined by the active instruction.

NOTE—The details of when this occurs are provided in 6.2.2 and 6.2.3 for level- and edge-detecting modes, respectively.

g) If the test receiver detects a valid input after having been cleared as specified in rule f) above, the output of the test receiver shall be captured in a boundary-scan register cell capture flip-flop at the rising edge of TCK in the *Capture-DR* TAP Controller state; otherwise the content of the boundary-scan register cell capture flip-flop shall be preserved.

h) Whenever the test receiver is provided with a programmable threshold voltage, hysteretic voltage, or bypassable on-chip coupling capacitor, the programming shall be accessible via the TAP.
NOTE—The fact of programmability and the TDR register fields used are to be documented in BSDL and the procedures for programming are to be documented in PDL per the requirements of Clause 7. Also, this does not preclude other access to the programmability.

i) Whenever the test receiver is provided with a programmable bypassable on-chip coupling capacitor, then the TDR field controlling the bypass function shall be reset by one of the test reset signals defined in IEEE Std 1149.1, and the reset value shall be the non-bypassed (ac-coupled) state.

NOTE—Additional test reset signals were initially defined in the 2013 version of IEEE Std 1149.1.

6.2.1.2 Permissions

The ac input pins and test receivers permissions include the following:

a) A test receiver may be isolated from its signal with a linear buffer as long as it still sees the actual waveform appearing on the mission receiver input pin.

b) Circuitry needed to implement the test receiver may be shared with that needed for the implementation of the mission circuitry, as long as all rules for this standard are observed.

c) If the mission receiver has no specification for a maximum transition time, then the value of T_{Trans} may be chosen to match the maximum expected transition time for an upstream driver that is in test mode, taking into account any expected ac attenuation in the path.

NOTE—The discussion in 6.2 on determining T_{Trans}. See also Annex C for examples using existing logic families.

d) User-defined instructions may also clear the output (hysteretic memory) of a test receiver to suit the needs of those instructions.

6.2.1.3 Description

The rules and permissions in 6.2.1 require that there be a bimodal test receiver on each and every ac input pin, and define five electrical parameters based on the expected mission-mode input signal, that are used to design the test receiver. Consider rule b) in 6.2.1.1. This rule applies to signal transitions within any common-mode or fixed voltage range definition of the mission of the pin. Changes outside of this range are not valid signal transitions. For example, a signal transition that forward biases input protection diodes would be outside the normal operating range of the receiver. In addition, the minimum valid signal transition is the smallest, slowest transition that represents a valid state change of the signal, i.e., that represents a change in signal data (and is to be differentiated from a signal decay seen in an ac-coupled channel). This could be determined by examining the characteristics of a driver implemented in the same technology, its minimum specifications, and the maximum amount of signal attenuation allowed between the driver and receiver. The transition parameters (T_{Trans} and ΔV_{Min}) are used in subsequent rules.

Permission c) in 6.2.1.2 allows a device designer to substitute the maximum transition time of an upstream driver in test mode that is expected to be connected to this test receiver. This handles the case where there is no maximum transition time specified for the mission receiver, for example, a low frequency analog input. A designer can thus avoid having to implement edge-detection for very slow edges. However the designer must assure that the upstream driver's parameters are known, which this often is the case for custom designs or standard logic families.

Rule c) in 6.2.1.1 defines a maximum input voltage swing. This value, the tolerable amount the receiver can be overdriven, is used in subsequent rules to improve the noise immunity of the test receiver.

Rule d) in 6.2.1.1 requires that the test receiver have a level-detection mode (familiar from IEEE Std 1149.1) and also an edge-detecting mode. The edge-detecting mode is required since ac coupling is a high-pass filter that will not propagate dc levels.

Rule e) in 6.2.1.1 states the test receiver will be implemented with voltage hysteresis levels, with a specification for level- and edge-detecting modes. While this hysteresis provides noise immunity, it also defines the test receiver as having three possible detection states: legal "one," legal "zero," and an "indeterminate" value in between. Due to the decay behavior of dc signals when transmitted on an ac-coupled net, the explicit detection of the "indeterminate" input state is critical to defect detection and diagnosis. More specifications on edge and level detection are given in the following subclauses as a function of these hysteresis parameters.

Rules f) and g) in 6.2.1.1 define the test window within which a valid input could change the test receiver output. This window starts when the test receiver output (hysteretic memory) is cleared of prior history, as specified in rule f) in 6.2.1.1, and ends with the rise of TCK in the *Capture-DR* TAP Controller state, as specified in rule g) in 6.2.1.1.

Rule g) in 6.2.1.1 also associates a test receiver with a boundary-scan register cell, particularly the capture flip-flop therein. The capture flip-flop will be loaded with a default data value during normal shifting in the

Shift-DR TAP Controller state. If no valid inputs are detected by the test receiver within the test window, this default value will be scanned back out. When valid inputs are detected during the test window, then the current output of the test receiver will be captured upon the rise of TCK in the *Capture-DR* TAP Controller state.

One structure that meets the requirements of these two rules, and is used for illustration throughout the text, figures, and timing diagrams of this standard preloads the content of the boundary-scan register cell capture latch into the hysteretic memory at the time the memory is to be cleared of previous history. Then the test receiver output could always be captured in the boundary-scan register cell capture flip-flop, since the preloaded default value will not be changed if there are no valid inputs to the test receiver. See 6.2.5 for such a possible integration of a test receiver with a boundary-scan register cell.

6.2.2 Level-detection behavior of input test receivers

The level-detection behavior is selected by the (DC) EXTEST and SAMPLE instruction from IEEE Std 1149.1 and any user instructions that select the boundary-scan register in a dc test mode.

6.2.2.1 Rules

The input test receiver level detection rules include the following:

a) Whenever a test receiver is operating in the level-detection mode on an ac input signal, it shall determine the value of the input signal by comparison with a fixed threshold voltage $V_{Threshold}$ and a hysteresis voltage offset value V_{Hyst_Level} set by one of two methods:
   1) By setting $V_{Threshold}$ to the mid-range (common-mode voltage) of the minimum input voltage swing ΔV_{Min} and setting V_{Hyst_Level} to one half of V_{Hyst_Edge}; or
   2) if a receiver is specified to be ac-coupled to its driver, the only expected use of level-sensitive mode in test is detection of a shorted coupling capacitor, and the possible range of drivers is such that no single $V_{Threshold}$ would allow reliable determination of the driven value, by setting $V_{Threshold}$ to a voltage at one or the other limit of the test receiver input range and V_{Hyst_Level} to any convenient level.

   NOTE 1— For the purpose of determining $V_{Threshold}$, "fixed threshold" is interpreted as explicitly prohibiting any scheme (such as a filtered average of the voltage on differential pins) that dynamically generates $V_{Threshold}$ from the signals being measured, but permitting programmable threshold voltages that are set during initialization.

   NOTE 2— The choice of option 1) or 2) is documented in BSDL per rule h) in clause 7.5.6.2. If there is no on-chip capacitor, then any requirement of ac coupling on the board would normally also be documented in the specification sheet for the component.

   NOTE 3— When section 2) of rule a) in 6.2.2.1 is chosen, $V_{Threshold}$ might be set as close to a voltage rail (VDD or GND) as the hysteretic comparator in the test receiver will tolerate, and V_{Hyst_Level} might be set equal to V_{Hyst_Edge}, to simplify the design of the test receiver. This option does not determine the driven value, but only whether there is a dc path from the driver to the receiver (a fault condition). The shorted board-level capacitor test will retain the initialized default state for a good capacitor (good machine) per part 3) of rule b) in 6.2.2.1 and as also discussed below around Figure 44 and Table 2 for ac-coupled interface 'dc test.'

b) Whenever a test receiver is operating in level-detection mode on an ac input pin, the test receiver shall compare the current voltage on the pin to a fixed threshold voltage , $V_{Threshold}$, for a minimum period of time called the hysteresis delay, T_{Hyst} , and performing one of the following actions:
   1) output a logic one only after this period has elapsed and if the voltage continuously compares greater than this threshold $V_{Threshold}$ plus the hysteresis voltage V_{Hyst_Level} , or

2) output a logic zero only after this period has elapsed and if the voltage continuously compares less than this threshold $V_{Threshold}$ minus the hysteresis voltage V_{Hyst_Level}, or

3) maintain its current output state if the voltage compares less than this threshold $V_{Threshold}$ plus the hysteresis voltage V_{Hyst_Level} and greater than this threshold $V_{Threshold}$ minus the hysteresis voltage V_{Hyst_Level}.

c) The device designer shall choose the period T_{Hyst} to be significantly longer than the slowest expected transition time T_{Trans}.

d) Whenever a test receiver is operating in the level-detection mode on an ac input pin, the test receiver output shall be cleared of prior history on the falling edge of TCK in the *Capture-DR* TAP Controller state.

e) When a net is ac-coupled and a dc testing instruction is active in any IC receiving the ac-coupled signal, then the time between falling TCK in the *Update_IR* or *Update_DR* state and the falling edge of TCK in the *Capture_DR* state shall be no less than three times the actual coupling filter time constant (T_{Test}) of any such connection between devices being tested.

NOTE— The T_{HP} parameter specified for each ac pin is a minimum for an off-chip filter and actual for an on-chip filter. T_{Test} must be calculated using actual coupling time constants.

6.2.2.2 Permissions

The input test receiver level detection permissions include the following:

a) Per rules d) in 6.2.2.1 and g) in 6.2.1.1, the output of the test receiver is only relevant during the window of time between the falling and rising edges of TCK in the *Capture-DR* TAP Controller state, and this output may be considered a “don’t care” at other times.

b) With respect to the SAMPLE instruction, the device designer may choose to have test receiver input capture registers capture a static value of 0/1 rather than attempt to sample the ac inputs on the rising edge of TCK in the Capture-DR state.

NOTE—Additional observe-only cells are allowed by IEEE Std 1149.1 on any pin (see IEEE Std 1149.1-2013, clause 11.8). This permission simply states that they are explicitly allowed on non-redundant ac test receiver input pins

6.2.2.3 Recommendations

The input test receiver level detection recommendations include the following:

a) The amount of hysteresis delay, T_{Hyst}, should be chosen by the designer to reject common noise sources such as ringing or over/undershoot that could occur on a signal pin.

b) Whenever system specifications such as common-mode voltage range indicate that operation of the dc mode of the Test Receiver could be problematic, then option 2) of rule a) of 6.2.2.1 should be taken and, further, a boundary cell conforming to IEEE Std 1149.1 should be added on the output of the mission mode receiver.

NOTE—Option 2) of rule a) of 6.2.2.1 assumes that the driver to test receiver connection is ac-coupled, but even in this case, the possible range of driver common-mode voltages must be taken into account in order to perform the test for a shorted board-level capacitor.

c) Whenever the test receivers and their associated boundary cells are designed so they do not support the SAMPLE instruction, then a boundary cell conforming to IEEE Std 1149.1 should be added on the output of the mission mode receiver.

NOTE—The 2013 edition of IEEE Std 1149.1 added a permission (11.8.1d) that allows a redundant observe-only boundary cell to capture static 0/1 data during SAMPLE instruction operation, rather than pin data. The boundary cell associated with the test receiver will generally be covered by this permission.

6.2.2.4 Description

Rule a) in 6.2.2.1 defines two different testing methods for dc tests such as the EXTEST instruction defined in IEEE Std 1149.1. The first method, given by part 1) of rule a), defines a level-sensitive test receiver that is a fixed-threshold single-ended logic comparator with hysteresis and is designed to conform to the requirements of IEEE Std 1149.1. The second method, given in part 2) of rule a), defines a level sensitive receiver with a threshold fixed at one limit of the test receiver input range an does not conform to the requirements of IEEE Std 1149.1.

NOTE 1—See paragraphs immediately following Figure 44 for an explanation of how this method works.

In both cases, per rule a) in 6.2.2.1, the designer defines the parameters of this comparator. As discussed in clause 4.6.5, the voltage swing seen on a single net of a pin defined in this standard as an ac pin often will be small and will ride on a relatively large and possibly unknown offset. In a level-sensitive test, there is no way of removing this offset, and the offset has to be known for the test to be able to detect both the '0' and '1' signal levels from the driver. In some cases this offset is known. For example, a test receiver that only monitors the output of a driver can usually be designed to share the offset with the driver. In application-specific integrated circuit (ASIC) components using the same technology and intended for board designs where they will only talk to each other, it is likely that the technology provides drivers and receivers using the same constant offset voltage, eliminating the problem. In other designs such as commodity parts, the offset seen at a receiver could be completely unknown at the time of chip design, and this method might not work (the edge-detection mode, described in 6.2.3, overcomes the unknown offset problem). See Figure 33 for an example of one way to implement the (DC) EXTEST capability.

Given the inability in all cases to predetermine the "correct" $V_{Threshold}$, it is tempting to try to generate an appropriate threshold from the actual voltage of the pin or pins to be measured. Such schemes were investigated with Spice simulations early in the development of this standard, and while they work well in the good machine case, they produce unpredictable results in the presence of faults and would make detection problematic and make diagnostic analysis of the results nearly impossible. Such schemes are therefore explicitly prohibited. Other means of providing a dynamic threshold voltage that is independent of the signals being measured, such as a voltage divider on the board, the component driving the bus providing an analog threshold reference voltage, or programmable threshold voltages in the receiving or driving components, would meet the requirements in this standard and provide better compatibility between components on the board.

In any case, including a traditional boundary cell on the output of the mission mode receiver in addition to the structures specified in this standard, as described in recommendation b) of 6.2.2.3, will help ensure, as a minimum, the EXTEST coverage achieved by IEEE Std 1149.1. Similarly, while changes to both this standard and IEEE Std 1149.1 would allow an ac pin to not support the SAMPLE instruction, a boundary cell on the mission receiver output that would capture in SAMPLE is recommended.

Rule b) in 6.2.2.1 defines a *valid logic level* as a signal persisting at one logic level at least a hysteresis level beyond the threshold for a minimum period of time. The IC designer could choose a threshold value at the midpoint of a voltage swing deemed optimal for the mission receiver. This could be the same voltage used to bias the mission receiver when it is ac-coupled as seen in Figure 13. The hysteresis voltage, V_{Hyst_Level}, will eliminate response to small-amplitude noise. The hysteresis delay period, T_{Hyst}, will eliminate response to short-duration large-amplitude pulses. For example, if the hysteresis delay is set to five times the value of

T_{Trans}, this will allow the receiver to ignore typical ringing that could occur on transitions or due to other coupled noise. The designer should contemplate the nature of noise sources that might impact signals received at a pin when choosing the hysteresis delay (see 4.11).

Rules d) in 6.2.2.1 and g) in 6.2.1.1 define the behavior of the hysteretic memory of this level detection. If during the time between the falling edge of TCK and the capture of data at the rising edge of TCK in the *Capture-DR* TAP Controller state there is a valid level, this level is captured. If the level is not valid, then a default initial value is captured (see "Capture Window" in Figure 44). This determines the result when an ac-coupled signal decays to an invalid level before the sample interval. The time between the falling and rising edges of TCK in the *Capture-DR* TAP Controller state is a window of input observation and of vulnerability to noise events with magnitude sufficient to disturb the validity of the level appearing on the pin.

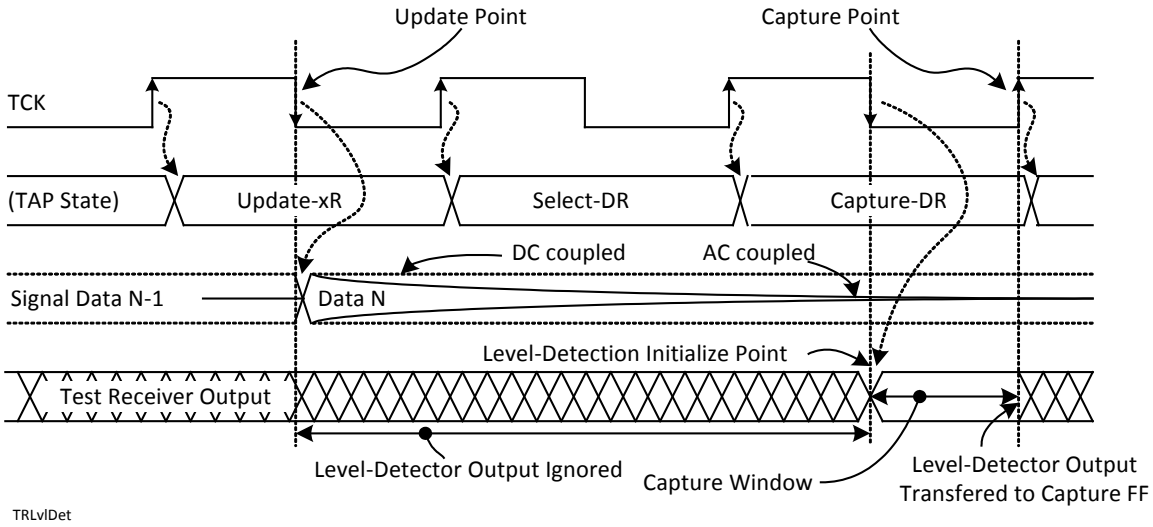


Figure 44— Timing of level-detection initialization and data capture in a test receiver

NOTE 2—The time from falling TCK in *Update-xR* to falling TCK in *Capture-DR* must be greater than T_{Test}. This can be accomplished by adding TCK cycles in the *Run-Test/Idle* state or by manipulating the TCK frequency.

Part 2) of rule a) in 6.2.2.1 gives the test receiver designer additional flexibility for designing a more robust threshold for the 'shorted-capacitor test' (which uses EXTEST) for ac pins that are required to be externally ac-coupled. The nature of ac coupling allows the transmitter's common-mode voltage (T_X V_{CM}) to be significantly offset relative to the test receiver's threshold voltage used for ac tests. Consider a component with a supply voltage of 3.6 volts capacitively coupled to a component with a supply voltage of 1.1 volts: there is a very low probability that these chips would have any overlap between driven voltages and received thresholds on their differential channels. The result, if part 1) of rule a) in 6.2.2.1 is chosen, is that this offset will often cause a shorted-capacitor fault to look like a 'stuck-at' fault in one direction regardless of the applied EXTEST 0/1 stimulus. Additionally the relatively small voltage swing around the threshold (half of ΔV_{Min} on either side) and hysteresis threshold of EXTEST (half of V_{Hyst_Edge}) could cause a good capacitor to be reported as a false failure in small amplitude protocols due to system noise.

By setting the threshold and bias voltages of the test receiver close to one or the other limit of the input voltage range of the test receiver (usually the voltage rails: VDD and ground), the good-machine detection of a float condition is unaffected. In most cases the shorted board-level capacitor case will cause a predictable and significantly greater voltage difference between the receiver threshold and the driven signal than a threshold set in the middle of the legal operating range. Generally, the test receiver will detect any voltage greater than the threshold when the threshold is set low, and less than the threshold when the

threshold is set high. This effect is enhanced when the stimulus data is set to give the greatest difference between the receiver threshold and the driven value. This results in a single sided (stuck-at) detection in the presence of the shorted-capacitor, and detection of the float condition in the absence of a shorted board-level capacitor, but with predictable behavior for detection and diagnostics.

This choice of part 2) of rule a) in 6.2.2.1 is documented in the BSDL file as detailed in 7.5.4. The BSDL modifier asserts that detection will occur in one specified polarity, and might require the use of only one polarity of stimulus. For example, if the test receiver is designed so that the threshold is set to ground, then the receiver boundary cell should be preloaded with a '0,' the driver stimulus preloaded with a '1' for testing the representative input and a '0' for testing the associated input, and the fault condition will result in the receiver boundary cell capturing a '1' as opposed to recapturing the preloaded value of '0.' Again, this method does not conform to IEEE Std 1149.1 EXTEST since it does not detect the digital value driven, but simply attempts to check for continuity between the driver and receiver.

For either choice of level sensitive operation, the threshold voltage of the test receiver is documented as an AIO_PIN_BEHAVIOR keyword-value pair as detailed in 7.5.4. This will permit determination of whether the test receiver will be able to reliably operate with a given board configuration of driver and coupling. When the threshold voltage is programmable, it will assist choosing the threshold voltage best suited for testing a particular board configuration.

Software tools will need to account for this behavior as described in A.3.4.2.1.

6.2.3 Edge-detection behavior of input test receivers

The edge-detection behavior is selected by an ac testing instruction, such as the EXTEST_PULSE or EXTEST_TRAIN instructions provided by this standard.

6.2.3.1 Rules

The input test receivers edge-detection rules include the following:

a) Whenever a test receiver is operating in edge-detection mode on an ac input signal, it shall determine the value of the input signal by one of two methods:

   1) By a method equivalent to comparing the instantaneous voltage of the signal to the recent average of the voltage of the signal determined by a simple low-pass filter, or
   NOTESee Figure 30 for an example of this filter.

   2) If a receiver is specified to be ac-coupled to its driver by comparing the instantaneous voltage of the signal to a fixed threshold voltage.

NOTE 1— Even when method 2 is permitted, method 1 could still be implemented.

NOTE 2— The opportunity to implement method 2) clearly exists in the case where the ac coupling is implemented on-chip. It can also apply to a device not containing ac coupling when the device designer assures that in all applications the input will be ac-coupled at the board or system level. The choice of method 1) or 2) is documented in BSDL per rule g) in 7.5.6.2. If there is no on-chip capacitor, then the requirement of ac coupling on the board would normally also be documented in the specification sheet for the component.

NOTE 3— When adopting method 2), the device designer can choose a threshold and bias value at the midpoint of a voltage swing deemed optimal for the mission receiver (e.g., the receiver common-mode voltage). The threshold and bias values must be the same in order to detect the float condition.

NOTE 4— If method 2) is adopted, then rule b) below must be interpreted as defining hysteresis centered at the fixed threshold voltage.

b) Whenever a test receiver is operating in edge-detection mode on an ac test input signal, the test receiver shall respond only to transitions having a magnitude greater than the hysteresis voltage V_{Hyst_Edge}, where V_{Hyst_Edge} is a voltage no less than 50% of ΔV_{Min} and no greater than 90% of ΔV_{Min}.
NOTE—See rule e) below for further definition of the meaning of the word "respond."

c) Whenever a test receiver is operating in edge-detection mode on an ac input signal, the test receiver shall respond to valid signal transitions defined in rule b) in 6.2.1.1 only when the new signal level persists beyond V_{Hyst_Edge} for at least a minimum period of time called the hysteresis delay, T_{Hyst}.
NOTE—There could also be a propagation delay in the test receiver. This time is not included in the value of the hysteresis delay.

d) Whenever a test receiver is operating in the edge-detection mode on an ac input signal, the test receiver output shall be cleared of prior history at a time between exiting the *Shift-DR* TAP Controller state and before entering the *Update-DR* TAP Controller state.
NOTE—This rule maximizes device design flexibility as the initialization signal could be derived from a signal already being distributed to the boundary-scan register. See recommendation b) in 6.2.3.2 for a choice that minimizes noise vulnerability.

e) In response to a valid signal transition as defined in rules b) and c) above, the test receiver shall output a logic one when the signal transition is rising or a logic zero when the signal transition is falling, and retain its state in response to an invalid transition as defined by rules b) and c) above [see also g) in 6.2.1.1]; and it shall retain that logic value at least until after the rising edge of TCK in the *Capture-DR* TAP Controller state, or until the next valid transition occurs, or it is initialized per rule d) above.

f) When an ac input pin is ac-coupled to a driver, the time constant, T_{HP}, of the effective high-pass filtering of the coupling shall be no less than HP_Mult times the value of the hysteresis delay, T_{Hyst}, where HP_Mult is chosen per rule i) below.

g) When method 1) of rule a) above has been exercised, the time constant, T_{LP}, of the (usually low-pass) edge-detection filter shall be no less than LP_Mult times the value of the hysteresis delay, T_{Hyst}, where LP_Mult is chosen via rule i) below.

NOTE 1— When method 2) rather than method 1) of rule a) above is exercised, this rule is unnecessary since recent signal history is not used as a reference. In such case, the coupling filter performs the edge detection function and there is no additional edge-detection filter in the implementation.

NOTE 2— Propagation delay is assumed to be small compared to the value of T_{LP}.

h) When rules f) and g) above have been exercised, the value of T_{HP} (coupling time constant) shall be adjusted to be no less than HPLP_Ratio times the value of T_{LP} (edge-detection time constant), where HPLP_Ratio is selected (or interpolated) from Table 4, to select an appropriate noise immunity.
NOTE—See discussion in 6.2.3.3 concerning the selection of this ratio. When rule g) above is not exercised, then there is no edge-detection filter implemented and no need for adjusting T_{HP}.

i) The choices of HP_Mult and (if needed) LP_Mult are found by selecting (or interpolating) values from Table 3 for the value of the ($V_{Hyst\_Edge} / \Delta V_{Min}$) percentage equal to or higher than the value selected via rule b) above.
NOTE—No value of LP_Mult is needed when method 2) rather than method 1) of rule a) above is exercised. See the discussion of Table 3 for choosing the value of HP_Mult when LP_Mult is not implemented.

j) When method 1) of rule a) above has been exercised, and when an ac testing instruction is active in any IC participating in those tests, then the time between subsequent test data changes, T_{Test}, shall be no less than three times the edge-detection filter time constant, T_{LP} of any IC.

k) When method 2) of rule a) above has been exercised, and when an ac testing instruction is active in any IC participating in those tests, then the time between subsequent test data changes, T_{Test}, shall be no less than three times the actual coupling filter time constant of any such connection between devices being tested.

NOTE—The T_{HP} parameter specified for each ac pin is a minimum for an off-chip filter and actual for an on-chip filter. T_{TEST} must be calculated using actual coupling time constants.

l) When method 1) of rule a) above has been exercised, the edge-detection time constant, T_{LP}, shall be documented in the BSDL, and when method 2) of rule a) above has been exercised, the coupling filter time constant, T_{HP}, shall be documented in the BSDL.

6.2.3.2 Recommendations

The input test receivers edge-detection recommendations include the following:

a) For better, small-amplitude noise immunity, the value of V_{Hyst_Edge} selected per rule b) in 6.2.3.1 should be set closer to 70% of ΔV_{Min}.

NOTE—Proper accounting for process variations that could affect the value of V_{Hyst_Edge} should be observed to stay within the limits given in rule b) in 6.2.3.1.

b) In order to minimize the time during which noise on an input pin might be falsely detected as a valid input, the prior history should be cleared at the rising edge of TCK in both *Exit1-DR* and *Exit2-DR* TAP Controller states.

6.2.3.3 Description

Rule a) in 6.2.3.1 specifies two methods for detecting edges, the first by comparing the instantaneous value of the signal against its recent history. See Figure 30 for an example of one way to do this comparison. The second method allows the comparison to be done versus a fixed reference when the signal is guaranteed to be ac-coupled. This can be the case where the coupling is integrated into the device. For devices that might be dc-coupled, since the transmitted voltage levels cannot be guaranteed, there is no way to determine a valid reference and therefore a fixed reference is not feasible. A designer is not required to use this second method, but could find it useful to reuse a portion of the circuitry needed for level-detection [see rule a) in 6.2.2.1].

Rule b) in 6.2.3.1 defines the range for the hysteresis voltage, V_{Hyst_Edge}, between 50% and 90% of ΔV_{Min} as the target for a designer to set a trip point value for test receiver response and allows for significant margin. See recommendation a) in 6.2.3.2. See also rule i) in 6.2.3.1, which uses this value to determine bounds on time constants.

When method 1) of rule a) in 6.2.3.1 is implemented, transitions typically are detected with a self-referenced comparison (see 3.1) as shown in Figure 45 rather than with a fixed-threshold voltage. The shaded regions in the figure show the range of effective switching points for allowed values of V_{Hyst_Edge} in either the ac- or dc-coupled cases. It is important to note that in these two cases, the starting points of a transition as measured by V_{Hyst_Edge} [see rule b) in 6.2.3.1] are quite different.

Rule c) in 6.2.3.1 defines the hysteresis delay, T_{Hyst}, used to provide noise rejection. If a signal change does not persist long enough (for example, it is a short noise pulse) the hysteresis delay will suppress the test receiver's response to it. This gives the test receiver two methods for ignoring noise. The hysteresis voltage will ignore small-amplitude noise and the hysteresis delay will ignore large-amplitude pulses of

insufficient duration. See Annex A for a discussion of how hysteresis delay could be controlled within an implementation.

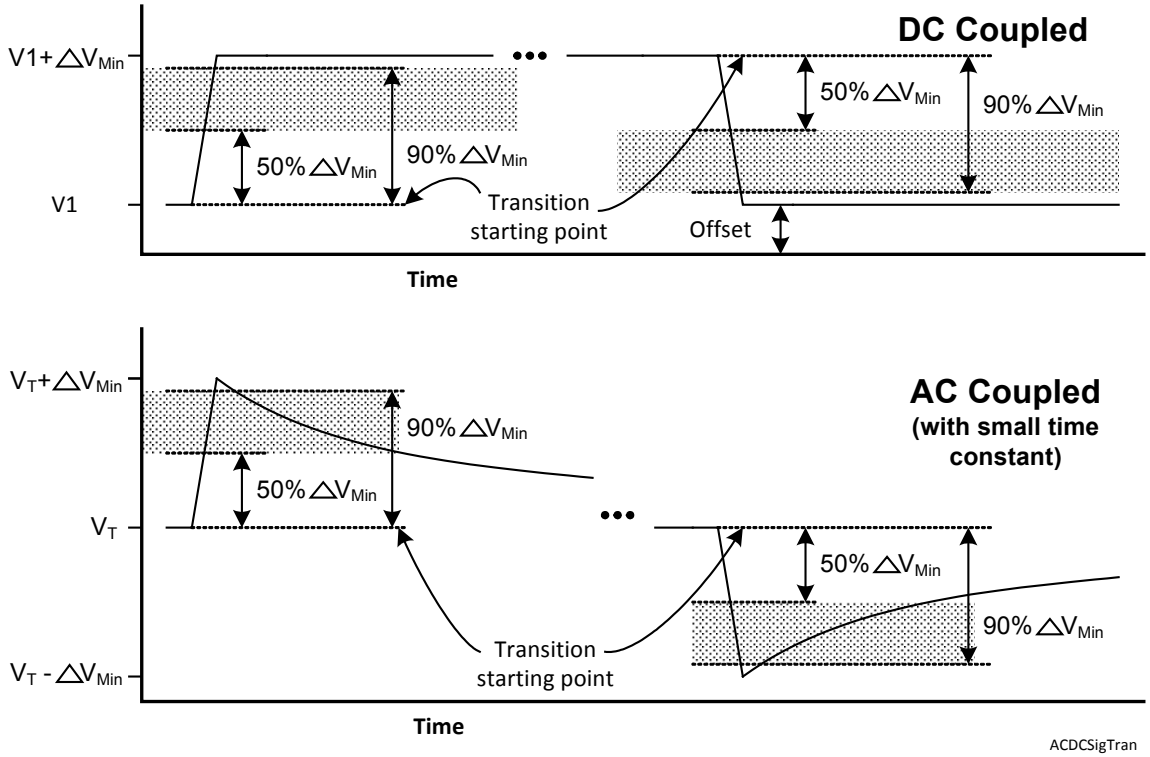


Figure 45—Allowable settings for V_{Hyst_Edge}, referenced to the starting point of a transition

Rule g) in 6.2.3.1 governs the edge-detection time constant. The edge-detection filter time constant is set large enough to assure that the recent history of the average input voltage is not unduly influenced by the rise time of the signal itself and small enough to settle before a new test-induced transition arrives.

Rules d) and e) in 6.2.3.1 and g) in 6.2.2.1 specify the behavior of the test receiver when transitions occur or do not occur, as shown in Figure 46, which also observes recommendation b) in 6.2.3.2. When no transitions have been received, for example, due to an open circuit or a stuck-at fault, the hysteretic memory of the test receiver will retain a default initial value.

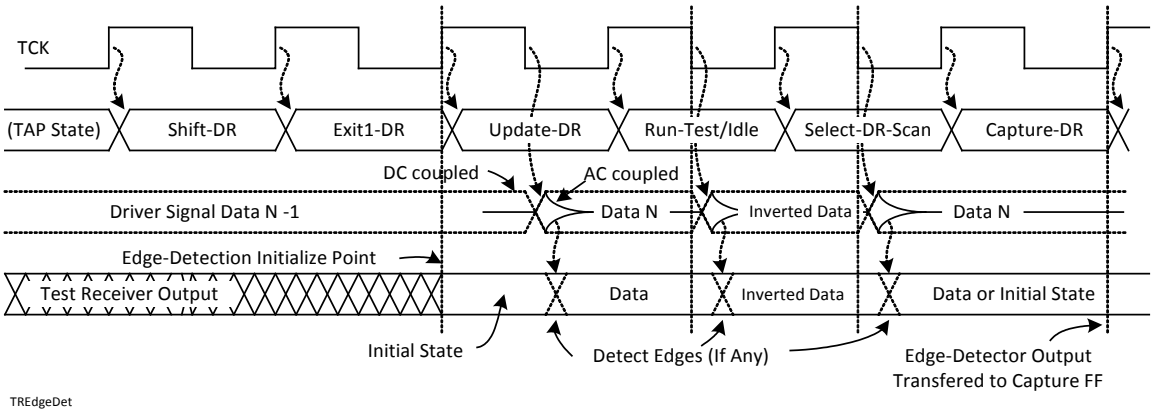


Figure 46—Timing of edge-detection initialization and data capture in a test receiver

NOTE—The time between each pair of transitions in the combined *Run-test/Idle* and *Select-DR-Scan* states must be greater than T_{Test}. When the EXTEST_PULSE instruction is active, this can be accomplished by adding TCK cycles in the *Run-Test/Idle* state, and when either the EXTEST_PULSE or EXTEST_TRAIN instruction is active, by changing the TCK frequency or the time between individual TCK edges.

The test receiver model of Figure 34, rule d) in 6.2.3.1 requires the loading of the hysteretic memory of the test receiver no later than the rising edge of TCK in the *Exit1-DR* and *Exit2-DR* TAP Controller states. This loads the hysteretic memory with the values supplied by the preceding boundary-scan register data shifting during EXTEST-type instructions. Note that there is no load of the hysteresis prior to the first scan immediately after an EXTEST-type instruction becomes active. Therefore, the data captured from test receivers prior to the first scan might not be useful.

Hysteresis loading occurs before reaching the *Update-DR* TAP Controller state. This has two consequences:

a) First, this permits a test receiver in ac test mode to capture transitions if the driving chip has the EXTEST instruction active, or the TAP does not pass through the *Run-Test/Idle* TAP Controller state, or the driving pins are dc pins or behaving as dc pins. In those cases, the possible transition at *Update-DR* is the only transition that can occur during the test period, and without a transition, the test receiver should return the default value previously loaded. In addition, this transition occurs at the same time as the large simultaneous switching noise transient, which can swamp some differential signals. Driving a receiving chip in edge detection mode from a driving chip that is (effectively) executing EXTEST is deprecated (see Table 2 and accompanying text), but will have to be tolerated. For example, a chip compliant only with IEEE Std 1149.1 might need to be ac-coupled to a chip compliant with this standard (if dc-coupled, EXTEST should be used to test the channel). Most interconnect test data sequences will not have a transition at the *Update-DR* TAP Controller state more than half the time on any given output pin, making this configuration difficult to test, at best. However, the following will improve the probability of the test receiver detecting the transition at the *Update-DR* TAP Controller state (see also A.3.4.5).

    1) Perform the interconnection tests with two scans per test pattern. In the first scan, the data for all dc drivers ac-coupled to ac receivers is inverted and the remainder of the data is normal. In the second scan, all data is normal. The data captured after the first scan is ignored, and the data captured after the second scan is used. This creates a transition for the nets that require them, and also minimizes the noise generated by switching drivers.

    2) If the test receiver uses the fixed reference mode given in part 2 of rule a) in 6.2.3.1, make the coupling capacitors on the board large enough so that the time before the transition decays below the hysteretic threshold is longer (by at least 3 times) than the duration of the worst case noise transients anticipated when the output drivers are switched during the *Update-DR* TAP Controller state.

    3) Aggressively reduce noise on the board, particularly from simultaneous switching at the *Update-DR* TAP Controller state.

b) The second consequence of loading the hysteretic memory prior to the *Update-DR* TAP Controller state is that the hysteretic memory could be subject to being changed by the noise transients anticipated in that state, primarily simultaneous switching noise injected into the power distribution. Disturbance of the hysteretic memory contents will make the diagnosis of certain defects (those that cause a float input condition, in particular) much more difficult. If noise continues to prevent reliable testing, then the tests might need to be performed with two scans per pattern, as described above (see also A.3.4.5).

Rule f) in 6.2.3.1 specifies the minimum value for the high-pass filter time constant when ac coupling is used. An upstream driver is expected to produce edges that are equal to or faster than the minimum input swing, ΔV_{Min}, within a maximum time, T_{Trans}, given an acceptable level of ac attenuation in the channel. These edges will pass through this high-pass coupling virtually unchanged. This rule assures the subsequent

decay will be much slower than a valid signal edge so it will not be confused with a valid signal transition, as depicted in Figure 42.

When the ac coupling high-pass is implemented on a board rather than inside an IC, the IC designer must communicate to the board designer this minimum time constant required for ac coupling. In a 50-Ω load termination environment, the minimum value of capacitance specified by this rule might be 1000 pF. If the same time constant were implemented on-chip, then the resistive component would likely be determined by a high-impedance biasing network and the capacitance could be much smaller, for example 10 kΩ and 5 pF, which are both amenable to on-chip integration.

In rule h) in 6.2.3.1, the value of HPLP_Ratio is a function of the maximum input swing, $\Delta V_{Max,}$ from rule c) in 6.2.1.1 divided by the V_{Hyst_Edge} e from rule b) of 6.2.3.1 (see subsequent discussion of Table 4 in 6.2.3.3, for more information). Values of the $HPLP\_Ratio$ closer to 1 yield poorer noise immunity. Ratios much larger than 1 have a higher value for noise immunity but force larger values of resistor/capacitor combinations in the coupling network.

Rules j) and k) in 6.2.3.1 govern the behavior of test signals so that transitions of test signals do not occur too close together in time (T_{Test}). These rules assure that the recent history of a signal is not influenced by the recent history of the test. Rule j) assures an edge-detection filter will charge/discharge sufficiently to be ready to detect the next signal edge. Similarly, rule k) assures sufficient charging/discharging of the coupling high-pass filter. Without sufficient coupling filter time, the average dc operating point of the signal will move versus the fixed logic threshold used for comparison, which risks missing the detection of a valid signal edge. If the coupling high-pass filter is implemented on-board with a larger time constant, then T_{Test} might have to be made larger. If the coupling high-pass filter is implemented on-chip, then it is likely to have a smaller time constant (to avoid a large on-chip capacitor) and thus T_{Test} could be relatively small compared to typical TCK periods.

Rule l) in 6.2.3.1 defines how the choice of method 1 or method 2 of rule a) in 6.2.3.1is documented in the BSDL. If the edge-detection time constant is defined, then method 1 has been implemented. Otherwise, method 2 has been implemented. Note that if both the edge-detection and coupling high-pass time constants are defined, then method 1 has been implemented.

NOTE—When the EXTEST_PULSE instruction is active, the charge/discharge time is governed by the time spent in the *Run-Test/Idle* TAP Controller state, but when the EXTEST_TRAIN instruction is active, the only method for controlling T_{Test} is by direct manipulation of the TCK period.

Given a model of the basic test receiver circuit that uses both high-pass coupling and low-pass edge-detection filtering to generate a self-referenced edge detector (one half shown in Figure 47) the effects of the rules given in 6.2.3 are plotted in Figure 48 for a single rising edge transition. Note that the hysteresis voltage has been omitted in Figure 47 and subsequent analysis.

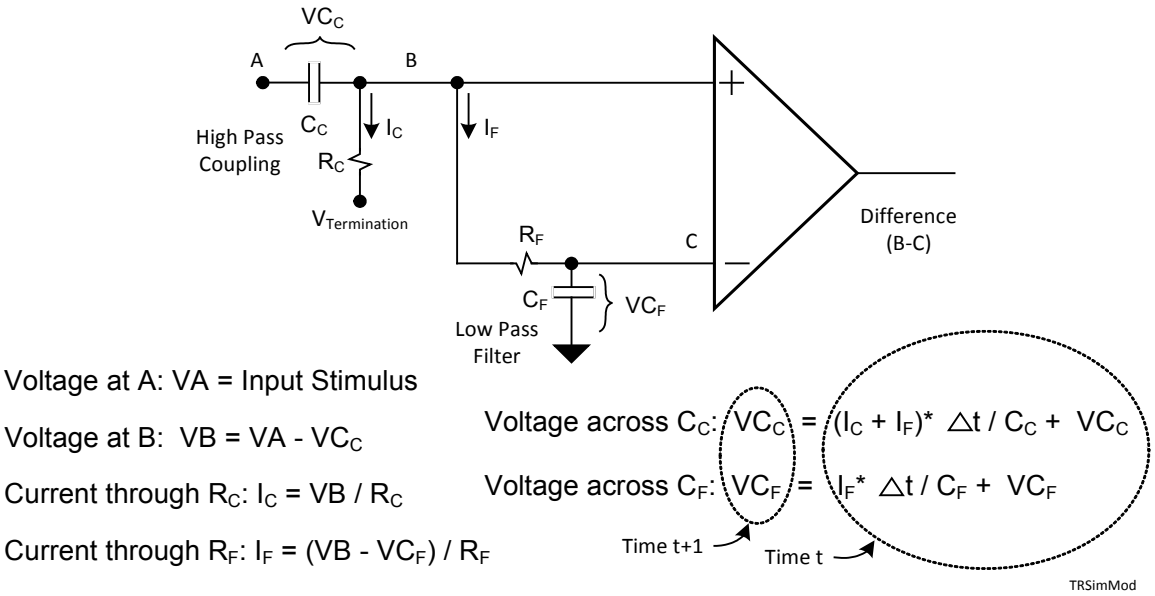


Figure 47—Model with recurrence equations for a portion of test receiver circuitry

Figure 48 shows the output of the non-hysteretic comparator modeled in Figure 47 to illustrate the effects of the two selected time constants and offsets for a single rising edge, on that portion of circuitry devoted to detecting a minimum transition voltage rising edge. Rule b) in 6.2.3.1 gives the two horizontal lines for 50% and 90% minimum input voltage swings. This example uses hysteresis voltage, V_{Hyst_Edge}, at 70% of the minimum input swing, ΔV_{Min}. Given a signal with a valid rising edge with rise time, T_{Trans}, the output of the coupling high-pass filter will decay as shown. There must be, by rule c) in 6.2.3.1, a hysteresis delay, T_{Hyst}, where the comparator does not respond. The coupling time constant, T_{HP} P, of decay due to the high-pass filter, by rule f) in 6.2.3.1 must be no less than 18 times (in this example) the value of T_{Hyst}. However, the decay of the difference signal is faster as shown, due to the low-pass filtering set to no less than nine times the value of T_{Hyst} by rule g) in 6.2.3.1. The choice of multipliers (9 and 18, from Table 3) assure that the minimum time that the voltage difference exceeds 70% of the minimum input swing is significantly greater than the hysteresis delay. If either or both multipliers are increased, this minimum time will increase. If the hysteresis voltage is decreased from 70%, the minimum time will increase. Finally, if the circuit is dc-coupled (equivalent to an infinite coupling time constant) the minimum time will increase.

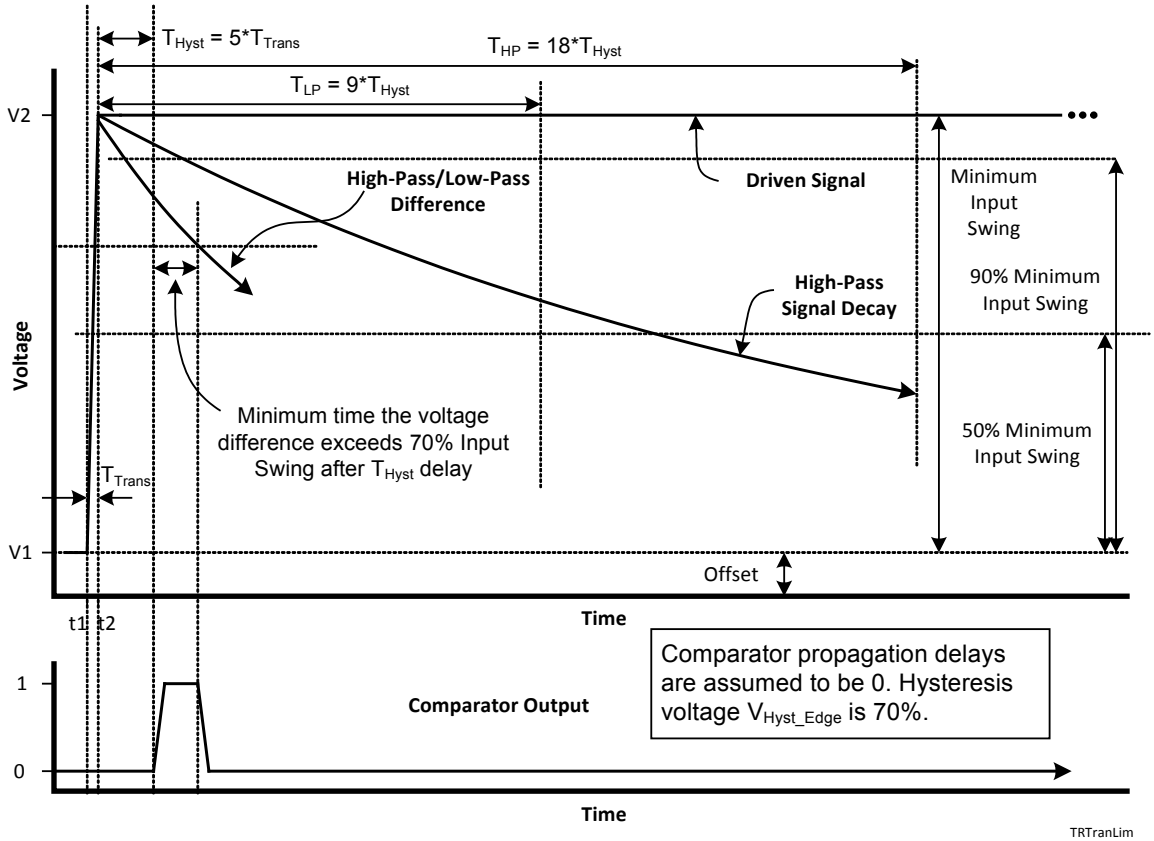


Figure 48—Key delay times and time constants for the detection of a rising edge

Rule i) in 6.2.3.1 selects minimum values for HP_Mult and (if needed) LP_Mult from Table 3 below, which determine coupling high-pass (HP) and edge-detection (LP) time constants. The HP_Mult column is divided into two subcolumns, one for the case where both time constants are required ("HP-LP") and one where the edge-detection time constant is not implemented ("no LP"). The "no LP" values are 33% of the "HP-LP" values since the decay of the system when only the high-pass coupling filter is present is slower, thus allowing a smaller time constant for performance equivalent to the case where both filters are present (this might be important when the high-pass filter is to be integrated on-chip). If both time constants are implemented, the "HP-LP" values must be used. This is consistent with rule h) in 6.2.3.1, which could have the effect of raising HP_Mult above the minimum value built into Table 3. If only the high-pass coupling time constant is implemented, then only HP_Mult ("no LP") is selected from this table and the LP_Mult values are ignored. The LP_Mult values are set to allow a test receiver to be either dc or ac-coupled.

Rule h) in 6.2.3.1 selects a minimum ratio of high-pass coupling time constant to edge-detection time constant (when both are implemented) via the content of Table 4 below. If the two series time constants are close to the same value, they will interact and the difference signal at the comparator can reverse polarity, that is, undershoot (this is seen in Figure 49, which depicts a complete transition and decay cycle, normalized to ΔV_{Min}). When a larger signal such as ΔV_{Max} is applied, this multiplies the opposite polarity excursion, or undershoot, by by a factor of $\Delta V_{MAX} / \Delta V_{MIN}$, approaching the opposite polarity hysterisis trigger point. In order to prevent this undershoot from eliminating noise immunity, or worse, re-triggering the test receiver in the opposite direction, the high-pass coupling time constant is adjusted to prevent this. As the ratio of these two constants gets larger, the undershoot is reduced. In the extreme case of an infinite ratio (dc coupling) there is no undershoot.

Table 3—Minimum values of multipliers HP_Mult and LP_Mult for selected values of V_{Hyst_Edge} as a percentage of ΔV_{Min}

V_{Hyst_Edge} percent of ΔV_{Min}	HP_Mult		LP_Mult if needed	Comments
	HP-LP	No LP		
50%	10	3.4	5	Lowest small signal noise immunity for positive noise events. Highest small signal noise immunity for negative noise events. Gives smallest values for on-chip filter components.
60%	12	4	6	
70%	18	6	9	Best small signal noise immunity for general noise events.
80%	30	10	15	
90%	75	25	37	Highest small signal noise immunity for positive noise events. Lowest small signal noise immunity for negative noise events. Gives highest values for on-chip filter components.
NOTE—The values shown in Table 3 were derived from simulations using the model shown in Figure 47.				

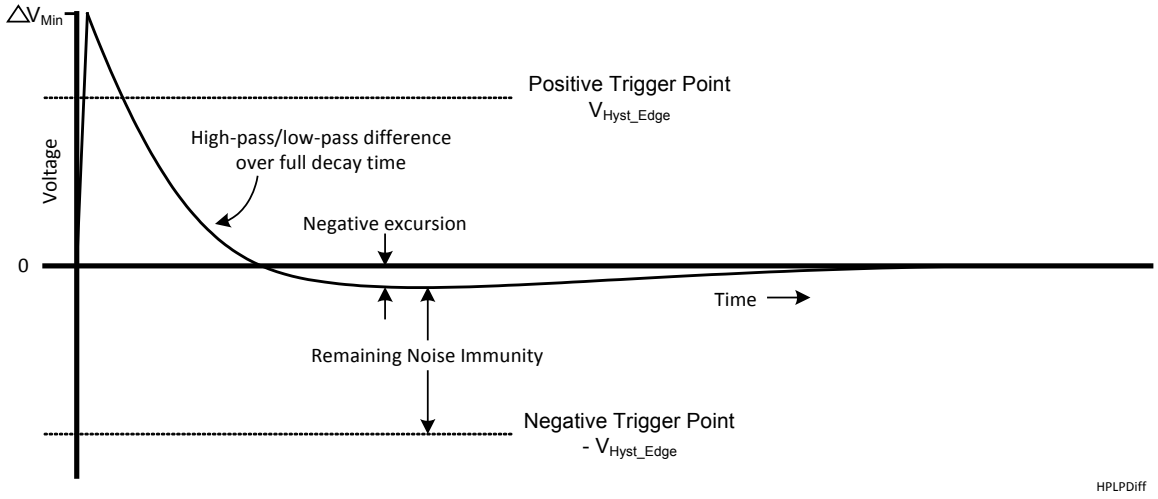


Figure 49—High-pass and low-pass filtered signal over the full decay time (note undershoot)

The values for HP_Mult shown in Table 3 apply when both the high-pass coupling and low-pass edge-detection filters exist and have an HPLP_Ratio of two (built into the table). In this case, the model in Figure 47 shows the maximum undershoot is about 12.5% of the transition input, leaving 87.5% of the noise immunity. The designer should compare the undershoot expected at ΔV_{Max} (in this example, $0.125 \times \Delta V_{Max}$) to the hysteresis voltage of the test receiver (50% to 90% of ΔV_{Min}). As the undershoot must be subtracted from the hysteresis voltage to determine noise margin, the undershoot at ΔV_{Max} should not be allowed to exceed more than about 1/2 of the hysteresis voltage. The board designer could choose to force a smaller under/overshoot, depending on his knowledge of the probable noise sources during interconnect testing.

NOTE—The values shown in Table 4 were derived from simulations using the model shown in Figure 47.

Table 4— Minimum HPLP_Ratio values to reduce under/overshoot and loss of noise margin seen in Figure 49

HPLP_Ratio	Over/ under shoot	Example loss of noise margin when the ratio ($\Delta V_{Max} / V_{Hyst\_Edge}$) is:			
		1.1, example: $\Delta V_{Max}=1.0\Delta V_{Min}$, $V_{Hyst\_Edge}=.9\Delta V_{Min}$	2, example: $\Delta V_{Max}=1.6\Delta V_{Min}$, $V_{Hyst\_Edge}=.8\Delta V_{Min}$	3, example: $\Delta V_{Max}=2.1V_{Min}$, $V_{Hyst\_Edge}=.7\Delta V_{Min}$	5, example: $\Delta V_{Max}=2.5\Delta V_{Min}$, $V_{Hyst\_Edge}=.5\Delta V_{Min}$
2	12.5%	14%	25%	38%	63%
4	9.9%	11%	20%	30%	50%
6	8.2%	9%	16%	25%	41%
8	6.9%	8%	14%	21%	35%
12	5.8%	7%	12%	17%	29%
16	4.3%	5%	9%	13%	22%
32	2.5%	3%	5%	8%	13%

The magnitude of the undershoot can be reduced by specifying a larger value for HPLP_Ratio than the value of two built into Table 3. Table 4 shows the maximum undershoot percentage for a range of HPLP_Ratio values. The under/overshoot percentage should be multiplied by the actual ($\Delta V_{Max} / V_{Hyst\_Edge}$) ratio to determine noise margin loss for the circuit being designed. For convenience, the noise margin loss for several example ratios is shown in this table (the shaded entries violate a 50% margin loss criterion). To comply with rule h) in 6.2.3.1, the designer should pick the minimum value of HPLP_Ratio that provides an appropriate noise margin and use that to calculate the minimum T_{HP} and minimum acceptable coupling capacitor. This value of minimum T_{HP} is documented in BSDL (see 7.5.4) for checking by board test and verification software.

If either the high-pass coupling or the edge-detection filter are not used (one or the other must be), then there cannot be an HPLP_Ratio and no undershoot is possible. When the channel is guaranteed to be ac-coupled, and the edge-detection filter is not implemented, the calculations for T_{HP} can be used to determine the size of the coupling capacitor and termination or bias resistor, depending on the configuration details.

6.2.4 Integration of level-sensitive and edge sensitive behaviors

Given the complexity of the rules and options in 6.2.2 and 6.2.3, it is worth reviewing the basic filter configurations which are allowed by this standard. In the figures and discussion below, only a single-ended net or one leg of a differential channel is shown, and the single hysteretic comparator is used for simplicity in the figures.

Figure 50 shows a single test receiver that illustrates one method of integrating both the level-sensitive (DC) and edge-sensitive (AC) behaviors. Options 1 and 2 in the figure show the edge-detecting low-pass filter in the negative path of the hysteretic comparator, per option 1) of rule a) of 6.2.3.1, with the component externally either dc (Option 1) or ac (Option 2) coupled to the driver. This was the focus during development of the standard, as reflected in the logical development of ideas in Clause 4 and Clause 6.

Switching from edge-sensitive behavior to level-sensitive behavior is accomplished by removing the low-pass filter in the path to the negative input of the hysteretic comparator and replacing it with a fixed voltage threshold (in all Options, the ac/dc switch is shown in the dc or level-sensitive position). Such a switch is mandatory and therefore does not need to be documented.

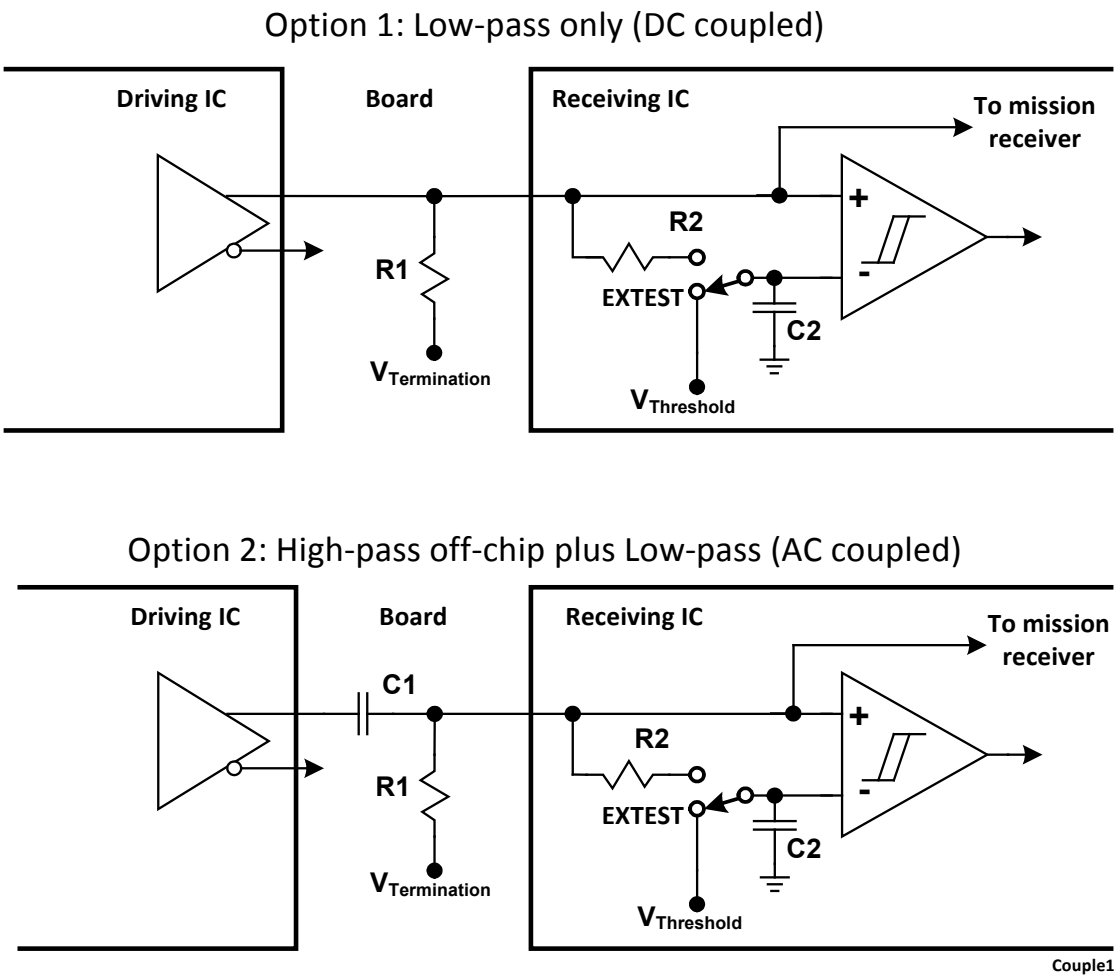


Figure 50—Integration of level-sensitive and edge-sensitive behavior with LP filter

It was also recognized that if the channel was guaranteed to be ac-coupled, then the LP filter could be omitted per option 2) of rule a) of clause 6.2.3.1, and a threshold voltage selected to match the termination voltage present at the positive hysteretic comparator input, as illustrated by Option 3 in Figure 51. The external coupling high-pass filter performs the edge-detection function just as well as the missing LP filter would.

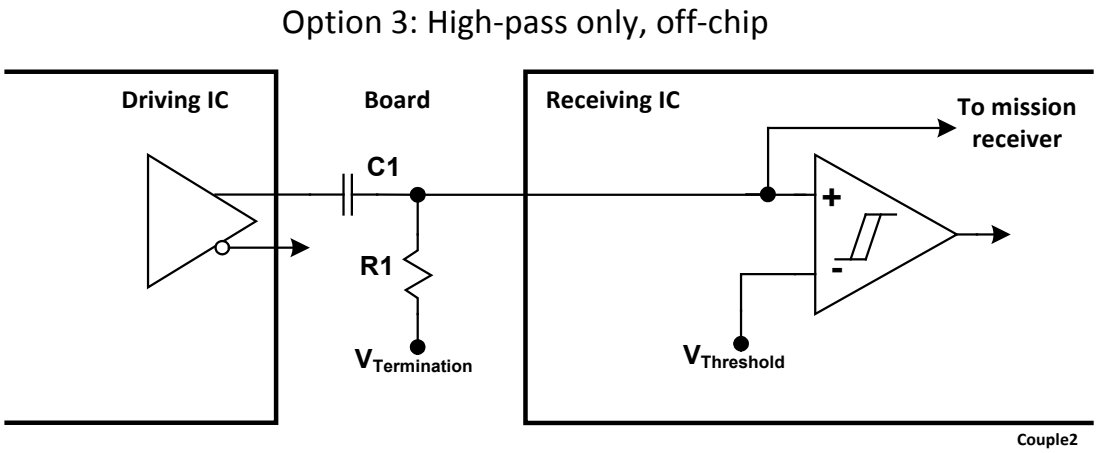


Figure 51—Mandatory ac coupling instead of edge-detecting filter

Further, it was recognized that the ac coupling could be moved on-chip, as shown in Option 4a of Figure 52. This did raise an issue for level-sensitive tests, as this configuration could not meet the requirements of 6.2.2. However, since the only purpose of level-sensitive mode was to test for a shorted external coupling capacitor, and the coupling capacitor was now on-chip, this violation was considered moot and ignored. Since bypass of the capacitor is not required in this situation, if one is present and bypassable, the bypass is documented in the BSDL.

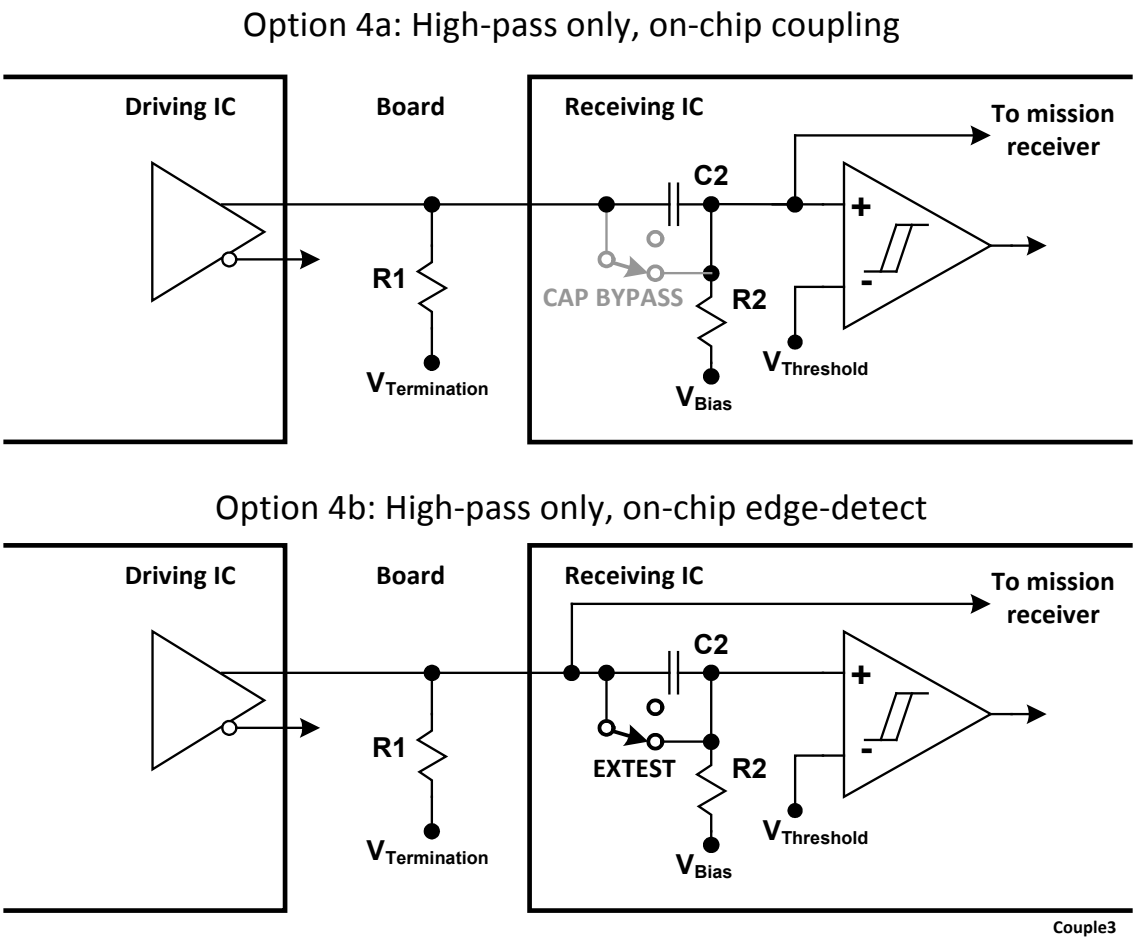


Figure 52—On-chip high-pass filter configurations

Note at this point that, from a filter theory point of view, there is total dynamic behavior equivalence between the low-pass configuration of Option 1 in Figure 50 and the on-chip high-pass only configuration of Options 4a and 4b in Figure 52, assuming the time constants of the two configurations are equal (the R2C2 value is the same in both options). Dynamic behavior equivalence means the difference voltage at the test receiver hysteretic comparator inputs will behave identically in both the time and frequency domains for both configurations. If the on-chip filter is to perform the coupling function, then the mission receiver must receive the same input as the positive input of the test receiver, as shown in Option 4a.

However, it is also legal, under the "equivalence" allowed by option 1) of rule a) of 6.2.3.1, to only use an on-chip high-pass filter as the edge detection filter, as shown in Option 4b of Figure 52. In this case, the mission receiver is connected to the input pad and is not directly affected by the on-chip high-pass filter going to the hysteretic comparator positive input. This identifies the on-chip high-pass filter as an edge-detection filter only, and such a high-pass filter would have to comply with all of the rules governing the low-pass filter shown in Option 1 of Figure 50. As the mission receiver in Option 4b of Figure 52 is now dc-coupled to the pad, both dc and ac coupling are allowed on the board as shown by Option 4b of Figure 52 for dc coupling and Option 5 of Figure 53 for ac coupling. In this sense, Option 1 and Option 4b are equivalent dc-coupled configurations, and Option 2 and Option 5 are equivalent ac-coupled configurations. Options 3 and 4a are also equivalent mandatory ac coupling configurations.

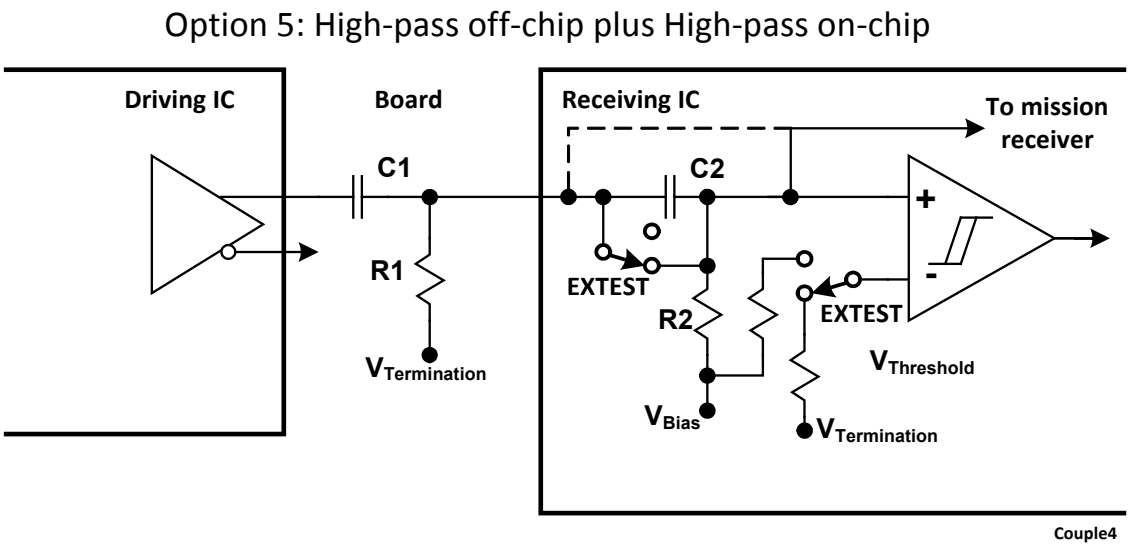


Figure 53—On-board and on-chip high-pass filter configurations

In some situations, connecting the mission receiver to the pad in Option 5 of Figure 53 could provide a design advantage by separating the specification of the coupling time constant to the mission receiver from the specification of the edge-detection time constant to the test receiver. The edge-detection time constant would still have to be the faster time constant.

One additional note on the switching between level-sensitive and edge-sensitive behaviors: in Option 4a of Figure 52, level-sensitive mode is not needed due to having the coupling filter being on-chip. A bypass of the on-chip capacitor is shown, but in gray to indicate it is optional. However, in Option 4b of Figure 52 and Option 5 of Figure 53, compliance with both the level-sensitive and edge-sensitive rules would require that the high-pass edge-detection filter be removed during level-sensitive testing, just as the edge-detection low-pass filter had to be removed in Options 1 and 2. The simplest way to do that would be the bypass switch shown that shorts the capacitor. Again, as the switch is required, it is not documented.

A quick discussion of the $V_{\text{Termination}}$ value in all of these options: the differential protocols used for simulations during development of this standard tended to rely on a 100 Ω termination resistor between the two differential channel legs. This is electrically equivalent to a 50 Ω resistor from each leg to a virtual common-mode voltage node, as mentioned before. Even in Spice simulations of 50 Ω termination resistors explicitly terminated to a voltage, the voltage was chosen to be in the middle of the receiver operating range.

However, there is a potential problem with all options when working with a protocol that sets $V_{\text{Termination}}$ to V_{dd} or to Ground. This might not be a problem if there is a dc path from the driver through the termination, as the driver might supply enough current to maintain a reasonable operating voltage across the termination resistor. However, if there is a coupling capacitor blocking the dc path from the driver, there is a low impedance (typically 50 Ω) dc path from $V_{\text{Termination}}$ to the positive input of the hysteretic comparator, at least during level-sensitive operation. If the comparator power supply is Vdd and Ground, then the dc level of at least one input voltage to the comparator will be centered at the edge of the legal input range for the comparator, and the test receiver might not operate properly. For this type of termination, there might need to be a dc block between the termination resistor and the hysteretic comparator inputs during edge-detection operation. This potential problem was not recognized when this standard was originally published.

Since then, this problem has arisen, and Option 5 of Figure 53 was developed as a solution. The on-chip high-pass filter in series with the positive leg of the hysteretic comparator replaces the low-pass filter in the negative leg of Option 2 of Figure 50, and since there are now two filters, would be considered the edge-detection filter regardless of whether it affects the mission receiver. As noted above, assuming the on-board time constants (C1R1) are the same and the on-chip time constants (C2R2) are the same, the behavior of the difference voltage at the comparator does not change when using a high-pass in series with the positive input instead of a low-pass in the negative leg. However, C2 now blocks $V_{\text{Termination}}$ in ac test mode, allowing the designers to select a good operating point for the comparator with V_{Bias} and $V_{\text{Threshold}}$. In level-sensitive mode, the on-chip high-pass filter must be removed, as shown, to comply with the rules in 6.2.2. Doing so un-blocks the dc path from $V_{\text{Termination}}$ to the hysteretic comparator and option 2) of rule a) in 6.2.2.1 must be used to accommodate this situation.

$V_{Threshold}$ in Options 4b and 5 might need to be switched between the values of V_{Bias} (in ac mode) and $V_{\text{Termination}}$ (in dc mode), as shown in Option 5 of Figure 53. 0, loosely based on the LVDS protocol, illustrates in a general way how a test receiver might be configured to perform the switching of $V_{Threshold}$ to take advantage of both rule a), option 2), of 6.2.2.1 and rule a), option 2), of 6.2.3.1, and use a fixed threshold for both ac and dc (EXTEST) behavior. The details of the illustrated switching will vary significantly depending on the details of the communication protocol and the filter configuration.

In addition to the problem illustrated above with $V_{\text{Termination}}$ equal to Vdd or Ground, this approach to testing for a shorted capacitor might be necessary any time the driver dc common-mode voltage is completely unknown. Again, this configuration is only applicable when the pin is guaranteed to be ac-coupled. For the LVDS protocol, which typically uses a 100 Ω termination between the representative and associated pins, and external to 0, the four resistors shown are high-impedance, relative to other impedances driving the input net. For other protocols, resistors R1 and R2 might be a low-impedance Thevenin equivalent pair providing both termination and the pin bias voltage in ac mode. The generated V_{Bias} and $V_{Threshold}$ must have the same value, but be separately generated since V_{Bias} will be overdriven by any signal applied to the pin. In other words, the ratio of R1 to R2 must equal the ratio of R3 to R4. When AC_MODE is set to ‘1,’ R1 and R3 are switched into the circuit and the voltage dividers set both V_{Bias} and $V_{Threshold}$ to the desired operating point for the test receiver hysteretic comparator. When AC_MODE is set to ‘0,’ R1 and R3 are switched out of the circuit and R2 and R4 pull both V_{Bias} and $V_{Threshold}$ to ground. If the coupling capacitor is shorted, the driver will pull the input pad up to some voltage greater than the V_{Hyst_Level} of the test receiver, and the short will be detected. This can only be detected as a ‘1’ captured in the boundary cell, which would normally be preloaded with a ‘0’ for this test.

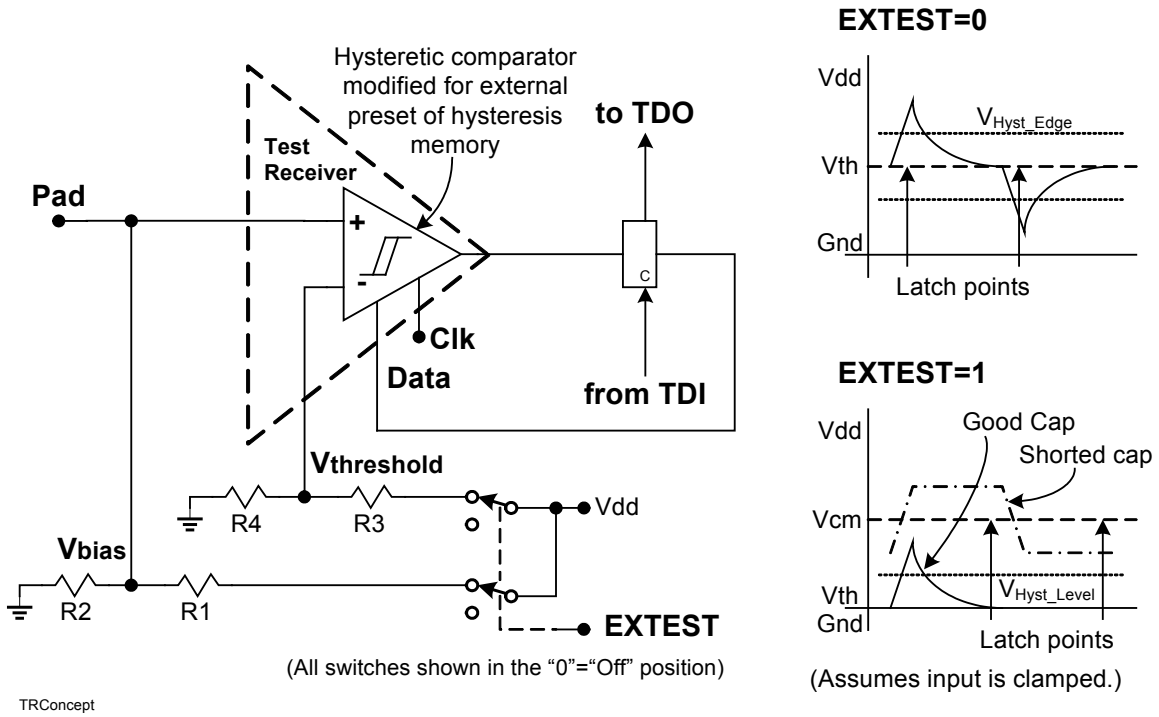


Figure 54—Conceptual test receiver conforming to option 2) of rule a) of both 6.2.2.1 and 6.2.3.1

The timing diagrams in Figure 54 show the nominal behavior in ac and dc test modes. In ac mode, the hysteretic memory is preset before the transition edge, and so the input value is latched T_{Hyst} time after the input exceeds the V_{Hyst_Edge} value above or below the threshold voltage and the latched result is captured during Capture-DR. Because the net is capacitively coupled, the bias voltage set by R1 and R2 establishes the common-mode value of the input signal at the same voltage as the threshold voltage. In dc mode, the bias and threshold voltages are set to ground, and the hysteretic memory is not preset until at least T_{Test} time after the edge, so the input value is latched at that time. If the coupling capacitor is good, then only rising edges will be seen on the input (assuming the input is clamped to ground) and the signal will have decayed away before capture. If the coupling capacitor is shorted, then the driver common-mode voltage (VCM) will force the input voltage well away from the threshold voltage and be above the V_{Hyst_Level} value to capture a '1,' especially when the driver is driving a '1.' To see how different VCM values would affect this circuit, the VCM value and shorted cap signal could be raised and lowered. If lowered enough, then a driven '0' would no longer be above the V_{Hyst_Level}, and would not detect the defect. Again, this circuit is loosely based on the LVDS protocol and is conceptual to illustrate the intended behavior for option 2) of rule a) in 6.2.2.1 and 6.2.3.1.

Note that in many applications, all four of the resistors shown in Figure 54 and Figure 55 would also be switched out of the circuit anytime the boundary-scan register does not control the I/O in order to reduce power. Such controls are not shown in these conceptual circuits.

Figure 55 shows a more flexible test receiver configuration that will dynamically set the threshold and bias to one limit or the other of the test receiver input range, depending on the value pre-loaded into the boundary cell. This allows for testing in board configurations not anticipated by the component designer. Again, the ratio of R1 to R2 must equal the ratio of R3 to R4 so that V_{Bias} equals $V_{Threshold}$ in all cases.

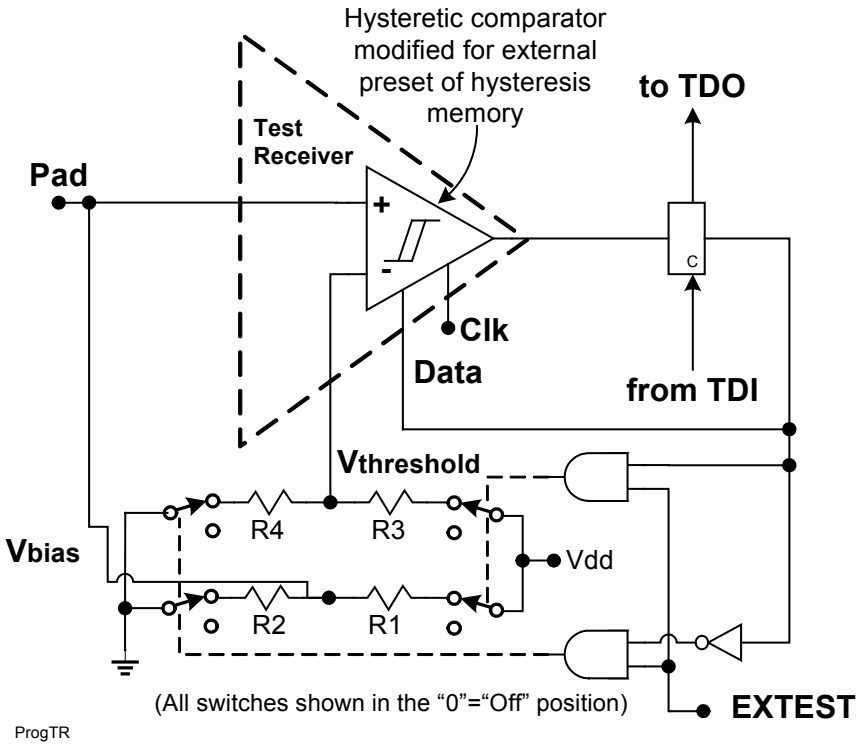


Figure 55—Programmable test receiver conforming to option 2) of both 6.2.2.1 and 6.2.3.1

Figure 54 and Figure 55 show possible implementations for individual ports. It is also possible to simplify the switch and resistor networks for port-grouped differential input pairs by sharing those circuits for both the representative and associated ports of a port-grouped pair. Figure 56 shows the pads, test receivers, boundary cells, and switching network for a port-grouped differential input pair implementing that sharing.

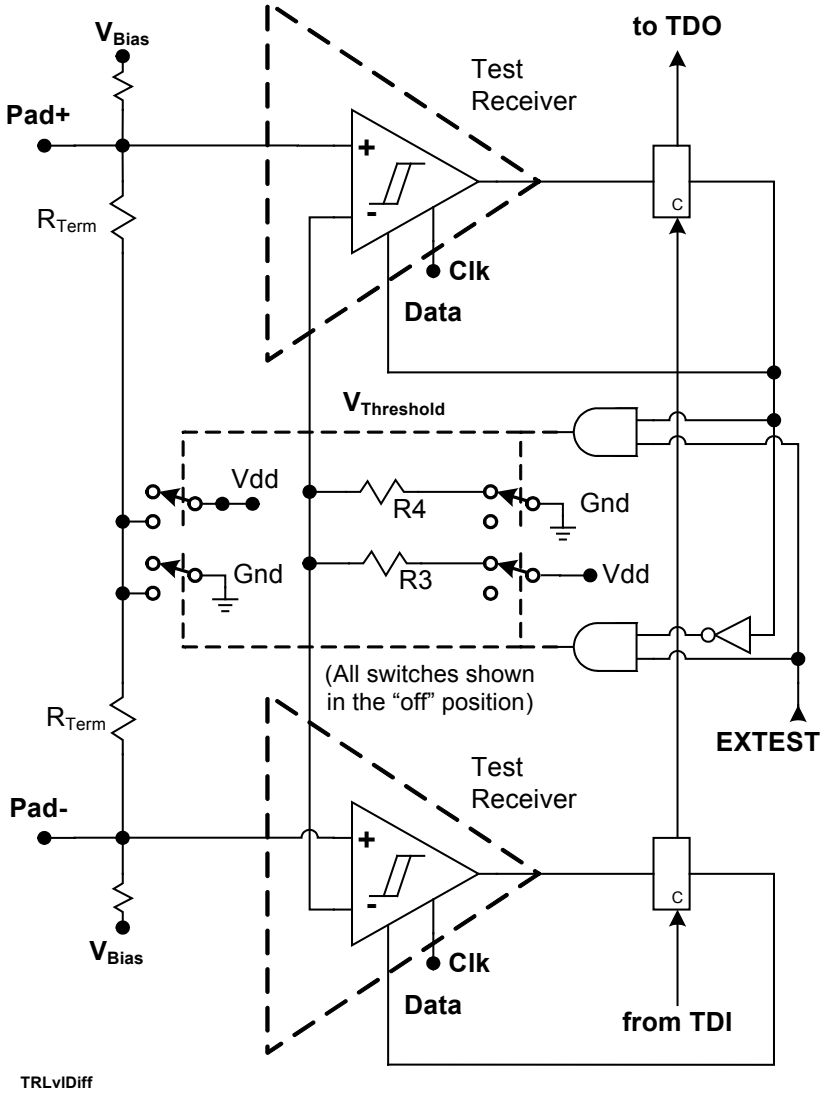


Figure 56—Programmable test receivers on a differential pin pair with shared circuitry

When operating in mission or ac test modes, all switches are held in the "off" position, as shown in Figure 56. The pads are terminated by the two R_{Term} resistors in series, forming an unreferenced (usually 100 Ω) termination between the pads. Bias voltage, V_{Bias}, provides the operating point for the mission and test receivers from a common source through relatively high-impedance resistors. The test receiver threshold voltage, $V_{Threshold}$, is set for both test receivers to the same voltage as the bias voltage by a single voltage divider (R3 and R4.)

When operating in dc test (EXTEST) mode, then one or the other AND gate shown in Figure 56 will turn "on" its respective switch. If the boundary cell associated with the representative port in the Figure is scanned to '0," then the bottom AND gate will turn "on," disconnecting R3 from Vdd and forcing $V_{Threshold}$ to ground through R4. At the same time, the center tap of the termination will be connected to ground, forcing V_{Bias} to ground as well. A shorted capacitor will be detected by capturing a high value. If the representative port boundary cell is scanned to '1,' then the threshold and bias voltages will similarly be forced to Vdd, and a shorted capacitor will be detected by capturing a low value.

The two test receivers share the threshold and bias voltages. This sharing saves circuitry but is only practical if the differential ports are port-grouped, and if the representative and associated port test receiver

boundary scan cells are both scanned to the same preload value. Port grouping is required in this case since the legs are terminated with an unreferenced termination in mission and ac test modes.

There are potential problems with implementations of this type, such as effects from the actual "on" resistance of the switches, which could cause termination impedance mismatches and reflections, and transient effects caused by the switching of the BC_4 boundary cell when scanning during EXTEST. It might be necessary to replace the observe-only BC_4 with an observe-only BC_1 or BC_2 and use the output of the Update latch to control the switches. This then also raises questions of any transients generated at the input pads in Update-DR, and whether they will dissipate prior to Capture-DR (the switching on the center tap of the termination will appear to the circuit as a driver). The behavior of a circuit like this would need to be thoroughly verified for both good and fail machine situations.

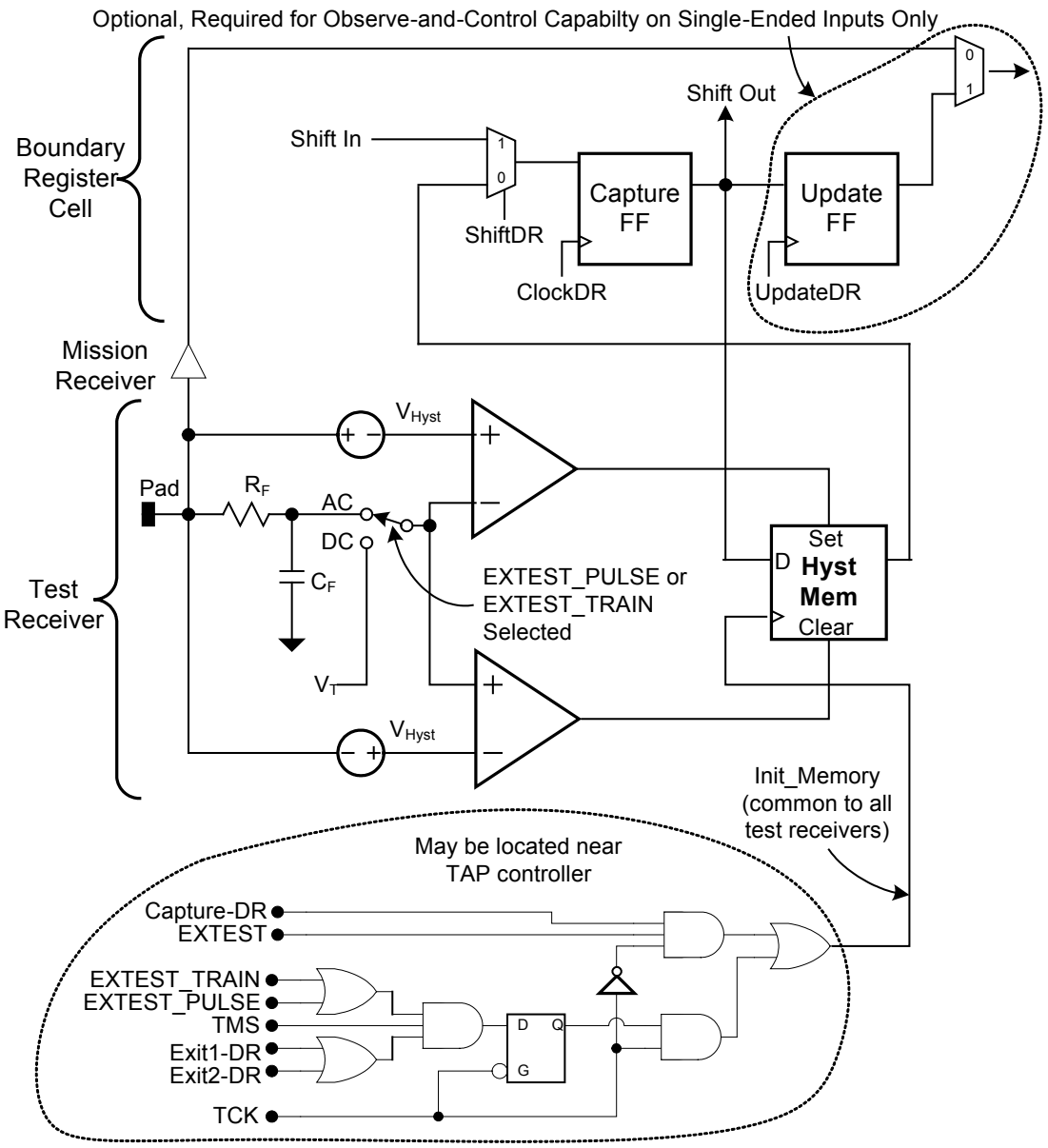
See 7.5.6 for information on using a BSDL <detection modifier> keyword to document the implementation of option 2) of rule a) of 6.2.2.1.

6.2.5 Integration of a test receiver with a boundary-scan register cell

Rules d) in 6.2.2.1 and in 6.2.3.1 govern the timing of when the hysteretic memory of a test receiver is initialized for SAMPLE (if supported), EXTEST, EXTEST_PULSE, and EXTEST_TRAIN. No initialization of the hysteretic memory is required for any other instruction defined in this standard or IEEE Std 1149.1, though it is permitted for user defined instructions. This initialized state remains until a valid test level or edge is detected. If no such test event is detected during testing, the initial state of the capture flip-flop will not be changed [per rule g) in 6.2.1.1].

Before any interconnect testing is conducted, it is necessary to preload all boundary-scan register cells with safe data via the PRELOAD instruction. Test data loaded into the boundary-scan register cells during the execution of the SAMPLE (if supported), EXTEST, EXTEST_PULSE, or EXTEST_TRAIN instructions will include the test stimuli and control data for the drivers and the default data for the test receivers. Data captured from test receivers into the boundary-scan register cells prior to the first scan in these test instructions might not be valid since the hysteresis is not cleared before that capture.

It is possible to integrate a full control-and-observe boundary-scan register cell into the design of a test receiver, which will satisfy these rules. One example using the test receiver model described earlier is shown in Figure 57. The capture flip-flop of this cell both captures the test receiver output (the hysteretic memory flip-flop) and provides initial state data for the hysteretic memory of the test receiver. The hysteretic memory is cleared, in accordance with rule f) in 6.2.1.1, by preloading the memory with the content of the boundary-scan register cell capture flip-flop. This default value will then be re-captured if there are no valid inputs to change it, preserving the capture cell contents in accordance with rule g) in 6.2.1.1.



NOTE: The generated clock (Init_Memory) shown is suitable for rising edge-sensitive behavior only.

TRwBreg

Figure 57—A possible full-featured integration of a test receiver model with its corresponding boundary-scan register cell

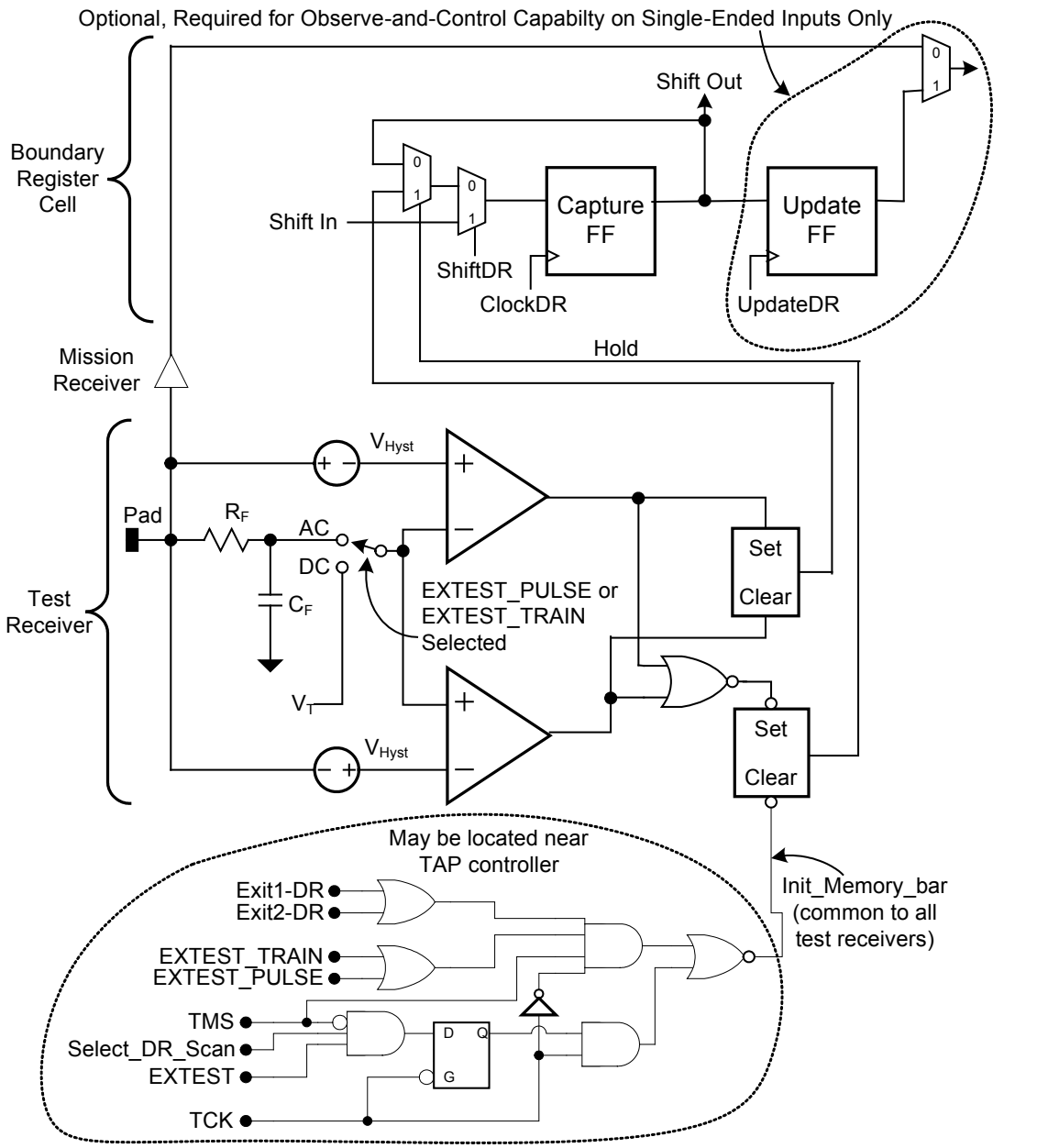
NOTE 1—The update stage shown in Figure 57 illustrates the ability to add optional control (INTEST/RUNBIST) capability to a single-ended input or to the representative leg of a differential input. In most cases, if the mission receiver produces a single-ended signal and can tolerate the signal multiplexor, as shown here, a full control-and-observe cell on the mission receiver combined with an observe-only cell on the test receiver output would provide more information to the test software and be preferred.

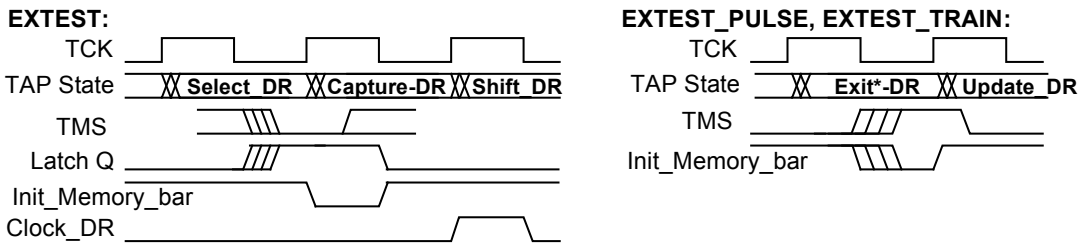
An alternative model of the test receiver and its integration with the control-and-observe boundary-scan register cell is shown in Figure 58. In this model, the hysteretic memory has two simple, asynchronous, set/reset flip-flop memory elements (cross-coupled NOR gates for example). One flip-flop retains the value of the last valid input; the other is set anytime there is a valid input. This second flip-flop is reset to clear the hysteretic memory in accordance with rule f) in 6.2.1.1, and in its off state it causes the boundary-scan register capture latch to recapture its own state, in accordance with rule g) in 6.2.1.1. As stated earlier, the test receiver models shown in this standard are for discussion, and the anticipated implementation of the test receiver is a hysteretic comparator. However, rules f) and g) in 6.2.1.1 require modification of the normal hysteretic comparator design. As always, it is not the intent of these rules to restrict the implementation of the test receiver hysteresis to any single design.

In both figures, the clocking for the Update flip-flop comes from the “UpdateDR” signal as defined by IEEE Std 1149.1 (see Figure 6.5 in IEEE Std 1149.1-2013). If the boundary-scan register cell is observe-only (e.g., for one pin of a differential pair, or for a single-ended input in a device that does not support the INTEST or RUNBIST instructions), then the Update flip-flop and multiplexer shown could be omitted. If part 2 of rule a) in 6.2.3.1 is implemented, then the analog selection switch at the input of the test receiver (along with its control by “EXTEST_PULSE or EXTEST_TRAIN”) can be omitted since the test receiver is always referenced to a level in that case, even when in edge-detection mode.

This standard only mandates the initialization of the hysteresis for EXTEST, EXTEST_PULSE, or EXTEST_TRAIN instructions. The clocking logic shown in Figure 57 and Figure 58 is typical: the “AND” of the appropriate phase of TCK, the TAP Controller state, and the instruction decode.

NOTE 2—Analysis of this combinational circuit shows there cannot be a critical race with respect to TCK.





NOTE: The generated clock (Init_Memory_bar) shown is suitable for both rising edge-sensitive (flop) or negative-active level-sensitive (latch) behaviors.

TRwBregAlt

Figure 58—An alternative test receiver model integrated with its corresponding boundary-scan register cell

6.3 Output drivers

All ac pins (see 4.1) that drive data from an IC are equipped with test circuitry that allows boundary-scan test instructions to take control of the mission driver and substitute a time-varying signal in place of mission data. The goal is to use the mission driver with its *native drive levels* and *edge rate* to produce test signals (as defined in 6.2). This is conceptualized in Figure 59 for a differential mission driver, but the concept is identical for single-ended drivers.

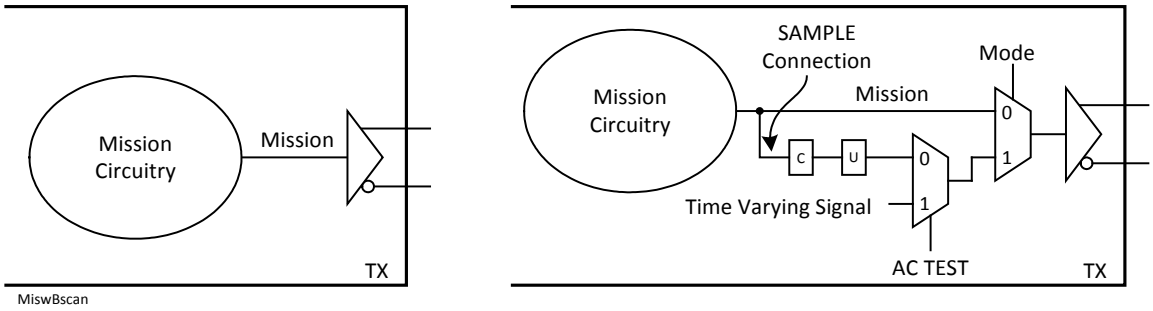


Figure 59—A mission driver (differential) and added test circuitry

The actual implementation could be different, since adding a multiplexer (as shown) in the mission signal path might introduce an unacceptable delay, but the effect is the same as shown. The test circuitry used to implement the functionality is not required to perform at the frequencies of the mission, except with respect to voltage levels and edge rates seen at the pin(s). The time between test edges will likely be much larger than for mission edges.

The mission circuitry could be digital or analog. If digital, it could operate at extremely high frequencies so as to have "analog" properties. In many cases, the signal states captured via the "SAMPLE connection" to the boundary-scan capture flip-flop might be indeterminate. This is especially true in cases where the TCK rate is much lower than the performance of the mission circuitry. Thus, this standard allows the relaxation of the requirement from IEEE Std 1149.1 to implement the SAMPLE instruction for ac output pins—the SAMPLE instruction must be implemented, but the SAMPLE connection shown could be omitted and replaced with a default value. If the mission of the device *is* a candidate for use with the SAMPLE instruction, then SAMPLE functionality should be implemented.

NOTE—The SAMPLE instruction must be implemented per IEEE Std 1149.1 on dc pins, and all inputs as consistent with permission 6.2.2.2, b) and recommendations 6.2.2.3, c).

6.3.1 ac pin driver behavior for IEEE Std 1149.1 test instructions

DC test mode instructions from IEEE Std 1149.1 include EXTEST and CLAMP, and within options contained within IEEE Std 1149.1, the RUNBIST and INTEST instructions could also provide data to the driver when they do not disable all drivers. The HIGHZ instruction always disables drivers. The rules below assume there is an appropriate load or termination available to allow a driver to operate as specified for its mission mode when enabled.

Non-test mode instructions (e.g., SAMPLE, PRELOAD, BYPASS, IDCODE, USERCODE) do not interfere with the mission function of the driver.

This standard provides two new instructions: EXTEST_PULSE and EXTEST_TRAIN. These instructions are ac test mode instructions that use the same boundary-scan register data and control cells already specified for the IEEE Std 1149.1 instructions, that is, the mapping of data and control cells does not change. This standard also provides for ac/dc selection boundary-scan register cells (see 6.5).

6.3.1.1 Rules

IEEE Std 1149.1 test instruction ac pin driver rules include the following:

a) The single-ended or differential driver for an ac output pin or channel shall be provided with data by a single associated boundary-scan register cell.

b) When a dc test mode instruction is active, the boundary-scan register data cells shall provide to the output driver, for an ac output pin, static values that cause an enabled driver to output the value stored in the Update flip-flop, per the rules within IEEE Std 1149.1.
NOTE—Depending on if permission a) in 6.3.1.2 is implemented, an output driver might always be enabled.

c) When an ac test mode instruction is active, the boundary-scan register data cells shall provide, to the output driver for an ac output pin, values that cause an enabled driver to output values as specified in rules e) and f) in 5.3.1 for EXTEST_PULSE, and rules e) and f) in 5.4.1 for EXTEST_TRAIN.

d) When any test mode instruction is active, an enabled output driver of a differential ac output channel shall initially produce on one leg pin a static value that matches the value provided by the boundary-scan register data cell and the opposite value on its other leg pin.

e) When any test mode instruction is active, an enabled output driver of an ac output pin (channel) shall produce on its output(s) a transition matching the levels and edge slew rate of the mission performance specified for the driver when the output of the associated boundary-scan register cell changes.

f) When any test mode instruction is active, a disabled output driver of an ac output pin (channel) shall produce on its output(s) a quiescent, nondriven state.
NOTE 1— There could be biasing supplied in the driver implementation that establishes a state as perceived by a downstream receiver, but this state is at most weakly driven. It is expected that if another enabled driver shared this channel, it would be unaffected by this weak state.

NOTE 2— A driver might be disabled [see permission a) in 6.3.1.2] by a control cell, or by an instruction such as RUNBIST, INTEST, or HIGHZ.

g) When the boundary-scan register cell associated with an ac output pin is part of an excludable boundary scan segment, and the segment is not included, then the ac output pin shall behave as specified in IEEE Std 1149.1-2013, 9.4, Design and control of test data register (TDR) segments.

6.3.1.2 Permissions

IEEE Std 1149.1 test instruction ac pin driver permissions include the following:

a) The single-ended or differential driver for an ac output pin (or channel) may be disabled or enabled via the content of a control cell in the boundary-scan register, per the rules within IEEE Std 1149.1.

b) With respect to the SAMPLE instruction, the device designer may choose to have the Capture flip-flop load a deterministic value rather than attempt to sample the state of the mission circuitry on the rising edge of TCK in the *Capture-DR* TAP Controller state.

c) Observe-only boundary-scan register cells may be connected to ac output pin(s) to observe the driven state of the pin(s).

NOTE—Additional observe-only cells are allowed by IEEE Std 1149.1 on any pin (see IEEE Std 1149.1-2013, 11.8, Redundant cells). This permission simply restates that they are explicitly allowed on all ac output pins, including negative as well as positive legs of differential ac channels. Per IEEE Std 1149.1, there is no data inversion in the path from either pad to its associated observe-only cell.

d) The state of a single-ended ac output or the positive leg of an ac differential output may be captured by the data cell for that output while the EXTEST, EXTEST_PULSE, or EXTEST_TRAIN instructions are active.

NOTE—This allows a driver to "self monitor" its ability to create a logic state on its pin by using the capture stage of its boundary-scan register data cell. This concept also comes from IEEE Std 1149.1-2013, 11.6, Provisions and operation of cells at system logic outputs.

6.3.1.3 Recommendations

IEEE Std 1149.1 test instruction ac pin driver recommendations include the following:

a) When the mission circuitry is digital, or is capable of slower digital performance, the designer should implement the SAMPLE instruction behavior per IEEE Std 1149.1.

b) Where applicable, permissions c) and d) in 6.3.1.2 should be implemented.

c) Whenever the driver is provided with a programmable common-mode voltage or programmable ΔV (peak-to-peak voltage), the programming should be accessible via the TAP.

NOTE—The fact of programmability and the TDR register fields used are to be documented in BSDL and the procedures for programming are to be documented in PDL per the requirements of Clause 7.

6.3.1.4 Description

For each single-ended or differential output ac pin or channel, there is one boundary-scan register data cell. That cell must provide to the output driver a signal that causes the driver to output values that conform to the rules in IEEE Std 1149.1, for dc test mode instructions, and conform to the rules in Clause 5 of this standard, for ac test mode instructions.

For ac pins only, permission b) in 6.3.1.2 allows a device designer to opt out of the requirement by IEEE Std 1149.1 that mandates capturing the digital state of the mission circuitry on the rising edge of TCK in the *Capture-DR* TAP Controller state while executing the SAMPLE instruction, since there are cases where this signal is analog in nature, or if digital, is changing so quickly as to make sampling at the much slower TCK rate meaningless. If, when SAMPLE is loaded, the mission mode output can be suitably observed (e.g., there is a single-cycle mode), the designer should implement SAMPLE as recommended in a) of 6.3.1.3.

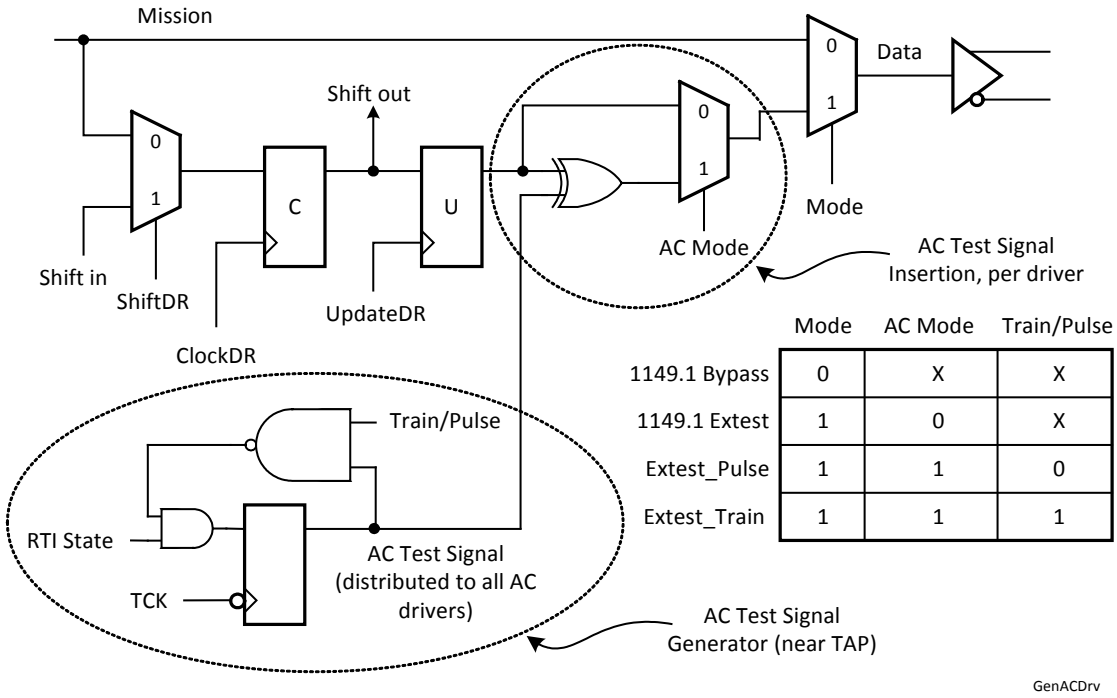
Permissions c) and d) in 6.3.1.2 allow a designer to add output pin monitoring capacity, useful in determining if a driver can successfully achieve a logic state on its pin(s). This is useful for detecting and diagnosing shorted driver pins. Permission c) in 6.3.1.2 does this with additional, redundant cells. Permission d) in 6.3.1.2 does this with the capture stage of the associated data cell.

When either the EXTEST_PULSE or EXTEST_TRAIN instruction is active, an enabled ac pin driver will produce an ac waveform (or its complement on the negative leg of a differential pair) with the same voltage range and edge rate that the driver produces in mission mode, when the TAP passes through the *Update-DR* and *Run-Test/Idle* TAP Controller state. This has important implications for any test receiver connected on downstream ac input pins. The test receiver contains a hysteretic memory that is first initialized to a default state. If the test receiver never sees a valid edge, this default state is retained. This can be used to detect open pins. If a driven edge is received, the hysteretic memory will be set to the logic value of the upstream driver even though this value could decay due to ac coupling. The ac test signal, described in Clause 5, assures that for any sequence of test data loaded into the Update flip-flop over time, including sequences of like data, there will be transitions generated that pass through the ac coupling for interpretation by the test receivers.

When the TAP does not traverse the *Run-Test/Idle* TAP Controller state, the effect of both the EXTEST_PULSE or EXTEST_TRAIN instructions is the same as (DC) EXTEST, because the ac test signal remains de-asserted.

6.3.2 Example ac pin driver structure

A possible implementation for a pin driver is shown in Figure 52 that supports IEEE Std 1149.1 instructions as well as ac test capabilities provided by this standard. This implementation is a simple adaptation of a commonly used boundary-scan register cell design (BC_1), where an extra multiplexer controlled by "ac Mode" is used to select either the Update register content or that same content modulated by an exclusive-OR gate with the ac test signal. See Annex B for a complete summary of boundary-scan register cells provided by this standard and how they are similar or different from the cells provided by IEEE Std 1149.1.



	Mode	AC Mode	Train/Pulse
1149.1 Bypass	0	X	X
1149.1 Extest	1	0	X
Extest_Pulse	1	1	0
Extest_Train	1	1	1

Figure 60 — A general ac pin driver structure

6.4 Bidirectional pins

Bidirectional pins are a merger of input and output pins and meet all rules for both types of pins.

6.4.1 Rules

Bidirectional pin rules include the following:

a) A bidirectional ac pin shall meet all the rules for input pins and output pins given in 6.2 and 6.3.

NOTE—This rule applies to single-ended pins and to both legs of a differential pair of pins.

b) A bidirectional ac pin shall have a test receiver monitoring activity during all test modes.

NOTE—This means the test receiver monitors data appearing on a pin regardless of whether data is supplied externally or by the driver for that pin.

6.4.2 Description

A set of resources for a single-ended ac bidirectional pin is shown in Figure 61, which will support the INTEST and RUNBIST instructions by virtue of the Update flip-flop and output multiplexer in the input path (these can be omitted if INTEST and RUNBIST are not implemented). A control cell determines if the output buffer is enabled to drive data.

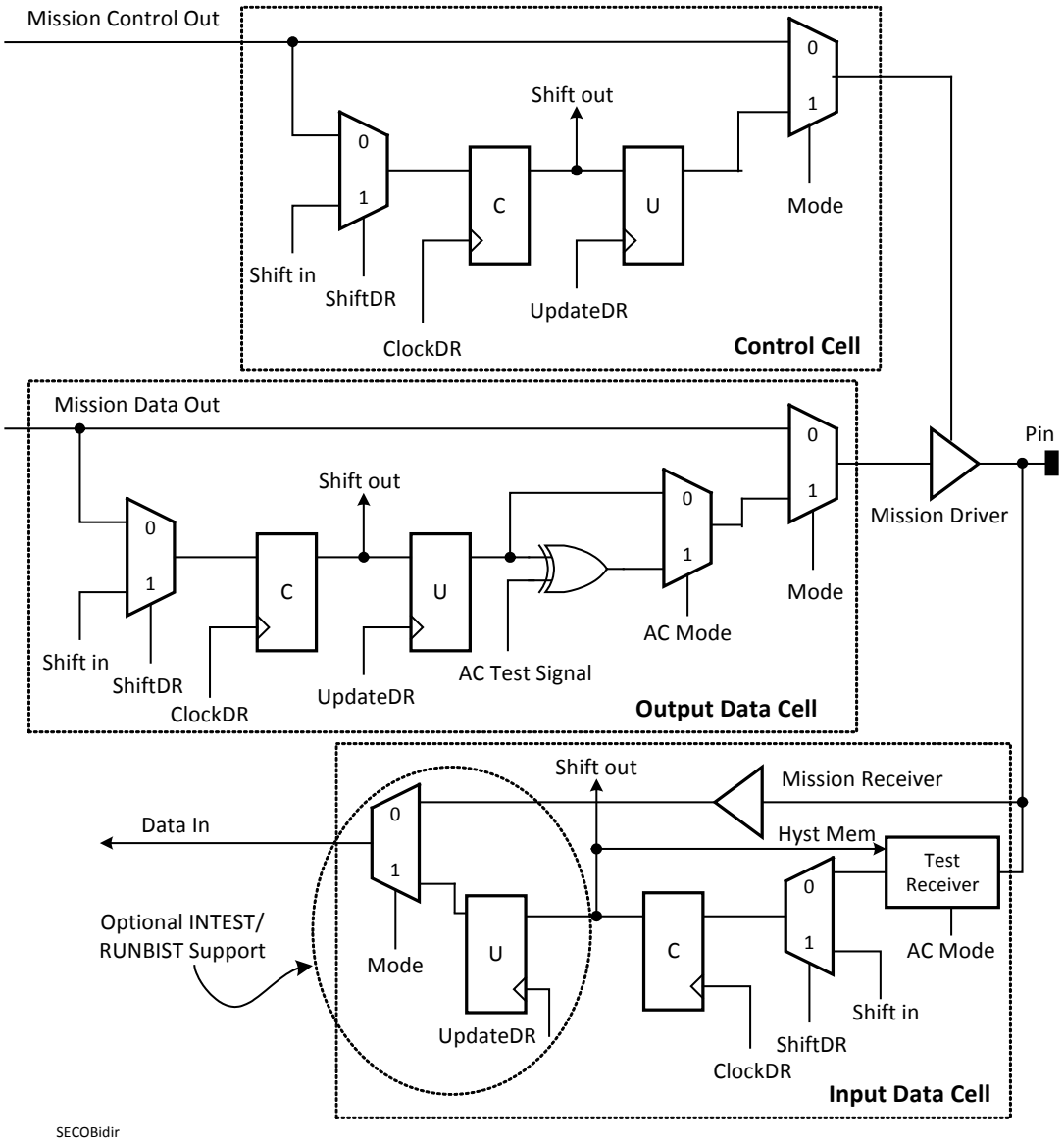


Figure 61—A single-ended bidirectional ac pin with drive enable, drive data, and test receiver monitor

NOTE—The update stage shown in Figure 61 illustrates the ability to add optional control (INTEST/RUNBIST) capability to a single-ended input or to the representative leg of a differential input. In most cases, if the mission receiver produces a single-ended signal and can tolerate the signal multiplexor, as shown here, a full control-and-observe cell on the mission receiver combined with an observe-only cell on the test receiver output would provide more information to the test software and be preferred.

Figure 62 shows a differential bidirectional pin pair with full support for ac testing, including support for INTEST and RUNBIST (if INTEST and RUNBIST are not implemented then the input path could have the Update flip-flop and output multiplexer omitted). A control cell determines if the output buffer is enabled to drive data. Per the rules of IEEE Std 1149.1, the entire boundary-scan register cell (observe-only or control-and-observe) on the mission receiver could be omitted.

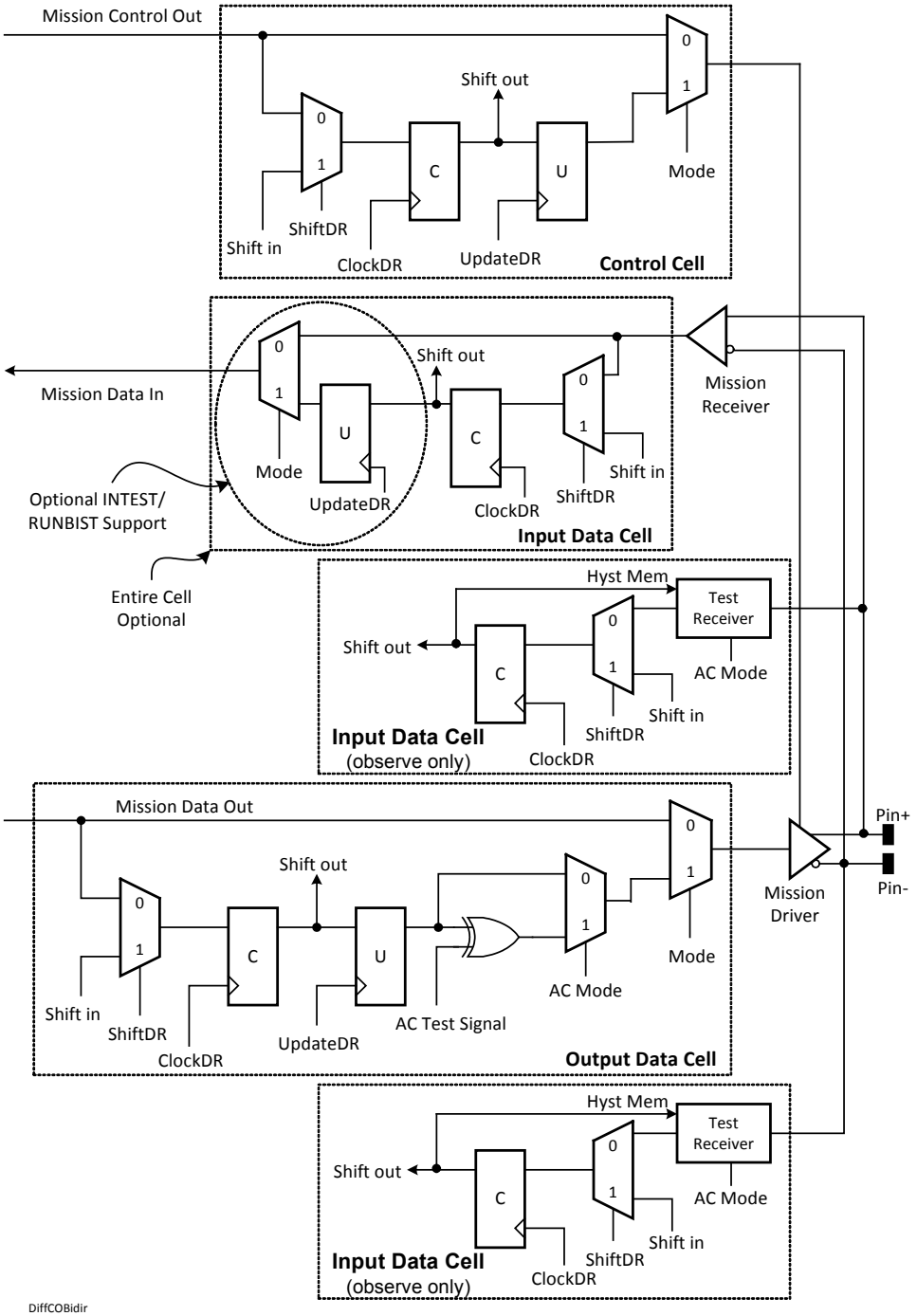


Figure 62—A differential bidirectional ac pin with drive enable, drive data and two test receiver monitors and an optional mission receiver input data cell with INTEST/RUNBIST support

A bidirectional differential ac pin with on-chip termination and on-chip ac coupling and bias generation for the receiver is shown in Figure 63. The termination provides source termination for the driver and load termination for a driver in another device connected to this device. The coupling capacitors along with the bias resistors set the time constant for the high-pass coupling filter.

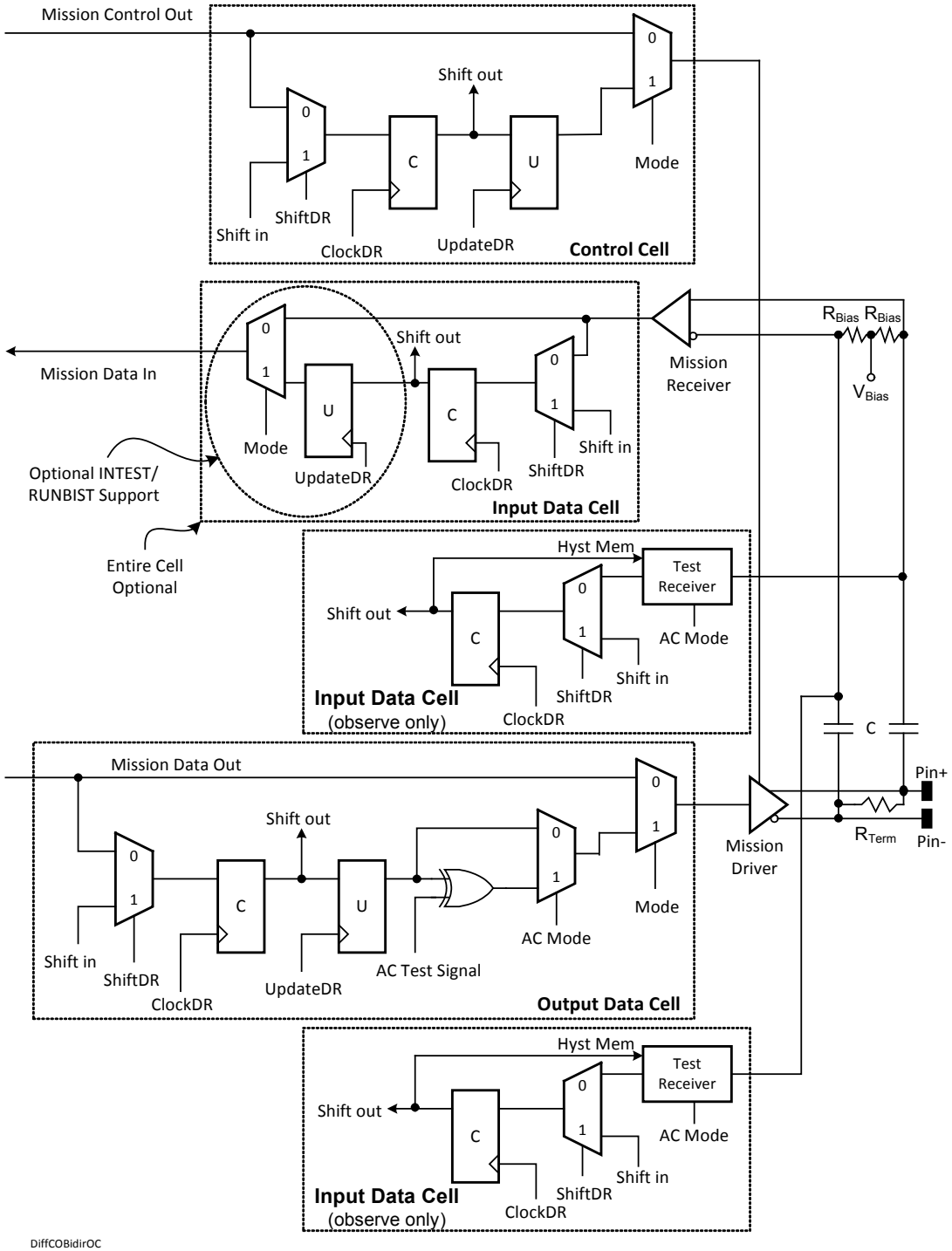


Figure 63—A differential bidirectional ac pin with on-chip termination and ac coupling with bias

6.5 AC/DC selection cells

There can be cases during the execution of tests where the ability to functionally disable the ac signal appearing on an ac pin driver might be desirable when the device is executing either the EXTEST_PULSE or EXTEST_TRAIN instructions. When the ac signal is functionally disabled, the dc behavior of that driver is reinstated, as if the EXTEST instruction were active. This allows fixed logic levels to be maintained on selected drivers while others are in ac mode. This necessitates a controlling structure, and this standard adopts a structure much like the driver enable/disable capability allowed by IEEE Std 1149.1.

NOTE—This standard also allows for electrically disabling a controlled driver (i.e., off or on) [see permission a) in 6.3.1.2]. AC/DC selection cells are in addition to these.

An ac/dc selection cell modifies the behavior of a data cell, not the driver itself, as shown in Figure 64, for a 2-state driver pin. The ac/dc selection cell simply prevents the modulation of data via the ac test signal, for those cells controlled by it. The ac/dc selection cell does not monitor any system logic outputs. The simple cell shown in Figure 64 does not capture any data, but it is recommended that it capture the output of its Update flip-flop if improved testability of these cells is desired.

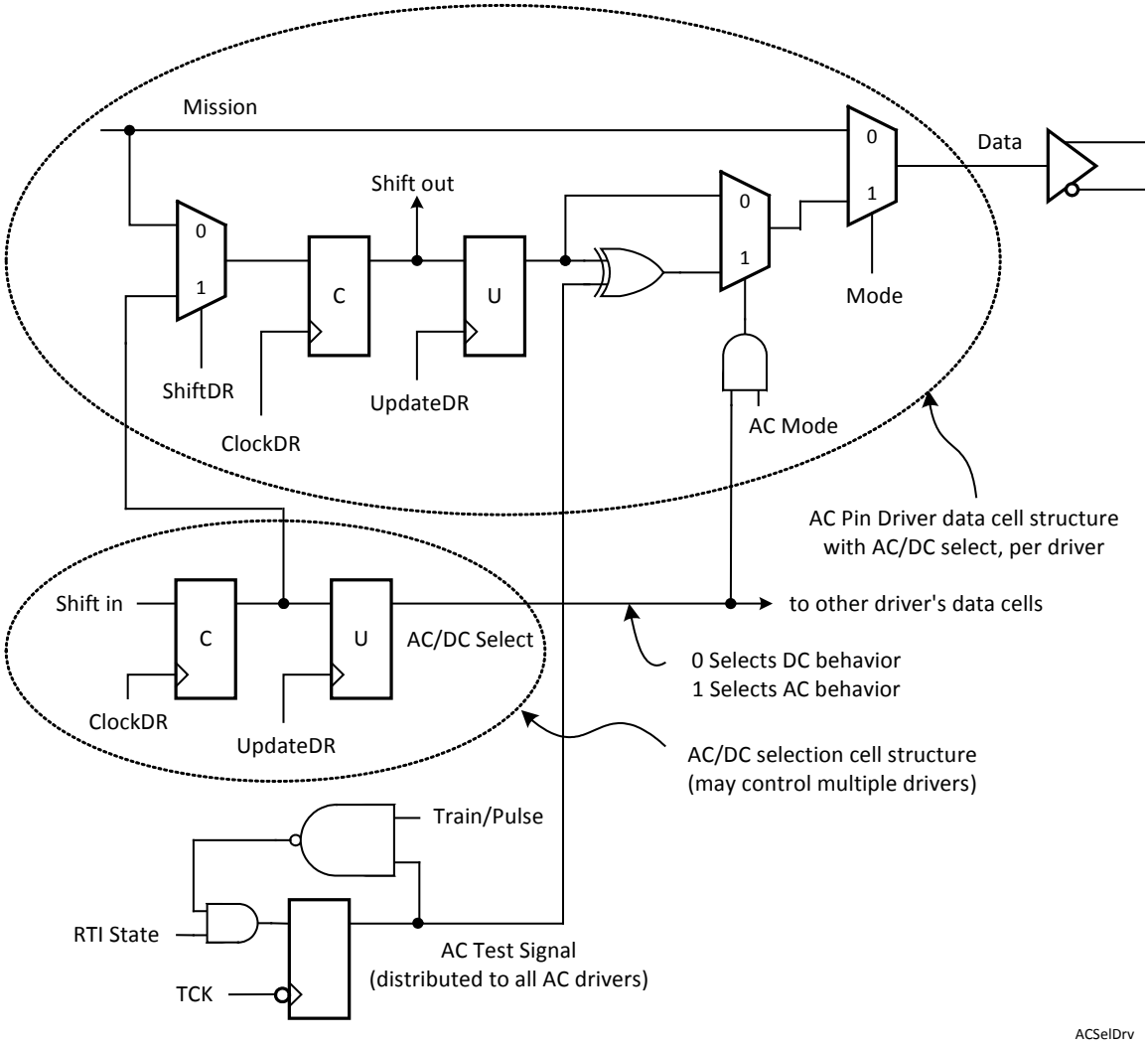


Figure 64—An ac pin driver structure with an ac/dc selection cell

6.5.1 Rules

AC/DC selection cell rules include the following:

a) If implemented, ac/dc selection cells shall be used to control one or more ac pin drivers.

b) An ac pin driver data cell shall be controlled by either 0 or 1 ac/dc selection cell, and if the Update latch of this cell is controlled by a TAP reset term, the Update latch shall be reset to 0.

c) When an ac/dc selection cell contains a 0, any associated ac pin driver data cells controlled by it will behave as if EXTEST is being executed when either the EXTEST_PULSE or EXTEST_TRAIN instructions are actually active.

NOTE—See 5.3.1 and 5.4.1 for rules on how ac pin data cells behave as conditioned by ac/dc selection cells.

d) When an ac/dc selection cell contains a 1, any associated ac pin driver data cells controlled by it will behave as specified for either the EXTEST_PULSE or EXTEST_TRAIN instructions when these are active.

e) If an ac pin driver data cell has no ac/dc selection cell associated with it, it shall always exhibit ac behavior in ac test mode.

f) AC/DC selection cells shall be composed of a Shift flip-flop and an Update flip-flop, identical to the control cell structure mandated by IEEE Std 1149.1.

g) When the boundary-scan register is selected by the effective instruction, ac/dc selection cells shall shift data from TDI towards TDO on the rising edge of TCK while in the *Shift-DR* TAP Controller state.

h) When the boundary-scan register is selected by the effective instruction, ac/dc selection cells shall transfer the content of the Shift flip-flop to the Update flip-flop on the falling edge of TCK in the *Update-DR* TAP Controller state.

i) If the boundary-scan register is segmented, then any ac/dc selection cell shall appear in the same boundary-scan register segment as all of the cells it controls.

6.5.2 Permissions

AC/DC selection cell permissions include the following:

a) AC/DC selection cells may be placed in any order within the boundary-scan register or boundary-scan register segment.

b) AC/DC selection cells may capture unspecified data or may capture the content of their Update flip-flops on the rising edge of TCK in the *Capture-DR* TAP Controller state, with the choice being the same for each of the SAMPLE, EXTEST, INTEST, EXTEST_PULSE, and EXTEST_TRAIN instructions, and any user instructions that select the boundary-scan register and expect to capture data.

6.5.3 Recommendations

AC/DC selection cell recommendations include the following:

a) AC/DC selection cells should capture the content of the Update flip-flop on the rising edge of TCK in the *Capture-DR* TAP Controller state, for the SAMPLE, EXTEST INTEST, EXTEST_PULSE and EXTEST_TRAIN instructions, or any user instructions that select the boundary-scan register and expect to capture data.

NOTE—This could improve the testability of these cells during device manufacture.

b) The quantity of ac/dc selection cells should be minimized to avoid significantly increasing the size of the boundary-scan register.

NOTE—This can be accomplished by assigning a single ac/dc selection cell to groups of logically related ac pin drivers.

6.5.4 Description

AC/DC selection cells can be added to a design to allow application software to force an ac pin to behave like a dc pin on selected ac pin drivers. Device designers are given the option to implement these controls on any ac pin drivers of their choosing. Designers of programmable devices might have to implement many of these cells if their devices contain highly flexible I/O pins.

The only constraint on placement of ac/dc selection cells in the Boundary Scan Register is that when the register is segmented, the ac/dc selection cells and all of the boundary cells it controls must be in the same segment.

7. Conformance and documentation requirements

7.1 Conformance

7.1.1 Specification

A component conforming to this standard shall comply with all rules provided herein.

NOTE 1—Due to the rule in 5.1.1, this also implies conformance with the rules set out in IEEE Std 1149.1-2013, except as provided herein.

NOTE 2—If compliance enable pins are used, the enabled state indicates compliance to all requirements of both IEEE Std 1149.1 and this specification.

7.1.2 Description

Conformance to the rules set out herein and in IEEE Std 1149.1 is essential for testing boards and other assemblies containing both dc and ac-coupled interconnections, allowing manufacturing defects such as shorted or open solder joints to be found and repaired before shipment. Conformance allows:

- IC vendors to provide testability features in a standardized way, so that each new IC design does not need new engineering investment to provide testability.
- Makers of Automatic Test Equipment to develop and continually refine standardized tools for the automation of test development, test execution, and diagnosis of failures.
- End-users to develop test methodologies in a way that is both standard and in line with a strategic direction, making full use of the automation provided by tools, to allow them to produce very large and complex boards and systems more rapidly and efficiently.

7.2 Documentation

Because adherence to this standard implies adherence to IEEE Std 1149.1, all devices conforming to this standard must have a description supplied in BSDL as specified in IEEE Std 1149.1-2013.

7.2.1 Specification

7.2.1.1 Rules

Documentation specification rules include the following:

a) A component conforming to this standard shall be documented with a BSDL description.

NOTE—See IEEE Std 1149.1-2013, Annex B for a description of IEEE BSDL. Note, versions of BSDL predating the 2013 version cannot be used for documentation of a component conforming to this standard since they do not support certain constructs such as PDL (see IEEE Std 1149.1-2013, Annex C) for programmable parameters that are used in this revision of this standard.

b) If EXTEST_PULSE and EXTEST_TRAIN instructions are implemented in a component, then these instructions shall be documented in the <instruction_opcode_stmt> element of the BSDL for that component as well as the optional BSDL extensions provided by this standard.

NOTE 1— New BSDL syntax (contained within a "BSDL extension") for describing concepts and structures introduced by this standard are given in 7.4. A BSDL extension is a mechanism provided by IEEE Std 1149.1-2013 (see User extensions to BSDL, B.8.24) which allows proprietary syntax to be provided that will allow tools to work that are unaware of this syntax.

NOTE 2— The <instruction_opcode_stmt> element is found in IEEE Std 1149.1-2013, B.8.11, Instruction Register description.

c) Prime and second source devices shall have nominally identical implementations of publicly accessible test circuitry, with the sole exception of the device identification code.

d) When either the output of a mission receiver connected to a differential input pin pair is monitored with a boundary-scan register cell, or the differential input pin pair is allowed to be terminated with a low impedance (e.g. 100 Ω) resistor between the two pins of the pair, then that differential input pin pair shall be documented within a Port_Grouping attribute of the BSDL description.

NOTE 1— Port grouping is documented in IEEE Std 1149.10-2013 in B.8.8, Grouped port identification.

NOTE 2— When the differential protocol requires termination of each leg independently to a low impedance voltage source, such as ground, then the termination process of this rule does not apply.

e) Other properties of each driver and test receiver on a device shall be documented by the manufacturer, including:
   1) The nominal value of ΔV_{Min} [see rule b) in 6.2.1.1]
   2) The nominal value of T_{Trans} [see rule b) in 6.2.1.1]
   3) The nominal value of ΔV_{Max} [see rule c) in 6.2.1.1]
   4) The nominal value of $V_{Threshold}$ [see rule a) in 6.2.2.1]
   5) The nominal value of V_{Hyst_Level} [see rule a) in 6.2.2.1]
   6) The nominal value of T_{Hyst} [see rule c) in 6.2.2.1]
   7) The nominal value of V_{Hyst_Edge} [see rule b) in 6.2.3.1]
   8) The minimum (off-chip) or nominal (on-chip) value (when implemented) of T_{HP} [see rule f) in 6.2.3.1]

9) The nominal value (when implemented) of T_{LP} [see rule g) in 6.2.3.1]
10) The nominal value of driver common-mode voltage
11) The nominal value of driver peak-to-peak voltage

NOTE—In addition, the manufacturer would also specify typical electrical parameters for device pins, including TAP signal voltage level requirements and the power requirements of the device.

f) If parameters of an ac pin are documented in the BSDL as programmable, then procedures for programming those parameters shall be documented with PDL descriptions.

NOTE—See IEEE Std 1149.1 Annex C for a description of IEEE PDL. Specific PDL procedures for documenting the use of programmable parameters are described in 7.7, with examples in 7.8.

7.2.1.2 Recommendations

If ac pins are provisioned with additional public test capabilities, which are useful at board or system test time, then these capabilities should be accessible via the TAP, and should have their access structures (if any) documented in BSDL and their access procedures (if any) documented in PDL, per IEEE Std 1149.1.

NOTE 1—Additional public test capabilities include near- or far-end loop-back, Bit-Error-Rate-Test (BERT), etc. These might be required in the I/O protocol or simply provided by the component designer. The use of "public" here is intended in the same sense as a "public" instruction, defined in IEEE Std 1149.1. A public instruction is documented for use, a private one is not. Making such additional test capabilities private (undocumented) restricts the usability of tests and would normally only be appropriate when the capability is reserved for component manufacturing characterization or test.

NOTE 2—PDL, and optional register description attributes, were introduced in the 2013 version of IEEE Std 1149.1.

7.2.2 Description

Consider a device with a full set of IEEE Std 1149.1 instructions and registers as well as the new ac testing instructions. Here are fragments of BSDL needed to document this device.

First, for devices that have ac differential inputs and/or outputs, a Port_Grouping attribute is given that identifies the positive and negative legs. For example, for a device with four pairs of differential data drivers, the Port_Grouping could be:

```
Attribute PORT_GROUPING of ACDEV:entity is
  "Differential_Voltage ( "   &
  "       (D(1), Dbar(1)), " &
  "       (D(2), Dbar(2)), " &
  "       (D(3), Dbar(3)), " &
  "       (D(4), Dbar(4))) ";
```

NOTE 1—Refer to IEEE Std 1149.1-2013, B.8.8, Grouped port identification, for precise information regarding the Port_Grouping attribute.

NOTE 2—The entity name in these examples is shown as "ACDEV." This would be replaced with a name unique to the device.

At a board or system level, differential pairs are interconnected with pair-wise wiring or coupling capacitors. With the information from the port grouping attribute, software can trace the pathways of the positive and negative legs of each pair.

NOTE 3—Note that some boards could swap the positive and negative legs of the pair-wise interconnections, with or without ac coupling. This might not affect functional operation, but test software must be aware of this possibility and record such occurrences when tracing pathways.

If an input differential pin pair is not observed by a differential to single-ended mission receiver with a boundary scan register cell, and each leg is separately terminated to a low impedance voltage source (as opposed to a termination resistor connecting the two pins), then that input pair can behave as independent inputs from a test perspective and does not need to be documented in the Port_Grouping attribute. The drivers for the two legs in a given board test could also be independent, allowing a more thorough test to be performed than if the inputs were included in the Port_Grouping attribute.

Next, an "Instruction_Opcode" attribute used to define instruction names and binary code assignments should be given:

```
Attribute INSTRUCTION_OPCODE of ACDEV:entity is
  "BYPASS      (111111), " &
  "EXTEST      (011000), " &
  "SAMPLE      (000001), " &
  "PRELOAD     (000001), " &  -- shared with SAMPLE
  "IDCODE      (010001), " &
  "USERCODE    (010010), " &
  "EXTEST_TRAIN(010011), " &
  "EXTEST_PULSE(000010)  ";
```

NOTE—Refer to IEEE Std 1149.1-2013, B.8.11, Instruction register description, for precise information regarding the Instruction_Opcode attribute.

Later, a "Register_Access" attribute is given:

```
Attribute REGISTER_ACCESS of ACDEV:entity is
  "Boundary  (EXTEST_PULSE, EXTEST_TRAIN)";
```

The Register_Access attribute defines the existence of optional registers, their length, the instruction(s) that target them and any consistent capture data they might be loaded with in the *Capture-DR* TAP Controller state. The above attribute documents the fact that ac testing targets the boundary-scan register.

NOTE 1—Refer to IEEE Std 1149.1-2013, B.8.13, Register access description, for precise information regarding the Register_Access attribute.

NOTE 2—This attribute can also (though redundantly) document the associations of EXTEST, SAMPLE, PRELOAD, BYPASS, USERCODE, and IDCODE as well. This was omitted here.

Advanced I/O, such as SerDes drivers and receivers, often have special test features built in. These might include near-end and far-end loop-back, Bit Error Rate Test (BERT), detection of open inputs, etc. In order to provide the best possible test and diagnosis at both component and board levels, it is recommended that the component be designed so that these test features are controllable and observable, as appropriate, via the TAP (this does not preclude other access methods in parallel). When this is done, then any test structures (such as register fields, register mnemonics, etc.), and any procedures, as needed to control and observe these features, can be documented using BSDL and PDL, respectively, as specified in IEEE Std 1149.1-2013, Annexes B and C. As these types of tests are not standardized, the tests themselves are not included in this standard, only the recommendation to provide TAP access and the requirement to document that access, when provided.

7.3 BSDL package for Advanced I/O description (STD_1149_6_2015)

The information provided in 7.2 showed how to document features of a device implementing ac testing instructions using the existing syntax of BSDL. This information is incomplete. BSDL has been defined (see IEEE Std 1149.1) to be extensible using a mechanism called a "BSDL Extension." A BSDL Extension for describing additional features of Advanced I/O devices is given here.

NOTE—Refer to IEEE Std 1149.1-2013, B.8.24, User extensions to BSDL, for precise information regarding BSDL extensions.

The extension mechanism chosen for describing devices is based on the definition of a VHSIC High-Level Design Language (VHDL) package with the name "STD_1149_6_2015," which contains the definitions of attributes that will be used to supply relevant data. Therefore, a compliant device BSDL will contain an additional "use" statement appearing just after the "standard use statement" as in this excerpt of a BSDL file:

```
  …
use STD_1149_1_2013.all;  -- Standard 'use' statement
use STD_1149_6_2015.all; -- BSDL Extension for AIO
  …
```

NOTE 2—Refer to IEEE Std 1149.1-2013, B.8.5, Use Statement, for precise information regarding references to additional packages.

Utilization of the extension mechanism of BSDL helps ensure that Advanced I/O information can be supplied to applications that are cognizant of this functionality without hindering other applications that might not be aware of this functionality. Noncognizant applications will simply ignore the extension.

The BSDL language allows cell constant information to describe cell capture information for the SAMPLE, EXTEST, and INTEST instructions defined by IEEE Std 1149.1. It does not allow new instructions (i.e., EXTEST_PULSE and EXTEST_TRAIN) to be used as instruction names. Rather than change the BSDL language to allow these new instructions, and because there is no difference in capture behavior allowed for any instruction by this standard, this standard adopts the convention that the EXTEST instruction name is representative of both dc EXTEST and the two ac test instructions provided by this standard. Thus in the cell definitions given below, only EXTEST will be listed, but it implies all three (EXTEST, EXTEST_PULSE, and EXTEST_TRAIN) instructions. This also applies to cell definitions given in the IEEE Std 1149.1 Standard BSDL Package (see clause B.9 of the IEEE 1149.1-2013), as well as all of the mode generation tables given in Clause 11 of that standard.

User-defined packages can be used to define new boundary-scan register cell definitions. This standard defines several new cell types to describe the cell behavior of ac/dc selection cells as defined in 6.5 and new data cells. The simplest selection cell is mandated by permission b) in 6.5.2, which allows the cell to capture nothing, or unspecified data ('X'). Recommendation a) in 6.5.3 advises the capture of the Update flip-flop content. In either case, this is true for the SAMPLE, EXTEST, INTEST, EXTEST_PULSE, and EXTEST_TRAIN instructions.

The cell context (see IEEE Std 1149.1-2013, B.10.1 Specifications) for ac/dc selection cells is, by convention set herein, "Internal" [see rule j) in 7.5.4.2]. Since the cell definitions given below do not support any other cell context, an attempt to use another context (e.g., CONTROL) will cause a BSDL semantic error and be rejected. This keeps common tools that are cognizant only of IEEE Std 1149.1 from associating a control capability (used to disable drivers) with ac/dc selection cells. Such software will ignore ac/dc selection cells, except to assure they are always loaded with the specified "safe values" (see IEEE Std 1149.1-2013, B.8.14.3.4, The <safe bit> element). Descriptions of ac/dc selection cells are found in Annex C.

A problem is here noted in IEEE Std 1149.1, between the versions released in 2001 and 2013. These two drafts show a reversal of two syntax items (see B.10.1) which appears to be a simple editorial error; they are <deferred constant> followed immediately by <extension declaration>. (This is the 2013 order.) Thus, the 2003 version of 1149.6, which precedes this version, is correct by the rules of the 2001 version of 1149.1, but not according to the 2013 version. The 2003 ordering is preserved in this version of the 1149.6 standard so as not to confuse the tools that are in place that understand this ordering. The order is not consequential to the meaning of what BSDL conveys. A potential future solution may be to change a future version of 1149.1 to allow either order for the two syntax items, again, since the order is inconsequential.

The VHDL package STD_1149_6_2015 contains the definition of additional attributes used to complete the description of the Advanced I/O features of a device. The content of this VHDL package is given here.

```
Package STD_1149_6_2015 is -- Attribute definitions for AIO
  use STD_1149_1_2013.all; -- Refer to BSDL definitions

  attribute AIO_Component_Conformance:BSDL_Extension;
  attribute AIO_EXTEST_Pulse_Execution:BSDL_Extension;
  attribute AIO_EXTEST_Train_Execution:BSDL_Extension;
  attribute AIO_Pin_Behavior:BSDL_Extension;
  attribute AIO_Port_Behavior:BSDL_Extension;

  constant AC_SelX : Cell_Info; -- AC/DC selection X
  constant AC_SelU : Cell_Info; -- AC/DC selection U
  constant AC_1 : Cell_Info; -- Output cell derived from BC_1
  constant AC_2 : Cell_Info; -- Output cell derived from BC_2
  constant AC_7 : Cell_Info; -- Output cell derived from BC_7
  constant AC_8 : Cell_Info; -- Output cell derived from BC_8
  constant AC_9 : Cell_Info; -- Output cell derived from BC_9
  constant AC_10 : Cell_Info; -- Output cell derived from BC_10
  constant AC_40 : Cell_Info; -- BC_4 that captures '0' in SAMPLE
  constant AC_41 : Cell_Info; -- BC_4 that captures '1' in SAMPLE
end STD_1149_6_2015;

Package Body STD_1149_6_2015 is
  use STD_1149_1_2013.all; -- Refer to BSDL definitions

  constant AC_SelX : Cell_Info:= -- Captures ‘X’, unknown data
    ((INTERNAL, SAMPLE, X),
     (INTERNAL, INTEST, X),
     (INTERNAL, EXTEST, X)); -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN

  constant AC_SelU : Cell_Info:= -- Captures ‘UPD’, the Update FF
    ((INTERNAL, SAMPLE, UPD),
     (INTERNAL, INTEST, UPD),
     (INTERNAL, EXTEST, UPD)); -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN

  constant AC_1 : Cell_Info:= -- Output cell derived from BC_1
    ((OUTPUT2, SAMPLE, PI),
     (OUTPUT2, EXTEST, PI), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OUTPUT2, INTEST, PI),
     (OUTPUT3, SAMPLE, PI),
     (OUTPUT3, EXTEST, PI), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OUTPUT3, INTEST, PI));

  constant AC_2 : Cell_Info:= -- Output cell derived from BC_2
  -- Note, corrects data errors in the 2003 draft of STD_1149_6_2003.
    ((OUTPUT2, SAMPLE, PI),    -- No support for INTEST
     (OUTPUT2, EXTEST, UPD),  -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OUTPUT3, SAMPLE, PI),
     (OUTPUT3, EXTEST, UPD)); -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN

  constant AC_7 : Cell_Info := -- BIDIR cell derived from BC_7
    ((BIDIR_IN, SAMPLE, PI),
     (BIDIR_IN, EXTEST, PI), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (BIDIR_IN, INTEST, UPD),
     (BIDIR_OUT, SAMPLE, PI),
```

```
    (BIDIR_OUT, EXTEST, PO), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
    (BIDIR_OUT, INTEST, PI));

  constant AC_8 : Cell_Info := -- BIDIR cell derived from BC_8
    ((BIDIR_IN, SAMPLE, PI), -- No support for INTEST
     (BIDIR_IN, EXTEST, PI), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (BIDIR_OUT, SAMPLE, PO),
     (BIDIR_OUT, EXTEST, PO)); -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN

  constant AC_9 : Cell_Info :=
  -- Self-monitoring Output cell derived from BC_9
    ((OUTPUT2, SAMPLE, PI),
     (OUTPUT2, EXTEST, PO), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OUTPUT2, INTEST, PI),
     (OUTPUT3, SAMPLE, PI),
     (OUTPUT3, EXTEST, PO), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OUTPUT3, INTEST, PI));

  constant AC_10 : Cell_Info :=
  -- Self-monitoring Output cell derived from BC_10
    ((OUTPUT2, SAMPLE, PO), -- No support for INTEST
     (OUTPUT2, EXTEST, PO), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OUTPUT3, SAMPLE, PO),
     (OUTPUT3, EXTEST, PO)); -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN

  constant AC_40 : Cell_Info :=
  -- Observe-only cell that captures 0 during SAMPLE
    ((OBSERVE_ONLY, EXTEST, PI),-- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OBSERVE_ONLY, SAMPLE, ZERO)); -- Intest on observe_only not
                                    -- supported

  constant AC_41 : Cell_Info :=
  -- Observe-only cell that captures 1 during SAMPLE
    ((OBSERVE_ONLY, EXTEST, PI),-- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OBSERVE_ONLY, SAMPLE, ONE)); -- Intest on observe_only not
                                   -- supported

end STD_1149_6_2015;
```

This VHDL package is “read-only” and might be maintained within a given system in the same location as the standard package STD_1149_1_2013.

7.4 BSDL extension structure

The following subclauses define the syntax and applicable semantics for the BSDL attributes defined by the Advanced I/O extension. The form of the syntax used is defined in IEEE Std 1149.1.

NOTE 1—See IEEE Std 1149.1-2013, Lexical elements of BSDL, B.5 and B.6, and Notes on syntax definitions for the conventions used herein to describe the parsing requirements for BSDL. Give particular notice to the convention j) specified by B.6.1 (BNF Conventions, IEEE Std 1149.1-2013) where character(s) used in the syntax descriptions are given syntactic names: for example, the “:” character is represented by <colon>.

NOTE 2—Syntactic items are shown surrounded in “< >” brackets when referenced in this text. Some of these items will be defined here, and some will be adopted from IEEE Std 1149.1. Those that are adopted will be underlined to indicate their source.

When an Advanced I/O extension exists in the extension area of a BSDL description, it must have the structure shown here. The various attributes, both mandatory and optional must appear in a prescribed order and not be intermixed with other statements.

NOTE 3—These other statements include BSDL attributes, attributes for other BSDL extensions, and general VHDL constructs.

The specific syntax item <real> is defined here and not adopted from IEEE Std 1149.1-2013, B.5.5, Numeric literals. This is to correct a problem with the definition of <real> given there, which precludes describing real numbers less than 1.0 using negative exponents. The definition of <real> used in this standard is:

> *A real number shall be represented in the syntax by the token <real> and shall be of the form<integer><period><integer> or <integer><period><integer><exponent>, where <exponent> is **E**<integer> or **E**<minus sign><integer>, all written contiguously without spaces, underscores, or format effectors; the E is case insensitive. Note that <exponent> can be positive or negative. Examples of real numbers are 125000.0, 1.25e5 and 37.5E-9.*

NOTE 4—This definition precludes describing negative real numbers (such as –2.3). In the event that a subsequent revision of this standard requires expressing negative reals, then the definition above will be revised, and places where the item <real> are used will need additional rules added when the the parameter being described is not to be a negative number. For example, a voltage could be negative or positive, but a time constant can only be a positive number.

7.4.1 Specification

7.4.1.1 Syntax

The BSDL extension structure specification syntax is as follows:

```
<AIO Extension> ::=
   <AIO component conformance statement>        (see 7.5.1)
  ├ [<AIO optional EXTEST_PULSE description>]   (see 7.5.2)
   [<AIO optional EXTEST_TRAIN description>] ┤  (see 7.5.3)
   <AIO behavior>

<AIO behavior> ::=
  <AIO optional pin behavior description>  |    (see 7.5.4)
  <AIO port link behavior description>          (see 7.5.5)
```

NOTE—In syntax expressions used here, the square bracket symbols '[' and ']' mean the enclosed syntactic item(s) can occur only once or can be omitted. Syntactic items enclosed between curly bracket symbols '{' and '}' can occur zero or more times. Syntactic items between '├' and '┤' symbols can occur in any order. Syntax items separated by "|" are choices, one or the other is allowed.

7.4.1.2 Rules

The BSDL extension structure specification rules are as follows:

a) The syntax for an <AIO Extension> as it appears in the extension area of a BSDL or User Pacakge description shall be that shown in 7.4.1.1.

   NOTE—The attributes must appear in the order shown in 7.4.1.1.

b) When an <AIO Extension> is placed in a user package, it shall not have an <AIO optional pin behavior description> included in it, and when an <AIO Extension> is placed in a component description, it shall not have an <AIO port link behavior description> included in it.

7.4.1.3 Permissions

An <AIO Extension> may appear anywhere in the extension area of a BSDL description, subject to any rules defined by other extensions as to ordering.

NOTE—If more than one extension exists, an <AIO Extension> can appear before or after any of them within the limits of their definitions.

7.4.2 Description

At this time there are up to five elements within an `<AIO Extension>`, though only four at a time may exist within either a component BSDL or a user package [see rule b) of 7.4.1.2]. No other statements are to be intermixed within the extension, as the syntax described herein specifies. This extension can coexist with, either preceding or following, other extensions.

7.4.3 Keywords for Advanced I/O BSDL

This subclause lists the keywords of Advanced I/O BSDL. These keywords are in addition to the reserved words of BSDL and VHDL (see BSDL reserved words and VHDL reserved and predefined words in IEEE Std 1149.1-2013, B.5.2). Keywords are not case-sensitive, but by convention are shown in bold type.

AC_0 to **AC_99**
AC_Select
AC_SelU
AC_SelX
AIO_Component_Conformance
AIO_EXTEST_PULSE_Execution
AIO_EXTEST_TRAIN_Execution
AIO_Pin_Behavior
AIO_Port_Behavior
AIO_VCM
AIO_VHyst
AIO_VPP
AIO_VTH
Bus_Programmable
Coupling_Time
Detect_EXTEST_High
Detect_EXTEST_Low
Detect_EXTEST_Both
Detect_EXTEST_Both_Grouped
Edge_Detect_Time
External
HP_Time
LP_Time
Maximum_Time
No_Pulse
On_Chip
On_Chip_Bus_Programmable
On_Chip_Programmable
On_Chip_AC

On_Chip_AC_Bus_Programmable
On_Chip_AC_Programmable
Port_Link_List
Programmable
STD_1149_6_2003
STD_1149_6_2015
Train

7.5 BSDL attribute definitions

The mandatory and optional attributes needed to describe the Advanced I/O properties of a device are given in the remainder of this subclause. They must appear in the order given in 7.4.1.1.

7.5.1 Attribute AIO_Component_Conformance

The mandatory AIO_Component_Conformance attribute is used to identify the version of this standard with which a given device conforms.

7.5.1.1 Syntax specification

The AIO_Component_Conformance attribute syntax is as follows:

<AIO component conformance statement> :: = **attribute AIO_Component_Conformance**
of <target> **is** <AIO conformance string> <semicolon>
<AIO conformance string> ::= <quote> <AIO conformance identification> <quote>
<AIO conformance identification> ::= **STD_1149_6_2003 | STD_1149_6_2015**

7.5.1.2 Rules

The AIO_Component_Conformance attribute rules are as follows:

a) The syntax for the mandatory AIO_Component_Conformance attribute shall be that shown in 7.5.1.1.
b) When the AIO_Component_Conformance attribute has a value of STD_1149_6_2015, the Component Conformance attributes specified in IEEE Std 1149.1 shall have a value of STD_1149_1_2013 or later.
c) When this attribute appears in a BSDL entity description, then <target> shall resolve to <component name> <colon> **entity** and the <component name> shall match the entity name; further, when this attribute appears in a User Package description, then <target> shall resolve to <user package name> <colon> **package** and the <user package name> shall match the name of the user package.

7.5.1.3 Description

The value of <AIO conformance identification> is STD_1149_6_2003 or STD_1149_6_2015 only. In future revisions of this standard new values could be defined. Certain features of this standard are dependent on features introduced in the IEEE Std. 1149.1-2013, and so both the Component_Conformance and the AIO_Component_Conformance attributes must have compatible values.

An AIO_Component_Conformance attribute can appear in either a package description or a BSDL entity description, per the provisions given by rule 7.5.1.2 c).

Examples

```
Attribute AIO_Component_Conformance of ACDEV:entity is
```

```
        "STD_1149_6_2003";

Attribute AIO_Component_Conformance of MyCorp_SERDES:package is
        "STD_1149_6_2015";
```

7.5.2 Optional attribute AIO_EXTEST_Pulse_Execution

This optional attribute is used to describe pulse width timing requirements for the EXTEST_PULSE instruction allowed by permission a) in 5.3.2. The omission of this attribute implies there is no such requirement.

7.5.2.1 Syntax specification

The AIO_EXTEST_PULSE attribute syntax is as follows:

<AIO optional EXTEST_PULSE description> :: = **attribute AIO_EXTEST_Pulse_Execution**
of <target> **is** <AIO EXTEST_Pulse string> <semicolon>
<AIO EXTEST_Pulse string> ::= <quote> <AIO EXTEST_Pulse spec> <quote>
<AIO EXTEST_Pulse spec> ::= **wait_duration** <time spec>
<time spec> ::= <real> | <clock cycles>

7.5.2.2 Rules

The AIO_EXTEST_PULSE attribute rules are as follows:

a) The syntax for the optional <AIO optional EXTEST_PULSE description> attribute shall be that shown in 7.5.2.1.

b) The value of <clock cycles> shall be the <port ID> in the <TCK stmt> element of the <scan port identification>, followed by an integer greater than 0.

c) When multiple <AIO optional EXTEST_PULSE description> statements exist within a component BSDL and all User Packages specified, the longest duration <time spec> across all such statements shall be used for the component.

   NOTE—Since some specifications may be in terms of seconds, and others in terms of TCK cycles, this may have to be resolved at test generation time assuming the tester, and the TCK period to be used in the test, is known.

d) If the <AIO optional EXTEST_PULSE description> appears in a user package, then the <time spec> shall be <real>.

   NOTE—This rule recognizes that a <portID> in the <TCK_stmt> element of the <scan_port_identification> and associated frequency are not known in a user package.

7.5.2.3 Description

Rule b) in 7.5.2.2 states the only port allowed to be listed in a <clock cycles> element is that identified as the TCK port for the device.

The value of <real> specifies the minimum wait time in the *Run-Test/Idle* TAP Controller state in seconds. The value of <integer> in a <clock cycles> element specifies the minimum wait time in terms of full cycles of TCK.

When an IP has specific requirements for the EXTEST_PULSE instruction, then the optional EXTEST_PULSE_EXECUTION attribute would be coded in the User Package. This means there could be multiple such attributes coded in the total structure of the component BSDL and all of the User Packages. Rule c) in 7.5.2.2 specifies how to resolve such multiple EXTEST_PULSE_EXECUTION statements.

Examples

```
Attribute AIO_EXTEST_Pulse_Execution of ACDEV:entity is
       "Wait_Duration 1.0e-3";
Attribute AIO_EXTEST_Pulse_Execution of ACDEV:entity is
       "Wait_Duration TCK 6";
```

7.5.3 Optional Attribute AIO_EXTEST_Train_Execution

This optional attribute is used to describe pulse train and possible timing requirements for the EXTEST_TRAIN instruction allowed by permissions a) and b) in 5.4.2. The omission of this attribute implies there are no such requirements.

7.5.3.1 Syntax specification

The AIO_EXTEST_TRAIN attribute syntax is as follows:

<AIO optional EXTEST_TRAIN description> :: = **attribute AIO_EXTEST_Train_Execution**
of <target> **is** <AIO EXTEST_Train string> <semicolon>
<AIO EXTEST_Train string> ::= <quote> <AIO EXTEST_Train spec> <quote>
<AIO EXTEST_Train spec> ::= <min pulse count> [<comma> <max time spec>]
<min pulse count> ::= **train** <pulse count>
<max time spec> ::= **maximum_time** <real>
<pulse count> ::= <integer>

7.5.3.2 Rules

The AIO_EXTEST_TRAIN attribute rules are as follows:

a) The syntax for the optional <AIO optional EXTEST_TRAIN description> attribute shall be that shown in 7.5.3.1.

b) The value of <pulse count> shall be an integer greater than 0.

c) In a <max time spec> the value of <real> shall be greater than 0.0 and in units of seconds.

d) When multiple <AIO optional EXTEST_TRAIN description> statements exist within a component BSDL and all used User Packages, the <min pulse count> for the component shall be set to the maximum of all <min pulse count> values in such statements, and the <max time spec> for the component shall be calculated by multiplying the component <min pulse count> (previously determined) times the minimum of the set of values calculated by dividing the <max time spec> by the <min pulse count> for each such statement.

7.5.3.3 Description

The value of <pulse count> specifies the minimum number of pulses to be issued immediately prior to exit of the *Run-Test/Idle* TAP Controller state (one pulse is issued for every two TCK clock cycles). The value

of <real> within the optional <max time spec> element specifies the maximum time (in units of seconds) prior to the exit of the *Run-Test/Idle* TAP Controller state within which the specified minimum number of pulses should occur. Note that more pulses could occur in that same maximum time frame, as well as additional pulses that could occur prior to the time frame. When a <max time spec> value is given, this implies a minimum TCK frequency, at least during the *Run-Test/Idle* TAP controller state.

When an IP has specific requirements for the EXTEST_TRAIN instruction, then the optional AIO_EXTEST_TRAIN_EXECUTION attribute would be coded in the User Package. This means there could be multiple such attributes coded in the total structure of the component BSDL and all of the User Packages. Rule d) specifies how to resolve such multiple AIO_EXTEST_TRAIN_EXECUTION statements. This same calculation would be made at the component or IP level to determine the original AIO_EXTEST_TRAIN_EXECUTION values anytime there are multiple I/O with differing EXTEST_TRAIN requirements. Note that we do not simply use the minimum of the specified <max time spec> values. The <max time spec> is intended to apply only for the specified minimum number of pulses, and essentially sets a minimum time per pulse (cycle time). That is the minimum that must be met from a timing perspective.

Examples

```
  Attribute AIO_EXTEST_Train_Execution of ACDEV:entity is
                       "train 30";

  Attribute AIO_EXTEST_Train_Execution of ACDEV:entity is
                       "train 30, maximum_time 5.0e-3";
```

Given the two following attribute statements in the BSDL and a used User Package, the correct values for EXTEST_TRAIN execution would be calculated as follows.

```
-- In IP User Package:
  Attribute AIO_EXTEST_Train_Execution of ACIP:package is
        "train 30, maximum_time 5.0e-3";
......
-- In component BSDL:
  Attribute AIO_EXTEST_Train_Execution of ACDEV:entity is
        "train 20, maximum_time 3.0e-3";
......
-- Set actual values as follows:
-- min pulse count = MAX(30, 20) = 30
-- max time spec = 30 * MIN(5.0e-3/30, 3.0e-3/20)
--               = 30 * MIN(1.67e-4, 1.5e-4)
--               = 30 * 1.5e-4
--               = 4.5e-3
-- This is equivalent, for the combined component and IP, to:
  Attribute AIO_EXTEST_Train_Execution of ACDEV:entity is
        "train 30, maximum_time 4.5e-3";
-- This simultaneously meets the requirements of both original
-- attributes.
```

7.5.4 Attribute AIO_Pin_Behavior (Entity)

AIO pin behavior may be documented at the entity level inside a BSDL file with this AIO_Pin_Behavior attribute, or may be documented in a supporting user package given by an IP supplier with the AIO_Port_Behavior attribute described later in 7.5.5. The documentation of the actual analog parameters, used by both of these attributes, is given in 7.5.6.

The optional AIO_Pin_Behavior attribute is used to enumerate those system pins of a component that are "AC" pins per rule a) of 6.1.1 and have been provided with Advanced I/O capability. For each ac pin, an appropriate set of parameters is defined to document the following characteristics of the pin:

— AC/DC select cells for these pins

— Time constants of any edge-detection and/or any high-pass coupling filters, when implemented

— The presence of an inline capacitor (for the high-pass coupling filter) on chip, and whether it is bypassable

— Common-mode voltage and minimum voltage swing for drivers, and threshold voltage and hysteresis voltage for receivers

— Receiver EXTEST detection behavior when part 2) of rule a) of clause 6.2.2.1 is implemented

NOTE—Future revisions of this standard could address other advanced I/O test issues in addition to those currently addressed. In that case, where a component in compliance with a future revision does not have any ac pins, this attribute would be made optional. For components compliant with the current standard, a behavior attribute would be needed in all cases.

7.5.4.1 Syntax specification

The AIO_Pin_Behavior attribute syntax is as follows:

NOTE—The term <equal> used below represents the single character "=".

<AIO optional pin behavior description> :: = **attribute AIO_Pin_Behavior of** <entity target> **is**
<quote> <AC entity info list> <quote> <semicolon>
<AC entity info list> ::= <entity pin info> { <semicolon> <entity pin info> }
<entity pin info> ::= <AC port list> <colon> [<AC/DC select cell>] <AC parameters>
<AC port list> ::= <AC ports> { <comma> <AC ports> }
<AC ports> ::= <AC port> [<input cell list>]
<AC port> ::= <port name> | <subscripted port name > | <ranged port name>
<subscripted port name> ::= <port name> <left paren> <integer> <right paren>
<ranged port name> ::= <port name> <left paren> <range> <right paren>
<input cell list> ::= <left bracket> <cell number list> <right bracket>
<cell number list> ::= <cell reference> { <comma> <cell number> }
<cell reference> ::= [<boundary segment name> <colon>] <cell number>
<AC/DC select cell> ::= **AC_Select** <equal> <cell reference>
<AC parameters> ::= <AC parameter list> | <AC port link list>

NOTE—The definition of <AC parameter list> is given in 7.5.6.

<AC port link list> ::= [**PACKAGE** <package hierarchy> <colon>] <port link reference list>
<port link reference list> ::= <port link reference> { <comma> <port link reference> }
<port link reference> ::= <port link name ref> [<input threshold>]

NOTE—The <input threshold> element is defined in 7.5.6

<port link name ref> ::= <port link name> | <subscripted port link name> | <ranged port link name>
<subscripted port link name> ::= <port link name> <left paren> <integer> <right paren>
<ranged port link name> ::= <port link name> <left paren> <range> <right paren>
<port link name> ::= <VHDL identifier>

7.5.4.2 Rules

The AIO_Pin_Behavior attribute rules are as follows:

a) The syntax for the optional <AIO optional pin behavior description> attribute shall be that shown in 7.5.4.1.

b) The <AIO optional pin behavior description> attribute shall be provided for components (entities) that contain ac pins, including ac pins contained in instantiated IP.

c) All ac pins in an entity and all instantiated IP shall be uniquely listed or implied in an <AC port list>.

d) The value of a <port name> shall be declared in the logical port description for the component, shall be of <port dimension> **bit** or **bit_vector**, and all bits shall be of the same <pin type>.

NOTE—A <port name> is described in IEEE Std 1149.1-2013, B.6.2, Commonly used syntactic elements. See also B.8.3, Logical port description statement, in that standard. A <port name> of dimension **bit_vector** used without any subscript or subrange is treated here as a list of subscripted ports with subscripts of the values and in the order (left to right) of the <range> specified in the logical port description.

e) A <subscripted port name> shall be composed of a <port name> declared as type **bit_vector** and an integer subscript that is an element of the <range> declared for that <port name> in the logical port list.

f) A <ranged port name> shall be composed of a <port name> declared as type **bit_vector** and a <range> that is a proper subset of the <range> declared for that <port name> in the logical port list, and all bits of the specified <range> shall be of the same <pin type>.

g) Each element of the <AC port list> shall refer to a system pin only and not to any pin that is a scan port signal or a compliance port pin; further, the <pin type> of the <AC port> shall not be any of the "**linkage_**" or "**power_**" BSDL reserved words.

NOTE—See IEEE Std 1149.1-2013, B.5.2 for BSDL reserved words. In previous revisions of this document which referred to earlier versions of IEEE Standard 1149.1, the reserved word "**linkage**" was the only disallowed port name type.

h) All explicit or implied elements of the <AC port list> shall be unique across all instances of an <AC port list>.

i) When an <entity pin info> element contains an <AC/DC select cell> element, then the <pin type> of the <port ID> being referenced shall be "**out**", "**inout**" or "**buffer**" only.

j) The value of <AC/DC select cell> shall be the <cell reference> of a boundary-scan register cell listed in a <cell table> of a <boundary register stmt> or a <boundary register segment>, with a <function> value of "**internal**" as recorded in the <cell spec> for that cell.

k) The value of the <safe bit> element in the <cell spec> for an <AC/DC select cell> shall be "0."

l) When the <associated port> of a <twin group> in a <grouped port identification> appears in an <AC port list> it shall appear in the same <AC port list> as the <representative port>.

NOTE—The <associated port> does not need to appear in the <AC port list> unless it has multiple boundary scan register cells associated with it. If it appears, it must be in the same <AC port list> as the representative port. If it does not appear, then the port characteristics are inherited from the representative port. In either case, the characteristics of the grouped representative and associated ports must be the same.

m) An <AC ports> element shall contain an <input cell list> whenever an <AC port> referenced in the <AC port list> is associated with more than one cell listed in the <boundary register stmt> or a <boundary register segment> with a <function> value of either "**input**" or "**observe_only**."

n) The <input cell list> shall list only boundary-scan register cells capturing the output of a test receiver.

o) Any <cell number> in an <input cell list> shall be the <cell number> of a boundary-scan register cell listed in the <cell table> of a <boundary register stmt> or a <boundary register segment>, with a <function> value of either "**input**" or "**observe_only**" as recorded in the <cell spec> for that cell.

p) When an <AC ports> element contains an <input cell list>, then the number of <cell number> elements in the <input cell list> shall equal the number of ports listed or implied in the <AC port list>, and individual ports shall be associated with individual cells by left-to-right order.

q) For each <cell number> given in an <input cell list> of a given <AC ports>, where the port in the <AC port list> associated with the cell:

1) is a <subscripted port name>, the boundary-scan register cell listed in the <boundary register stmt> or <boundary register segment> with matching <cell number> also shall have the matching <subscripted port name>; or

2) is a <port name> of dimension **bit**, the boundary-scan register cell listed in the <boundary register stmt> or <boundary register segment> with matching <cell number> also shall have the matching <port name>; or

3) is a <port name> of dimension **bit_vector**, the boundary-scan register cell listed in the <boundary register stmt> or <boundary register segment> with matching <cell number> also shall have a <port ID> that is a <subscripted port name> wherein the <port name> is matching and the <subscript> corresponds to the same position within the port's declared (if just a <port name>) or specified (if a <ranged port name>) <range> as the <cell number> within the <input cell list>.

r) When the boundary-scan register is defined by multiple boundary segments, then a <cell reference> shall be composed of the <boundary segment name>, a colon separator, and a valid <cell number> within the <cell table> of that named segment; and further, when the boundary-scan register is defined by the single boundary register statement, then a <cell reference> shall be composed of just a valid <cell number> within the <cell table> of that definition.

s) When the boundary-scan register is defined by multiple boundary segments, all <cell number> items in the <cell number list> shall be part of the same boundary segment as the initial <cell reference>.

t) When an <entity pin info> element contains an <AC port link list>, then the number of <port link reference> bits listed or implied in the <AC port link list> either shall be one, or shall be a number equal to the number of ports listed or implied in the <AC port list> of that <entity pin info>, in which case the individual ports shall be associated with individual links by left-to-right order.

u) A <port link reference> shall include an <input threshold> if and only if the <AC parameter list> of the <package pin info> included an <input threshold> with a <VTH choice> value of **external** for that <port link declaration>.

NOTE—See ac Parameters in 7.5.6 for definitions of syntax items in rule u).

v) The value of a <port link name> in the <port link reference list> shall have been declared in the <port link declaration list> of the associated package <AIO port link behavior description>, shall be of <port dimension> **bit** or **bit_vector**, and all bits shall be of the same <pin type>.

NOTE—The <AIO port link behavior description> is given in 7.5.5.

w) A <subscripted port link name> shall be composed of a <port link name> declared as type **bit_vector** and an integer subscript that is an element of the <range> declared for that <port link name> in the <port link declaration>.

x) A <ranged port link name> shall be composed of a <port link name> declared as type **bit_vector** and a <range> that is a proper subset of the <range> declared for that <port link name> in the <port link declaration>, and all bits of the specified <range> shall be of the same <pin type>.

7.5.4.3 Description

There are a number of options allowed in the design of the test circuitry designed for ac pins. In addition, some of the analog characteristics of the drivers and test receivers are important to initializing, understanding, and diagnosing the test results. The purpose of the <AIO optional pin behavior description> attribute is to associate each ac pin of a component with a set of parameters documenting those options and characteristics. For differential channels, only the ac pin of the representative leg needs to be documented as both representative and associated legs must have the same values.

When the ac pin is instantiated directly in the component, then the <entity pin info> element of the <AIO optional pin behavior description> attribute associates one or more system pins with a single set of analog parametric values (see 7.5.6) and, optionally, with an ac/dc select cell for controlling the ac behavior of a driver. The analog parametric values and any ac/dc select cell apply to all system pins listed in that <entity pin info>. The analog parameters are specifically required for use by test and diagnostic software to allow a high level of automation and success for test. Multiple <entity pin info> elements could be listed to allow for different sets of analog parameters. The system pins in the list can be single ports (a port name of type **bit**, or a subscripted port name of type **bit_vector**), or a collection of system ports (a port name of type **bit_vector** with or without a specified range).

When the circuitry (drivers, mission and test receivers, etc.) for a system pin is contained within a third party (IP) macro on the component, then the association of the IP ports to their analog parameters will be described in a User Package using the AIO_Port_Behavior attribute (see 7.5.5) using a local name called a port link name. Then, in the <entity pin info> element, the system pins will be associated with port link names and multiple system pins can be associated with either one port link name (assuming the IP has been multiply instantiated) or with a list of port link names. Again, the port link names in the list can be single ports (a port link name of type **bit**, or a subscripted port link name of type **bit_vector**), or a collection of IP ports (a port link name of type **bit_vector** with or without a specified range). When both the system ports and the port links are lists, and all system pin port names and all IP port link names are fully listed as single ports by expanding the ranges of a collection of ports or port links, the total number of ports in each list must be the same and individual ports are associated with port link names in a left-to-right order, as defined in rule r) of 7.5.4.2. All port link names used must be defined in the AIO_Port_Behavior attribute in the specified IP User Package, as defined in rule t) of 7.5.4.2.

Rule i) in 7.5.4.2 states that only cells with output driver capability can have ac/dc selection cells associated with them. Rule j) in 7.5.4.2 requires that an ac/dc selection cell must be described as an "internal" cell in the boundary-scan register description. Internal cells, as known to BSDL, are "place keeper" cells with an unknown function. This standard provides a particular function known only to it. Rule k) in 7.5.4.2 requires the "safe bit" of an ac/dc selection cell be set to "0," which becomes a default value for tools that do not compute a value for this cell. This allows tools that are not cognizant of ac testing to assure that ac pin behavior is not selected, even if for some reason they were to make the EXTEST_PULSE or EXTEST_TRAIN instruction active.

Rule l) in 7.5.4.2 states that if the associated port of a grouped pair is documented in the AIO_PIN_BEHAVIOR attribute, then it must appear in the same port list as the representative port. Normally, the associated port is not documented in this attribute and it inherits the characteristics of the representative port. However, IEEE Std 1149.1, starting with the 2013 version, allows multiple OBSERVE_ONLY boundary scan register cells to be associated with a single port, and in this case the associated port must be listed in order to identify the boundary scan register cell associated with the test receiver. In either case, the rules require that the grouped pair of ac pins have identical test receivers.

Rule m) in 7.5.4.2 governs the special situation where an ac input pin has more than one boundary-scan register cell of type "**input**" or "**observe_only**" associated with it. For example, the pin could have a test receiver and there could also be an input cell monitoring the mission receiver. In such a case, tools will need to know which cell actually monitors the output of the test receiver for that input pin. The cell that actually monitors the test receiver is identified by its inclusion in the <input cell list>. When multiple

input/observe-only cells do not exist (the more common case) then an <input cell list> can be omitted, though it is not an error (just redundant) to include it when only a single input/observe-only cell exists.

Rules n) through p) in 7.5.4.2 assure that there is a one-to-one mapping between ports and <cell number> elements in an <input cell list> whether the <port ID> is a single bit or a bit vector, and that each listed cell is of type "**input**" or "**observe_only**."

Rule q) in 7.5.4.2 gives three cases where, in each, the <port ID> element(s) implied in a given <AC port> must match the <port ID> element(s) listed the boundary-scan Cell(s) with matching <cell number> element(s). The three cases are:

a) Where the <port ID> is subscripted representing a single port, for example, PORT_X(5); here the port is "bit_vector" dimensioned, with subscript 5 being specified.
b) Where the <port ID> is not subscripted and has a dimension of "bit", for example, PORT_Y.
c) Where the <port ID> is not subscripted or has a specified <range>, and has a dimension of "bit_vector", then it represents an ordered list of ports with subscripts that lie within the declared or specified <range>, and these map positionally one-to-one with the <input cell list> elements.

As an example of case c) above, the port statement could be:

```
port PORT_Z: in bit_vector (4 downto 0);
```

and <AC port> could then be:

```
        ...
PORT_Z(4 downto 2) [22, 25, 28]
        ...
```

In this example, PORT_Z(4) maps to cell number 22, PORT_Z(3) maps to cell number 25 and PORT_Z(2) maps to cell number 28. In each of these numbered entries in the <boundary register stmt>, the <port ID> in each must match the mapped subscripted port. This could look like:

```
        ...
"22 (BC_1, PORT_Z(4), observe_only, X), " &
        ...
"25 (BC_1, PORT_Z(3), observe_only, X), " &
        ...
"28 (BC_1, PORT_Z(2), observe_only, X), " &
```

Rules p) through s) in 7.5.4.2 define the format for referencing a boundary scan register cell in either a segmented or unsegmented boundary scan register. If the boundary scan register is segmented, then the list of cells must be preceeded by the segment instance name and all of the cells in a single <entity pin info> element must be in the same boundary scan register segment.

Rules t) through x) of 7.5.4.2 define how port-link names, which must be defined in an AIO_Port_Behavior attribute in a User Package, are associated with ac pins in the component BSDL. The ac parameters for these ports are described in the User Package, and the port-link name is used to associate a set of parameters with each ac pin. The only ac parameter that may be defined with a port-link name is the threshold voltage, as it may be supplied from outside the IP. Port-link names may represent a single set of parameters (scalar) or may represent a vector of parameter sets. In any case, each ac pin is associated with one and only one set of parameters, while multiple ac pins may be associated with a single set of parameters, as when the IP was instantiated multiple times.

A note on hierarchy; when an IP is nested within another IP, to whatever depth, and then instantiated in the component, and all IP are provided with a User Package, then the <AC port link list> in the component BSDL will declare the package hierarchy directly to the level of the IP where the parameters of the driver

or test receiver are described, effectively skipping IP levels in between. These port-link names are simply placeholders, not signals or other connections that would be passed upward through each level of User Package hierarchy.

7.5.5 Attribute AIO_Port_Behavior (package)

The 2013 revision of the IEEE Std 1149.1 extended BSDL to allow the documentation in Package files of register segments inside intellectual property (IP) macros that would later be instantiated in components. This standard extends the documentation of ac pins to allow documentation of drivers and test receivers compliant with this standard in the IP macros by using the AIO_Port_Behavior attribute. As actual component port names are not known at the time that the IP is created and documented, the ac ports within the IP are given temporary names that will be used at the component level to link the component port name to the ac pin analog parameters documented in the IP Package file. These temporary, or port link names, are declared first in the attribute, and then the analog parameters of the ports are documented.

7.5.5.1 Syntax specification

The AIO_Port_Behavior attribute syntax is as follows:

<AIO port link behavior description> :: = **attribute AIO_Port_Behavior of** <package target> **is**
    <quote> <port link declaration list > <AC package info list> <quote> <semicolon>
<port link declaration list> ::= **port_link_list** <left paren> <port link declaration>
    { <semicolon> <port link declaration> } <right paren> <semicolon>
<port link declaration> ::= <port links> <colon> <port link type> <port dimension>
<port links> ::= <port link name> { <comma> <port link name> }
<port link name> ::= <VHDL identifier>
<port link type> ::= **in** | **buffer** | **out** | **inout**
<AC package info list> ::= <package pin info> { <semicolon> <package pin info> }
<package pin info> ::= <port link list> <colon> <AC parameter list>

NOTE—The definition of <AC parameter list> is given in 7.5.6.

<port link list> ::= <port link> { <comma> <port link> }
<port link> ::= <port link name> | <subscripted port link name> | <ranged port link name>

7.5.5.2 Rules

The AIO_Port_Behavior attribute rules are as follows:

a) The syntax for the optional <AIO port link behavior description> attribute shall be that shown in 7.5.5.1.

b) When a User Package is provided for an IP directly instantiating ac pins, then the AIO_Port_Behavior attribute shall be included and shall describe the ac parameters of each ac pin I/O driver or test receiver in the IP.

   NOTE—An IP can be instantiated without any BSDL Package documentation, but the responsibility for documenting the IP then rests with the provider of the component BSDL as if it were an integral part of the component design. See clause 7.5.4.

c) For all ac pins in an IP macro, the following shall apply:

   1) They shall be declared explicitly or implicitly in the <port link declaration list>

   2) They shall be assigned unique names, possibly within a **bit_vector**

   3) They shall be listed or implied in a <port link list>

NOTE—When a <port link name> of dimension **bit_vector** is listed in a <port link list> without a subscript or range, then all bits of that vector are implied to be part of the list in the order specified by the declared <range>, listing from left to right.

d) For each <port link name> appearing more than once in a <port links> of a <port link declaration>, all of the following shall apply:

1) It shall have a <port dimension> of **bit_vector**

2) The <range> of each appearance shall not overlap with other appearances

3) No gaps shall exist in the total sequence of all such appearances

4) The <range> of each appearance shall have the same direction (**to** or **downto**)

7.5.5.3 Permissions

Each <port link name> appearing more than once in a <port links> of a <port link declaration> may have a different <pin type> for each subrange listed in the <identifier list> and can occur in any order in the input.

7.5.5.4 Description

The <AIO port link behavior description> attribute is similar to the <AIO optional pin behavior description> attribute defined in 7.5.4. This attribute is required if the IP is documented in a User Package, and provides any ac pins, in which case all ac pins in the IP must be documented.

One or more port link names are first declared in a format similar to the logical port description of the component in the BSDL entity. As with the logical port description, a port link name can be declared with multiple, contiguous subranges, all with the same direction (**to** or **downto**). The various subranges can be the same or different types, though all declarations are limited to the **in**, **inout**, **out**, or **buffer** types.

These declared port link names are then associated with a set of analog parameters (see 7.5.6), and the port link names in the <port link list> can be single ports (a port link name of type **bit**, or a subscripted port link name of type **bit_vector**), or a collection of IP ports (a port link name of type **bit_vector** with or without a specified range).

This attribute is not hierarchical. That is, the <package pin info>element can only associate port link names with an <AC parameter list> (see 7.5.6) and not an <AC port link list>. Where the port-link names are used in the BSDL, a package name hierarchy will point directly to the User Package where the port-link name is declared and described.

7.5.6 AC parameters

AC parameters document the design characteristics of an ac pin driver or test receiver that are significant to the board test or diagnostic processes. They also allow documentation of characteristics that are programmable and that would need to be set during test initialization. See 7.7 for the PDL documentation of the actual programming procedures.

7.5.6.1 Syntax specification

In the syntax given below a <negative integer> is an <integer> preceded with a minus sign character '-,' without any space appearing between the sign and the number.

<AC parameter list> ::= [<LP time constant>] [<HP time constant>] <voltage parameters> [**No_pulse**]
<voltage parameters> ::= <voltage parm> <voltage parm> [<voltage parm> <voltage parm>]
<voltage parm> ::= <output VCM> | <output VPP> | <input threshold> | <input hysteresis>
<LP time constant> ::= <LP key> **<**equal> <time constant>
<LP key> ::= **LP_time** | **edge_detect_time**
<HP time constant> ::= <HP key> <equal> <time constant> [<cap spec>] [<detection modifier>]
<HP key> ::= **HP_Time** | **coupling_time**
<time constant> ::= <real>
<cap spec> ::= **On_Chip** | **On_Chip_Programmable | On_Chip_ac |**
On_Chip_AC_Programmable | <group on-chip>
<group on-chip> ::= <group keyword> <left paren> <group name> <right paren>
<group keyword> ::= **On_Chip_Bus_Programmable | On_Chip_AC_Bus_Programmable**
<detection modifier> ::= **Detect_EXTEST_High** | **Detect_EXTEST_Low** | **Detect_EXTEST_Both |**
Detect_EXTEST_Both_Grouped
<output VCM> ::= **AIO_VCM** <equal> <Volt choice>
<Volt choice> ::= <millivolts> **| programmable** | <group programmable>
<group programmable> ::= **bus_programmable** <left paren> <group name> <right paren>
<output VPP> ::= **AIO_VPP** <equal> <Volt choice>
<input threshold> ::= **AIO_VTH** <equal> <VTH choice>
<VTH choice> ::= <millivolts> | <source pin> | **programmable** | <group programmable> |
AIO_VCM | external
<input hysteresis> ::= **AIO_VHyst** <equal> <Volt choice>
<millivolts> ::= <integer> | <negative integer>
<source pin> ::= <port_ID>
<group name> ::= <VHDL identifier>

7.5.6.2 Rules

The ac parameter rules are as follows:

a) The syntax for the optional <AC parameter list> parameter documentation shall be that shown in 7.5.6.1.

b) The value of <time constant> shall be a positive, nonzero real number expressing the actual edge-detection and actual or minimum high-pass coupling time constant of the test receiver on an input pin, in units of seconds.

c) The value of <millivolts> shall be an integer expressing the nominal level-sensitive threshold (**AIO_VTH**) and edge sensitive hysteresis (**AIO_VHyst**) of the test receiver on an input pin, and the common-mode voltage (**AIO_VCM**) and minimum peak-to-peak voltage swing (**AIO_VPP**) of a driver on an output pin, in units of millivolts.

d) When an <AC port> element references a <port name> with <pin type> equal to "**in**" or "**inout**," then one or both <LP time constant> and <HP time constant> elements and the <input threshold> and <input hysteresis> elements shall be documented in the specified or linked <AC parameter list>, and these elements shall not be specified otherwise.

e) When an <AC port> element references a <port name> with <pin type> equal to "**out**", "**buffer**", or "**inout**," then the <output VCM> and <output VPP> elements shall be documented in the specified or linked <AC parameter list>, and these elements shall not be specified otherwise.

f) No more than one (1) occurrence each of <output VCM>, <output VPP>, <input threshold> or <input hysteresis> shall be specified as <voltage parameters> within a single <AC parameter list>.

g) When part 1) of rule a) in 6.2.3.1 is implemented for a port, then the <LP time constant> element shall be documented in the specified or linked <AC parameter list>; further, when part 2 of rule a) in 6.2.3.1 is implemented, the <LP time constant> element shall not be specified.

NOTE—When either of these options is implemented, the <LP time constant> is also used to document the on chip 'edge detection' time constants (whether high pass or low pass filtering was used) even when part 2) of rule a) in 6.2.3.1 is implemented.

h) When part 2) of rule a) in 6.2.2.1 is implemented for a port, then a <detection modifier> shall be documented in the specified or linked <HP time constant> element for that port; further, when part 1) of rule a) in 6.2.2.1 is implemented, then a <detection modifier> shall not be specified.

i) A <detection modifier> value indicates that a shorted board-level capacitor shall be detected during EXTEST by capturing a data value in the boundary cell associated with the test receiver, that meets one of the following situations:

1) A '1' is captured when **Detect_EXTEST_High** is specified and the boundary cell was pre-loaded with a '0.'

2) A '0' is captured when **Detect_EXTEST_Low** is specified and the boundary cell was preloaded with a '1.'

3) A '1' is captured when the boundary cell was preloaded with '0' and by a '0' when the boundary cell was preloaded with '1', and when **Detect_EXTEST_Both** is specified.

4) A '1' is captured when the boundary cells associated with both the representative and associated ports of a port-grouped differential pair were preloaded with '0' and by a '0' when those boundary cells were preloaded with '1', and when **Detect_EXTEST_Both_Grouped** is specified.

j) If the <time constant> value of an <HP time constant> element is based on an on-chip, inline capacitor, then a <cap spec> keyword shall be documented in the specified or linked <HP time constant> element.

k) A <cap spec> value of:

1) **On_chip_programmable** shall be documented only if the on-chip coupling capacitor can be bypassed with a TDR control field per <AC port> in the <entity pin info> element or <port link list> in the <package pin info> element.

2) **On_chip_bus_programmable** shall be documented only if the on-chip coupling capacitor can be bypassed with a TDR control field where the TDR control field controls all ports listed in the <entity pin info> or <package pin info> element.

3) **On_chip_ac** shall be documented only if the on-chip coupling capacitor is bypassed when the EXTEST instruction is active and not by a TDR control field.

4) **On_chip_AC_programmable** shall be documented only if the on-chip coupling capacitor is bypassed by the EXTEST instruction being active exclusive ORed with a TDR control field bit per <AC port> in the <entity pin info> element or <port link list> in the <package pin info> element.

5) **On_chip_AC_bus_programmable** shall be documented only if the on-chip coupling capacitor can be bypassed by the EXTEST instruction being active exclusive ORed with a TDR control field bit where the TDR control field controls all ports listed in the <entity pin info> or <package pin info> element.

6) **On_chip** shall be documented only if the on-chip coupling capacitor is not bypassable by any control accessible through the TAP, and this value shall be permitted but deprecated.

NOTE—A <cap spec> value containing the word "**programmable**" requires a PDL routine to control the capacitor bypass. See rule j) in clause 7.7.2.

l) The <VTH choice> value of **external** shall be:

1) Documented only within a <package pin info> element.

2) Specified only when the threshold voltage is provided externally to the IP.

3) Resolved by the inclusion of a <VTH choice> element in the <port link reference> element of an <AC entity info list>

m) The <VTH choice> value of **AIO_VCM** shall only be documented for the <input threshold> of a port of type **inout** when <output VCM> is also documented within the given <AC parameter list>, and the input threshold is always the same as the common-mode voltage.

n) A <VTH choice> value of <source pin> shall be documented only within an <entity pin info> element, and shall be the <port ID> of a system pin supplying the level-sensitive threshold voltage to the test receiver.

NOTE—This pin can have any input or analog <pin type> including any of the "**linkage_**" or "**power_**" pin types defined in clause B.8.3 of IEEE Std 1149.1-2013. The value of <port ID> must reference a bit-dimensioned port or a <subscripted port name> for a bit_vector-dimensioned port, as given in IEEE Std 1149.1-2013 in B.6.2.1, rules a) and c) respectively.

o) The <VTH choice> and <Volt choice> value of **bus_programmable** shall be specified when all ports in the port list have the respective value set in unison by a TDR field, and the value of **programmable** shall be specified when each port in the port list has its respective value set by separate TDR field.

NOTE—It can happen that all ports programmed by a single TDR field are not in a single port list. This will happen any time that any other parameter for these ports is not identical. In this case, the full list of ports programmed by a single TDR field is determined by listing all ac ports having the same <group name>. The lists for different parameters could be different.

p) When a single TDR field controls multiple ac ports, each unique <group name> shall associate all ac ports controlled by a single TDR register field.

q) Each <group name> shall be unique within a <user package body> or <BSDL description>.

r) A <millivolts> value for an <output VPP> or an <input hysteresis> specification shall be greater than 0.

s) When the driver or test receiver of an ac pin will provide significantly better test performance with the multiple pulses generated by the EXTEST_TRAIN instruction instead of the single pulse generated by the EXTEST_PULSE instruction, then the **No_Pulse** keyword shall be documented for that port, and further, the AIO_EXTEST_TRAIN_EXECUTION attribute shall be provided.

7.5.6.3 Recommendations

The ac parameter recommendations are as follows:

a) The <HP time constant> phrase of an <AC parameter list> element should be documented in all cases where it is permitted.

b) Whenever an AIO port is provided with a programmable ac parameter, the REGISTER_MNEMONICS, REGISTER_FIELDS, REGISTER_ASSEMBLY, and REGISTER_ASSOCIATION attributes should be documented where appropriate.

7.5.6.4 Permissions

The ac parameter permissions are as follows:

A test receiver with a programmable <VTH choice> may be designed to comply with both options of rule a) in 6.2.2.1 as follows:

a) By providing one or more programmable values in the middle of the input range of the test receiver in support of normal dc test (EXTEST), per option 1).

b) By additionally providing programmable values at either or both limits of the input range of the test receiver in support of a continuity test per option 2).

NOTE—The continuity test allowed above would be the same as if a <detection modifier> had been coded; **Detect_EXTEST_Low** if a value at the most positive end of the test receiver input range is programmed, or **Detect_EXTEST_High** if a value at the most negative end of the test receiver input range is programmed. The programmed <VTH choice> would have to be set separately for ac test and for dc test, in these cases.

7.5.6.5 Description

The rules of 7.5.6.2 govern the documentation of the analog parameters of the advanced I/O channel that are significant for test and diagnostics. Most of these are parameters of the design of the test receiver, but others are characteristics of an output driver that are important to the ability of the test receiver to detect ac test signals.

Rules b), d), and g) through k) in 7.5.6.2 and recommendation a) in 7.5.6.3 govern the documentation of the nominal edge-detection and nominal or minimum high-pass coupling time constants for a test receiver. Included with the high-pass coupling time constant are documentation of optional items including whether there is an on-chip coupling capacitor and any special EXTEST modes for detecting shorted board-level capacitors. This allows software:

— to know which options from rules a) of 6.2.2.1 and 6.2.3.1 are implemented,

— to know if a device is internally ac-coupled,

— to know if a device must be externally ac-coupled,

— to verify that the board ac coupling meets test receiver requirements,

— to compute TCK frequency and/or time spent in the *Run-Test/Idle* TAP Controller state during board test (see A.3.4.1.3 for discussion of T_{Test} time calculations), and

— to determine any special coding of EXTEST values for a shorted board-level capacitor test.

Rule d) in 7.5.6.2 requires that all pins with test receivers have documentation of their ac coupling and edge-detection filters, when they exist.

It is recommended [see 7.5.6.3, recommendation a)] that the <HP time constant> always be included. The optional <cap spec> and <detection modifier> elements of the <HP time constant>, and optional <LP time constant>, affect the interface options that can be employed on a board-level interface. The presence of any of the <cap spec> keywords implies that a device test receiver includes a series-connected ac coupling capacitor. The value of the on-chip coupling capacitor is always used in the computation of the <HP time constant>.

When there is also a coupling capacitor on the board, then any on-chip filter must be documented as an <LP time constant> and the <cap spec> is not used. The <HP time constant> is then used to document the minimum value required for the on-board coupling capacitor.

As seen in the discussion in Clause 4 and Clause 6, this standard was developed around the idea of an edge-detection low-pass filter in the negative leg of the comparator in the test receiver, plus a high pass coupling filter between the driver at one end and the receiver and test receiver at the other end of the net. The actual time constants were documented using the simple "**HP_time**" and "**LP_time**" keywords. Developments in the meantime have taken advantage of the fact that the edge detection function can also be performed with a high pass filter, and designs with two high pass filters in series have been used, making those simple terms confusing. So the longer equivalent, but more accurate, keywords "**edge_detect_time**" and "**coupling_time**" have been added to remove any confusion.

NOTE 1—If a <cap spec> modifier is included, the <HP time constant> must be included due to syntactical requirements.

NOTE 2—The term **HP_time** is equivalent to **edge_detect_time**, and the term **LP_time** is equivalent to **coupling_time**.

Either the absence of a <cap spec> keyword or the inclusion of the <LP time constant> allows either ac or dc coupling for a board implementation. If, however, a <cap spec> keyword modifier and <LP time constant> are both omitted, then board-level ac coupling is required.

The following situations describe ac coupling and edge-detection filter possibilities:

a) The component pin test receiver contains an edge-detection filter and this pin can be ac or dc-coupled to a driver as shown in Figure 50. In this case, the <AC parameter list> must contain a <LP time constant> phrase, and would normally contain a <HP time constant> phrase to allow verification software to check the board design. The lack of a <cap spec> keyword indicates that the <HP time constant> phrase is a minimum value for board checking:

"PinName:LP_time=0.75e-8 HP_time=1.5e-8"

b) The component pin test receiver has neither an edge-detection filter, nor an on-chip coupling capacitor, as shown in Figure 51, then this pin requires external ac coupling on the board. Board rule-checking software will need to verify both the existence of the coupling and also that its time constant exceeds that specified for the <HP time constant> (which is a minimum). In this case, the <AC parameter list> would not contain an <LP time constant> element, nor a <cap spec> element.

"PinName:HP_time=1.5e-8"

c) The component pin test receiver has no edge-detection filter but does have an on-chip coupling capacitor as shown in Option 4a of Figure 52, then the <AC parameter list> element would contain an <HP time constant> element with the <cap spec> modifier, and the time constant value would be the nominal value for the on-chip coupling filter:

"PinName:HP_time=1.5e-8 On_chip_programmable"

d) The component pin test receiver has a high-pass edge-detection filter, as shown in Option 4b of Figure 52 and Option 5 of Figure 53, then the <AC parameter list> must contain an <LP time constant> phrase, and would normally contain a <HP time constant> phrase to allow verification software to check the board design. This is equivalent to Options 1 and 2 of Figure 50, above:

e) "PinName:LP_time=0.75e-8 HP_time=1.5e-8"

f) The component pin test receiver has both a edge-detection filter and an on-chip coupling capacitor, then the <AC parameter list> element would contain both an <LP time constant> element and a <HP time constant> element with the <cap spec> modifier:

"PinName:LP_time=0.75e-8 HP_time=1.5e-8 On_Chip_Programmable"

The <cap spec> keywords document that there is a series coupling capacitor between the component pin and both the test and mission receivers. These keywords are defined as follows:

— **On_Chip**: This capacitor cannot be bypassed by any means accessible through the TAP. **Deprecated**: This keyword value could be removed in a future revision of the standard, and is included primarily to permit documenting existing components with the BSDL extentions defined in this version of the standard.

— **On_Chip_AC**: This capacitor will be bypassed in test mode any time the EXTEST instruction is active, and otherwise is not bypassable by other means accessible through the TAP.

— **On_Chip_Programmable**: This capacitor is bypassable by seting a field in a TDR, preferably the init_data TDR, and otherwise is not bypassable by other means accessible through the TAP.

— **On_Chip_AC_Programmable**: This capacitor is bypassable and the bypass is controlled by the exclusive-OR of the EXTEST instruction decode and a field in a TDR. Essentially, this is the same as **On_Chip_AC**, except that the bypass could be toggled by setting the TDR field to '1.' This allows override of the bypass when EXTEST is active, and also allows bypassing the capacitor when either of the EXTEST_PULSE or EXTEST_TRAIN instructions are active. See Figure 65 for a simple test-only logic example that illustrates this option. Mission-mode control of the coupling capacitor bypass could be added in a number of ways, if desired.

— **On_Chip_Bus_Programmable** or **On_Chip_AC_Bus_Programmable**: The same as **On_Chip_Programmable** or **On_Chip_AC_Programmable**, respectively, except that a group name is provided with the keyword and all ac ports associated with the same group name are programmed by a single TDR control field. For example, if there was a single control bit exclusive-ORed with the EXTEST instruction decode that controlled the bypass of all of the on-chip capacitors in the chip, then all AIO with on-chip capacitors would have the **On_Chip_AC_Bus_Programmable** keyword with the same group name in their HP time constant phrase.

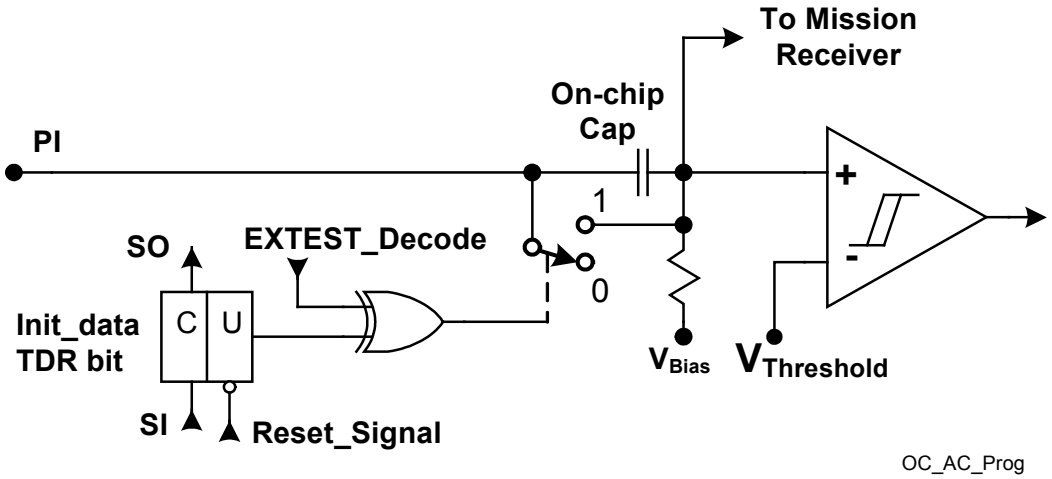


Figure 65 An on-chip capacitor bypassed by EXTEST but overriden by a TDR bit

NOTE—In Figure 65, the signal labeled "EXTEST_Decode" has a value of '1' when the EXTEST instruction is active, and is '0' at all other times.

The four keyword values with the suffix string **"_Programmable**" in them require a PDL procedure to document how the bypass is programmed. See 7.7.

The reason that the **On_Chip** value of the <cap spec> modifier is deprecated is that there are situations in board test where defects cannot be detected if the capacitor is not bypassed. For example, if an on-chip coupling capacitor is documented, and an on-board capacitor is also present in the connected net, then the on-board capacitor cannot be tested for a shorted capacitor. Another example exists where a component input is driven from a board connector and there is no on-board capacitor. Very few ATE can provide the correct ac waveform to the board for ac testing such nets, so such nets could be tested using the EXTEST instruction with the capacitor bypassed and treating the independent test receivers as single ended receivers.

The <detection modifier> keywords are required if and only if option 2) of rule a) of 6.2.2.1 is exercised, which requires that the pin be ac-coupled, and the keywords only have meaning when the EXTEST instruction is active. For example, any of these HP_Time phrases could appear:

```
HP_time=0.1e-6 Detect_EXTEST_High
HP_time=0.1e-6 Detect_EXTEST_Low
HP_time=0.1e-6 Detect_EXTEST_Both
HP_time=0.1e-6 Detect_EXTEST_Both_Grouped
```

Detect_EXTEST_High indicates that for a bad capacitor (shorted), the short will be captured as a logic ‘1’ at the output, overwriting the preloaded ‘0,’ when an input high (V_{ih}) stimulus is applied by EXTEST. No predicted expected value is possible when an input low (V_{il}) stimulus is applied. See Figure 54 for a conceptual example circuit.

Detect_EXTEST_Low indicates that for a bad capacitor (shorted), the short will be captured as a logic ‘0’ at the output, overwriting the preloaded ‘1,’ when an input low (V_{il}) stimulus is applied by EXTEST. No predicted expected value is possible when an input high (V_{ih}) stimulus is applied.

Detect_EXTEST_Both indicates that for a bad capacitor (shorted), the short will be captured as a logic 1 if the boundary cell associated with the port is preloaded with a ‘0’ and an input high (V_{ih}) stimulus is applied by EXTEST, and be captured as a logic ‘0’ if the boundary cell associated with the port is preloaded with a ‘1’ and an input low (V_{il}) stimulus is applied by EXTEST. See Figure 55 for a conceptual example circuit. Note that the associated boundary scan cell provides both the test receiver initialization data and a control signal to select the threshold voltage.

Detect_EXTEST_Both_Grouped indicates that both the representative and associated ports of a grouped differential pair share the same threshold and bias voltages, and the same values must be loaded into both boundary cells associated with the two test receivers. Otherwise, this is the same as the **Detect_EXTEST_Both** specification. See Figure 56 for a conceptual example circuit.

The default behavior, if none of these keywords is specified, is the level-sensitive behavior defined by part 1) of rule a) of 6.2.2.1. That is, the value captured in a shorted board-level capacitor test will be dependent on the driver common-mode voltage, the receiver threshold voltage, and the value driven.

All of these detection modifier keywords describe test receiver structures that provide a more robust shorted capacitor test when board coupling is required, but the board environment of the component is otherwise unknown. The driver common-mode voltage may be too low or too high for EXTEST communication through a shorted capacitor.

In the case of a structure described by the **Detect_EXTEST_Both** and **Detect_EXTEST_Both_Grouped** keywords, it is tempting to think of the behavior as equivalent to EXTEST, since the switching between high and low level detection can be done dynamically on a scan-by-scan basis. However, the driver and receiver are not communicating digital values, they are running continuity tests where not every combination of driver preload and test receiver preload will produce a valid, predictable, captured value. A

particular test vector may produce a valid result on the positive leg of a differential pair, but not the negative leg, for instance.

However, having both detection levels (high and low) available improves the flexibility and testability of the component, though the test may only use the appropriate high or low detection levels as dictated by the board environment. In the case that the driver common-mode voltage is well within the operating range of the test receiver, then a structure described by the **Detect_EXTEST_Both** or **Detect_EXTEST_Both_Grouped** keywords may allow a test that uses both levels and mimics EXTEST, but not all captured data is valid. The test vectors would need to take this into account.

The most flexible solution to possible common-mode voltage incompatibility in level-sensitive tests is a programmable test receiver threshold voltage, offering one or more settings supporting conventional EXTEST, plus the values supporting both high and low continuity test, as described in Permission a) of 7.5.6.4. The most appropriate value could then be chosen based on a given board configuration.

It is permitted to build the test receiver so that both options of rule a) in 6.2.2.1 are supported. The AIO_VTH parameter would have to be programmable, and the possible values would have to include both mid-range values suitable for conventional EXTEST, as well as either or both limits of the input range suitable for a continuity test. No <detection modifier> would be provided, implying that normal EXTEST is the default (and probably preferred, if it will work) test. If one of the mid-range values is programmed, then the tests would be performed as a normal EXTEST, per option 1), and the same programmable value could be used for both ac and dc tests. If a value at the test receiver input limits is programmed, then the test would be performed as if the appropriate <detection modifier> had been provided: **Detect_EXTEST_Low** for a value at the high end of the test receiver input range; and **Detect_EXTEST_High** for a value at the low end of the test receiver input range. The programmable value for dc test would have to be programmed separately from the value for the ac test, as ac test will not work with a threshold voltage at the limit of the test receiver input range. This would allow the decision of whether to attempt EXTEST to detect a shorted capacitor or to use the continuity test to be made at board test generation time based on the board configuration.

In addition to the time constants, rules d) and e) in 7.5.6.2 require certain analog voltage levels to be documented. In order to properly interpret the results of a level-sensitive test (such as EXTEST), particularly when dc-coupled, input thresholds of the test receivers (**AIO_VTH**) and output common-mode voltage of the drivers (**AIO_VCM**) must be specified. Also, in order to properly interpret the results of an edge-sensitive test (such as EXTEST_PULSE), the input hysteresis voltage (**AIO_VHyst**) of the test receivers and the minimum output peak-to-peak voltage swing (**AIO_VPP**) of the drivers must be specified. In both cases, the preferred specification would be that the values are **programmable**, in which case specific PDL procedures must be supplied. This allows test software to set up tests in ways that help produce predictable tests. Otherwise, a fixed voltage in integer millivolts, per rule c) in 7.5.6.2, would be specified. For example, a bidirectional port might be specified as:

```
HP_time=0.1e-6 AIO_VCM=850 AIO_VPP=350 AIO_VTH=825 AIO_VHyst=180
```

It is worth taking a moment to inspect this set of voltage parameters. If it is assumed that, on a board, this bi-directional channel is connected to another bi-directional channel with the same specifications, then the robustness of both ac and dc tests can be analyzed.

There is no <detection modifier> specified, so option 1) of rule 6.2.2.1 a) is indicated, meaning that this port will attempt the classical, level-sensitive EXTEST. Comparing the AIO_VCM of the driver (850 mV) and the AIO_VTH of the test receiver (825 mV), they are not identical, but close. The receiver AIO_VHyst is specified as 180 mV, meaning the V_{Hyst_Level} for the test receiver is 90 mV (half of V_{Hyst_Edge}), and so the difference between AIO_VCM and AIO_VTH is still well within the hysteretic zone, allowing proper EXTEST operation as defined in this standard, though with 25 mV loss of noise margin.

For the ac test, we can compare the AIO_VPP of the driver (350 mV) to the AIO_VHyst of the receiver (180 mV). Assuming minimal dc attenuation of the signal on the board, the hysteresis appears to be within the required range of 50% to 90% of the expected ΔV_{Min} at the receiver, so ac test should operate as defined in this standard. There even is leeway for attenuation of the driver signal at the receiver input down to a ΔV_{Min} of 200 mV, making the test very robust.

If some or all of these parameters were programmable, then values for these parameters could be set at board test time to help ensure a valid test. Even if there was no way to set compatible VCM and VTH levels for a channel, it would be possible to perform the shorted capacitor test if VTH could be set to a voltage rail, to operate dc test in the continuity check mode of option 2) of rule a) of 6.2.2.1rather than classic EXTEST mode of option 1) of rule a) of 6.2.2.1.

In some protocols, the input threshold is supplied from off-chip, and in that case rule n) of 7.5.6.2 requires that the port ID of the pin receiving the threshold voltage would be specified instead of the actual voltage. This can only be done at the component level where port IDs are declared. For example, an input port with an externally supplied threshold voltage might be specified (where HSI_IN2 is the port name of the externally supplied threshold) as:

```
HP_time=0.1e-6 AIO_VTH=HSI_IN2 AIO_VHyst=150
```

In an IP macro documented in a user package, the input threshold should be documented as **external** per rule l) of 7.5.6.2, meaning that it will be supplied from outside the IP. At the component level, such an input threshold must then be documented per rule u) of 7.5.4.2 with either a fixed value, a programmable value, or as coming from a port ID. See examples 6 and 7 in 7.5.7 for one use of this keyword.

In some protocols, and in particular for bi-directional AIO ports, the driver common-mode voltage and the test receiver threshold are always the same, by design, even when programmable. In this case, rule m) of 7.5.6.2 allows the input threshold to be documented as equal to the common-mode voltage using the value **AIO_VCM**. Such a bidirectional port might be specified as:

```
HP_time=0.1e-6 AIO_VCM=850 AIO_VPP=350 AIO_VTH= AIO_VCM AIO_VHyst=150
```

Any of these voltage parameters could be programmable using TDR fields. In some cases, each port is independently programmable, having separate TDR fields for each, and in other cases all of a set of ports are programmed using a single TDR field. Rule o) of 7.5.6.2 requires the use of the value **programmable** in the first case, and the value **bus_programmable** with a group name in the latter case. See examples 6 and 7 in 7.5.7 for an example of the use of these keywords.

Both the EXTEST_PULSE and EXTEST_TRAIN instructions are assumed to work as specified in this standard. The "**No_pulse**" keyword allows the specification that the EXTEST_PULSE instruction might not work in all situations, and that the EXTEST_TRAIN instruction would be the preferred instruction for the listed ports. EXTEST_TRAIN places requirements on the test equipment (specifically, the ability to vary TCK frequency during the *Run-Test/Idle* TAP controller state in order to control T_{Test}), which cannot always be met, so EXTEST_PULSE is normally the preferred instruction. However, the I/O circuitry might not be fully static, and the delay between transitions can be quite long when scanning the boundary-scan register. The undesired dynamic behavior could be due to the driver or test receiver being implemented with dynamic design techniques, or the full board net configuration might include some unintended leakage that forces circuit values away from the normal operating point. In any case, restoration of the normal operating point might require multiple transitions, and this keyword allows identification of that situation. See 5.2.2 for additional explanation. Whenever the **No_Pulse** keyword is specified, then the optional EXTEST_TRAIN_EXECUTION attribute must be specified, as well.

7.5.7 AIO_Pin_Behavior examples

Example 1:

A component has four outputs (A, B, C, D) with no ac/dc select cells and four outputs (E, F, G, H) with a common ac/dc select cell which is cell 13 of a boundary-scan register segment named "quad4."

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "A, B, C, D: AIO_VCM=510 AIO_VPP=250;" &
  "E, F, G, H: AC_Select=quad4:13 AIO_VCM=480 AIO_VPP=220" ;
```

Example 2:

A component has ac input test receivers on four inputs (I, J, K, L) and each has identical time constants. This component also has two other inputs (M and N) with different time constants, and with an on-chip capacitor. Inputs M and N might not have proper EXTEST_PULSE behavior in all cases, so the "**No_pulse**" keyword is included.

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "I, J, K, L: LP_Time = 5.0e-9 HP_Time = 15.0e-9 AIO_VTH=500 "&
            " AIO_VHyst=90;"&
  "M, N: HP_Time = 100.0e-9 On_Chip_ac AIO_VTH=750 AIO_VHyst=50 "&
       " No_pulse" ;
```

Example 3:

A component has two ac input pins: input P that is associated with two cells with cell 37 of a non-segmented boundary scan register being the one that observes the pin's test receiver, and input Q that is associated with only a single cell. Both test receivers have the same time constants and other parameters.

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "P[37], Q: Edge_Detect_Time = 5.0e-9 Coupling_Time = 15.0e-9 "&
  "AIO_VTH=500 AIO_VHyst=90" ;
```

Example 4:

A component has four ac input pins described as a bit_vector (1 to 4) of four elements named "Data." Each member has a test receiver observed by input cells 11, 12, 13, and 16 respectively (while the mission receivers have their own input cells). Each test receiver has identical time constants.

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "Data[11,12,13,16]:Edge_Detect_Time=5.0e-9 HP_Time=15.0e-9" &
  "AIO_VTH=500 AIO_VHyst=90";
```

This same configuration could be described less compactly (but more easily by simple software) as:

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "Data(1)[11]: Edge_Detect_Time=5.0e-9 HP_Time=15.0e-9 " &
              "AIO_VTH=500 "AIO_VHyst=90; " &
  "Data(2)[12]: Edge_Detect_Time=5.0e-9 HP_Time=15.0e-9 " &
              "AIO_VTH=500 "AIO_VHyst=90; " &
  "Data(3)[13]: Edge_Detect_Time=5.0e-9 HP_Time=15.0e-9 " &
              "AIO_VTH=500 "AIO_VHyst=90; " &
  "Data(4)[16]: Edge_Detect_Time=5.0e-9 HP_Time=15.0e-9 " &
              "AIO_VTH=500 "AIO_VHyst=90 " ;
```

Finally, this same example could be expressed this way:

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "Data(1)[11], Data(2)[12], Data(3)[13], Data(4)[16] : " &
  " Edge_Detect_Time=5.0e-9, HP_Time=15.0e-9 " &
  " AIO_VTH=500 AIO_VHyst=90" ;
```

Example 5:

A device with four bidirectional ac pins associated with a bit_vector (1 to 4) named "Bidi," all having identical test receivers, each with a driver having a shared ac/dc select cell 44, and each being associated with multiple input cells, of which cells 103, 105, 107, and 98 monitor the test receivers. The driver common-mode voltage is programmable, with separate programming fields for each ac pin, and the test receiver threshold voltage is always matched to the driver common mode voltage. This could be described as:

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "Bidi[103,105,107,98]:AC_Select=44 LP_Time=5.0e-9 HP_Time=15.0e-9"&
  "  AIO_VCM=programmable AIO_VPP=250 AIO_VTH=AIO_VCM AIO_VHyst=90 ";
```

Example 6:

An IP macro contains one transmitter and one receiver, named Lane(1) and Lane(0), respectively. They are differential, though only one port link name is declared for each. The "Lane" is intended to be used as one of multiple communication lanes in a component. The driver has a programmable common-mode voltage, and the test receiver has its threshold voltage supplied from outside the IP. The ports are required by the protocol to be ac-coupled and take option 2 of rules a) in 6.2.2.1 and 6.2.3.1, and the IP provider has built-in a bypassable on-chip capacitor for the receiver. Note that the IP provider will also need to supply PDL routines for setting the on-chip capacitor bypass and the driver common-mode voltage. The AIO could be documented in the package file as:

```
attribute AIO_Port_Behavior of ACPKG : package is
 "port_link_list (Lane : in bit_vector (0 to 0) ; "&
 "                Lane : out bit_vector (1 to 1) ) ; "&
 "Lane(0):Coupling_Time=15.0e-9 on_chip_programmable " &
 "        Detect_EXTEST_High AIO_VTH=external AIO_VHyst=90;"&
 "Lane(1):AIO_VCM=programmable AIO_VPP=350 ";
```

Example 7:

A component using the IP from Example 6 to provide four lanes of communication, declared as TX_Nibble(3 downto 0) and RX_Nibble(3 downto 0), with no select cell on the driver. In addition, there is a separate differential port of type inout, dimension bit, and called Cmd_Status, which might be ac or dc-coupled. This could be documented as:

```
attribute AIO_Pin_Behavior of ACDEV : entity is
  "TX_Nibble  : package ACPKG:Lane(1) ; "&
  "RX_Nibble  : package ACPKG:Lane(0) AIO_VTH=bus_programmable ; "&
  "cmd_status : Edge_Detect_Time=5.0e-9 Coupling_Time=15.0e-9 "&
  "  AIO_VCM=510 AIO_VPP=250 AIO_VTH=500 AIO_VHyst=90 ";
```

Note that, at the component level, the threshold voltage of the lane receivers is documented as bus_programmable, meaning that a single TDR field will program all test receivers on RX_Nibble pins. The component supplier therefore must supply the PDL routine to control the threshold voltage, even though the test receivers are inside an IP macro. Note also, that TX_Nibble and RX_Nibble names are used without subscript or range. That means the entire vector, as declared in the logical port list, is used. So all

four bits of those two vectors, and both representative and associated ports, get the analog parameters defined for the appropriate Lane bit.

7.6 Example BSDL

The first BSDL example illustrates several possible Advanced I/O input, output, and BIDI configurations. The second BSDL example is a typical small component, a serialize/deserialize (SERDES) device with differential parallel and serial ports. First, a series of graphic symbols is introduced, needed to understand the examples.

7.6.1 Symbology

IEEE Std 1149.1-2013, B.8.14, Figures B.5 and B.6, defined the simplified boundary-scan register symbols (repeated below in Figure 66). Each supports some subset of the signals boundary-scan in and out (SI, SO), Parallel in and out (PI, PO), and either Preset (at the top of the Update latch) or Clear (at the bottom of the Update latch). The Preset/Clear capabilities are asynchronous and typically used for output enable control cells that are initialized to non-controlling states during reset operations (e.g., Test-Logic-Reset). The numeric value at the top of the capture cell (00 in each example in Figure 66) is the cell number in the Boundary-Scan Register description or Boundary-Scan Register segment description. These symbols are intended to illustrate the connections defined in the BSDL, so global signals that go to a large subset of the boundary register cells are not shown but understood to exist.

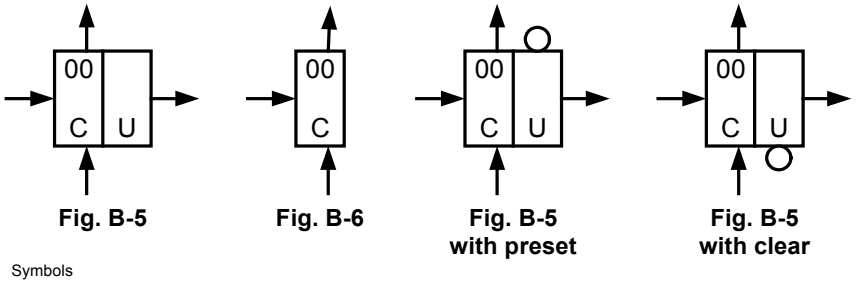


Figure 66—Symbols adapted from IEEE Std 1149.1-2013, B.8.14

These symbols covered the cases of a boundary register cell without an Update latch (Figure B.6), a normal cell with an Update latch (Figure B.5), and cells with an Update latch with either a “preset” or “clear” input (Figure B.5 with preset and B.5 with clear). These symbols covered most of the cases in the subsequent Figures B-9 and B-10, but the symbology was extended in common sense ways, without comment, to cover boundary register cells with a function of BIDIR, which must have a PI/PO signal pair for both the input and output directions. A further extension of this symbology is shown in Figure 67.

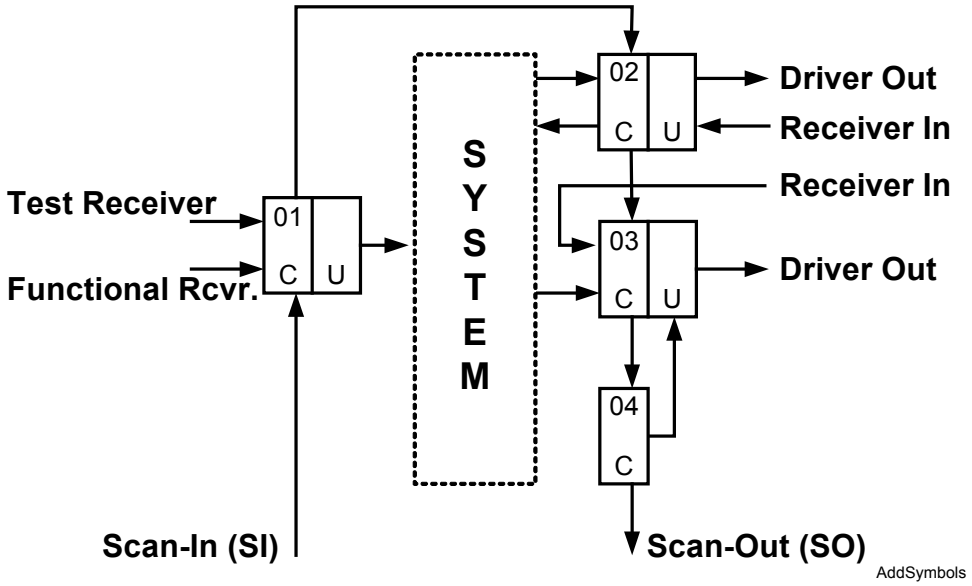


Figure 67—Additional symbols for boundary-scan register cells

Cell 01 is a cell merging an INPUT type boundary cell with an OBSERVE_ONLY boundary-scan register cell, and is only used in combination with both a functional receiver and a test receiver. In this type of cell, during test the test receiver output is captured and the Update cell drives the cell output. During functional operation, the functional receiver output is passed to the cell output.

Cell 02 is the boundary-scan register cell with function BIDIR. Cell 03 is a self-monitoring boundary-scan register cell (basically an OUTPUT2 or OUTPUT3 cell merged with an OBSERVE_ONLY cell) used only in combination with an output driver. Note that both of these cells also require an input from their control cell for operation, but these inputs are not shown explicitly for simplicity (the pairing of the CONTROL type cell with the OUTPUT or BIDIR cell is already clearly shown in the diagrams and explicit in the BSDL).

Cell 04 is an ac/dc selection cell, having only a shift stage and no capture input. Its parallel output is connected to the bottom of the Update portion of other boundary register cells used in output or bidirectional applications, as shown here for cell 03. This connection location conflicts with the IEEE Std 1149.1 symbology, which uses this position for the “Clear” input. The difference should be obvious in use since “Preset” and “Clear” inputs are TAP global signals (and usually represented by the presence of a “bubble” without any line connecting to it) where ac/dc selection must always come from another boundary-scan register cell.

7.6.2 Boundary-scan configuration descriptions

Figure 68 and Figure 69 show several possible configurations of input, output, and bidirectional I/O and boundary-scan configurations. All input and bidirectional ports have both mission and test receivers.

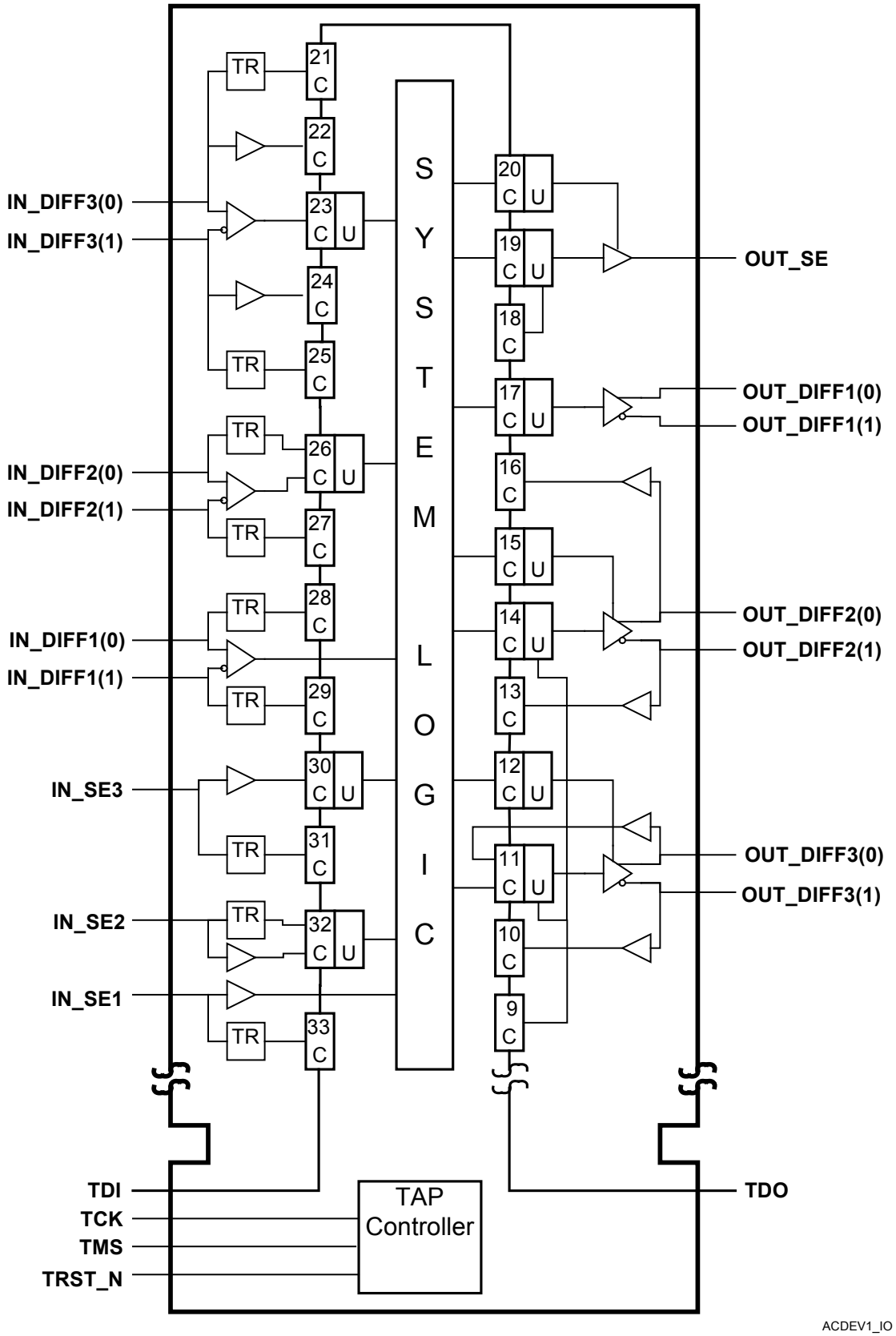


Figure 68—INPUT and OUTPUT boundary-scan structures

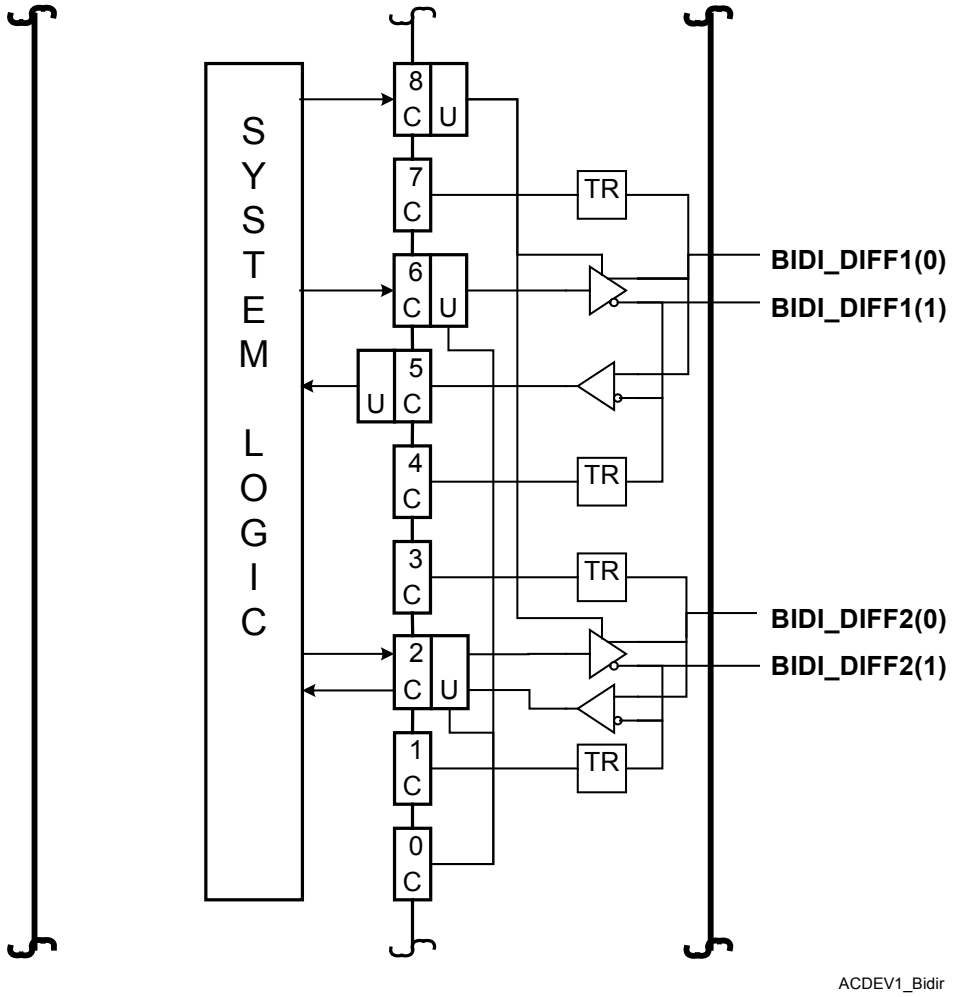


Figure 69—Bidirectional boundary-scan configurations

Starting from the TDI input and following the boundary-scan chain around to TDO, we have the following configurations:

— Cell 33 shows a single-ended input with the test receiver observed by a BC_4 cell, and there is no control on the mission receiver output. In the BSDL, the single OBSERVE_ONLY cell is coded for the port (this is also shown in Figure 25).

— Cell 32 shows a single-ended input with the test receiver observed and the mission receiver output controlled by a BC_1 type cell, as shown in Figure 57. In the BSDL, the single INPUT cell is coded for the port.

— Cells 31 and 30 show a single ended input with the test receiver observed by a BC_4 cell, and the mission output observed and controlled by a BC_1 cell. This situation is differentiated from the previous two in the BSDL by the presence of both INPUT and OBSERVE_ONLY entries for the same port.

— Cells 29 and 28 show the simplest differential input configuration: separate BC_4 type cells observe the two test receivers, and there is no observation or control on the mission receiver. In the BSDL, an OBSERVE_ONLY cell is coded for each leg of the differential pair (this is also shown in Figure 26).

— Cells 27 and 26 show a differential input with the test receiver on the representative (positive) port observed and the mission receiver output controlled by a BC_1 INPUT type cell, and the test

receiver of the associated (negative) port observed by a BC_4 OBSERVE_ONLY type cell. The con- trol of the output of the mission receiver is similar to Cell 32 on the single-ended input.

— Cells 25, 23, and 21 show a differential input with both test receivers observed by BC_4 OBSERVE_ONLY cells, and the mission receiver output observed and controlled by a separate BC_1 INPUT cell. In addition, cells 24 and 22 are redundant observe-only cells observing some test function (perhaps input not connected). Both the BC_1 INPUT (cell 23) and two BC_4 OBSERVE_ONLY cells (22 and 21) are coded for the representative (positive) port, and two BC_4 OBSERVE_ONLY cells (24 and 25) are coded for the associated (negative) port of the differential pair. Since there are multiple cells coded for the associated (negative) port, this port must be included in the port list of the AIO_PIN_BEHAVIOR attribute in order to specify which cell observes the test receiver.

— Cells 20, 19, and 18 show an unobserved single ended three-state output. A conventional BC_1 type CONTROL cell, an AC_1 type OUTPUT3 cell, and an AC_SELU INTERNAL cell are coded in the BSDL providing enable, data, and ac/dc selection, respectively, for the port (this is also shown in Figure 22).

— Cell 17 shows a simple differential output with a single AC_1 type OUTPUT2 cell coded in the BSDL for the representative (positive) port.

— Cells 16 through 13 show a self-monitoring, three-state differential output. An AC_1 type OUTPUT3 cell (14) connected to a BC_1 type CONTROL cell (15), and a BC_4 type OBSERVE ONLY cell (16) are coded for the representative (positive) port and a BC_4 type OBSERVE_ONLY cell (13) is coded for the associated (negative) port. AC/DC selection is provided by a shared cell (9). Since the observing receivers are intended only for self-monitoring, not bidirectional use, they are simple receivers with thresholds fixed at the common-mode voltage of the differential driver.

— Cells 12 through 10 show another form of a self-monitoring, three-state differential output. An AC_9 self-monitoring type OUTPUT3 cell (11) is coded for the representative (positive) port and connected to a BC_1 type CONTROL cell (12), and a BC_4 type OBSERVE_ONLY cell (10) is coded for the associated (negative) port. AC/DC selection is provided by a shared cell (9). Again, the observing receivers are simple receivers with thresholds fixed at the common-mode voltage of the differential driver.

— Cell 9 provides the shared ac/dc selection for the previous two outputs using an AC_SELU cell.

— Cell 8 provides a shared enable for both bidirectional differential ports using a BC_1 cell.

— Cells 7 through 4 show an unmerged bidirectional differential port. A BC_4 type OBSERVE_ONLY, an AC_1 type OUTPUT3, and a BC_1 type INPUT cell are all coded for the representative (positive) port of the differential output, and a BC_4 type OBSERVE_ONLY cell is coded for the associated (negative) port. Driver enable and ac/dc selection are provided by shared cells 8 and 0, respectively.

— Cells 3 through 1 show a merged bidirectional differential port. A BC_4 type OBSERVE_ONLY and an AC_7 type BIDIR cell are coded for the representative (positive) port of the differential pair, and a BC_4 type OBSERVE_ONLY cell is coded for the associated (negative) port. Driver enable and ac/dc selection are provided by shared cells.

— Cell 0 provides a shared ac/dc selection for both bidirectional differential ports using an AC_SELU cell.

In this example, cells were used that support INTEST or RUNBIST. If that support is not needed, BC_2, AC_2, AC_8, and AC_10 cells could be substituted for BC_1, AC_1, AC_7, and AC_9 cells, respectively. AC_SELX cells could be substituted for AC_SELU cells.

7.6.3 boundary-scan configuration BSDL

This BSDL documents the example given in 7.6.2.

```
------------------------------------
-- BSDL Example for 1149.6 Standard.
------------------------------------
entity ACDEV1 is

-- Generic Parameter
generic (PHYSICAL_PIN_MAP : string := "ACDEV1_DIP32");

-- Logical Port Description
port (
   TDI : in bit;
   TMS : in bit;
   TCK : in bit;
   TDO : out bit;
   TRST_N : in bit;
   IN_SE1 : in bit;
   IN_SE2 : in bit;
   IN_SE3 : in bit;
   IN_DIFF1 : in bit_vector(0 to 1);
   IN_DIFF2 : in bit_vector(0 to 1);
   IN_DIFF3 : in bit_vector(0 to 1);
   OUT_SE : out bit;
   OUT_DIFF1 : buffer bit_vector(0 to 1);
   OUT_DIFF2 : out bit_vector(0 to 1);
   OUT_DIFF3 : out bit_vector(0 to 1);
   BIDI_DIFF1 : inout bit_vector(0 to 1);
   BIDI_DIFF2 : inout bit_vector(0 to 1);
   GND : power_0 bit_vector(0 TO 3);
   VDD : power_pos bit_vector(0 TO 2)
   );

   -- Use Statements
   use STD_1149_1_2013.all;
   use STD_1149_6_2015.all;

   -- Component Conformance Statement
   attribute COMPONENT_CONFORMANCE of ACDEV1 : entity is
      "STD_1149_1_2013";

   -- Device Package Pin Mappings
   attribute PIN_MAP of ACDEV1 : entity is PHYSICAL_PIN_MAP;
   constant ACDEV1_DIP32:PIN_MAP_STRING:=
      " TDI : 031," &
      " TMS : 003," &
      " TCK : 004," &
      " TDO : 030," &
      " TRST_N : 002," &
      " IN_SE1 : 005," &
      " IN_SE2 : 006," &
      " IN_SE3 : 007," &
      " IN_DIFF1 : (011,010)," &
      " IN_DIFF2 : (013,012)," &
      " IN_DIFF3 : (015,014)," &
```

```
      " OUT_SE : 018," &
      " OUT_DIFF1 : (019,020)," &
      " OUT_DIFF2 : (021,022)," &
      " OUT_DIFF3 : (023,024)," &
      " BIDI_DIFF1 : (026,027)," &
      " BIDI_DIFF2 : (028,029)," &
      " GND : (001,008,017,025)," &
      " VDD : (009,016,032)" ;

   -- Grouped Port Identification
   attribute PORT_GROUPING of ACDEV1: entity is
      "DIFFERENTIAL_VOLTAGE (" &
         "(IN_DIFF1(0), IN_DIFF1(1)), " &
         "(IN_DIFF2(0), IN_DIFF2(1)), " &
         "(IN_DIFF3(0), IN_DIFF3(1)), " &
         "(OUT_DIFF1(0), OUT_DIFF1(1)), " &
         "(OUT_DIFF2(0), OUT_DIFF2(1)), " &
         "(OUT_DIFF3(0), OUT_DIFF3(1)), " &
         "(BIDI_DIFF1(0), BIDI_DIFF1(1)), " &
         "(BIDI_DIFF2(0), BIDI_DIFF2(1)) )" ;

   -- Scan Port Identification
   attribute TAP_SCAN_CLOCK of TCK : signal is (50.0e6, BOTH);
   attribute TAP_SCAN_IN of TDI : signal is true;
   attribute TAP_SCAN_MODE of TMS : signal is true;
   attribute TAP_SCAN_OUT of TDO : signal is true;
   attribute TAP_SCAN_RESET of TRST_N : signal is true;

   -- Compliance-Enable Description
   -- none

   -- Instruction Register Description
   attribute INSTRUCTION_LENGTH of ACDEV1: entity is 4;
   attribute INSTRUCTION_OPCODE of ACDEV1: entity is
      -- IEEE Std 1149.1
      -- BYPASS gets all unused codes
      "EXTEST (0001)," &
      "SAMPLE (0010)," &
      "PRELOAD (0010)," &
      "IDCODE (1000)," &
      "CLAMP (0100)," &
      "HIGHZ (0101)," &
      --IEEE Std 1149.6
      "EXTEST_PULSE (0110)," &
      "EXTEST_TRAIN (0111)" ;
   attribute INSTRUCTION_CAPTURE of ACDEV1: entity is "0001";

   -- Optional Register Description
   attribute IDCODE_REGISTER of ACDEV1 : entity is
      "11110000111100001111000011110001";

   -- Register Access Description
   attribute REGISTER_ACCESS of ACDEV1: entity is
      "BOUNDARY (EXTEST_PULSE, EXTEST_TRAIN)" ;

-- Boundary-Scan Register Description
attribute BOUNDARY_LENGTH of ACDEV1 : entity is 34;
```

```
attribute BOUNDARY_REGISTER of ACDEV1 : entity is
   "33 (BC_4, IN_SE1,         OBSERVE_ONLY, X)," &
   "32 (BC_1, IN_SE2          INPUT, X, OPENX)," &
   "31 (BC_4, IN_SE3,         OBSERVE_ONLY, X)," &
   "30 (BC_1, IN_SE3,         INPUT, X, OPENX)," &
   "29 (BC_4, IN_DIFF1(1),    OBSERVE_ONLY, X)," &
   "28 (BC_4, IN_DIFF1(0),    OBSERVE_ONLY, X)," &
   "27 (BC_4, IN_DIFF2(1),    OBSERVE_ONLY, X)," &
   "26 (BC_1, IN_DIFF2(0),    INPUT, X, OPENX)," &
   "25 (BC_4, IN_DIFF3(1),    OBSERVE_ONLY, X)," &
   "24 (BC_4, IN_DIFF3(1),    OBSERVE_ONLY, X)," &
   "23 (BC_1, IN_DIFF3(0),    INPUT, X, OPENX)," &
   "22 (BC_4, IN_DIFF3(0),    OBSERVE_ONLY, X)," &
   "21 (BC_4, IN_DIFF3(0),    OBSERVE_ONLY, X)," &
   "20 (BC_1, *,              CONTROL, 0)," &
   "19 (AC_1, OUT_SE,         OUTPUT3, X, 20, 0, Z)," &
   "18 (AC_SELU, *,               INTERNAL, 0)," &
   "17 (AC_1, OUT_DIFF1(0),   OUTPUT2, X)," &
   "16 (BC_4, OUT_DIFF2(0),   OBSERVE_ONLY, X)," &
   "15 (BC_1, *,              CONTROL, 0)," &
   "14 (AC_1, OUT_DIFF2(0),   OUTPUT3, X, 15, 0, Z)," &
   "13 (BC_4, OUT_DIFF2(1),   OBSERVE_ONLY, X)," &
   "12 (BC_1, *,              CONTROL, 0)," &
   "11 (AC_9, OUT_DIFF3(0),   OUTPUT3, X, 12, 0, Z)," &
   "10 (BC_4, OUT_DIFF3(1),   OBSERVE_ONLY, X)," &
   " 9 (AC_SELU, *,           INTERNAL, 0)," &
   " 8 (BC_1, *,              CONTROL, 0)," &
   " 7 (BC_4, BIDI_DIFF1(0), OBSERVE_ONLY, X)," &
   " 6 (AC_1, BIDI_DIFF1(0), OUTPUT3, X, 8, 0, Z)," &
   " 5 (BC_1, BIDI_DIFF1(0), INPUT, X, OPENX)," &
   " 4 (BC_4, BIDI_DIFF1(1), OBSERVE_ONLY, X)," &
   " 3 (BC_4, BIDI_DIFF2(0), OBSERVE_ONLY, X)," &
   " 2 (AC_7, BIDI_DIFF2(0), BIDIR, X, 8, 0, Z)," &
   " 1 (BC_4, BIDI_DIFF2(1), OBSERVE_ONLY, X)," &
   " 0 (AC_SELU, *,           INTERNAL, 0)" ;

   -- Advanced I/O Description

   attribute AIO_COMPONENT_CONFORMANCE of ACDEV1 : entity is
      "STD_1149_6_2015";

   attribute AIO_Pin_Behavior of ACDEV1 : entity is
      "IN_SE1, IN_SE2, IN_SE3[30] : LP_time=5.0e-9 " &
                  "HP_time=15.0e-9 AIO_VTH=500 AIO_VHyst=90;" &
      "IN_DIFF1(0), IN_DIFF2(0), IN_DIFF3(0)[21], " &
      "   IN_DIFF3(1)[25] : LP_time=5.0e-9 HP_time=15.0e-9 " &
            " AIO_VTH=500 AIO_VHyst=90;" &
      "OUT_SE : AC_Select=18 AIO_VCM=500 AIO_VPP=250;" &
      "OUT_DIFF1(0) : AIO_VCM=500 AIO_VPP=250; " &
      "OUT_DIFF2(0), OUT_DIFF3(0) : AC_Select=9 AIO_VCM=500
AIO_VPP=250; " &
      "BIDI_DIFF1(0)[7], BIDI_DIFF2(0) [3] : AC_Select=0 " &
      "   LP_time=5.0e-9 HP_time=15.0e-9 AIO_VTH=AIO_VCM " &
          "  AIO_VHyst=90 AIO_VCM=500 AIO_VPP=250 ";
end ACDEV1;
```

7.6.4 Typical part description

Figure 70 shows a small SERDES part with two unidirectional 10-b parallel ports (one read and one write) and four pairs of high speed serial ports, an input and output port in each pair, plus appropriate clocks and controls. All of the data ports are differential. The parallel data ports are low-voltage pseudo emitter-coupled logic (LVPECL), and can be ac- or dc-coupled. The serial data ports are custom current mode logic (CML), and for performance reasons are required to be ac-coupled on the board.

Both 1X and 10X single-ended clocks are provided for the parallel and serial ports, respectively. Single-ended write port selection and read port selection and enablement inputs are provided that allow the parallel read and write ports to be associated with any of the four serial read and write ports. Idle serial write ports maintain a "null" data pattern, per the custom-encoding scheme used. Four single-ended "status" bits are provided to indicate which serial read ports are active (have legal patterns, including null, being received). This allows system detection of any disconnected serial port pairs.

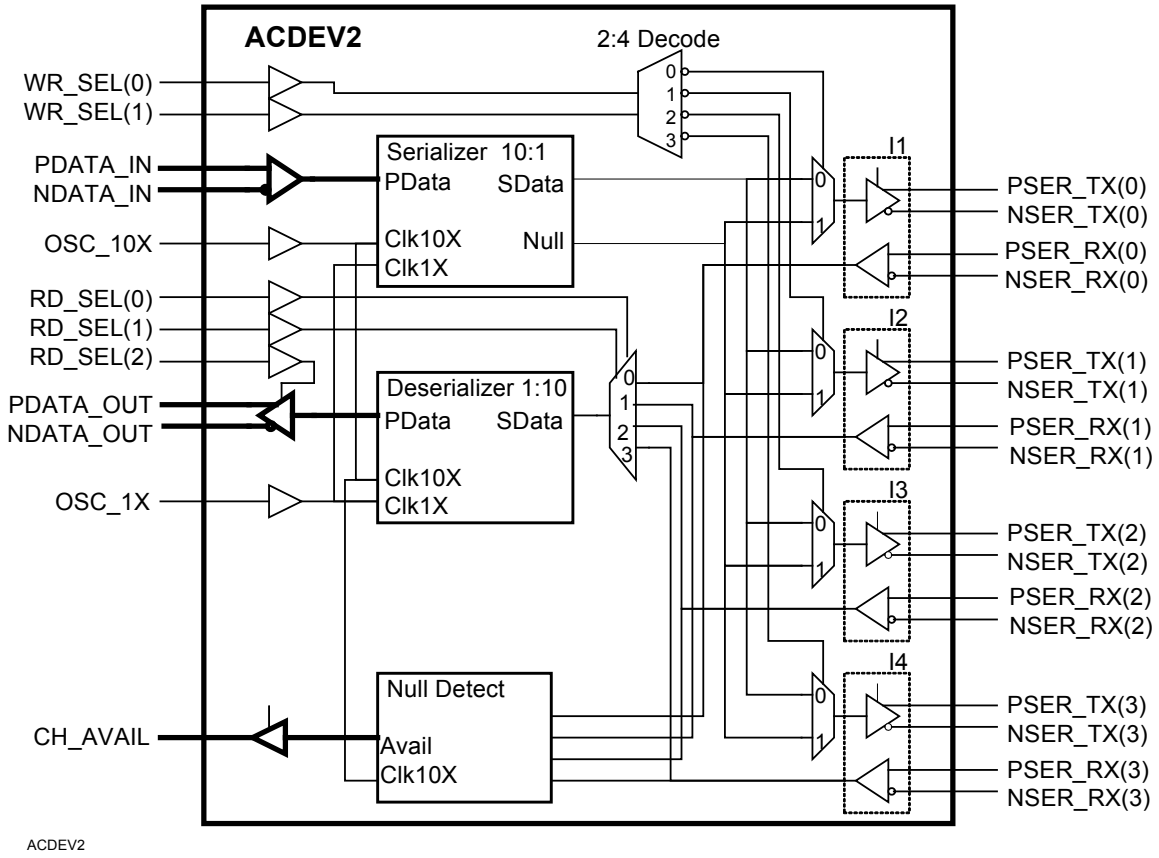


Figure 70—Typical ac-capable component block diagram

Note that the boundary-scan structures are not shown in the block diagram. Given the BSDL, below, as input, most Design-For-Test synthesis tools should be able to build the appropriate structures. The following are a few notes on the coding of the boundary-scan chain:

— CH_AVAIL, PSER_TX and NSER_TX are all two-state outputs. Their dangling enable pins will be connected to the TAP Controller HIGHZ control.

— RD_SEL(2) input is really just a read port enable for the parallel data output. Its boundary-scan is shown as a merged input and control (cell 46).

— None of the differential outputs are monitored, and none of the differential mission mode receivers are monitored or controlled. Neither INTEST nor RUNBIST are supported.

— The differential serial data ports are required to be ac-coupled, so there are no edge-detection filters in the associated test receivers.

— The differential parallel data ports can be ac- or dc-coupled, so the test receivers do have the edge-detection filter, and the outputs are provided a common ac/dc selection cell (cell 30).

The serial transmit/receiver pair are a multiply instantiated IP, as shown by the dashhed box around each pair, with the instance name above the box. The test receiver and driver in this IP have no programmable parameters. In additiion, the short segment of boundary-scan register in this IP uses custom input and output cells in support of the long and short loop-back tests. These cells are defined in the package file along with the ac parameters of the ac I/O.

The following is an example BSDL package and component descriptions.

```
------------------------------------------------------------
-- BSDL Package File for Acme LVDS Duplex Channel version 1.3.9
------------------------------------------------------------
package Acme_LVDS_Duplex_1_3_9 is

  -- Use Statements
  use STD_1149_1_2013.all;
  use STD_1149_6_2015.all;

  -- Boundary cell deferred constants (see package body)
  -- These cells, designed by Acme, provide additional test
  -- capabilities in support of the user looping instructions.
  constant Acme_Out : CELL_INFO;
  constant Acme_In  : CELL_INFO;

end package Acme_LVDS_Duplex_1_3_9 ;

package body Acme_LVDS_Duplex_1_3_9 is

  -- Use Statements
  use STD_1149_1_2013.all;
  use STD_1149_6_2015.all;

  constant Acme_Out : Cell_Info := -- Output cell derived from AC_2
((OUTPUT2, SAMPLE, PI),   -- No support for INTEST
     (OUTPUT2, EXTEST, UPD),  -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OUTPUT3, SAMPLE, PI),
     (OUTPUT3, EXTEST, UPD)); -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN

  constant Acme_In : CELL_INFO := -- Input cell derived from BC_4
    ((INPUT, EXTEST, PI),        -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (INPUT, SAMPLE, PI),
     (OBSERVE_ONLY, EXTEST, PI), -- EXTEST, EXTEST_PULSE, EXTEST_TRAIN
     (OBSERVE_ONLY, SAMPLE, PI));

  -- Advanced I/O Description 4129
  attribute AIO_COMPONENT_CONFORMANCE of
    Acme_LVDS_Duplex_1_3_9 : package is "STD_1149_6_2015";

  attribute AIO_Port_Behavior of Acme_LVDS_Duplex_1_3_9 : package is
    "port_link_list (RX : in bit ; TX : out bit ) ) ; " &
```

```
    "RX : Coupling_time=5.0e-9 AIO_VTH=500 AIO_VHyst=90; " &
    "TX : AIO_VCM=500 AIO_VPP=250" ;

-- The boundary-scan segment in this IP consists of three cells:
  --TDO
  -- "i   (Acme_Out, TX,       output2,      X)," &
  -- "i+1 (Acme_In , RX_Plus,  observe_only, X)," &
  -- "i+2 (Acme_In , RX_Minus, observe_only, X)," &
  --TDI

end package Acme_LVDS_Duplex_1_3_9 ;

-------------------------------------
-- BSDL Example for P1149.6 Standard.
-------------------------------------
entity ACDEV2 is

-- Generic Parameter
generic (PHYSICAL_PIN_MAP : string := "ACDEV2_PQFP96");

-- Logical Port Description
port (
   TDI : in bit;
   TMS : in bit;
   TCK : in bit;
   TDO : out bit;
   TRST_N : in bit;
   OSC_10X : in bit;
   OSC_1X : in bit;
   CE0_TEST : in bit;
   CE0_SCAN : in bit;
   RD_SEL : in bit_vector(0 to 2);
   WR_SEL : in bit_vector(0 to 1);
   -- The parallel data ports are LVPECL
   -- and can be ac or dc-coupled.
   PDATA_IN : in bit_vector(0 to 9);
   NDATA_IN : in bit_vector(0 to 9);
   PDATA_OUT : out bit_vector(0 to 9);
   NDATA_OUT : out bit_vector(0 to 9);
   -- The serial data ports are custom CML
   -- and MUST be ac-coupled.
   PSER_RX : in bit_vector(0 to 3);
   NSER_RX : in bit_vector(0 to 3);
   PSER_TX : buffer bit_vector(0 to 3);
   NSER_TX : buffer bit_vector(0 to 3);
   CH_AVAIL : buffer bit_vector(0 to 3);
   GND : power_0 bit_vector(0 TO 7);
   V25 : power_pos bit_vector(0 TO 7);
   V33 : power_pos bit_vector(0 TO 2)
   );

-- Use Statements
use STD_1149_1_2013.all;
use STD_1149_6_2015.all;
use Acme_LVDS_Duplex_1_3_9.all;

-- Component Conformance Statement
```

```
attribute COMPONENT_CONFORMANCE of ACDEV2 : entity is
   "STD_1149_1_2013";

-- Device Package Pin Mappings
attribute PIN_MAP of ACDEV2 : entity is PHYSICAL_PIN_MAP;
constant ACDEV2_PQFP96:PIN_MAP_STRING:=
   " TDI       : N01," &
   " TMS       : N02," &
   " TCK       : N03," &
   " TDO       : N04," &
   " TRST_N    : N06," &
   " OSC_10X   : N09," &
   " OSC_1X    : N19," &
   " CE0_TEST  : N21," &
   " CE0_SCAN  : N22," &
   " RD_SEL    : (N16,N17,N18)," &
   " WR_SEL    : (N07,N08)," &
   " PDATA_IN  : (W01,W03,W06,W08,W11,W13,W16,W18,W21,W23)," &
   " NDATA_IN  : (W02,W04,W07,W09,W12,W14,W17,W19,W22,W24)," &
   " PDATA_OUT : (S01,S03,S06,S08,S11,S13,S16,S18,S21,S23)," &
   " NDATA_OUT : (S02,S04,S07,S09,S12,S14,S17,S19,S22,S24)," &
   " PSER_RX   : (E01,E03,E21,E23)," &
   " NSER_RX   : (E02,E04,E22,E24)," &
   " PSER_TX   : (E06,E09,E13,E18)," &
   " NSER_TX   : (E07,E11,E14,E19)," &
   " CH_AVAIL  : (N11,N12,N13,N14)," &
   " GND       : (N10,N20,E10,E20,S10,S20,W10,W20)," &
   " V25       : (N05,N15,E05,E15,S05,S15,W05,W15)," &
   " V33       : (E08,E12,E17)" ;

-- Grouped Port Identification
attribute PORT_GROUPING of ACDEV2: entity is
"DIFFERENTIAL_VOLTAGE (" &
   "(PDATA_IN(0), NDATA_IN(0)), " &
   "(PDATA_IN(1), NDATA_IN(1)), " &
   "(PDATA_IN(2), NDATA_IN(2)), " &
   "(PDATA_IN(3), NDATA_IN(3)), " &
   "(PDATA_IN(4), NDATA_IN(4)), " &
   "(PDATA_IN(5), NDATA_IN(5)), " &
   "(PDATA_IN(6), NDATA_IN(6)), " &
   "(PDATA_IN(7), NDATA_IN(7)), " &
   "(PDATA_IN(8), NDATA_IN(8)), " &
   "(PDATA_IN(9), NDATA_IN(9)), " &
   "(PDATA_OUT(0), NDATA_OUT(0)), " &
   "(PDATA_OUT(1), NDATA_OUT(1)), " &
   "(PDATA_OUT(2), NDATA_OUT(2)), " &
   "(PDATA_OUT(3), NDATA_OUT(3)), " &
   "(PDATA_OUT(4), NDATA_OUT(4)), " &
   "(PDATA_OUT(5), NDATA_OUT(5)), " &
   "(PDATA_OUT(6), NDATA_OUT(6)), " &
   "(PDATA_OUT(7), NDATA_OUT(7)), " &
   "(PDATA_OUT(8), NDATA_OUT(8)), " &
   "(PDATA_OUT(9), NDATA_OUT(9)), " &
   "(PSER_RX(0), NSER_RX(0)), " &
   "(PSER_RX(1), NSER_RX(1)), " &
   "(PSER_RX(2), NSER_RX(2)), " &
   "(PSER_RX(3), NSER_RX(3)), " &
```

```
   "(PSER_TX(0), NSER_TX(0)), " &
   "(PSER_TX(1), NSER_TX(1)), " &
   "(PSER_TX(2), NSER_TX(2)), " &
   "(PSER_TX(3), NSER_TX(3)) )" ;

-- Scan Port Identification
attribute TAP_SCAN_CLOCK of TCK : signal is (50.0e6, BOTH);
attribute TAP_SCAN_IN of TDI : signal is true;
attribute TAP_SCAN_MODE of TMS : signal is true;
attribute TAP_SCAN_OUT of TDO : signal is true;
attribute TAP_SCAN_RESET of TRST_N : signal is true;

-- Compliance-Enable Description
attribute COMPLIANCE_PATTERNS of ACDEV2 : entity is
   "(CE0_TEST, CE0_SCAN) (00)";

-- Instruction Register Description
attribute INSTRUCTION_LENGTH of ACDEV2: entity is 4;
attribute INSTRUCTION_OPCODE of ACDEV2: entity is
   --IEEE Std 1149.1
   -- BYPASS gets all unused codes
   "EXTEST (0001)," &
   "SAMPLE (0010)," &
   "PRELOAD (0010)," &
   "IDCODE (1000)," &
   "CLAMP (0100)," &
   "HIGHZ (0101)," &
   --IEEE Std 1149.6
   "EXTEST_PULSE (0110)," &
   "EXTEST_TRAIN (0111)," &
   --User
   "Loop_Short (1100)," &
   "Loop_Long (1101)" ;

attribute INSTRUCTION_CAPTURE of ACDEV2: entity is "0001";
attribute INSTRUCTION_PRIVATE of ACDEV2: entity is
   "Loop_Short, Loop_Long";

-- Optional Register Description
attribute IDCODE_REGISTER of ACDEV2 : entity is
   "11110000111100001111000011110001";

-- Register Access Description
attribute REGISTER_ACCESS of ACDEV2: entity is
   "BOUNDARY (EXTEST_PULSE, EXTEST_TRAIN)," &
   "BYPASS (CLAMP, HIGHZ, Loop_Short, Loop_Long)" ;

-- Boundary-Scan Register Description
attribute BOUNDARY_LENGTH of ACDEV2 : entity is 54;
attribute BOUNDARY_REGISTER of ACDEV2 : entity is
   "53 (BC_1, WR_SEL(0),    INPUT, X, OPENX)," &
   "52 (BC_1, WR_SEL(1),    INPUT, X, OPENX)," &
   "51 (BC_4, OSC_10X,      CLOCK, X)," &
   "50 (BC_1, CH_AVAIL(0),  OUTPUT2, X)," &
   "49 (BC_1, CH_AVAIL(1),  OUTPUT2, X)," &
   "48 (BC_1, CH_AVAIL(2),  OUTPUT2, X)," &
   "47 (BC_1, CH_AVAIL(3),  OUTPUT2, X)," &
```

```
   "46 (BC_5, RD_SEL(2),    INPUT, X, OPENX)," &
   "46 (BC_5, *,            CONTROL, 0)," &
   "45 (BC_1, RD_SEL(1),    INPUT, X, OPENX)," &
   "44 (BC_1, RD_SEL(0),    INPUT, X, OPENX)," &
   "43 (BC_4, OSC_1X,       CLOCK, X, OPENX)," &
   "42 (BC_4, PSER_RX(0),   OBSERVE_ONLY, X)," &
   "41 (BC_4, NSER_RX(0),   OBSERVE_ONLY, X)," &
   "40 (BC_4, PSER_RX(1),   OBSERVE_ONLY, X)," &
   "39 (BC_4, NSER_RX(1),   OBSERVE_ONLY, X)," &
   "38 (AC_1, PSER_TX(0),   OUTPUT2, X)," &
   "37 (AC_1, PSER_TX(1),   OUTPUT2, X)," &
   "36 (AC_1, PSER_TX(2),   OUTPUT2, X)," &
   "35 (AC_1, PSER_TX(3),   OUTPUT2, X)," &
   "34 (BC_4, PSER_RX(2),   OBSERVE_ONLY, X)," &
   "33 (BC_4, NSER_RX(2),   OBSERVE_ONLY, X)," &
   "32 (BC_4, PSER_RX(3),   OBSERVE_ONLY, X)," &
   "31 (BC_4, NSER_RX(3),   OBSERVE_ONLY, X)," &
   "30 (AC_SELU, *,         INTERNAL, 0)," &
   "29 (AC_1, PDATA_OUT(0), OUTPUT3, X, 46, 0, Z)," &
   "28 (AC_1, PDATA_OUT(1), OUTPUT3, X, 46, 0, Z)," &
   "27 (AC_1, PDATA_OUT(2), OUTPUT3, X, 46, 0, Z)," &
   "26 (AC_1, PDATA_OUT(3), OUTPUT3, X, 46, 0, Z)," &
   "25 (AC_1, PDATA_OUT(4), OUTPUT3, X, 46, 0, Z)," &
   "24 (AC_1, PDATA_OUT(5), OUTPUT3, X, 46, 0, Z)," &
   "23 (AC_1, PDATA_OUT(6), OUTPUT3, X, 46, 0, Z)," &
   "22 (AC_1, PDATA_OUT(7), OUTPUT3, X, 46, 0, Z)," &
   "21 (AC_1, PDATA_OUT(8), OUTPUT3, X, 46, 0, Z)," &
   "20 (AC_1, PDATA_OUT(9), OUTPUT3, X, 46, 0, Z)," &
   "19 (BC_4, PDATA_IN(0),  OBSERVE_ONLY, X)," &
   "18 (BC_4, NDATA_IN(0),  OBSERVE_ONLY, X)," &
   "17 (BC_4, PDATA_IN(1),  OBSERVE_ONLY, X)," &
   "16 (BC_4, NDATA_IN(1),  OBSERVE_ONLY, X)," &
   "15 (BC_4, PDATA_IN(2),  OBSERVE_ONLY, X)," &
   "14 (BC_4, NDATA_IN(2),  OBSERVE_ONLY, X)," &
   "13 (BC_4, PDATA_IN(3),  OBSERVE_ONLY, X)," &
   "12 (BC_4, NDATA_IN(3),  OBSERVE_ONLY, X)," &
   "11 (BC_4, PDATA_IN(4),  OBSERVE_ONLY, X)," &
   "10 (BC_4, NDATA_IN(4),  OBSERVE_ONLY, X)," &
   " 9 (BC_4, PDATA_IN(5),  OBSERVE_ONLY, X)," &
   " 8 (BC_4, NDATA_IN(5),  OBSERVE_ONLY, X)," &
   " 7 (BC_4, PDATA_IN(6),  OBSERVE_ONLY, X)," &
   " 6 (BC_4, NDATA_IN(6),  OBSERVE_ONLY, X)," &
   " 5 (BC_4, PDATA_IN(7),  OBSERVE_ONLY, X)," &
   " 4 (BC_4, NDATA_IN(7),  OBSERVE_ONLY, X)," &
   " 3 (BC_4, PDATA_IN(8),  OBSERVE_ONLY, X)," &
   " 2 (BC_4, NDATA_IN(8),  OBSERVE_ONLY, X)," &
   " 1 (BC_4, PDATA_IN(9),  OBSERVE_ONLY, X)," &
   " 0 (BC_4, NDATA_IN(9),  OBSERVE_ONLY, X)" ;

-- Advanced I/O Description
attribute AIO_COMPONENT_CONFORMANCE of ACDEV2 : entity is
   "STD_1149_6_2015";

-- The following two "Execution" attributes are shown as examples.
-- They are rarely needed.

attribute AIO_EXTEST_Pulse_Execution of ACDEV2 : entity is
```

```
   "Wait_Duration TCK 15";

attribute AIO_EXTEST_Train_Execution of ACDEV2 : entity is
   -- To be used only for custom test
   "train 30, maximum_time 120.0e-6" ;

attribute AIO_Pin_Behavior of ACDEV2 : entity is
   "PDATA_IN  : Edge_Detect_time=11.25e-9 Coupling_time=22.5e-9 " &
   "  AIO_VTH=500 AIO_VHyst=90; " &
   "PDATA_OUT : AC_Select=30 AIO_VCM=500 AIO_VPP=250; " &
   "PSER_RX : package Acme_LVDS_Duplex_1_3_9 : RX ; "&
   "PSER_TX : package Acme_LVDS_Duplex_1_3_9 : TX ; "&

end ACDEV2;
```

7.6.5 Example for On-chip Capacitor Bypass Documentation

The following illustration and example Package and BSDL shows how to document an on-chip capacitor bypass control. The Package file is supplied by an IP supplier for the SerDes channel using the register description attributes, all as introduced by IEEE Standard 1149.1-2013. The register description attributes could be coded the same way for a component BSDL, if there was no IP Package supplied. Note that this circuit assumes capacitive coupling is present (an HP_TIME construct is included). For the Test Receivers to work on the board there would have to be an external capacitor if the internal capacitors shown are bypassed.

The circuit shown in Figure 71 is an example of a hard IP with a package file supplied by the IP designer.

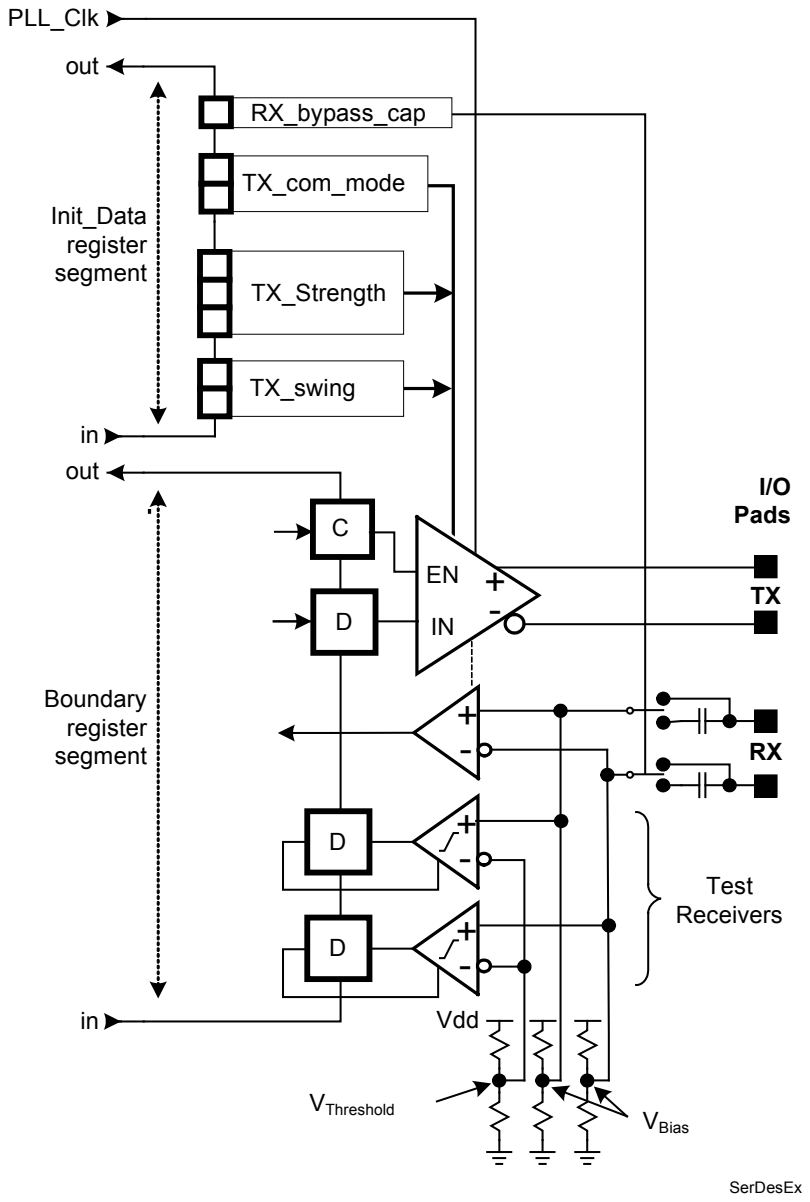


Figure 71 – SerDes channel with bypassable on-chip capacitor.

Figure 71 shows a single SerDes Channel with separate TX and RX I/O pads (as opposed to a single BIDI channel), and with Test Receivers on each receiver plus bypassable on-chip capacitors. Within the init-data register segment provided by the IP designer, there is a bit called RX_bypass_cap to control whether the on-chip capacitor is bypassed or not. The capacitor will be bypassed when this bit is set to '1.' The bypass switching is shown as an idealized switch, and there are several ways of configuring CMOS transistors to perform this function. Other fields in the init-data register control additional characteristics of the TX driver.

It should be noted that this example shows an init-data register bit controlling the bypass on a single differential pair of inputs, but a single such bit could control the bypass on multiple inputs, and a multi-bit field could control bypass as one of its functions. The REGISTER_MNEMONICS attribute allows easy identification of the bypass and reset values.

In the example BSDL Package shown below, provided by the designer of the IP in Figure 71, we see the mnemonics defined for each field, and in particular for the capacitor bypass bit. Then the init-data register

fields are defined. As a courtesy, the IP designer has also shown (in comments) how the included boundary segment should be coded for a single bit. This, of course, cannot be defined in the Package because the actual chip I/O pin names are required and these are not known to the IP designer, but rather, they are generated later by the IC designer making use of the IP.

In the top level BSDL that follows, we see a 9-b bus defined in the PORT statement, followed by the Port_Grouping and a boundary register segment for this bus. Then the init-data and boundary registers are assembled. Finally, the fields of the init-data register are associated with the correct I/O ports, and the AIO_PinBehavior attribute is defined including the fact that there is an on-chip capacitor. Note that the HP_time still applies whether the cap bypassed or not and whether an external capacitor is used or not.

Test software must note that the capacitor is bypassable, and adjust the test based on the settings in the init-data RC_cap_bypass bits. It must also check for an external capacitor if the on-chip capacitor is bypassed and the LP_Time value is not given (required capacitive coupling).

First, the hard IP description is provided in a package file provided by the IP designer.

```
package MyCorp_SERDES_1_2_3 is
   use STD_1149_1_2013.all;
   use STD_1149_6_2015.all;
end MyCorp_SERDES_1_2_3;

package body MyCorp_SERDES_1_2_3 is
   use STD_1149_1_2013.all;
   use STD_1149_6_2015.all;

   attribute REGISTER_MNEMONICS of MyCorp_SERDES_1_2_3 : package is
      "SerDes_TX_Strength ( "&
         "   off  (0b000) <Powered down>, "&
         " Resvd1 (0b001) <Reserved for future use>, "&
         " 13_seg (0b010) <4.5 ma -- Lowest strength>, "&
         " 15_seg (0b011) <5.5 ma>, "&
         " 17_seg (0b100) <6.5 ma>, "&
         " 19_seg (0b101) <7.5 ma>, "&
         " 22_seg (0b110) <9.0 ma>, "&
         " 26_seg (0b111) <10.5 ma -- Highest strength> "&
         " ), "&

      "SerDes_TX_Swing ( "&       -- Output driver swing level
         "         off (0b00) <Powered down>,  "&
         "   Max_Swing (0b01) <50%   Vdd Swing>, "&
         "   75%_Swing (0b10) <37.5% Vdd Swing>, "&
         "   50%_Swing (0b11) <25%   Vdd Swing> "&
         " ), "&

      "SerDes_TX_ComMode ( "&       -- Output driver swing level
         "          off (0b00) <Powered down>,  "&
         "     Centered (0b01) <50% Vdd>, "&
         "    One-third (0b10) <33% Vdd>, "&
         "  One-quarter (0b11) <25% Vdd> "&
         " ), "&

      "SerDes_RX_CapBypass ( "&     -- Only 2 valid settings
         "  Inline_cap (0b0), "&
         "  Bypass_cap (0b1), "&
         " )" ;
```

```
   attribute REGISTER_FIELDS of MyCorp_SERDES_1_2_3 : package is
      -- Init data register segment
      "Channel[8] ( "&
        "(TX_Swing[2] IS (7,6)    SAFE(SerDes_TX_Swing(*))), "&
        "(TX_Strength[3] IS(5,4,3)SAFE(SerDes_TX_Strength(17_seg))),"&
        "(TX_Common_mode[2] IS (2,1) SAFE(SerDes_TX_ComMode(*))), "&
        "(RX_Bypass_Cap[1] IS (0)    SAFE(SerDes_RX_CapBypass(*))) "&
        " )" ;

-----------------------------
-- NOTE: in each Channel, there are four boundary scan cells:
-- num cell  port            function safe [ccell disval rslt]
--TDO
--   n(BC_2,   *,            control,   0),
-- n+1(BC_8, TX_pos_port_id, output3,   X,     n,    0,    Z),
-- n+2(BC_4, RX_pos_port_id, observe_only, X),
-- n+3(BC_4, RX_neg_port_id, observe_only, X),
--TDI
-----------------------------
-- Advanced I/O Description 4129
  attribute AIO_COMPONENT_CONFORMANCE of MyCorp_SERDES_1_2_3:package is
     "STD_1149_6_2015";

-- The following assume a Vdd of 1500mv, 75%_swing VPP, and
-- One_Third VCOM
  attribute AIO_Port_Behavior of MyCorp_SERDES_1_2_3:package is
    "port_link_list (SERDES_RX : in bit ; SERDES_TX : out bit )) ;"&
    "SERDES_RX:HP_time=5.0e-9 On_Chip_Programmable "&
              "AIO_VTH=500 AIO_VHyst=240;"&
    "SERDES_TX:AIO_VCM=programmable AIO_VPP=programmable ";
...

end MyCorp_SERDES_1_2_3 ;

-- <EOF>
```

Note that the VCOM, VPP, and capacitor bypass parameters above are programmable. PDL routines (not shown) would be required for those parameters. See 7.7 for PDL requirements, and 7.8 for combined BSDL and PDL examples.

Next, a partial BSDL is given that uses this hard IP description from the package above.

```
-- Chip name is INIT_Example.
entity INIT_Example IS
   ...
   PORT (
      ...
     WBus7_P, WBus7_N, RBus7_P, RBus7_N
         : inout bit_vector (8 downto 0); -- TX Pos
            ...
   ) ;
   ...
   USE STD_1149_1_2013.all;
   USE STD_1149_6_2015.all;
   USE Mycorp_SERDES_1_2_3.all;
   ...
```

```
attribute PORT_GROUPING of INIT_Example : entity is
   "Differential_Voltage ( " &
   "   (WBus7_P(8), WBus7_N(8)),"&
   "   (WBus7_P(7), WBus7_N(7)),"&
   ...
   "   (WBus7_P(0), WBus7_N(0)),"&
   "   (RBus7_P(8), RBus7_N(8)),"&
   "   (RBus7_P(7), RBus7_N(7)),"&
   ...
   "   (RBus7_P(0), RBus7_N(0)))";
...

attribute BOUNDARY_SEGMENT of INIT_Example : entity is
     "byte_bus [45] ( "&
   -- num  cell  port      function safe [input/ccell disval rslt]
     " 44 (BC_4, RBus7_N(8), observe_only, X),  "&
     " 43 (BC_4, RBus7_P(8), observe_only, X),  "&
     " 42 (BC_2, RBus7_P(8), input,   X),       "&
     " 41 (AC_2, WBus7_P(8), Output3, X,         40,    0,    Z), "&
     " 40 (BC_2,     *,      control, 0),       "&
     " 39 (BC_4, RBus7_N(7), observe_only, X),  "&
     " 38 (BC_4, RBus7_P(7), observe_only, X),  "&
     " 37 (BC_2, RBus7_P(7), input,   X),       "&
     " 36 (AC_2, WBus7_P(7), Output3, X,         35,    0,    Z), "&
     " 35 (BC_2,     *,      control, 0),       "&
       ...
     "  4 (BC_4, RBus7_N(0), observe_only, X),  "&
     "  3 (BC_4, RBus7_P(0), observe_only, X),  "&
     "  2 (BC_2, RBus7_P(0), input,   X),       "&
     "  1 (AC_2, WBus7_P(0), Output3, X,          0,    0,    Z), "&
     "  0 (BC_2,     *,      control, 0),       "&
     "   )";
   ...

attribute REGISTER_ASSEMBLY of INIT_EXAMPLE : entity is
   -- Register Assembly of BOUNDARY register
   "boundary ( "&
      -- TDI
      ...
      "(upper_data_bus IS byte_bus), "&  -- 9*5=45 cells
      ...
      -- TDO
            "), " &
   -- Register Assembly of INIT_DATA register
   "init_data ( "&
   -- TDI
      ...
      -- Point to the IP package
      "(USING MyCorp_SERDES_1_2_3), "&
      "(array Serdes_Init(8 DOWNTO 0) IS Channel), "&
         -- Values are deferred
      ...
   -- TDO
            ")";

attribute Register_Association of INIT_EXAMPLE : entity is
```

```
   "Serdes_Init(8).RX_Bypass_Cap  : PORT (RBus7_P(8), RBus7_N(8)), "&
   "Serdes_Init(8).TX_Common_mode : PORT (WBus7_P(8), WBus7_N(8)), "&
   "Serdes_Init(8).TX_Strength    : PORT (WBus7_P(8), WBus7_N(8)), "&
   "Serdes_Init(8).TX_Swing       : PORT (WBus7_P(8), WBus7_N(8)), "&
   "Serdes_Init(7).RX_Bypass_Cap  : PORT (RBus7_P(7), RBus7_N(7)), "&
   "Serdes_Init(7).TX_Common_mode : PORT (WBus7_P(7), WBus7_N(7)), "&
   "Serdes_Init(7).TX_Strength    : PORT (WBus7_P(7), WBus7_N(7)), "&
   "Serdes_Init(7).TX_Swing       : PORT (WBus7_P(7), WBus7_N(7)), "&
   ...
         ")";
...
attribute AIO_Pin_Behavior of INIT_EXAMPLE : entity is
-- Capacitive coupling required, no LP_Time specified.
  "RBus7_P : Package MyCorp_SERDES_1_2_3 : SERDES_RX ; "&
  "WBus7_P : Package MyCorp_SERDES_1_2_3 : SERDES_TX ; "&
   ...
      ;" &
   ...
...
--  <EOF>
```

7.7 PDL procedures for programmable ac pins

In the **AIO_Pin_Behavior** or **AIO_Port_Behavior** attributes (see 7.5.4, 7.5.5 and 7.5.6), there are voltage parameters (**AIO_VCM**, **AIO_VPP**, **AIO_VTH**, and **AIO_VHyst**) that could be specified as programmable and a bypass for an on-chip capacitor that could also be specified as programmable. In this standard, programmable means that the value can be changed by setting a TDR bit through the TAP, and any such parameter must have an appropriate set of PDL procedures to document how the programming is performed. PDL, and the notation used to define PDL syntax, are defined in the current version of IEEE Std 1149.1. This standard does not add or change PDL in any way; it simply defines specific procedure names, arguments, actions, and return values. If a component or an IP contains ac pins with one or more programmable parameters, then the PDL procedures must be supplied with the component BSDL or the IP User Package.

Specifically, each programmable voltage parameter must have three PDL routines: one to set a value, a second to get and return the current write value, and a third to return a list of all possible values. A binary parameter only requires the first two, as the list of possible values is obvious. The procedure names are pre-defined so that they can be recognized and used automatically, and are listed in Table 5:

Table 5—PDL procedure names

Parameters	Set procedures	Get procedures	Getall procedures
AIO_VCM=programmable	AIO_set_VCM	AIO_get_VCM	AIO_getALL_VCM
AIO_VPP=programmable	AIO_set_VPP	AIO_get_VPP	AIO_getALL_VPP
AIO_VTH=programmable	AIO_set_VTH	AIO_get_VTH	AIO_getALL_VTH
AIO_VHyst=programmable	AIO_set_VHyst	AIO_get_VHyst	AIO_getALL_VHyst
On_chip_programmable	AIO_set_DC_couple	AIO_get_DC_couple	(N/A)
On_chip_ac_programmable	AIO_set_DC_couple	AIO_get_DC_couple	(N/A)

The syntax notation for PDL procedures is defined in C.3.1, C.3.2, and C.3.3 of IEEE Std 1149.1-2013. Certain syntax items adopted from that standard are underlined. As a quick reminder, literal text, in bold, is required and is case sensitive. The tokens <L_Brace>, and <R_Brace> are used for the literal characters **{** and **},** respectively. All of these procedures are Level 1 PDL, and return values as specified in the rules. Note PDL is designed to be compatible with Tool Command Language (Tcl) as described at length in IEEE 1149.1-2013, Annex C.

The PDL procedure definitions that follow all take a "port" argument, which is defaulted to the null string. For a component, this will be the port name defined for an ac port in the entity description. For an IP, this will be the port link name defined in the **AIO_Port_Behavior** BSDL attribute. The term “port argument” will refer to this argument regardless of whether it is in a component or IP context.

7.7.1 Syntax Specification

The programmable ac pin syntax specification is as follows:

<set_voltage> ::= **iProc** <set_voltage_proc_name> { <AIO_proc_option> }
<L_brace> **voltage** <L_brace> **port ""** <R_brace>
<L_brace> **vscale "1.0"** <R_brace> <R_brace> <executables> <ct>
<set_voltage_proc_name> ::= **AIO_set_VCM** | **AIO_set_VPP** | **AIO_set_VTH** | **AIO_set_VHyst**

<get_voltage> ::= **iProc** <get_voltage_proc_name> { <AIO_proc_option> }
<L_brace> <L_brace> **port ""** <R_brace>
<L_brace> **vscale "1.0"** <R_brace> <R_brace> <executables> <ct>
<get_voltage_proc_name> ::= **AIO_get_VCM** | **AIO_get_VPP** | **AIO_get_VTH** | **AIO_get_VHyst**

<getall_voltage> ::= **iProc** <getall_voltage_proc_name> { <AIO_proc_option> }
<L_brace> <L_brace> **port ""** <R_brace>
<L_brace> **vscale "1.0"** <R_brace> <R_brace> <executables> <ct>
<getall_voltage_proc_name> ::= **AIO_getALL_VCM** | **AIO_getALL_VPP** |
AIO_getALL_VTH | **AIO_getALL_VHyst**

<set_binary> ::= **iProc AIO_set_DC_couple** { <AIO_proc_option> }
<L_brace> **state** <L_brace> **port ""** <R_brace>
<R_brace> <executables> <ct>

<get_binary> ::= **iProc AIO_get_DC_couple** { <AIO_proc_option> }
<L_brace> <L_brace> **port ""** <R_brace> <R_brace> <executables> <ct>

<AIO_proc_option> ::= **-export** | **-mission** | <information>
<executables> ::= <L_brace> <proc_command> { <proc_command>} <R_brace>

7.7.2 Rules

The programmable ac pin rules are as follows:

a) All of the procedure definitions in 7.7.1, when required, shall follow an **iPDLLevel** command setting the PDL level to “1” and the version to STD_1149_1_2013.

b) All of the above procedure definitions, when required, shall follow an **iProcGroup** command associating the procedures with the <component name> or <user package name> containing the register fields used to control the programmable parameters.

NOTE—In the case of hierarchical IP, where the register segment and the port are not in the same BSDL or User Package, this may or may not be the same component or IP where the actual port is directly instantiated. Contrast this with rule b) in 7.5.5.2.

c) When one of the above procedures is used in an **iCall** command, the argument **port** shall resolve to either an <AC port> for a procedure associated with a component BSDL, or a <port link> for a procedure associated with an IP user package, or the null string (“”, the default) if the associated component or IP has only a single ac pin.

d) When one of the above procedures is used in an **iCall** command, the argument **vscale** shall resolve to a <real>, as defined in this standard.

e) When a <set_voltage> procedure is used in an **iCall** command, the argument **voltage** shall resolve to a <millivolts> scaled integer value expressed in millivolts and shall be divided by the supplied scaling factor **vscale** within the called procedure when necessary to recover the nominal value.

f) When a <set_binary> procedure is used in an **iCall** command, the argument **state** shall resolve to either the string "**ON**" or the string "**OFF.**"

g) All of the procedure definitions in 7.7.1 that are defined within an iProcGroup associated with an entity <component name> shall have an implied <AIO_proc_option> value of **-export**.

NOTE—The **-export** option is still permitted, and it has value as a reminder to those reading or writing the PDL, but it is redundant. Procedures associated with an IP <user package name> are not normally exported, much less assumed to be exported.

h) None of the procedure definitions in 7.7.1 shall contain an **iTRST** or **iTMSReset** command at any level of the procedure hierarchy.

i) For any component containing TDR fields controlling the programmable parameters of an ac pin, either by direct instantiation in the component or by instantiation of IP containing an ac pin, and also for any IP directly instantiating an ac pin, the <set_voltage>, <get_voltage>, and <getall_voltage> procedures shall be provided for each voltage parameter (**AIO_VCM**, **AIO_VPP**, **AIO_VTH**, or **AIO_VHyst**) that has a value of **programmable** or **bus_programmable** in an <AC parameter list>.

j) For any component containing TDR fields controlling the programmable parameters of an ac pin, either by direct instantiation in the component or by instantiation of IP containing an ac pin, and also for any IP directly instantiating an ac pin, the <set_binary> and <get_binary> procedures shall be provided whenever a <cap spec> keyword in an <AC parameter list> ends in the string "**_programmable**."

NOTE—The previous two rules, i) and j), do not apply to an IP that contains an ac pin only because it instantiates another IP containing an ac pin. It does apply to the component level any time the component contains an ac pin, whether or not that ac pin is also documented for an IP.

k) All <set_voltage> procedures shall return a Tcl list of all port names (<AC port> for components, <port link> for IP) that were set by the call to the procedure; or optionally the <group name> if the port was part of a bus-programmable group; or the string "AIO_ERROR" in case the passed port is not recognized by this routine, or the passed voltage or scaling factor are not supported for this named port.

l) All <get_voltage> procedures shall return a <millivolts> value expressed in millivolts, corresponding to the current write setting of the named port; or the string "AIO_ERROR" in case the passed port is not recognized by this routine.

NOTE 1—The "current write setting" would normally be the nominal value, implied by the encoded value in the TDR recovered by an "**iGet –si**" command, multiplied when appropriate by the scaling factor **vscale**.

NOTE 2—The result of an "iGet -si" command is the value to be written by the next iApply command. See IEEE Std 1149.1-2013 rule d) in C.3.7.1 and rule d) in C.4.2. In other words, it is the value currently scheduled to be written, and the "iGet -si" command allows the procedure to determine whether that value needs to be changed. The result of a <get voltage> procedure may be completely different than the current actual register setting.

m) All <set_binary> procedures shall set the controlling field so that the on-chip coupling capacitor is not bypassed during ac test if the input state is "**OFF**," and so that it is bypassed during ac test if the input state is "**ON**."

n) All <set_binary> procedures shall return a Tcl list of all port names (<AC port> for components, <port link> for IP) that were set by the call to the procedure; or optionally the <group name> if the

port was part of a bus-programmable group; or the string "AIO_ERROR" in case the passed port is not recognized by this routine, or the passed on/off parameter is not the string "**ON**" or "**OFF**."

o) All <get_binary> procedures shall return the string "**ON**" if the named port is currently dc-coupled during ac test; "OFF" if the named port is currently ac-coupled during ac test; or the string "AIO_ERROR" in case the passed port is not recognized by this routine.

p) All <getall_voltage> procedures shall return a Tcl list of <millivolts> values expressed in millivolts, listing all of the scaled actual values that the named port can be set to; or the string "AIO_ERROR" in case the passed port is not recognized by this routine, or the passed scaling factor **vscale** is not supported for this port.

7.7.3 Description

Rules 7.7.2 a) and b), above, require all PDL procedures mandated by rules 7.7.2 h) and i) to be described as Level 1 PDL conforming to IEEE Std 1149.1-2013, Annex C, using the **iPDLLevel** command, and to be associated with either the component BSDL or an IP User Package using the **iProcGroup** command.

NOTE— The <AIO_proc_option> has a subset of values provided by <proc_option> given in IEEE Std 1149.1-2013, C.3.5.4.

Rules 7.7.2 i) and j) effectively mandate that there be component level PDL procedures for the used programmable parameters (AIO_VCC, AIO_VPP, AIO_VTH, AIO_VHyst, or a programmable <cap spec>) whenever any port on the component has one or more programmable parameters. This includes ports that are instantiated in an IP for which there already is an IP level PDL procedure. The reason for this is that the component level procedure needs to call the correct instance of the IP instantiation in order for the system to find the field names that are used in the IP PDL, and the component level procedure needs to convert the component port name to the correct IP port-link name used in the IP PDL. This makes the component level PDL procedures defined in this clause the sole entry point for programming the ports, whether such programming is being done as part of the init_setup procedure for the component, or for verification or diagnostic work. Calls directly to the IP PDL procedures from a tool would not necessarily have all of the needed information.

The field names are defined in the IP User Package and used in the IP PDL. REGISTER_ASSEMBLY attributes will assemble the segments in the IP into the component level TDRs, creating unique instance names for each instantiation. REGISTER_ASSEMBLY attributes in the component BSDL can also create instance hierarchies within the component as well. All of the defined instance hierarchy needs to be part of the call of the IP PDL.

Similarly, port-link names are defined in the IP User Package and used in the IP PDL. The component level AIO_PIN_BEHAVIOR attribute includes the mapping of port name to the correct port-link name, but that information cannot be retrieved by a PDL procedure, so it must also be encoded in the component PDL call to the IP PDL. Unlike the register instance hierarchy, port-link names are not subject to hierarchy. If an IP instantiates a port with programmable parameters and the controlling register segment, and that IP is instantiated in another IP (User Package) which is in turn instantiated in the component, there are none of the programming PDL procedures defined in this clause for the intermediate IP (unless it instantiates another port). The component AIO_PIN_BEHAVIOR attribute directly references the port-link name in the User Package where the port was instantiated; it does not have to reference a hierarchical reference.

Note that if an IP provider instantiates a programmable port, but does not include a register segment with fields for providing the programming data, then it is not possible to code PDL procedures for that IP. Such IP would be instantiated in a higher level IP (perhaps just a wrapper with the register segment) or the component, and the register fields defined there. All of the procedures defined in this clause must be defined at the level (IP User Package or component BSDL) where the register fields are defined.

Every procedure defined in this clause takes as an argument an optional name of a port argument, per rule 7.7.2 c), which defaults to the null string. This argument can be defaulted only if there is exactly one ac pin

in the component or IP, otherwise it is required. In a User Package for an IP block, the name would be a port link name as defined in the **AIO_Port_Behavior** attribute <port link declaration list>. In a BSDL for a component, the name would be an <AC port> as defined in the **AIO_Pin_Behavior** attribute. The PDL procedure will use the port name, if required, to select the correct method for responding to the call. The procedure will return the null string ("") if the name does not match any of the ports where that parameter is programmable, and would return the string "AIO_ERROR" for other errors, as specified in rules 7.7.2 k) through p).

The **AIO_set_*** PDL procedures also take a value as the first argument. For the voltage parameters, the value will be a non-negative integer expressed in millivolts, per rule e). For the binary parameters, the value will be either the keyword **ON** or **OFF**.

Every voltage procedure either accepts (**AIO_set_***) or returns (**AIO_get_*** or **AIO_getall_***) voltages expressed as an integer number of millivolts. In some cases, and possibly on a port by port basis, the current value of some of these voltage arguments could depend on some external parameter, such as the Vdd voltage supplied to the I/O circuits (see Figure 71 for an example circuit using resistor dividers to set VTH). The PDL is expected to normally be coded for voltage values at nominal, if arbitrary, conditions, and each voltage procedure takes as its last argument an optional scaling factor **vscale**, defaulted to 1.0, which is used to adjust the nominal voltage values for such dependencies. Note that, depending on the design, a given parameter of a given I/O could be completely independent of outside factors, so use or not of the scaling factor is determined inside the above procedures.

For components, the actual scaling needed might not be known until the component is instantiated on a board, and so the scaling factor might need to be supplied at the time board tests are generated, similar to other initialization parameters first described in IEEE Std 1149.1-2013 (see 8.17 and following clauses of that standard for additional discussion of board-level initialization). For an IP, the scaling factor might be fixed when the IP is instantiated in a component, or the value determined by a board design might need to be passed down through the component PDL procedures to the IP procedures. In all cases, the voltage values supplied to or returned by the procedure is the actual, scaled value, not the nominal value (when there is a difference), so these values should be used without adjustment for comparison to values from other chips during board test generation.

In some cases, the dependency could be on a discrete set of values, and in other cases, on a range of values. This implies scaling factors might have discrete legal values, or a legal range of values, and these can be checked inside the above procedures. As a general "good practice," the header of these PDL routines would include documentation of the conditions upon which the nominal values are based, any dependencies on external values, and the legal discrete scaling factors or legal ranges of scaling factors for the dependent conditions. This will help users of the IP or component for which the PDL is written.

Per rule 7.7.2 k), if a call to one of the **AIO_set_*** PDL procedure was able to successfully set the value, it will return a list of all of the names of the ports that were set. Normally, if the parameter has a value of **programmable**, there will be only one name in the list, the same name as was supplied or defaulted on the call, and if the parameter has a value of **bus_programmable**, then it will be the list of all of the ports that are programmed together.

Note that any two port attributes that have even a single parameter not in common can not be coded together in a single BSDL <entity pin info> or <package pin info> element. In order to state in the BSDL that such pins are, in fact, programmed together by a call, a group name is added in parenthesis to the **bus_programmable** keyword. Consider the following example: an IP block with three port links, A, B, and C. A is type **in**, B is type **inout**, and C is type **out**. For the input and bidirectional test receivers, VTH is set by a single TDR field and the on-chip bypass is set by a single TDR field, and for the outputand bidirectional drivers, VCM is set by a single TDR field. By rule, the **AIO_Port_Behavior** would have to provide separate parameters for each of the three ports since they are different types, as follows.

```
-- Excerpts from BSDL Package Body
attribute AIO_PORT_BEHAVIOR of StrangePkg : package is
```

```
  "port_link_list ( "&
    "A : in bit; "&
    "B : inout bit; "&
    "C : out bit ); "&
"A : HP_time=1.3e-9 on_chip_bus_programmable(grp1) "&
"    AIO_VTH=bus_programmable(grp2) AIO_VHyst=140; "&
"B : HP_time=1.3e-9 on_chip_bus_programmable(grp1) "&
"    AIO_VCM=bus_programmable(grp3) AIO_VPP=200 "&
"    AIO_VTH=bus_programmable(grp2) AIO_VHyst=140; "&
"C : AIO_VCM=bus_programmable(grp3) AIO_VPP=200 " ;
```

In this situation, a call to **AIO_set_VTH** or **AIO_set_DC_couple** with port link A or B as an argument will return the list "A B," and a call to **AIO_set_VCM** with port link B or C as an argument will return the list "B C." Which ports are controlled together by a single PDL call, when one or more parameter has the value **bus_programmable**, is made explicit by the group name. All ports having the same unique group name for a bus-programmable parameter are programmed by the same register field. This information is also documented unambiguously in the PDL procedure by returning a list of all port links, or optionally the group name, that were set by that call.

Per rules k) and m), a call to one of the **AIO_get_*** PDL procedures will return the current write value for that parameter. "Current" is defined the same way as the "**iGet –si**" command (that is, retrieved from the internal map of the current write settings) and the value will be either a non-negative integer expressed in millivolts for the voltage parameters or the string "**ON**" or "**OFF**" for a binary parameter. Note that these values returned by the AIO_get_* procedures, per the definition of the iGet command in IEEE Std 1149.1-2013, are the values to be written to the register rather than the actual current value in the hardware register field. This will normally not matter, as an iApply will always be done after setting any of the fields in the TDR, and the write data will then become the actual register setting.

Per rule p), a call to one of the AIO_getALL_* PDL procedures will return a list of all legal values possible for the port argument. Again, the values are non-negative integers expressed in millivolts. This information is only documented in these PDL procedures, not in the BSDL.

Rules k) through p) also require that PDL procedures check the calling parameters to help ensure they are not incorrect and issue the string "AIO_ERROR" when such an error occurs. For example, a port name might be misspelled or otherwise not recognized by the procedure, or a requested voltage setting or scaling factor is not supported by that port.

These PDL procedures do not have all of the capabilities defined for a general **iProc** command. Specifically, the reduced list of proc-options does not allow the use of the **–TRSTreset** or **–TMSreset** options. In addition, the use of the **iTRST** or **iTMSreset** commands are not allowed in any of these PDL procedures, or any sub-procedure at any level of the hierarchy below these procedures. This helps ensure that these procedures do not reset the test in mid-stream.

7.8 Example PDL procedures for programmable ac pins

This clause provides integrated examples of PDL procedures with supporting segments of BSDL and/or BSDL packages. The first shows a board with three components and two differential channels, with partial BSDL and full PDL for each component, and partial PDL for the board. The second shows some additional features using an IP that instantiates four duplex lanes of differential I/O, some parameters of which are bus programmable. Partial BSDL Package and full PDL are provided for the IP, with partial BSDL and PDL for the chip.

Implied in the rules and from the information available to the PDL, and just good programming practice, any of the standard iProcs defined in this standard have a few basic things that they would be expected to do:

- Verify that the port or port-link name supplied is actually valid for this procedure. This includes dealing with the fact that VHDL identifiers are case insensitive, and PDL strings are all case sensitive.
- For component procedures calling IP procedures, convert the port name to a port-link name.
- Convert the port or port-link name into the correct instance of the register field name to be written or read.
- For a "set procedure, verify that the voltage or state argument supplied is a legal value for the register field.
- Convert between the voltage or state argument and the encoded value to be written to or having been read from the register field.
- For voltage procedures, perform voltage scaling as required.
- Print human-readable error messages.

Note that all this applies whether or not all the BSDL register attributes were coded. The register field might just have to be referenced as a bit range within a TDR, and the actual binary value rather than a mnemonic name used as the value to be written. In this case, the information required to write the PDL procedure would have to be gleaned from other documentation of the design than the BSDL.

7.8.1 Example of a single programmable differential path

This example shows a board containing three components with instance names of U1, U2, and U3. U1 is an instance of component IC1, and is designed without any programmability on its differential advance I/O, but is documented in compliance with this standard. U2 and U3 are two instances of component IC2, and the differential advanced I/O drivers have programmable common-mode voltage (VCM), and the test receivers have programmable threshold voltage (VTH). The example will focus on the drivers and receivers for the two board level differential channels Chan1 and Chan2, shown in Figure 72.

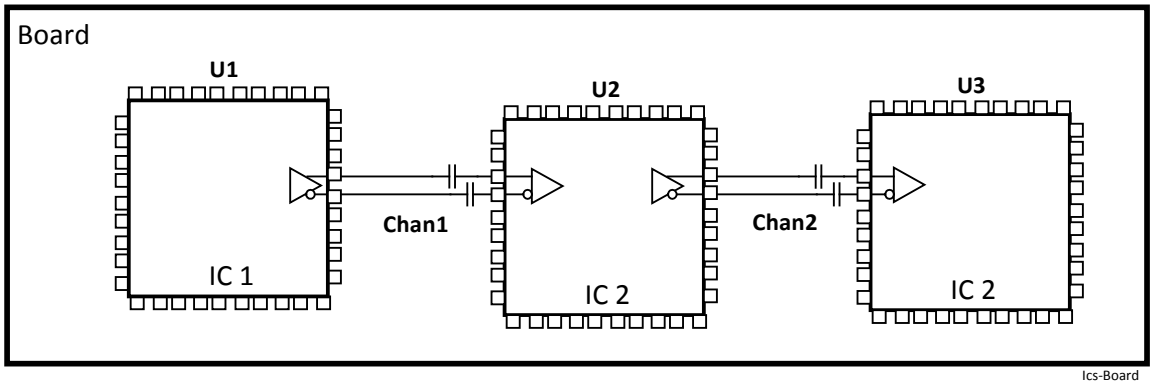


Figure 72—A board with two differential channels between three ICs.

The drivers and receivers follow the LVDS standard. IC1 is a 2.5V Vdd component, so the common-mode voltage on its driver is 1250mv, and the peak-to-peak voltage (VPP) is 400mv. Partial BSDL for this component is shown below, but no PDL is required for this component since there are no programmable parameters.

IC2 is a 1.2V Vdd component, so the test receiver programmable threshold voltage defaults to 600mv, with a fixed edge-detection hysteresis of 240mv. The test receiver threshold voltage can be programmed to 0, 450, 600, 750, and 1200mv to give some flexibility in different board configurations. The IC2 differential input receivers are shown in Figure 73.

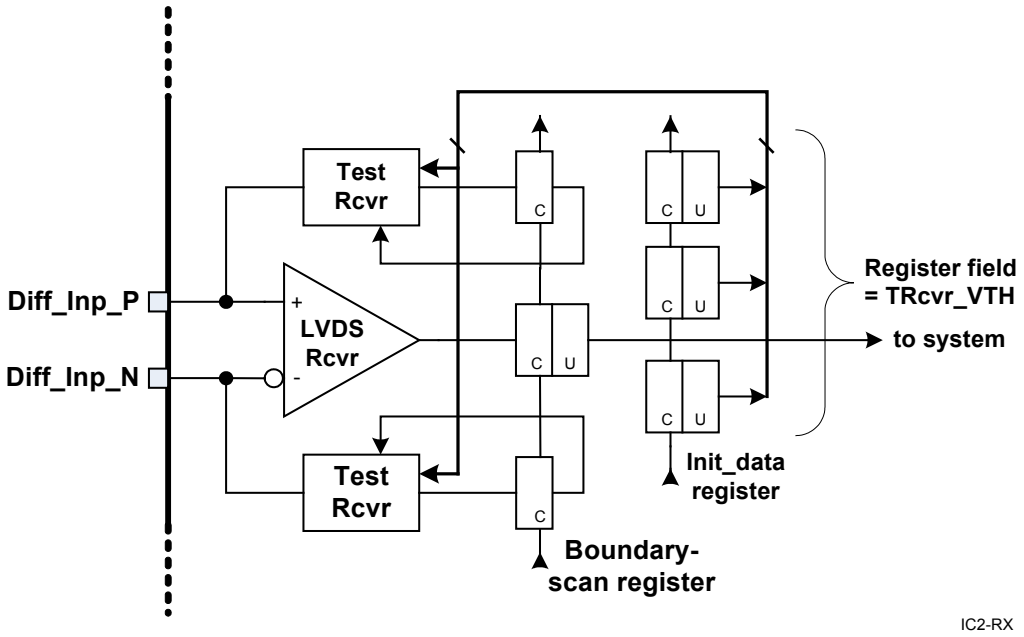


Figure 73—Differential input receiver design in IC2

The IC2 driver common-mode programmable voltage defaults to 600mv, and the driver peak-to-peak voltage is fixed at 350mv. The driver common-mode voltage can be programmed to 450, 600, or 750mv. The IC2 driver is shown in Figure 74.

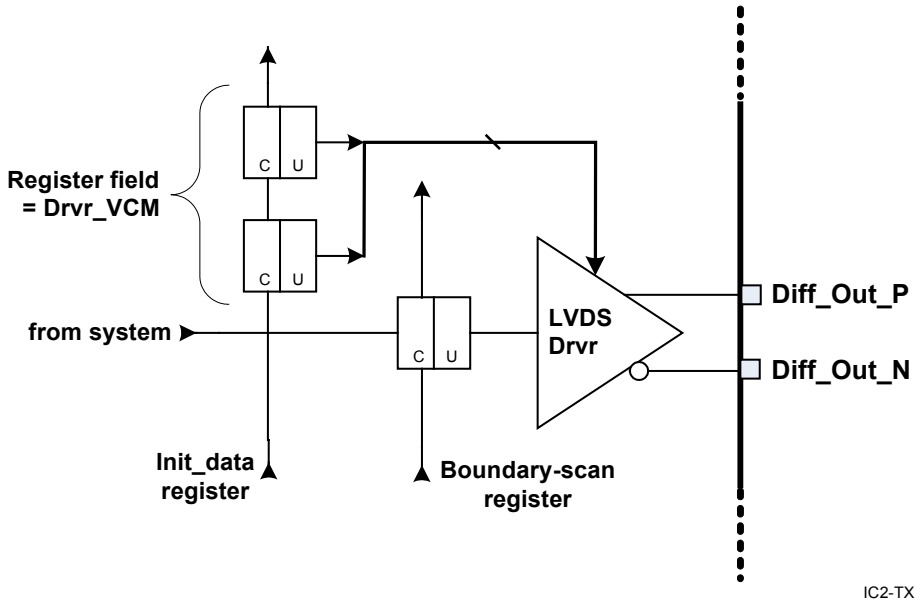


Figure 74—Differential driver design in IC2

Partial BSDL and complete PDL are shown for IC2 in the following sections.

When analyzing Chan1 between the U1 driver and the U2 receiver, the ac test mode appears to have no problem. Driver VPP is 400mv, test receiver edge-detection hysteresis is 240mv, which is 60-70% of the receiver input delta-V, allowing for some small amount of board attenuation. However, the dc test (EXTEST) does have an obvious problem detecting a shorted capacitor, since the U1 driver common-mode voltage is 1250mv, above the Vdd of IC2. In this case, the U2 test receiver can be programmed for the shorted capacitor test to a threshold voltage of 0mv, and the receiver treated as if the **Detect_EXTEST_High** keyword were encoded in the AIO_Port_Behavior attribute for the receiver port, as allowed by Permission a) in 7.5.6.4. This will allow detection for a shorted capacitor even though EXTEST does not work.

Note that if U1 had a Vdd of 1.5V, then the fixed driver common-mode voltage would be 750mv and a matching threshold voltage of 750mv could be programmed into the U2 test receiver. In this case EXTEST would be expected to work to detect a shorted capacitor.

When analyzing Chan2, it is apparent that there should be no problem with either ac test or dc test. The default values should be programmed into the U2 driver and the U3 test receiver.

7.8.1.1 Example IC1 Partial BSDL

An example of the IC1 Partial BSDL is as follows:

```
------------------------------------------------------
-- Partial BSDL for Component IC1 of design example.
--   2.5V Vdd
--   No AIO programmability
--   Elipsis (...) represent required but omitted text.
------------------------------------------------------
entity IC1 is
  ...
  port (
    ...
    DIFF_DRVR_P, DIFF_DRVR_N : buffer bit;
    ...
  );
  use STD_1149_1_2013.all ;
  use STD_1149_6_2015.all ;
  attribute COMPONENT_CONFORMANCE of IC1 : entity is "STD_1149_1_2013"
;
  ...
  --Differential driver outputs are grouped because they are
  --  controlled by a single boundary cell
  attribute PORT_GROUPING of IC1 : entity is
    ...
    "differential_current ( " &
    "  (DIFF_DRVR_P, DIFF_DRVR_N) " &
    ") , " &
    ...
  ;
...
  -- Establish the length of the standard INIT_DATA register
  attribute REGISTER_ACCESS of IC1 : entity is
    ...
    "INIT_DATA[89] (INIT_SETUP, INIT_SETUP_CLAMP), " &
```

```
    ...
  ;
  -- Describe the Boundary-scan register
  attribute BOUNDARY_LENGTH of IC1 : entity is 81 ;
  attribute BOUNDARY_REGISTER of IC1 : entity is
  -- num cell  port/*      function safe [ccell disval rslt]
    ...
    "17 (BC_2, DIFF_DRVR_P, output2, X), " &
    ...
  ;
  -- Advance I/O description
  attribute AIO_COMPONENT_CONFORMANCE of IC1 : entity is
    "STD_1149_6_2015";
  -- Document the parameters of the AIO driver.
  attribute AIO_PIN_BEHAVIOR of IC1 : entity is
    ...
    "DIFF_DRVR_P: aio_vcm=1250 aio_vpp=400; " &
    ...
  ;
end IC1 ;
```

7.8.1.2 Example IC2 Partial BSDL

An example of the IC2 Partial BSDL is as follows:

```
------------------------------------------------------
-- Partial BSDL for Component IC2 of design example.
--   1.2V Vdd
--   AIO programmability on test receivers and differential driver
--   Elipsis (...) represent required but omitted text.
------------------------------------------------------
entity IC2 is
  ...
  port (
    ...
    DIFF_INP_P, DIFF_INP_N : in bit;
    DIFF_OUT_P, DIFF_OUT_N : buffer bit;
    ...
  );
  use STD_1149_1_2013.all ;
  use STD_1149_6_2015.all ;
  attribute COMPONENT_CONFORMANCE of IC2 : entity is "STD_1149_1_2013"
;
  ...
  -- Differential inputs are grouped in this case because they have
  --   a full input cell on the mission differential receiver.
  -- Differential driver outputs are grouped because they are
  --  controlled by a single boundary cell
  attribute PORT_GROUPING of IC2 : entity is
    ...
    "differential_current ( " &
    "  (DIFF_INP_P, DIFF_INP_N)," &
    "  (DIFF_OUT_P, DIFF_OUT_N)" &
    ") , " &
    ...
```

```
  ;
...
  -- Establish the length of the standard INIT_DATA register
  attribute REGISTER_ACCESS of IC2 : entity is
    ...
    "INIT_DATA[101] (INIT_SETUP, INIT_SETUP_CLAMP), " &
    ...
  ;
  -- Describe the Boundary-scan register
  attribute BOUNDARY_LENGTH of IC2 : entity is 84 ;
  attribute BOUNDARY_REGISTER of IC2 : entity is
  -- num cell  port/*      function safe [ccell disval rslt]
    ...
    "34 (BC_2, DIFF_OUT_P, output2,      X), " &
    ...
    "60 (BC_4, DIFF_INP_P, observe_only, X), " &
    "61 (BC_2, DIFF_INP_P, input,        X, OPENX) " &
    "62 (BC_4, DIFF_INP_N, observe_only, X), " &
    ...
  ;
  -- The following register attributes are optional, but make
  --   writing and reading PDL much easier.
  -- Describe values that may be used with register fields
  attribute REGISTER_MNEMONICS of IC2 : entity is
    "AIO_INP_THRESH ( " &
    "  GROUND   (0b000) <Detect_EXTEST_High behavior>, "&
    "  450MVTH  (0b0X1) <EXTEST 450 mv threshold>, "&
    "  600MVTH  (0bX10) <EXTEST 600 mv threshold>, "&
    "  750MVTH  (0b1X1) <EXTEST 750 mv threshold>, "&
    "  VDD_IO   (0b100) <Detect_EXTEST_Low behavior> ), "&
    "AIO_OUT_COMMON ( " &
    "  450MVCM  (0b0X) <450 mv common-mode>, "&
    "  600MVCM  (0b10) <600 mv common-mode>, "&
    "  750MVCM  (0b11) <750 mv common-mode> )"
  ;

  -- Describe register fields in a REGISTER_ASSEMBLY attribute.
  --   Associate each field with a mnemonic group and default value.
  attribute REGISTER_ASSEMBLY  of IC2 : entity is
    "init_data ( " &
    ...
    "  (Drvr_VCM [2] default (AIO_OUT_COMMON (600MVCM))), " &
    ...
    "  (TRcvr_VTH [3] default (AIO_INP_THRESH (600MVTH))), " &
    ...
    ")" ;
  -- Advance I/O description
  attribute AIO_COMPONENT_CONFORMANCE of IC2 : entity is
    "STD_1149_6_2015";
  -- Document the parameters of the AIO ports.
  attribute AIO_PIN_BEHAVIOR of IC2 : entity is
    ...
    -- Programmable common-mode, fixed peak-to-peak voltages
    "DIFF_OUT_P: aio_vcm=programmable aio_vpp=340; " &
    -- Required board level coupling, no <cap spec>
    --   or <detection modifier>
    --   Programmable threshold, fixed hysteresis voltages
```

```
    "DIFF_INP_P: coupling_time=1.5e-8 " &
    "            aio_vth=programmable aio_vhyst=240; " &
    ...
  ;
end IC2 ;
```

7.8.1.3 Example IC2 Component PDL

An example of the IC2 Component PDL is a follows:

```
#############################################################
# Component IC2 has programmable AIO drivers and test receivers.
# The programmable parameters are not sensitive to Vdd, so
#   no scaling of voltages is required (vscale will default.)
# BSDL register descriptions include mnemonic values,
#   so these can be used by name in the PDL.
#############################################################
# first, define the context
iPDLLevel 1 -version STD_1149_1_2013
iProcGroup IC2

#############################################################
# The driver common-mode voltage procedure documentation
# This iProc sets a desired common-mode voltage for one or more
#   ports and returns a list of ports set.
# A port name is required as there are multiple AIO ports.
iProc AIO_set_VCM -export { voltage { port "" } { vscale "1.0" } } {

  # VHDL names are case insensitive.  Convert port names to all
  #   upper (or lower) case before comparisons.
  set uport [string toupper $port]

  # Verify that the port name is valid
  if {$uport == "DIFF_OUT_P" || $uport == "DIFF_OUT_N"} {

    # Verify that the supplied voltage value is legal for the
identified
    #   port and convert that value to a digital encoding and write it
    #   to the appropriate register field.
    if {$voltage==450} {
      # The PDL has to know the field name associated with each port
      iWrite Drvr_VCM 450MVCM
    } elseif {$voltage==600} {
      iWrite Drvr_VCM 600MVCM
    } elseif {$voltage==750} {
      iWrite Drvr_VCM 750MVCM

    } else {
      # Print a meaningful error message if the voltage is not valid
      puts "ERROR: The requested common-mode voltage '$voltage mv' is\
            unsupported on $PDL_CONTEXT_PATH.$port.\n"
      return "AIO_ERROR"
    }

  # If there are additional driver ports with programmable VCM,
```

```
  #   they would be handled in turn here.

  } else {
    # Print a meaningful error message if the port is not recognized.
    puts "ERROR: The requested port '$port' is unsupported for setting\
            common-mode voltage on $PDL_CONTEXT_PATH.\n"
    return "AIO_ERROR"
  }

  # Normally, many iApply cmds will be executed at once, so the iApply
  #   here defines a synchronization point and suspends this iProc
  #   until the iApply is actually executed.
  iApply
  return $port
}; # End of AIO_set_VCM procedure.

#####################################################################
# This iProc returns the common-mode voltage ready to be written
#   for a port.
# A port name is required as there are multiple AIO ports.
iProc AIO_get_VCM -export { { port "" } { vscale "1.0" } } {

  # VHDL names are case insensitive.  Convert port names to all
  #   upper (or lower) case before comparisons.
  set uport [string toupper $port]

  # Verify that the port name is valid
  if {$uport == "DIFF_OUT_P" || $uport == "DIFF_OUT_N"} {

    # iGet -si returns the value to be written to the register field.
    # iGet -mnem returns the mnemonic name whose value matches the
    #   register field contents.
    set common [iGet -si -mnem Drvr_VCM]
    # Force upper (or lower) case for comparisons
    set common [string toupper $common]

    # Convert the mnemonic names to actual voltages
    if {$common=="450MVCM"} {
      return "450"
    } elseif {$common=="600MVCM"} {
      return "600"
    } elseif {$common=="750MVCM"} {
      return "750"

    } else {
      puts "ERROR: Undefined error in $PDL_CONTEXT_PATH!\n"
      return "AIO_ERROR"
    }

  # If there are additional driver ports with programmable VCM,
  #   they would be handled in turn here.

  } else {
    # Print a meaningful error message if the port is not recognized.
    puts "ERROR: The requested port '$port' is unsupported for getting\
            common-mode voltage on $PDL_CONTEXT_PATH.\n"
    return "AIO_ERROR"
```

```
  }
}; # End of AIO_get_VCM procedure.

#####################################################################
# This iProc returns a list of the common-mode voltage settings
#   for a port.
# A port name is required as there are multiple AIO ports.
iProc AIO_getALL_VCM -export { { port "" } { vscale "1.0" } } {

  # VHDL names are case insensitive.  Convert port names to all
  #   upper (or lower) case before comparisons.
  set uport [string toupper $port]

  # Verify that the port name is valid
  if {$uport == "DIFF_OUT_P" || $uport == "DIFF_OUT_N"} {
    return "450 600 750"

  # If there are additional driver ports with programmable VCM,
  #   they would be handled in turn here.

  } else {
    # Print a meaningful error message if the port is not recognized.
    puts "ERROR: The requested port '$port' is unsupported for getting\
            all common-mode voltages on $PDL_CONTEXT_PATH.\n"
    return "AIO_ERROR"
  }
}; # End of AIO_getALL_VCM procedure.

#############################################################
# The receiver threshold voltage procedure documentation
#
#############################################################
# This iProc sets a desired threshold voltage for one or more
#   ports and returns a list of ports set.
# A port name is required as there are multiple AIO ports.
iProc AIO_set_VTH -export { voltage { port "" } { vscale "1.0" } } {

  # VHDL names are case insensitive.  Convert port names to all
  #   upper (or lower) case before comparisons.
  set uport [string toupper $port]

  # Verify that the port name is valid
  if {$uport == "DIFF_INP_P" || $uport == "DIFF_INP_N"} {

    # Verify that the supplied voltage value is legal for the
identified
    #   port and convert that value to a digital encoding and write it
    #   to the appropriate register field.
    if {$voltage==450} {
      # The PDL has to know the field name associated with each port
      iWrite TRcvr_VTH 450MVTH
    } elseif {$voltage==600} {
      iWrite TRcvr_VTH 600MVTH
    } elseif {$voltage==750} {
      iWrite TRcvr_VTH 750MVTH
    } elseif {$voltage==0} {
      puts "Warning: Detect_EXTEST_High mode set on \
```

```
            $PDL_CONTEXT_PATH.$port.\n"
      iWrite TRcvr_VTH Ground
    } elseif {$voltage==1200} {
      puts "Warning: Detect_EXTEST_Low mode set on \
            $PDL_CONTEXT_PATH.$port.\n"
      iWrite TRcvr_VTH VDD_IO

    } else {
      puts "ERROR: The requested threshold voltage '$voltage mv' is\
            unsupported on $PDL_CONTEXT_PATH.$port.\n"
      return "AIO_ERROR"
    }

  # If there are additional receiver ports with programmable VTH,
  #   they would be handled in turn here.

  } else {
    # Print a meaningful error message if the port is not recognized.
    puts "ERROR: The requested port '$port' is unsupported for setting\
            threshold voltage on $PDL_CONTEXT_PATH.\n"
    return "AIO_ERROR"
  }

  # Normally, many iApply cmds will be executed at once, so the iApply
  #   here defines a synchronization point and suspends this iProc.
  iApply
  return $port
}; # End of AIO_set_VTH procedure.

#####################################################################
# This iProc returns the threshold voltage ready to be written
#   for a port.
# A port name is required as there are multiple AIO ports.
iProc AIO_get_VTH -export { { port "" } { vscale "1.0" } } {

  # VHDL names are case insensitive.  Convert port names to all
  #   upper (or lower) case before comparisons.
  set uport [string toupper $port]

  # Verify that the port name is valid
  if {$uport == "DIFF_OUT_P" || $uport == "DIFF_OUT_N"} {

    # iGet -si returns the value to be written to the register field.
    # iGet -mnem returns the mnemonic name whose value matches the
    #   register field contents.
    set threshold [iGet -si -mnem TRcvr_VTH]
    # Force upper (or lower) case for comparisons
    set threshold [string toupper $threshold]

    # Convert the mnemonic names to actual voltages
    if {$threshold=="450MVTH"} {
      return "450"
    } elseif {$threshold=="600MVTH"} {
      return "600"
    } elseif {$threshold=="750MVTH"} {
      return "750"
    } elseif {$threshold=="GROUND"} {
```

```
      return "0"
    } elseif {$threshold=="VDD_IO"} {
      return "1200"
    } else {
      puts "ERROR: Undefined error in $PDL_CONTEXT_PATH!\n"
      return "AIO_ERROR"
    }

  # If there are additional receiver ports with programmable Vth,
  #   they would be handled in turn here.

  } else {
    # Print a meaningful error message if the port is not recognized.
    puts "ERROR: The requested port '$port' is unsupported for getting\
            threshold voltage on $PDL_CONTEXT_PATH.\n"
    return "AIO_ERROR"
  }
}; # End of AIO_get_VTH procedure.

#####################################################################
# This iProc returns a list of the threshold voltage settings
#   for a port.
# A port name is required as there are multiple AIO ports.
iProc AIO_getALL_VTH -export { { port "" } { vscale "1.0" } } {

  # VHDL names are case insensitive.  Convert port names to all
  #   upper (or lower) case before comparisons.
  set uport [string toupper $port]

  # Verify that the port name is valid
  if {$uport == "DIFF_OUT_P" || $uport == "DIFF_OUT_N"} {
    return "0 450 600 750 1200"

  # If there are additional driver ports with programmable VTH,
  #   they would be handled in turn here.

  } else {
    # Print a meaningful error message if the port is not recognized.
    puts "ERROR: The requested port '$port' is unsupported for getting\
            all threshold voltages on $PDL_CONTEXT_PATH.\n"
    return "AIO_ERROR"
  }
}; # End of AIO_getALL_VTH procedure.
```

7.8.1.4 Example Board Level PDL

PDL is procedure documentation. The board test software (or test engineer) has to select programmable values for the drivers and receivers that are compatible in both ac (EXTEST_PULSE or EXTEST_TRAIN) and dc (EXTEST) modes. For the board shown in Figure 72, it has been shown that Chan1 will require that the VTH of the test receiver on U2 will have to be set to 0mv for dc test, because the VCM of the U1 driver exceeds U2 VDD. It has also been shown that for Chan2, the default values will work for dc test. The parameters that might be set to ensure ac test robustness, namely AIO_VPP for the driver and AIO_VHyst for the test receiver, are not programmable in either IC1 or IC2, but their fixed values are compatible.

Somewhere in the initialization procedure at the board level, statements equivalent to the following would be expected:

```
#Board channel Chan1 requires a continuity check for shorted capacitor.
iCall U2.AIO_set_VTH 0 "Diff_Inp_P"
# Board channel Chan2 only requires default values
iCall U2.AIO_set_VCM 600 "Diff_Out_P"
iCall U3.AIO_set_VTH 600 "Diff_Inp_P"
```

7.8.2 Example of multiple programmable ports

The following example shows the IP deliverables (User Package and PDL) for a hard IP that supports four full duplex transmit and receive pairs of high-performance, programmable, differential I/O, each pair called a "Lane", with a single lane shown in Figure 75. Four lanes are then grouped, with the controlling TDR segment, to form the actual IP, as shown in Figure 76. This IP can be used to build as many lanes, in multiples of four, as desired in a component. Functional details are omitted here to focus on test, and in particular, board test using IEEE Std 1149.1 and 1149.6.

This example provides an opportunity to discuss some of the subtleties of coding the TDR fields controlling the AIO parameters, and AIO parameters themselves. First, assume for the moment that, in the logical hierarchy, the single lane AIO shown in Figure 75 is a hard IP, and the circuit of Figure 76 is a soft IP wrapper instantiating four of the AIO hard macros and the register segment programming the AIO. The soft IP could then be instantiated in a component. It would be difficult to code the AIO_PORT_BEHAVIOR attribute at the AIO hard macro level for several reasons. It may not be known to the designer of the AIO hard macro how the programming fields will be configured, for instance. Are all of the parameters to be controlled individually, or are some to be bus-programmed, as shown in Figure 76. In addition, there are register segment instance names resulting from REGISTER_ASSEMBLY, which allow pointing to specific instantiations of such register segments, but no similar mechanism for pointing to instances of IP that do not include register segments, such as the hard AIO IP. In the case where an AIO IP does not contain the controlling register segment, it is better to provide both the AIO_PORT_BEHAVIOR attribute documenting the parameters as well as the PDL documenting the procedures at the IP level where the register segment is implemented.

One can note that synthesis tools may very likely flatten the soft IP into the component netlist. From a test point of view, this does not preclude having the User Package and the PDL written for the soft IP that no longer exists as a separate entity in the netlist. The board test software does not have access to the component netlist and will not notice the difference. On the other hand, if the soft IP wrapper and the component are designed by the same team, it may be just as easy to simply document the AIO parameters in the component AIO_PIN_BEHAVIOR attribute, and to document the PDL behavior in the component level PDL procedure just as if the AIO had been instantiated directly in the component. Both ways are acceptable.

In order to illustrate coding bus-programmable AIO parameters, here we provide documentation at the level instantiating the controlling TDR segment, assuming that the circuits shown in both Figure 75 and Figure 76 are a single hard IP, as might happen if the AIO and the wrapper were being supplied by a third party.

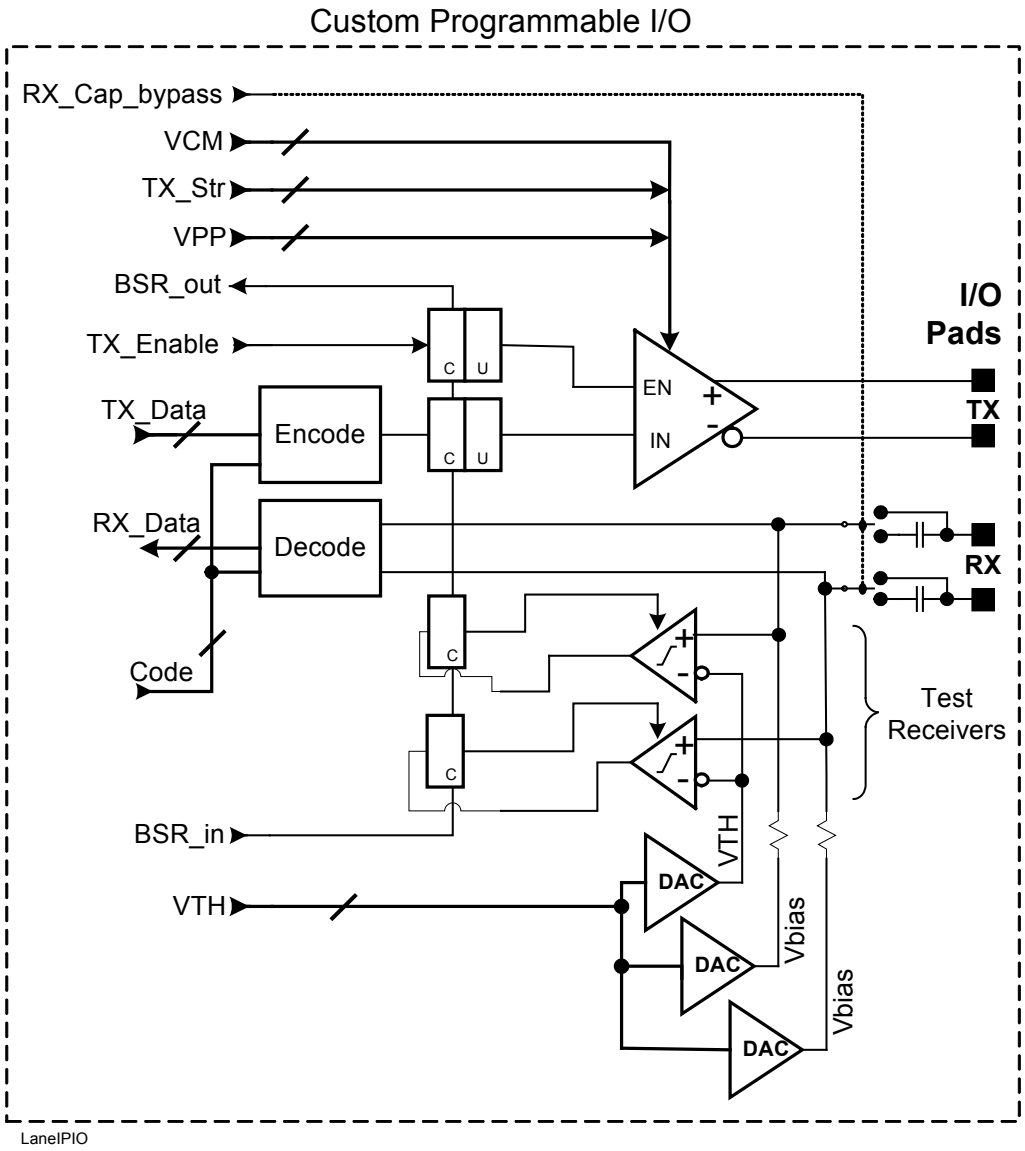


Figure 75—A single lane, showing TDRs, drivers and receivers

Note that the three DAC blocks in Figure 75 could simply be resistor divider networks switched by the register value. Resistors in CMOS are often implemented as a transistor with the appropriate "on" resistance. This makes it simple to implement resistor networks where individual resistors can be switched from a nominal value to a very high, essentially open, value. In this case, the actual value in millivolts achieved for VTH and VBias could be dependent on the I/O supply voltage, VDD. Similarly, depending on the design of the driver, the common mode voltage and the driver swing might have some dependence on VDD, especially for a commodity part designed to be used across a relatively wide range of VDD values.

Figure 76 shows four copies of the duplex lane (one driver, one receiver) of Figure 75, instantiated four times, plus the init_data register segment that controls the programmable parameters of the drivers and test receivers. Note that all of the parameters except VTH are bus programmable, that is, one register field feeds all four lanes.

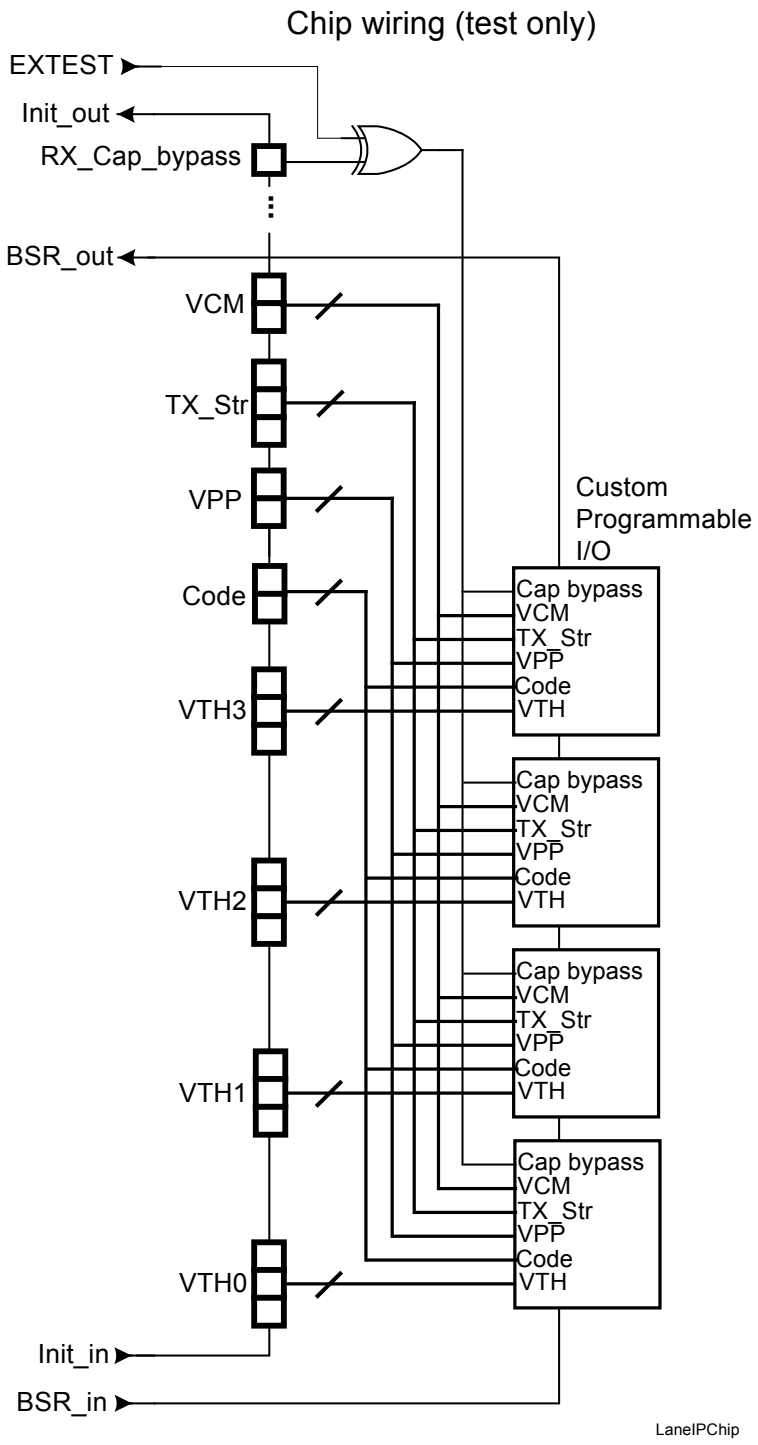


Figure 76—Instantiation of IP into the chip level

7.8.2.1 Example IP User Package file

In the following example, a fictional “Wendex Corp” defines IP for use in their own IC designs.

```
Package Wendex_Gen5_Lanes is
------------------------------------------------------------------------
--
-- The Wendex Corp fifth generation Lanes provides the very latest in
-- high-performance transmit and receive pairs. These are programmable
-- to use with multiple protocols, and a hard IP macro is defined for
-- a nibble of 4 independent lanes.
--
-- This IP is for the internal use of Wendex Corp for their line of
-- bridge components.
------------------------------------------------------------------------
--
  Use STD_1149_1_2013.all;
  Use STD_1149_6_2015.all;
End Wendex_Gen5_Lanes;

Package Body Wendex_Gen5_Lanes is
  Use STD_1149_1_2013.all;
  Use STD_1149_6_2015.all;
  attribute Register_Mnemonics of Wendex_Gen5_Lanes:package is
    -- Nominal VDD for this component is 1800mv.
    -- Other VDD values can require adjustment of the following values.
    "RX_Threshold ("&
      "Ground (0b000) <For Detect_EXTEST_High behavior>, "&
      "600mvTH (0b001), 900mvTH (0b010), 1200mvTH (0b100), "&
      "VDD (0b111) <For Detect_EXTEST_Low behavior> ), "&
    "Encoding ( "&
      "None (0b00), 6B8B (0b01), 8B10B (0b10), 64B66B (0b11)), "&
    "TX_Swing ( "&
      -- These values are for nominal VDD_IO of 1.8 Volts.
      "210mvPP (0b00), 300mvPP (0b01), 420mvPP (0b1x)), "&
    "TX_Strength ( "&
      "Disable  (0b000) "&
                "<Disable drive regardless of functional enable>,"&
      "14slices (0b001), "&
      "15slices (0b010), "&
      "16slices (0b011), "&
      "17slices (0b100), "&
      "18slices (0b101), "&
      "19slices (0b110), "&
      "20slices (0b111)), "&
    "TX_Common_Mode ( "&
      "Illegal (0b00) <Will produce unpredictable results>, "&
      -- These values are for nominal VDD_IO of 1.8 Volts.
      "600mvCM (0b01), 900mvCM (0b10), 1200mvCM (0b11)), "&
    "ON_OFF (ON (1), OFF (0))" ;

-- This segment should be part of the INIT_DATA register.

  attribute Register_Fields of Wendex_Gen5_Lanes:package is
    "lanex4_setup[22] ( "&
      -- TDI
      " (VTH0[3] IS (21 DOWNTO 19) NOPI TRSTRESET "&
      " RESETVAL (RX_Threshold(900mvTH))), "&
      " (VTH1[3] IS (18 DOWNTO 16) NOPI TRSTRESET "&
      " RESETVAL (RX_Threshold(900mvTH))), "&
      " (VTH2[3] IS (15 DOWNTO 13) NOPI TRSTRESET "&
```

```
      " RESETVAL (RX_Threshold(900mvTH))), "&
      " (VTH3[3] IS (12 DOWNTO 10) NOPI TRSTRESET "&
      " RESETVAL (RX_Threshold(900mvTH))), "&
      " (Code [2] IS (9 DOWNTO 8) NOPI TRSTRESET "&
      " RESETVAL (Encoding(None))), "&
      " (VPP[2] IS (7 DOWNTO 6) NOPI TRSTRESET "&
      " RESETVAL (TX_Swing(300mvPP))), "&
      " (TX_Str[3] IS (5 DOWNTO 3) TRSTRESET "&
      " RESETVAL (TX_Strength(17slices))), "&
      " (VCM[2] IS (2 DOWNTO 1) NOPI TRSTRESET "&
      " RESETVAL (TX_Common_Mode(900mvCM))), "&
      " (RX_Cap_Bypass[1] IS (0) NOPI TRSTRESET "&
      " RESETVAL (ON_OFF(Off))) )";;
      -- TDO

-- Each Lane (I/O cell) contains 4 boundary cells, as follows, using
-- local port-link names. Duplicate as needed for each lane (16
-- boundary cells total for this IP of 4 lanes.)
--
--  TDI
-- num cell  port/*  function      safe [ccell disval rslt]
--  3 (BC_4, RX_Neg, observe_only, X,    OPENX),
--  2 (BC_4, RX_Pos, observe_only, X,    OPENX),
--  1 (AC_1, TX_Pos, output3,      X,    0,     0,     Z),
--  0 (BC_1, *,      control,      0)
--  TDO

-- REGISTER_CONSTRAINTS could be defined here, if needed.

-- AIO attributes

  attribute AIO_Component_Conformance of Wendex_Gen5_Lanes:package is
      "STD_1149_6_2015";

-- Since all four inputs (Gen5_RX) and all four outputs (Gen5_TX)
-- are identical, they may be bussed and described together.
-- Note that the mapping from TDR fields (above) to port-link
-- names (below) is not coded here, but in the PDL.

  attribute AIO_Port_Behavior of Wendex_Gen5_Lanes:package is
    "port_link_list (Gen5_RX: in bit_vector (0 to 3); "&
      "              Gen5_TX: out bit_vector (0 to 3)); "&
    "Gen5_RX : HP_Time=15.0e-9 on_chip_AC_bus_programmable(grp1) "&
      "AIO_VTH=programmable AIO_VHyst=150;"&
    "Gen5_TX : AIO_VCM=bus_programmable(grp2) "&
      "AIO_VPP=bus_programmable(grp3) ";

End Wendex_Gen5_Lanes;
```

7.8.2.2 Example IP PDL

Even though this is an in-house IP, separate PDL is provided for the IP in order to better support reuse in different components and multiple instantiations in any one component.

```
############################################################
iPDLLevel 1 -version STD_1149_1_2013
iProcGroup Wendex_Gen5_Lanes

############################################################
# AIO_set_VCM.
# VCM is independent of the VDD_IO, so scaling is not used.
iProc AIO_set_VCM { voltage { port "" } { vscale "1.0" } } {
  # VHDL names are case insensitive.  To avoid confusion, convert
  # to upper case before comparisons.
  set upper_port [string toupper $port]
  set PORT_LIST { GRP2 GEN5_TX(0) GEN5_TX(1) GEN5_TX(2) GEN5_TX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port/Group $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
  if { $voltage == 600 } {
    iWrite VCM 600mvCM
  } elseif { $voltage == 900 } {
    iWrite VCM 900mvCM
  } elseif { $votage == 1200 } {
    iWrite VCM 1200mvCM
  } else {
    puts "The common-mode voltage $voltage is unsupported on \
Port/Group $PDL_CONTEXT_PATH.$port.\n"
    return "AIO_ERROR"
  }
  iApply
  return "grp2"
}; # End AIO_set_VCM procedure

############################################################
# AIO_get_VCM returns a voltage.
iProc AIO_get_VCM { { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
  set PORT_LIST { GRP2 GEN5_TX(0) GEN5_TX(1) GEN5_TX(2) GEN5_TX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port/Group $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
# iGet -mnem returns the mnemonic name whose value matches the
# register field contents.
  set myVCM [iGet -si -mnem VCM]
  # These comparisons to mnemonic names are CASE SENSITIVE.
  if {$myVCM == "600mvCM"} {
    return "600"
  } elseif {$myVCM == "900mvCM"} {
    return "900"
  } elseif {$myVCM == "1200mvCM"} {
    return "1200"
  } else {
# This should never happen due to reset on register field.
```

```
    puts "The AIO_get_VCM procedure detected an error on Port/Group\
 $PDL_CONTEXT_PATH.$port\n"
    return "AIO_ERROR"
  }
}; # End AIO_get_VCM procedure

############################################################
# AIO_getALL_VCM returns a list of voltages.
iProc AIO_getALL_VCM { { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
  set PORT_LIST { GRP2 GEN5_TX(0) GEN5_TX(1) GEN5_TX(2) GEN5_TX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "PortGroup $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
  return "600 900 1200"
}; # End AIO_getALL_VCM procedure

############################################################
# AIO_set_VPP
# VPP is independent of the VDD_IO, so scaling is not used.
iProc AIO_set_VPP { voltage { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
  set PORT_LIST { GRP3 GEN5_TX(0) GEN5_TX(1) GEN5_TX(2) GEN5_TX(3)}
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port/Group $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
  if { $voltage == 210 } {
    iWrite VPP 210mvPP
  } elseif { $voltage == 300 } {
    iWrite VPP 300mvPP
  } elseif { $voltage == 420 } {
    iWrite VPP 420mvPP
  } else {
    puts "The peak-to-peak voltage $voltage is unsupported for \
Port/Group $PDL_CONTEXT_PATH.$port.\n"
    return "AIO_ERROR"
  }
  iApply
  return "grp3"
}; # End AIO_set_VPP procedure

############################################################
# AIO_get_VPP returns a voltage.
iProc AIO_get_VPP { { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
  set PORT_LIST { GRP3 GEN5_TX(0) GEN5_TX(1) GEN5_TX(2) GEN5_TX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port/Group $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
# iGet -mnem returns the mnemonic name whose value matches the
# register field contents.
  set deltaV [iGet -si -mnem VPP]

  # These comparisons to mnemonic names are CASE SENSITIVE.
```

```
  if {$deltaV == "210mvPP"} {
    return "210"
  } elseif {$deltaV == "300mvPP" } {
    return "300"
  } elseif {$deltaV == "420mvPP" } {
    return "420"
  } else {
# This should never happen due to reset on register field.
    puts "The AIO_get_VPP procedure detected an error for Port/Group\
 $PDL_CONTEXT_PATH.$port\n"
    return "AIO_ERROR"
  }
}; # End AIO_get_VPP procedure

############################################################
# AIO_getALL_VPP returns a list of voltages.
iProc AIO_getALL_VPP { { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
  set PORT_LIST { GRP2 GEN5_TX(0) GEN5_TX(1) GEN5_TX(2) GEN5_TX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port/Group $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
  return "210 300 420"
}; # End AIO_getALL_VPP procedure

############################################################
# AIO_set_VTH
# VTH scales linearly with VDD_IO.  Nominal VDD_IO is 1.8V.
#   vscale should be set to (actual_VDD_IO/1.8)
iProc AIO_set_VTH { voltage { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
  set PORT_LIST { GEN5_RX(0) GEN5_RX(1) GEN5_RX(2) GEN5_RX(3) }
  set FIELD_LIST { VTH0 VTH1 VTH2 VTH3 }
  set findex [lsearch -exact $PORT_LIST $upper_port]
  if { $findex < 0 } {
    puts "Port $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  } else {
    set field [lindex $FIELD_LIST $findex]
  }
  # Scale the requested voltage to the nominal voltage
  set scaled_mv {%voltage/%vscale}
  if {$scaled_mv >= 590 && $scaled_mv <= 610} {
    iWrite $field 600mvTH
  } elseif {$scaled_mv >= 890 && $scaled_mv <= 910} {
    iWrite $field 900mvTH
  } elseif {$scaled_mv >= 1190 && $scaled_mv <= 1210} {
    iWrite $field 1200mvTH
  } elseif {$scaled_mv < 10 } {
    iWrite $field Ground
  } elseif {$scaled_mv > 1790 } {
    iWrite $field VDD
  } else {
    puts "The threshold voltage $voltage is unsupported for Port \
$PDL_CONTEXT_PATH.$port with current scaling factor.\n"
    return "AIO_ERROR"
```

```
  }
  iApply
  return $port
}; # End AIO_set_VTH procedure

############################################################
# AIO_get_VTH returns a voltage.
iProc AIO_get_VTH { { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
#  Could also be done with lsearch and lindex, as above.
  if {$upper_port == "GEN5_RX(0)"} {
# iGet -mnem returns the mnemonic name whose value matches the
# register field contents.
    set threshold [iGet -si -mnem VTH0]
  } elseif {$upper_port == "GEN5_RX(1)"} {
    set threshold [iGet -si -mnem VTH1]
  } elseif {$upper_port == "GEN5_RX(2)"} {
    set threshold [iGet -si -mnem VTH2]
  } elseif {$upper_port == "GEN5_RX(3)"} {
    set threshold [iGet -si -mnem VTH3]
  } else {
    puts "Port $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
  # These comparisons to mnemonic names are CASE SENSITIVE.
  if {$threshold == "600mvTH"} {
    return "[expr round(600*vscale)]"
  } elseif {$threshold == "900mvTH" } {
    return "[expr round(900*$vscale)]"
  } elseif {$threshold == "1200mvTH" } {
    return "[expr round(1200*$vscale)]"
  } elseif {$threshold == "Ground" } {
    return "0"
  } elseif {$threshold == "VDD" } {
    return "[expr round(1800*vscale)]"
  } else {
# This should never happen due to reset on register field.
    puts "The AIO_get_VTH procedure detected an error for Port\
 $PDL_CONTEXT_PATH.$port\n"
    return "AIO_ERROR"
  }
}; # End AIO_get_VTH procedure

############################################################
# AIO_getALL_VTH returns a list of voltages.
iProc AIO_getALL_VTH { { port "" } { vscale "1.0"} } {
  set upper_port [string toupper $port]
  set PORT_LIST { GEN5_RX(0) GEN5_RX(1) GEN5_RX(2) GEN5_RX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  }
  return "0\
   [expr round(600*$vscale)]\
   [expr round(900*$vscale)]\
   [expr round(1200*$vscale)]\
   [expr round(1800*$vscale)]"
```

```
}; # End AIO_getALL_VPP procedure

#########################################################
# AIO_set_DC_couple returns a list of ports.
iProc AIO_set_DC_couple { state { port "" } } {
  if { $state != "ON" && $state != "OFF" } {
      puts "Invalid state value for Port/Group \
$PDL_CONTEXT_PATH.$port.\n"
      return "AIO_ERROR"
     }
  set upper_port [string toupper $port]
  set PORT_LIST { GRP1 GEN5_RX(0) GEN5_RX(1) GEN5_RX(2) GEN5_RX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port/Group $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  } else {
    -- Since the mnemonic value names are the same as the legal state
    --   values, they may be used without conversion.
    iWrite RX_Cap_Bypass $state
  }
  return "grp1"
}; # End AIO_set_DC_couple

#########################################################
# AIO_get_DC_couple returns state of "ON" (DC) or "OFF" (AC).
iProc AIO_get_DC_couple { { port "" } } {
  set upper_port [string toupper $port]
  set PORT_LIST { GRP1 GEN5_RX(0) GEN5_RX(1) GEN5_RX(2) GEN5_RX(3) }
  if { [lsearch -exact $PORT_LIST $upper_port] < 0 } {
    puts "Port/Group $PDL_CONTEXT_PATH.$port is not recognized.\n"
    return "AIO_ERROR"
  } else {
# iGet -mnem returns the mnemonic name whose value matches the
# register field contents.
    return [iGet -si -mnem Rx_Cap_Bypass]
  }
}; # End AIO_get_DC_couple

# END Wendex_Gen5_Lanes iProcGroup

#########################################################
```

7.8.2.3 Example component partial BSDL

This example shows only those portions of the BSDL necessary to illustrate the instantiation of eight lanes (two IP) from the Wendex_Gen5_Lanes IP. There could be multiple other instantiations of this and other I/O IP. Elipses (...) show where BSDL is left out and are not themselves part of BSDL.

```
entity HystNetBridge_4_0 is
-- This file documents the test features of the fourth generation of
-- the Hysterical Networks High Speed I/O Bridge for large networks
-- by Wendex Corp.

generic (PHYSICAL_PIN_MAP : string := "BGA80");
port (
```

```
  ...
  West_HSIOTX_P, West_HSIOTX_N: out bit_vector(3 DOWNTO 0);
  West_HSIORX_P, West_HSIORX_N: in bit_vector(3 DOWNTO 0);
  East_HSIOTX_P, East_HSIOTX_N: out bit_vector(3 DOWNTO 0);
  East_HSIORX_P, East_HSIORX_N: in bit_vector(3 DOWNTO 0);
  ...
  GND: power_0 bit_vector (0 TO 14);
  VCC: power_pos bit_vector (0 TO 8);
  VDD_IO: power_pos bit_vector (0 TO 3);
  TDO:out bit; TMS, TDI, TCK, TRST:in bit);
  );

--Get IEEE Std 1149 attributes and definitions
use STD_1149_1_2013.all;
use STD_1149_6_2015.all;
use Wendex_Gen5_Lanes.all;

attribute COMPONENT_CONFORMANCE of HystNetBridge_4_0 : entity is
  "STD_1149_1_2013";

constant BGA80: PIN_MAP_STRING:=
  ...
  "West_HSIOTX_P:(B6,B8,E8,H8), "&
  "West_HSIOTX_N:(C6,C8,F8,J8), "&
  "West_HSIORX_P:(E6,A7,D7,G7), "&
  "West_HSIORX_N:(F6,B7,E7,H7), "&
  "East_HSIOTX_P:(A1,D1,G1,B2), "&
  "East_HSIOTX_N:(B1,E1,H1,C2), "&
  "East_HSIORX_P:(B0,E0,H0,E2), "&
  "East_HSIORX_N:(C0,F0,J0,F2), "&
  ...
  "GND:(D0,G0,F1,A2,G2,F3,A4,G4,F5,A6,G6,F7,A8,D8,G8), "&
  "VCC:(D2,C3,J3,D4,C5,J5,D6,C7,J7), "&
  "VDD_IO:(C1,J1,C7,J7), "&
  "TDO:H6, TMS:J6, TCK:J2, TDI:H2, TRST:J4";

attribute PORT_GROUPING of HystNetBridge_4_0 : entity is
  "Differential_Voltage ( "&
  -- Test receivers are independently terminated, and so are
  -- not grouped.
  ...
  "(West_HSIOTX_P(3),West_HSIOTX_N(3)),"&
  "(West_HSIOTX_P(2),West_HSIOTX_N(2)),"&
  "(West_HSIOTX_P(1),West_HSIOTX_N(1)),"&
  "(West_HSIOTX_P(0),West_HSIOTX_N(0)),"&
  "(East_HSIOTX_P(3),East_HSIOTX_N(3)),"&
  "(Esst_HSIOTX_P(2),East_HSIOTX_N(2)),"&
  "(Esst_HSIOTX_P(1),East_HSIOTX_N(1)),"&
  "(East_HSIOTX_P(0),East_HSIOTX_N(0)))";

attribute TAP_SCAN_IN of TDI: signal is true;
attribute TAP_SCAN_MODE of TMS: signal is true;
attribute TAP_SCAN_OUT of TDO: signal is true;
attribute TAP_SCAN_RESET of TRST: signal is true;
attribute TAP_SCAN_CLOCK of TCK: signal is (20.0e6, BOTH);

attribute COMPLIANCE_PATTERNS of HystNetBridge_4_0 : entity is
```

```
...

attribute INSTRUCTION_LENGTH of HystNetBridge_4_0 : entity is 8;

attribute INSTRUCTION_OPCODE of HystNetBridge_4_0 : entity is
  "EXTEST (10000000)," &
  "EXTEST_PULSE (10001000)," &
  "EXTEST_TRAIN (10001100)," &
  ...
  "INIT_SETUP (00001111, 10001111)";

attribute INSTRUCTION_CAPTURE of HystNetBridge_4_0 : entity is
  "10000001";

attribute IDCODE_REGISTER of HystNetBridge_4_0 : entity is
   ...

attribute REGISTER_ACCESS of HystNetBridge_4_0 : entity is
  "BOUNDARY (EXTEST_PULSE, EXTEST_TRAIN, ...)," &
  ...
  "INIT_DATA[*] (INIT_SETUP)";

attribute BOUNDARY_LENGTH of HystNetBridge_4_0 : entity is 148;

attribute BOUNDARY_REGISTER of HystNetBridge_4_0 : entity is
-- num cell port function safe [input/ccell disval rslt]
-- TDI
  "147 (...), " &
  ...
  -- Copied and modified from package Wendex_Gen5_Lanes
  "103 (BC_4, East_HSIORX_N(3), observe_only, X, OPENX),"&
  "102 (BC_4, East_HSIORX_P(3), observe_only, X, OPENX),"&
  "101 (AC_1, East_HSIOTX_P(3), output3, X, 100, 0, Z),"&
  "100 (BC_1, *, control, 0),"&
  " 99 (BC_4, East_HSIORX_N(2), observe_only, X, OPENX),"&
  " 98 (BC_4, East_HSIORX_P(2), observe_only, X, OPENX),"&
  " 97 (AC_1, East_HSIOTX_P(2), output3, X, 96, 0, Z),"&
  " 96 (BC_1, *, control, 0),"&
  " 95 (BC_4, East_HSIORX_N(1), observe_only, X, OPENX),"&
  " 94 (BC_4, East_HSIORX_P(1), observe_only, X, OPENX),"&
  " 93 (AC_1, East_HSIOTX_P(1), output3, X, 92, 0, Z),"&
  " 92 (BC_1, *, control, 0),"&
  " 91 (BC_4, East_HSIORX_N(0), observe_only, X, OPENX),"&
  " 90 (BC_4, East_HSIORX_P(0), observe_only, X, OPENX),"&
  " 89 (AC_1, East_HSIOTX_P(0), output3, X, 88, 0, Z),"&
  " 88 (BC_1, *, control, 0),"&
  -- END Package Wendex_Gen5_Lanes
  ...
  -- Copied and modified from package Wendex_Gen5_Lanes
  " 53 (BC_4, West_HSIORX_N(3), observe_only, X, OPENX),"&
  " 52 (BC_4, West_HSIORX_P(3), observe_only, X, OPENX),"&
  " 51 (AC_1, West_HSIOTX_P(3), output3, X, 50, 0, Z),"&
  " 50 (BC_1, *, control, 0),"&
  " 49 (BC_4, West_HSIORX_N(2), observe_only, X, OPENX),"&
  " 48 (BC_4, West_HSIORX_P(2), observe_only, X, OPENX),"&
  " 47 (AC_1, West_HSIOTX_P(2), output3, X, 46, 0, Z),"&
  " 46 (BC_1, *, control, 0),"&
```

```
  " 45 (BC_4, West_HSIORX_N(1), observe_only, X, OPENX),"&
  " 44 (BC_4, West_HSIORX_P(1), observe_only, X, OPENX),"&
  " 43 (AC_1, West_HSIOTX_P(1), output3, X, 42, 0, Z),"&
  " 42 (BC_1, *, control, 0),"&
  " 41 (BC_4, West_HSIORX_N(0), observe_only, X, OPENX),"&
  " 40 (BC_4, West_HSIORX_P(0), observe_only, X, OPENX),"&
  " 39 (AC_1, West_HSIOTX_P(0), output3, X, 38, 0, Z),"&
  " 38 (BC_1, *, control, 0),"&
  -- END Package Wendex_Gen5_Lanes
  ...
  -- TDO

...

attribute REGISTER_ASSEMBLY of HystNetBridge_4_0 : entity is
  "init_data ( "&
  -- TDI
    ...
    "(USING Wendex_Gen5_Lanes), "&
    "(East_Lanes IS Lanex4_setup), "&
    "(USING -), "&
    ...
    "(West_Lanes IS PACKAGE Wendex_Gen5_Lanes:Lanex4_setup), "&
    ...
  -- TDO
  ") ";

-- IEEE STD 1149.6 attributes

attribute AIO_Component_Conformance of HystNetBridge_4_0 : entity is
  "STD_1149_6_2015";

-- AIO_EXTEST_PULSE_EXECUTION could be defined here, if needed.
-- AIO_EXTEST_TRAIN_EXECUTION could be defined here, if needed.

attribute AIO_PIN_BEHAVIOR of HystNetBridge_4_0 : entity is
  ...
  " East_HSIORX_P : PACKAGE Wendex_Gen5_Lanes : Gen5_RX ; "&
  " East_HSIOTX_P : PACKAGE Wendex_Gen5_Lanes : Gen5_TX ; "&
  " West_HSIORX_P : PACKAGE Wendex_Gen5_Lanes : Gen5_RX ; "&
  " West_HSIOTX_P : PACKAGE Wendex_Gen5_Lanes : Gen5_TX  ";

end HystNetBridge_4_0;
```

7.8.2.4 Example component PDL

This example shows only those portions of the PDL necessary to control and observe the instantiation of eight lanes (two IP) from the Wendex_Gen5_Lanes IP. There could be multiple other instantiations of this and other I/O IP. Ellipses (...) show where PDL is left out and are not themselves part of PDL.

Note that all three of the VCM procedures are essentially identical except for procedure names. Rather than continue to repeat this pattern, the required VPP, VTH, and DC_couple procedures would be the same except for the changed parameter name, and for VTH, which is not bus-programmed, the procedure would have to recognize each of the eight ports individually and call the appropriate port-link name instead of a group name.

If there were AIO instantiated directly in the component instead of being instantiated in an IP, then these routines would have to have code similar to the IP code in 7.8.2.3, though a callable procedure could be used to handle them, depending on coding style. For instantiated IP with supplied PDL, the main task for the component level routines is converting the actual port names as defined in the BSDL to the corresponding port-link names defined in the PDL supplied with the IP. This is the same correspondence as in the AIO_Pin_Behavior attribute in the BSDL. In addition, the IP procedure must be called with the correct instance name or names (East_Lanes or West_Lanes, in this case) as defined in the REGISTER_ASSEMBLY attributes in the component BSDL.

```
#########################################################

iPDLLevel 1 -version STD_1149_1_2013
iProcGroup HystNetBridge_4_0

#########################################################
# AIO_set_VCM.
iProc AIO_set_VCM -export { voltage { port "" } { vscale "1.0" } } {
  set pport [string toupper $port]
  # Convert real port names to IP port-link names
  set PORT_LIST "EAST_HSIOTX_P EAST_HSIOTX_P(0) EAST_HSIOTX_P(1) \
EAST_HSIOTX_P(2) EAST_HSIOTX_P(3)"
  if { [lsearch -exact $PORT_LIST $pport] >= 0 } {
    return [iCall East_Lanes.AIO_set_VCM $voltage grp2 $vscale]
  }
  set PORT_LIST "WEST_HSIOTX_P WEST_HSIOTX_P(0) WEST_HSIOTX_P(1) \
WEST_HSIOTX_P(2) WEST_HSIOTX_P(3)"
  if { [lsearch -exact $PORT_LIST $pport] >= 0 } {
    return [iCall West_Lanes.AIO_set_VCM $voltage grp2 $vscale]
  }
# ... Handle other AIO
  } elseif {$port == ""} {
    puts "Null port name is not permitted in component \
$PDL_CONTEXT_PATH\n"
    return "AIO_ERROR"
  } else {
    puts "Port $PDL_CONTEXT_PATH.$port not recognized.\n"
    return "AIO_ERROR"
  }
}; # End AIO_set_VCM procedure

#########################################################
# AIO_get_VCM returns a voltage.
iProc AIO_get_VCM -export { { port "" } { vscale "1.0" } } {
  set pport [string toupper $port]
  # Convert real port names to IP port-link names
  set PORT_LIST "EAST_HSIOTX_P EAST_HSIOTX_P(0) EAST_HSIOTX_P(1) \
EAST_HSIOTX_P(2) EAST_HSIOTX_P(3)"
  if { [lsearch -exact $PORT_LIST $pport] >= 0 } {
    return [iCall East_Lanes.AIO_get_VCM grp2 $vscale]
  }
  set PORT_LIST "WEST_HSIOTX_P WEST_HSIOTX_P(0) WEST_HSIOTX_P(1) \
WEST_HSIOTX_P(2) WEST_HSIOTX_P(3)"
  if { [lsearch -exact $PORT_LIST $pport] >= 0 } {
    return [iCall West_Lanes.AIO_get_VCM grp2 $vscale]
  }
# ... Handle other AIO
  } elseif {$port == ""} {
```

```
    puts "Null port name is not permitted in $PDL_CONTEXT_PATH\n"
    return "AIO_ERROR"
  } else {
    puts "Port $PDL_CONTEXT_PATH.$port not recognized.\n"
    return "AIO_ERROR"
  }
}; # End AIO_get_VCM procedure

############################################################
# AIO_getALL_VCM returns a list of voltages.
iProc AIO_getALL_VCM -export { { port "" } { vscale "1.0" } } {
  set pport [string toupper $port]
  # Convert real port names to IP port-link names
  set PORT_LIST "EAST_HSIOTX_P EAST_HSIOTX_P(0) EAST_HSIOTX_P(1) \
EAST_HSIOTX_P(2) EAST_HSIOTX_P(3)"
  if { [lsearch -exact $PORT_LIST $pport] >= 0 } {
    return [iCall East_Lanes.AIO_getALL_VCM grp2 $vscale]
  }
  set PORT_LIST "WEST_HSIOTX_P WEST_HSIOTX_P(0) WEST_HSIOTX_P(1) \
WEST_HSIOTX_P(2) WEST_HSIOTX_P(3)"
  if { [lsearch -exact $PORT_LIST $pport] >= 0 } {
    return [iCall West_Lanes.AIO_getALL_VCM grp2 $vscale]
  }
  ... Handle other AIO
  } elseif {$port == ""} {
    puts "Null port name is not permitted in component  \
$PDL_CONTEXT_PATH\n"
    return "AIO_ERROR"
  } else {
    puts "Port $PDL_CONTEXT_PATH.$port not recognized.\n"
    return "AIO_ERROR"
  }
}; # End of AIO_getALL_VCM

# ...

# END HystNetBridge_4_0 iProcGroup

############################################################
```

Annex A

(informative)

Applications and tools

A.1 Chip compliance checking and BSDL and PDL verification

Compliance checking requires verifying that the behavior of a design, as observed from its I/O pins, conforms to the rules and permissions of this standard. It is recommended that compliance checking be performed as part of both pre-silicon and post-silicon verification efforts. Pre-silicon compliance checking is generally done using simulation and helps to assure that the IC complies with the standard prior to fabrication. Post-silicon compliance checking is recommended as a final certification step and to verify that the BSDL file matches the silicon.

NOTE—Whereas pre-silicon compliance checking for IEEE Std 1149.1 has required only digital simulation, the input test receivers for this standard mandate certain analog circuitry and characteristics that could require other methods for compliance verification. These could include analog simulations and/or structural checks of the design.

As the standard described herein utilizes and is compatible with the existing IEEE Std 1149.1, it must first comply with IEEE Std 1149.1. The additional compliance checking required for this standard is summarized in the following subclauses and is intended to serve as a guide for new compliance checks. The summary outlines the additional behaviors introduced by this standard that must be checked, but does not necessarily describe how to perform the compliance checking. The methods of compliance checking are left up to the individual tool vendors and designers. Refer to Clause 7 for comprehensive conformance and documentation requirements. Refer to Clause 5 and Clause 6 for information on instructions and pin implementation detail.

In practice, automated tools that read the BSDL description of a device will generate the TAP Controller sequences and TDI/TDO data needed to perform compliance checking. BSDL provides these tools with the details of a device's implementation, such as the bit patterns for each instruction, the length of various registers, and the association of boundary-scan register cells with pins. This has the desirable side-effect of verifying the BSDL description matches the implementation. In many cases, if the BSDL is inaccurate the verification process for the device will fail. Verification tools can be used to maximize the checking of the correspondence between the BSDL description and the actual implementation.

A.1.1 Verification of ac test instructions

Two instructions are mandated for testing of ac pins. These instructions provide an edge-detecting test for signal paths containing ac pins. The EXTEST_PULSE instruction provides a pulse of data on an ac driver with the pulse width controlled by the time spent in the *Run-Test/Idle* TAP Controller state. This produces a single "wide" pulse of variable, controllable period. The EXTEST_TRAIN instruction provides a pulse train the edges of which are generated by each falling edge of TCK while in the *Run-Test/Idle* TAP Controller state, thus generating a train of pulses. DC pins behave according to the IEEE Std 1149.1 EXTEST instruction when either the EXTEST_PULSE or EXTEST_TRAIN instructions are active.

A.1.1.1 EXTEST_PULSE

The following behavior should be verified for the EXTEST_PULSE instruction.

a) Verify that the EXTEST_PULSE instruction is provided in the BSDL description.

b) Verify that the EXTEST_PULSE instruction becomes active at the falling edge of TCK in the *Update-IR* TAP Controller state.

c) Verify that all output pins or bidirectional pins in output mode behave in accordance with the rules in 5.3.1 while the EXTEST_PULSE instruction is active. AC pins should be tested with both ac behavior selected and dc behavior selected (if ac/dc selection cells are provided), as well as enabled and disabled (if CONTROL cells are provided). DC pins should be tested both enabled and disabled (if CONTROL cells are provided.) Verification should be of all pins in parallel. To recap the behavior of an ac output pin that is ac selected and enabled, the output should be observed as follows:

    1) Single-ended ac pins and the positive leg of differential ac drivers should drive the logic value as loaded in the Update stage of the associated boundary-scan Register cell. The negative leg of differential ac drivers should drive the opposite value.

    2) AC pins should transition to their inverted logic values on the first falling edge of TCK after entering the *Run-Test/Idle* TAP Controller state. The ac pins should then remain at these values as long as the TAP Controller remains in the *Run-Test/Idle* state.

    3) AC pins should be restored to their original logic values on the first falling edge of TCK after leaving the *Run-Test/Idle* TAP Controller state.

    4) If the *Run-Test/Idle* TAP Controller state is not entered, then the ac pins should not toggle, but should behave as if the IEEE Std 1149.1 EXTEST instruction is active.

d) Check that system values are driven out onto ac pins in the Test-Logic-Reset state of the TAP Controller.

A.1.1.2 EXTEST_TRAIN

The following behavior should be verified for the EXTEST_TRAIN instruction.

a) Verify that the EXTEST_TRAIN instruction is provided in the BSDL description.

b) Verify that the EXTEST_TRAIN instruction becomes active at the falling edge of TCK in the *Update-IR* TAP Controller state.

c) Verify that all output pins or bidirectional pins in output mode behave in accordance with the rules in 5.4.1 while the EXTEST_TRAIN instruction is active. AC pins should be tested with both ac behavior selected and dc behavior selected (if ac/dc selection cells are provided), as well as enabled and disabled (if CONTROL cells are provided). DC pins should be tested both enabled and disabled (if CONTROL cells are provided.) Verification should be of all pins in parallel. To recap the behavior of an ac output pin that is ac selected and enabled, the output would be observed as follows:

    1) Single-ended ac pins and the positive leg of differential ac drivers should drive the logic value as loaded in the Update stage of the associated boundary-scan register cell. The negative leg of differential ac drivers should drive the opposite value.

    2) AC pins should transition to their inverted logic values on the first falling edge of TCK after entering the *Run-Test/Idle* TAP Controller state.

    3) With each successive falling edge of TCK while in the *Run-Test/Idle* TAP Controller state the ac pins should toggle their logic value (i.e., the inverse of the value generated on the previous falling edge of TCK).

4) AC pins should be restored to their original logic values (i.e., the value observed at the start of this sequence) no later than the first falling edge of TCK after leaving the *Run-Test/Idle* TAP Controller state.

5) If the *Run-Test/Idle* TAP Controller state is not entered, then the ac pins should not toggle, but should behave as if the IEEE Std 1149.1 EXTEST instruction is active.

6) This entire sequence should be executed twice, once with an even number of TCK cycles while in the *Run-Test/Idle* TAP Controller state, and again with an odd number of TCK cycles. For the even TCK cycles, the driver will be noninverted upon exit from the *Run-Test/Idle* TAP Controller state. For the odd TCK cycles, the driver will still be inverted upon exit from the *Run-Test/Idle* TAP Controller state, and will switch to the noninverted state in the *Select-DR-Scan* TAP Controller state.

A.1.2 Verification of ac pin behavior

This standard specifies input test receivers, output drivers, and ac/dc selection cells for ac pins that are designed to support testing of both dc- and ac-coupled signals. AC pin output drivers are provisioned with test circuitry such that they can deliver a test signal in place of the normal mission mode data. It is expected that the generated test signals be driven out of the mission mode driver at the normal system drive levels and edge rates. Input test receivers reconstruct the signal, as driven by an output driver, when an ac test instruction is active. However, when the IEEE Std 1149.1 EXTEST instruction is in effect, the test receivers should behave as a level detector. AC/DC selection cells are optional and provide the ability to disable the ac test signal on individual output drivers, causing them to behave as dc EXTEST drivers.

It is assumed in this discussion that if any of the characteristics of the driver or test receiver are programmable, an appropriate value has been selected and initialized before the following tests are performed.

A.1.2.1 Input test receivers

The following properties and behavior should be verified for input test receivers. Verify that all ac input and bidirectional pins have a test receiver monitoring the associated pin. Single-ended pins should have one test input receiver, and differential pin pairs should have one input test receiver per leg. This check requires verifying that each input test receiver operates in both level-detect and edge-detect modes. Additionally, it should be verified that each input test receiver meets the required designer specified parameters for ΔV_{Min}, T_{Trans}, V_{Hyst_Level}, V_{Bias} and V_{Hyst_Edge}. Verification of level-detection and edge-detection operation is described below. Unless otherwise stated, the input voltage should be held or returned to $V_{Threshold}$. ($V_{Threshold}$ is the termination or bias voltage established by the receiver for the pin and is documented in the BSDL.)

Whenever either of the analog parameters (threshold voltage and hysteretic voltage) for the test receiver are documented as programmable in the BSDL, the following tests should be repeated for each programmable value. Note that if a threshold voltage has programmable values that essentially match the supply voltage or ground, those programmable values should be treated as having the **Detect_EXTEST_Low** or **Detect_EXTEST_High** keywords, respectively.

NOTE—The voltage and time parameters named above, and used in the following tests, are not all specified in the BSDL. They must be provided separately. It might be appropriate for detailed parametric verification of the I/O circuit to be performed by the I/O circuit designers prior to silicon, and then incorporated into the chip manufacturing test. Then verification of this standard for the chip could use relaxed parametric values simply to verify that the I/O circuits, boundary cells, and TAP are connected and behaving properly (see A.2). These values will still have to be communicated via means other than BSDL.

a) Check that the input test receiver for each ac input pin operates in level-detection mode. This can be done using the EXTEST instruction. The following verification procedure assumes that any on-chip coupling capacitor has been bypassed.

1) Verify that a valid logic one is detected and captured in the capture stage of the boundary-scan register cell associated with the ac input pin. Preload the associated boundary-scan register cell with the logic value of zero. In the *Capture-DR* TAP Controller state, after the falling edge and prior to the next rising edge of TCK, the input pin should be driven to a voltage level greater than $V_{Threshold} + V_{Hyst\_Level}$ and held for a minimum duration of T_{Hyst}. A logic one should have been captured in the associated boundary-scan register cell and scanned out. This test is not performed if this pin has the **Detect_EXTEST_Low** keyword in its description.

2) Verify that a valid logic zero is detected and captured in the capture stage of the boundary-scan register cell associated with the ac input pin. Preload the associated boundary-scan register cell with the logic value of one. In the *Capture-DR* TAP Controller state, after the falling edge and prior to the next rising edge of TCK, the input pin should be driven to a voltage level less than $V_{Threshold} - V_{Hyst\_Level}$ for a minimum duration of T_{Hyst}. A logic zero should have been captured in the associated boundary-scan register cell and scanned out. This test is not performed if this pin has the **Detect_EXTEST_High** keyword in its description.

3) Verify that an invalid logic one is not detected and captured, and that the value previously scanned onto the capture stage of the boundary-scan register cell associated with the ac pin is returned unchanged. Preload the associated boundary-scan register cell with the logic value of zero. After the *Update-DR* TAP Controller state but prior to the *Capture-DR* TAP Controller state, the input pin should be driven to a voltage level greater than $V_{Threshold} + V_{Hyst\_Level}$ and held for a minimum duration of T_{Hyst}. The voltage is then reduced to less than $V_{Threshold} + V_{Hyst\_Level}$ and greater than $V_{Threshold} - V_{Hyst\_Level}$, and held until the end of the *Capture-DR* TAP Controller state. A logic zero should have been captured in the associated boundary-scan register cell. This test is not performed if this pin has the **Detect_EXTEST_Low keyword** in its description.

4) Verify that an invalid logic zero is not detected and captured, and that the value previously scanned into the capture stage of the boundary-scan register cell associated with the ac pin is returned unchanged. Preload the associated boundary-scan register cell with the logic value of one. After the *Update-DR* TAP Controller state but prior to the *Capture-DR* TAP Controller state, the input pin should be driven to a voltage level less than $V_{Threshold} - V_{Hyst\_Level}$, and held for a minimum duration of T_{Hyst}. The voltage is then raised to more than $V_{Threshold} - V_{Hyst\_Level}$ and less than $V_{Threshold} + V_{Hyst\_Level}$, and held until the end of the *Capture-DR* TAP Controller state. A logic one should have been captured in the associated boundary-scan register cell. This test is not performed if this pin has the **Detect_EXTEST_High keyword** in its description.

NOTE— This verification can be done by means of an analog simulation.

5) Where optionally implemented, verify that boundary-scan register cells on input receivers of ac input pins capture a fixed logic one or zero value with the SAMPLE instruction.

b) Check that the input test receiver for each ac input or bidirectional pin operates in edge-detection mode. This should be done for both the EXTEST_PULSE and EXTEST_TRAIN instructions.

1) Verify that with no transition on the input, the chip captures the initialization value for both one and zero.

2) Verify that a valid rising (i.e., low to high) transition is detected and a logic one is captured into the boundary-scan register cell associated with the ac input pin. Preload the associated

boundary-scan register cell with the logic value of zero. In the *Run-Test/Idle* TAP Controller state, hold the input at $V_{Threshold}$ for the duration of T_{Test} and then apply a rising transition of magnitude greater than V_{Hyst_Edge} with a rise time appropriate for the technology. Hold the high voltage value for a minimum duration of T_{Hyst}. A logic one should have been captured in the associated boundary-scan register cell.

3) Verify that a valid falling (i.e., high to low) transition is detected and a logic zero is captured into the boundary-scan register cell associated with the ac input or bidirectional pin. Preload the associated boundary-scan register cell with the logic value of one. In the *Run-Test/Idle* TAP Controller state, hold the input at $V_{Threshold}$ for a duration of T_{Test} and then apply a falling transition of magnitude greater than V_{Hyst_Edge} with a fall time appropriate for the technology. Hold the low voltage value for a minimum duration of T_{Hyst}. A logic zero should have been captured in the associated boundary-scan register cell.

4) Verify that the test input receiver does not respond to an invalid rising transition. Preload the associated boundary-scan register cell with the logic value of zero. Apply each of the following invalid rising transitions in the *Run-Test/Idle* TAP Controller state. Check that a logic zero is captured in the associated boundary-scan register cell in each case.

5) Transition magnitude less than V_{Hyst_Edge} but with appropriate rise time and duration greater than T_{Hyst}.

6) Transition magnitude equal to ΔV_{Max} and with rise time greater than T_{HP} and duration greater than T_{Hyst}.

7) Transition magnitude equal to ΔV_{Max} and with appropriate rise time and duration less than T_{Hyst}.

c) Verify that the test input receiver does not respond to an invalid falling transition. Preload the associated boundary-scan register cell with the logic value of one. Apply each of the following invalid falling transitions in the *Run-Test/Idle* TAP Controller state. Check that a logic one is captured in the associated boundary-scan register cell in each case.

1) Transition magnitude less than V_{Hyst_Edge} but with appropriate fall time and duration greater than T_{Hyst}.

2) Transition magnitude equal to ΔV_{Max} and with fall time greater than T_{HP} and duration greater than T_{Hyst}.

3) Transition magnitude equal to ΔV_{Max} and with appropriate fall time and duration less than T_{Hyst}.

A.1.2.2 Output drivers

In addition to the driver behavior verified during compliance checking of the EXTEST_PULSE and EXTEST_TRAIN instruction, the following additional properties and behavior should be verified for output drivers.

a) For IEEE Std 1149.1 instructions that drive outputs (EXTEST, CLAMP, and optionally RUNBIST or INTEST), verify the following:

   1) That single-ended and differential drivers for an ac output pin or channel drive data from a single associated boundary-scan register cell. For example, enable all output drivers and update a single boundary-scan register cell with a unique data logic value (i.e., all other register cells are the opposite logic value). The unique logic value should be observed on a single ac output pin or differential channel. Single-ended ac output pins should output a value that matches the value stored in the Update stage of the associated boundary-scan register cell. A differential ac output channel should drive a logic value that matches the value stored in the Update stage of the associated boundary-scan register cell on its positive leg pin, and the opposite value on its negative leg pin, and these pins should correspond to the representative and associated ports of the Port_Grouping statement, respectively.

   2) Check that, when the content of the Update flip-flop of the associated boundary-scan register cell changes, each enabled output driver of an ac output pin or channel produces the correct transitions on its output. The transitions should match the levels and edge slew rates of the mission performance specified for the driver.

   3) Check that the measured high and low output voltages are valid for the specified common-mode and peak-to-peak voltages specified for the port. Whenever either of those parameters are documented as programmable in the BSDL, verify each programmable value.

   4) Check that a disabled output driver of an ac output pin or channel produces a quiescent (i.e., nondriven) state on its output(s).

   5) Where implemented, verify that each such single-ended or differential driver for an ac output pin or channel are disabled or enabled via the content of a control cell in an associated boundary-scan register cell, per the rules within IEEE Std 1149.1.

   6) Where optionally implemented, verify that boundary-scan register cells on output drivers of ac output pins capture a fixed logic one or zero value with the SAMPLE instruction.

b) For the standard EXTEST_PULSE and EXTEST_TRAIN ac test instructions, verify the following:

   1) Check that at least one single-ended and differential driver for an ac output pin or channel drives data from a single associated boundary-scan register cell and that the cell is the same register cell as the one associated with the pin in item 1) above.

   2) Check that at least one disabled output driver of an ac output pin or channel produce a quiescent, i.e., nondriven state on its output(s).

NOTE—These checks need not be done on all ac outputs since correspondence mapping between pins and cells is already known.

A.1.2.3 AC/DC selection cells

This standard provides for optional ac/dc selection cells to allow the ability to disable the ac test signal generated at output drivers during the EXTEST_PULSE or EXTEST_TRAIN instructions. When the ac behavior is disabled, the output driver provides a fixed logic level, as if the dc EXTEST instruction was active.

The following behavior should be verified for all ac/dc selection cells:

— Verify that each ac/dc selection cell controls the data cell of at least one ac pin driver and that each data cell associated with an ac pin driver has no more than one ac/dc selection cell.

— Verify that ac/dc selection cells are compliant to the control cell structure mandated by IEEE Std 1149.1. Each cell should be composed of a Shift stage and an Update stage and should shift data from TDI towards TDO on the rising edge of TCK while in the *Shift-DR* TAP Controller state. AC/DC selection cells should subsequently transfer the content of the Shift stage to the Update stage on the falling edge of TCK in the *Update-DR* TAP Controller state.

— Sequentially update each ac/dc selection cell with a 0 and verify that any associated ac pin data cells controlled by the ac/dc selection cell drive their fixed logic levels. This should be done for both the EXTEST_PULSE and EXTEST_TRAIN instructions, verifying that ac pin output drivers behave as if EXTEST is in effect.

NOTE—This proves both the operation of the ac/dc selection cells and their mapping to ac output pins.

— AC/DC selection cells could capture fixed values or the content of their Update stages. Verify that each ac/dc selection cell captures the data as specified in the BSDL. Capture should occur on the rising edge of TCK in the *Capture-DR* TAP Controller state.

A.1.2.4 PDL routines for programmable features

Some PDL code may also be supplied to support the setup of programmable I/O features (see 7.7). These should be (where possible) executed to verify that they perform as intended. If a tester is used that cannot directly read PDL, then a fallback would be to translate the PDL into that tester's language before execution.

A.2 Functional verification

Functional verification takes place at volume production of devices that are known to be properly designed. Many of the tests (or physical implementations of simulations) from A.1 could be repeated as a functional verification, but these will likely be overly time-consuming and focused on problems (like the correspondence of BSDL to silicon) that should no longer be of concern.

Production testing should focus on proving the testability circuitry was fabricated properly, and this can be divided into two main tasks: verifying the testability logic and the analog characteristics of the test receivers.

The testability logic could very well be subordinated into a higher level testability scheme such as internal scan, and thus tested as a portion of the general logic of the device. If this is not the case, then some of the test processes from A.1 can be adapted and streamlined for production test of the logic.

The analog characteristics of test receivers can be tested by controlling the edge rates and voltage swings of waveforms applied to them. Referring to Figure 42, this amounts to setting up input waveforms with known valid edges and known invalid edges, where transitions should be detected and transitions should be rejected respectively.

A.3 Board interconnection testing

This subclause specifies upgrades required to current IEEE Std 1149.1-based board test processes to support features defined in this standard. These recommendations assume a typical existing set of board test tools as a starting point, with basic interconnect test capability but no differential capability.

Test generation is usually largely automated, but then requires checking to verify it will run correctly. This is usually performed with a "known good" UUT, which should pass since it is defect-free. However, some nets might show errors, particularly if certain ac parameters of their drivers and/or receivers are not

appropriately matched for signal transmission. The 2015 revision of this standard offers much more information in BSDL to help anticipate and/or avoid such problems. At the very least, each transmission pathway should be analyzed by test generation software to see if the driver and test receiver parameters are compatible, and where there might be problems. In the cases of programmable choices of parameters, these should be considered as well. Note that in some cases of bus programmability, it can happen that a group of nets is affected by a single setting of parameters, and that not every pathway in the group can be adequately set up, so some coverage may be lost or more than one test should be generated to address the problematic nets separately.

The Init_Data register is the preferred TDR for driver/receiver parameter controls. Therefore, before entering into testing with an EXTEST* instruction, the test generator should load the Init_Data register before doing the usual PRELOAD/EXTEST* routines of testing, assuming that the initialization instructions and registers introduced in IEEE Std 1149.1-2013 are implemented in the component.

It is also desirable to verify that defects are detected by the test. However, this is often impractical due to the large number of possibilities for defects. One area that should benefit from defect analysis (perhaps on a sampling basis) is that of board-level capacitor shorts, which may be tested with EXTEST. The BSDL for test receivers in such pathways may show Detect_EXTEST_* keywords, which indicate potential issues for shorted-capacitor tests. Problems may also be anticipated when I/O parameters between driver and receiver are radical mismatches (which is one reason why ac coupling exists). It would be wise to experiment with a sampling of such pathways by deliberately inserting shorts and seeing if the test performs adequately, before going into volume production where such troubles are unwelcome.

A.3.1 BSDL parsing

The BSDL parser software must recognize and process the resource descriptions given in the BSDL extension for this standard. It must identify and differentiate ac and dc pins and record time constant, voltage references, programmable voltages, programmable in-line capacitors, segmented Boundary registers, and ac/dc selection cell information for each ac pin.

A.3.2 Understanding differential signaling

Both the netlist analysis and the basic pattern generator software must understand differential signaling. Differential signals are carried as complementary values on a pair of nets. Thus, a bit shifted into the single data cell for a differential driver will result in separate true and complementary logic levels. For ac pins, those complementary values are later captured individually.

Netlist analysis must identify all differential channels and determine whether or not they are ac testable and record this information for use by the test pattern generator.

When assigning count values (or other unique identification sequences) to nets for the purpose of detecting shorts, the test pattern generation software must be aware that when it assigns a sequence of bits to be driven by a differential driver, it is explicitly assigning that sequence of bits to the driver's positive differential net and implicitly assigning the complement of that sequence of bits to the driver's negative differential net. For ac-testable channels, each leg is captured separately and complementary expect values must be assigned.

It is possible that the positive and negative legs of a differential channel may be swapped in the board netlist. This may be done for a number of reasons, usually to ensure that a very high speed channel is properly balanced, and has no negative effects in mission mode. Such swaps must be detected when analyzing the board netlist and recorded for use by the test generator.

A.3.3 Topological analysis of ac coupling

The analysis software must identify ac-coupled nets and build separate interconnect lists for dc and ac test patterns.

In addition to the simple situations described here, the software might recognize more complex instances of ac coupling where additional, less straightforward types of testing could be applied. See A.3.4.3 and A.3.4.4 below for details.

A.3.3.1 Identification of ac-coupled nets

When analyzing a netlist, the software must find all relevant coupling capacitors that pass signals from drivers to receivers, including:

— All on-board capacitors in series between two I/O signal pins (ac or dc).

— All on-chip coupling capacitors, as declared in the BSDL (these do not appear in the board netlist).

— All on-board capacitors in series between an I/O signal pin (ac or dc) and a board connector contacted directly by the tester.

Some device receivers may have on-chip coupling capacitors that are bypassable. These offer a choice during testing to have them in-line or bypassed. Test analysis should recognize these options for use in testing, as appropriate.

A.3.3.2 Identification of incomplete ac-coupled nets

If an ac net is incomplete (e.g., an ac driver net proceeds off-board via an edge connector, or a net enters a board from an edge connector and proceeds to a test receiver), then software must determine if the resources of the tester hardware will provide the necessary testability for such nets.

A.3.3.3 Generation of dc and ac interconnection lists

This standard provides instructions that test a pair of nets connected with a coupling capacitor as a single ac-coupled net. In addition, EXTEST can be used to test such a pair of nets as separate, unconnected nets. Both of these types of tests should be performed for a thorough test of the board.

The analysis software should make two lists of interconnects, those that are independent interconnects for dc testing and those that are independent interconnects for ac testing. Test generators will need these lists depending on whether they are using the EXTEST instruction or one of the EXTEST_PULSE or EXTEST_TRAIN instructions. Note that where bypassable on-chip coupling capacitors are present, some of the nets might belong in both lists; but care should be taken when a single bypass setting might control a group of pins since all of these would be bypassed (or not) as a group.

A.3.4 Interconnect test generation

Each pair of nets connected by a coupling capacitor and having ac test capability should be tested in two ways:

— Testing as a single net, through the capacitor, using the new ac test instructions, as described in A.3.4.1 below. This can detect lack of continuity and many types of shorts.

— Testing the separate nets on either side of the capacitor independently, using an enhanced dc interconnect test using the EXTEST instruction, as described in A.3.4.2 below. This can detect shorted board-level capacitors and other types of shorts.

A.3.4.1 ac interconnect testing

The traditional interconnection tests for dc interconnections should be modified to use the ac test instructions where possible. This will test the ac-coupled signals in the list of ac-testable interconnections (see A.3.3). This tests ac and dc channels simultaneously for interactions within and between the two groups.

These ac instructions also test for faults on differential signals, even if they are not ac-coupled, since testing enabled by this standard can detect faults that are not detectable by the minimal testing of differential signals that is provided by the EXTEST instruction.

There are several new considerations when generating these tests.

A.3.4.1.1 AC/DC selection cells

During this test of ac-coupled interconnects, the test generation software could set up ac/dc selection cells to force dc operation for those ac drivers that must be held steady throughout the test. The software should also set up the data cells associated with the drivers to drive the desired steady-state values.

This is useful for disabling non-boundary-scan devices. It could be possible to analytically determine the cells for which this should be done and their desired drive states. It would also make sense to allow explicit user specification of these settings.

A.3.4.1.2 Hysteresis presets

For each test vector, determine the default values (hysteresis preset values) for all test receivers used in the test.

These values will be shifted into each boundary-scan cell associated with a test receiver at the same time the driver test data is shifted in. These values will be captured and subsequently shifted out if the test receivers detect no transition(s) or valid level(s). The default value for each receiver should be varied so that all four combinations of default value and expect value occur at some time during the pattern set, and the details of which default values were used must be passed to the diagnostic software.

Since the hysteresis is not cleared between the *Shift-IR* and *Capture-DR* TAP Controller states, the test receiver values captured prior to the first boundary-scan register scan for any invocation of the interconnect test instructions will not be valid. (See discussion in 6.2.3.3.)

A.3.4.1.3 T_{Test} calculation

Calculate a value for T_{Test}, the minimum time between signal transitions defined in 6.2.3, for each test receiver, as follows:

1) If the calculation is for an ac test instruction, and there is a T_{LP} time included in the receiver BSDL, then set T_{Test} for this receiver to three times this value, otherwise:

2) If there is a T_{HP} time with either the ON_CHIP or ON_CHIP_PROGRAMMABLE modifier included in the receiver BSDL (and the on-chip capacitor is not currently bypassed), then set T_{Test} for this receiver to three times the T_{HP} value, otherwise:

3) Calculate the time constant of the board-level coupling circuit and set T_{Test} for this receiver to three times this calculated time constant.

NOTE 1—This time constant is the product of the coupling capacitance times the effective resistance that it sees. Test software can analyze the circuit to compute these values. Alternatively, the test software could avoid calculating the resistance value by using an upper bound in controlled impedance environments. The resistance seen by the capacitor invariably matches the characteristic impedance of the board interconnects. This is usually 100 Ω or less, and it must be less than 377 Ω (the characteristic impedance of empty space). Thus the test software could choose a value such as 300 Ω as a practical upper bound on the resistance seen by the capacitor. While using an upper bound this way has the effect of lengthening T_{Test}, this typically will not lengthen the overall test time appreciably since it need only affect the time spent in the *Run-Test/Idle* TAP Controller state, which is typically much less than the time spent shifting data.

NOTE 2—If the above calculation gets to step 3, and if the BSDL for the receiver declares a T_{HP} without a <cap spec> modifier, and if this T_{HP} value is greater than the value calculated in step 3, then this indicates an error in which the coupling time constant provided on this board does not meet the minimum required by the chip for proper operation.

NOTE 3—These calculations apply to IC testing as well as board test. If the receiver in question has neither a T_{LP} nor a T_{HP} with a <cap spec> modifier, then the IC test environment must provide capacitive coupling for this discussion to have meaning. This does not preclude other test strategies that could be employed by the IC manufacturer and which are not discussed here.

NOTE 4— Option 4b of Figure 52, and option 5 of Figure 53 use the on chip high pass filter as an ‘edge detection’ filter, which is documented using the <LP time constant> field of the BSDL. It is not to be confused with the on-chip HP ‘coupling’ capacitors that are documented with the <HP time constant>. The important point here is that both board test and IC testing are required to calculate a T_{Test} value of some kind in order to apply the necessary decay time for Test Receiver verification whether or not the ac coupling is external or on chip.

The maximum of the values calculated for various subsets of test receivers will be used as T_{Test} for the whole board for various tests. More flexible software might also allow explicit user specification of T_{Test} for each test, overriding the values calculated above.

A.3.4.1.4 TAP Controller state trajectories and timings

The test pattern generation software should perform specific TAP Controller navigation as specified in 5.3 and 5.4 in those portions of the interconnect test that make use of ac test instructions.

At a minimum, the state trajectory should always pass through the *Run-Test/Idle* TAP Controller state between each update and capture when either the EXTEST_PULSE or EXTEST_TRAIN instructions are active in one or more ICs.

The number of TCK cycles spent in the *Run-Test/Idle* TAP Controller state, and the TCK cycle time, depend on T_{Test} and the execution parameters, if any, specified for the instructions.

The simplest approach occurs when only the EXTEST_PULSE instruction is active in all ac-testable devices. In that case, the test must provide a time interval in the *Run-Test/Idle* TAP Controller state at least as long as the maximum T_{Test} for all test receivers receiving EXTEST_PULSE pulses, calculated above. This can be achieved either by spending multiple TCK cycles or by stopping TCK temporarily, depending on the TCK cycle time. If a “wait duration” value is specified in the BSDL attribute “AIO_EXTEST_Pulse_Execution,” the greater of that specification and the maximum T_{Test} should be

used to determine the minimum time interval to spend in the *Run-Test/Idle* TAP Controller state. Test software could allow for a user override of this interval, as well.

The calculation gets more complicated if EXTEST_TRAIN is required in one or more of the ICs. EXTEST_TRAIN is provided to support designs that might require a train of pulses to condition a driver or for any other reason that dictates an ac stability condition instead of a dc stability condition, and its use could be specified by the board designer, or could be specified by the component designer with the **no_pulse** keyword for a port. The test software must do a more complicated determination of TCK cycle time, ac test instruction to use in each device, and time to spend in the *Run-Test/Idle* TAP Controller state. This is needed to satisfy the pulse-train timing requirements while still satisfying the minimum time interval requirements described under EXTEST_PULSE above.

The required actions depend on the relationship between the minimum interval required for EXTEST_PULSE (the maximum value of T_{Test} or the "wait duration" value of the AIO_EXTEST_Pulse_Execution BSDL attribute), and the maximum TCK cycle time implied by the "train" and "maximum_time" values of the AIO_EXTEST_Train_Execution BSDL attribute.

First,

— if the EXTEST_TRAIN execution specification does not require a maximum time for the pulse train, or

— if the minimum interval required for the EXTEST_PULSE instruction is less than the TCK cycle time implied by the EXTEST_TRAIN execution specification,

then:

— set the TCK cycle time for the Run-Test/Idle TAP Controller state to a value between the minimum required for EXTEST_PULSE and the maximum (if any) implied for EXTEST_TRAIN,

— set the number of TCK cycles for the *Run-Test/Idle* TAP Controller state to the value required for EXTEST_TRAIN, and

— use the EXTEST_TRAIN instruction in all ac-testable devices.

In this case, the requirement for EXTEST_PULSE is met by each and every pulse generated in the train, and the requirements for EXTEST_TRAIN are met by the full train of pulses. There is no conflict between the timings of the two instructions.

If the minimum interval required for the EXTEST_PULSE instruction is greater than the TCK cycle time implied by the EXTEST_TRAIN execution specification, then:

— set the TCK cycle time for the *Run-Test/Idle* TAP Controller state to no more than the maximum value implied by the EXTEST_TRAIN execution specification, or to at least the maximum T_{Test} for the subset of test receivers that will be receiving the pulse trains, if that value is less, and

— set the number of TCK cycles in the *Run-Test/Idle* TAP Controller state to the number of pulses required by the EXTEST_TRAIN execution specification, or to the number of cycles needed to exceed the maximum T_{Test} for the board, if that number is greater, and

— use the EXTEST_PULSE in all devices except those driving interfaces requiring the use of the EXTEST_TRAIN instruction.

If the EXTEST_TRAIN instruction specifies a maximum amount of time that is less than the T_{Test} value calculated for the subset of test receivers receiving single pulses, then the number of pulses in the train should be increased, without changing TCK, until the total time in the *Run-Test/Idle* TAP Controller state exceeds that in T_{Test}.

It would make sense for the test software to allow explicit user specification of the parameters affecting operation in the *Run-Test/Idle* TAP Controller state, including TCK period and number of pulses.

NOTE—If driver conditioning (or any other IC characteristic) requires some minimum number of pulses and possibly a maximum time while in the *Run-Test/Idle* TAP Controller state, these parameters should be included in BSDL using the AIO_EXTEST_Train_Execution extension attribute (see 7.5.3).

A.3.4.1.5 Multiple parallel scan chains

If there are multiple chains operating in concert, then they must be controlled so that they all go through the *Update-DR* (or *Update-IR*), *Run-Test/Idle*, and *Select-DR-Scan* TAP Controller states simultaneously. If this requirement is not observed, then it is possible for test receivers to capture data before or after drivers change state, such that testing will lose synchronization, causing erroneous failures.

A.3.4.2 Enhanced dc interconnect testing

In addition to the ac test described in the previous subclause, the test pattern generation software should also generate an EXTEST interconnect test using the list of dc-independent interconnections. This test will treat ac drivers as if they were disconnected from their ac receivers, which should therefore capture default values. Failures at these ac receivers during this test will indicate shorts, either across the coupling capacitor or to other nets or to power planes. Note that this test is enhanced relative to a standard boundary-scan interconnect test because explicit default values permit the test receivers defined by this standard to detect shorts even when they are undriven.

A.3.4.2.1 Test receiver expect values

The undriven test receivers on a good board will always capture the default values in this test. Each such receiver should be given a unique combination of both 1 and 0 default values as if the default value was a weak driver for the receiver net, while each driven net on the board should go through its own unique sequence of bits, to permit identification of the shorted net in the event of a failure. A shorted board-level capacitor will simply show up as a short between two nets in this test.

For cases where a design uses one of the HP_time constant modifiers **Detect_EXTEST_High**, **Detect_EXTEST_Low, Detect_EXTEST_Both** and **Detect_EXTEST_Both_Grouped** defined in 7.5.4.1 per rules 7.5.6.2 h) and 0, the fault diagnostics will need to account for this behavior as follows.

For a good capacitor, the default initialized value will be observed at the output, regardless of input state, for each of the detection modifiers. For a shorted board-level capacitor:

a) When a test receiver is marked **Detect_EXTEST_High**, and if the driver is driving a 1 into the shorted capacitor, the test receiver produces a ‘1’ output, overwriting the initialized default of ‘0’. However when the driver is driving 0 into the shorted capacitor, the initialized default value might or might not be overwritten, regardless of the default's initial value;

b) When a test receiver is marked **Detect_EXTEST_Low**, and if the driver is driving a 0 into the shorted board-level capacitor, the test receiver produces a ‘0’ output, overwriting the initialized default of ‘1.’ However when the driver is driving 1 into the shorted capacitor, the initialized default value might or might not be overwritten, regardless of the default's initial value;

c) The **Detect_EXTEST_Both** and **Detect_EXTEST_Both_Grouped** modifiers will behave as **Detect_EXTEST_High** in 1), above, when the appropriate boundary cell is scanned to a '0', and will behave as **Detect_EXTEST_Low** in 2), above, when the appropriate boundary cell is scanned to a ‘1.’

d) When no modifier for a port is specified, then the port supports normal EXTEST and the value driven will be captured, assuming compatible driver common-mode and test receiver threshold voltages. Those voltages are documented in the BSDL AIO attributes for the ac pins. An analysis should be performed to determine if the driver common-mode voltage is compatible with the test receiver threshold voltage, that is, if the mismatch is sufficiently small that the test receiver will properly execute EXTEST, or whether it will reliably behave similar to one marked with either the **Detect_EXTEST_High** or **Detect_EXTEST_Low** detection modifiers, and the expected capture values must be adjusted as described above.

A.3.4.2.2 Special TAP Controller navigation

This test also requires special TAP Controller navigation. Between each Update and Capture, the system must spend enough time in the *Run-Test/Idle* TAP Controller state to allow any signals propagated through the coupling capacitors to decay. To calculate the minimum time to spend, use the procedure to calculate T_{Test} in A.3.4.1.3 above, but always skip step 1 because the fixed-reference test receivers used in this test do not make use of the T_{LP} time constant used in step 1. Use the maximum calculated values as the value for the whole board. Note that the time interval calculated here might be longer than the maximum T_{Test} previously calculated.

It is recommended that test software allow explicit user specification of this interval, overriding the calculated value.

A.3.4.2.3 AC/DC selection cells

All ac/dc selection cells must be loaded with their safe-value, as specified in the BSDL boundary-scan register description.

A.3.4.3 Testing mixtures of devices adhering to IEEE Std 1149.1 and to this standard

The topological analysis software should perform extra analysis to do as much testing as possible on interconnects where some device pins are ac pins, and other pins are only dc pins. Many ac-coupled interconnects will consist of a single driver and a single receiver. Reliable continuity testing (through the coupling) can only be done in such a case if both the driver and receiver are ac-capable. Cases that are more complex are possible, such as multiple drivers, receivers, or bidirectional pins, perhaps with a mix of ac and dc pins.

If a channel has multiple drivers, each driver must drive both states at least once during the test to verify continuity.

For each driver, do all testing allowed by Table 2, which is a comprehensive list of the tests that can be performed in various situations. A quick summary is as follows:

— When driving with an ac driver, it is valid to test for continuity at all receivers on either side of any coupling capacitor, except for dc-only receivers on the far side of any coupling capacitor.

— When driving with a dc-only driver, reliable continuity testing can only be expected if the receivers are on the same side of any coupling capacitor and all devices are using the EXTEST instruction. Also see the next subclause.

In any event, separate dc tests should be done on each side of the coupling capacitor, as described in A.3.4.2 above.

A.3.4.4 Additional mixed dc and ac testing (deprecated)

Testing of deprecated net configurations will be less robust than testing the fully supported net configurations. Test engineers should work closely with the board design engineers to help ensure that the steps suggested in 6.2.3.3 have been taken, when possible, and should take extra care verifying the reliability of these tests, and might have to disable those that prove unreliable or unworkable because of noise or other problems.

A.3.4.4.1 dc-only drivers ac-coupled to ac-capable receivers

It could also be possible to do additional testing of nets where dc driver pins are ac-coupled to ac receiver pins. Testing in this situation is deprecated by this standard (see Table 2 in 4.10 and the discussion following Figure 46) because, unlike the more robust tests for which this standard was intended, testing of continuity cannot be guaranteed and the results are susceptible to the simultaneous switching noise at *Update-DR* and *Update-IR*. However, in certain situations, such a test might work well enough to get additional defect coverage that is otherwise unavailable.

In this situation, dc output driver states can be detected by an ac receiver only if they cause an edge, so these tests must produce such edges. One way to do this is to pre-condition the boundary-scan register cells associated with these dc drivers with the complement of the test pattern data with an additional Shift and Update before the Shift and Update that applies each test. This produces a transition. Other drivers should be preconditioned with their test values, not the complement, in order to minimize the simultaneous switching noise during the test.

If the board coupling capacitors for these nets are sufficiently large that the signal will not decay significantly between the transition in the *Update-DR* TAP Controller state and the rise of TCK in the *Capture-DR* TAP Controller state, then consider using the EXTEST instruction for the receivers. This avoids the noise generated during the *Update-DR* TAP Controller state. Otherwise, use either the EXTEST_PULSE or EXTEST_TRAIN instruction for the receivers.

Even if these precautions are taken, these tests will be less robust than the other types of tests described earlier. This is because of the problems discussed in 6.2.3.3 concerning noise susceptibility. Test engineers should work closely with the board design engineers to help ensure that the steps suggested in 6.2.3.3 have been taken, when possible, and should take extra care verifying the reliability of tests of this type, and might have to disable those that prove unreliable because of noise or other problems.

A.3.4.4.2 Any driver ac-coupled to dc-only receivers

Testing in this situation is deprecated by this standard (see Table 2 and the discussion following Figure 46) because, unlike the more robust tests for which this standard was intended, testing of continuity cannot be guaranteed, and any tests would require the imposition of a minimum TCK based on the coupling time constant and TAP navigation. However, in certain situations, such a test might work well enough to get additional defect test coverage that is otherwise unavailable.

In this situation, the data being driven can be detected only if the signal has not decayed to an ambiguous level. Depending on the receiver technology and type (single-ended, differential, CMOS, Bipolar, etc.), the behavior of the ac coupling decay must be determined for each deprecated net configuration, either by analog simulation or by direct observation. The resulting worst case (fastest) decay time before the signal becomes ambiguous must be longer than the time from the last transition of the driver (falling TCK in the *Select-DR* TAP Controller state for an ac driver in EXTEST_PULSE, preferred because it is shorter, or falling TCK in the *Update-DR* TAP Controller state for a driver performing a dc test) to the capture of receiver data (rising TCK in the *Capture-DR* TAP Controller state.) This implies a minimum frequency for

TCK during that interval must be calculated, and that TCK cannot be paused during the TAP sequence expected.

A.3.4.5 *Update-DR* noise avoidance

Since the hysteretic memory is loaded prior to the *Update-DR* TAP Controller state, there is the possibility that the noise generated during that state could change the memory contents, even if there is no transition on that net in that state. If this occurs often enough to reduce the test effectiveness, and normal steps to minimize the noise are ineffective, then it might be necessary to perform the interconnection tests with a double scan. Each test pattern would be scanned into the boundary-scan registers twice, and the results captured between the two scans would be ignored. After the first scan, all the outputs would assume the desired static values, and after the second, the hysteretic memory would be reloaded but the outputs would not switch, eliminating a major source of noise. There, clearly, is no need to transit the *Run-Test/Idle* TAP Controller state after the first scan, though it would not hurt.

It might be desirable to provide a user selectable mode in the tester software to apply all or selected test patterns with double scan. This mode could also be useful for determining what nets are affected by noise, by comparing the results captured after the first and second scans of the same pattern.

A.3.5 Diagnostic routines

Many of the processes for testing ac-coupled channels will require new diagnostic routines, particularly to interpret captured data that matches data preset in hysteretic test receiver memories, interpreting captured data when any receiver is described with any of the **Detect_EXTEST_*** keywords, on-chip capacitor specifications, and for identifying shorted board-level capacitors. See rule 7.5.6.2 0.

A.3.6 Test system hardware requirements

Special drive and detect circuitry might be necessary for ac-coupled signals that go to the edge connector of the board:

— Tester drivers will have to meet specifications for ΔV and T_{Trans} when driving edge connector signals that are received by an IC that contains test receivers.

— Tester drivers will have to meet arbitrary common-mode and threshold specifications when driving edge connector signals that do not go through a capacitor to be received by an IC that contains test receivers.

— Traditional tester detectors, which detect levels rather than edges, might not be able to detect the transitions on the signals at the board edge that go through a capacitor after being driven by an IC driver (this should happen rarely, since when a connector separates a driver and receiver, the coupling capacitor is most likely to be on the receiver side of the connector).

— In some cases, test engineers might have to resort to special circuitry on the fixture to address the above problems (such as circuitry that mimics the capabilities this standard mandates for ICs).

Annex B

(informative)

Noise rejection in edge-detecting mode

The hysteresis delay T_{Hyst} is used to provide noise rejection during edge-detection mode of the test receiver (see 6.2.3). The hysteresis voltage setting will reject a noise pulse with small amplitude. But if a noise pulse with amplitude greater than the hysteresis voltage is encountered, the test receiver is expected to reject those with duration less than T_{Hyst}. There are two techniques that can be utilized separately or in combination to accomplish this (other techniques could also be applied to achieve this result). These techniques are *bandwidth limitation* and *slew rate limitation.*

B.1 Noise rejection by bandwidth limitation

Bandwidth limitation can be added to the test receiver to limit its response time (this analysis is a linear small signal analysis and specifically assumes no saturation effects). Consider a rectangular pulse waveform of height V_{PW} and pulse width duration T_{PW}, where V_{PW} is greater than the hysteresis voltage (see Figure B.1). For analysis purposes, assume that the lower amplitude of the pulse is at 0 volts, although the results are applicable to the general case of an arbitrary baseline voltage. Furthermore, assume the rise and fall times of the pulse are much faster than those of the test receiver input stage so they can be ignored in this analysis, but the test receiver input stage is adequately fast enough to track the pulse itself.

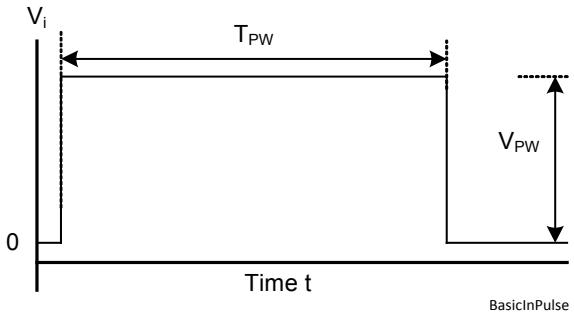


Figure B.1— Pulse input to test receiver

The test receiver can be modeled in the small-signal domain as a single-pole low-pass amplifier with a frequency response (as designated by the lowercase nomenclature) given in Equation (B.1), frequency response of a simple single-pole low-pass amplifier, where v_o and v_i are the output and input voltages of the amplifier and A_0 is its dc gain. Let $\tau_{BW} = 1/2\pi f_{BW}$ be the time constant of the bandwidth frequency, f_{BW} and S is a complex number of the form: $S = \sigma + J\omega$, where both σ and ω are real numbers and j is an imaginary unit that is equal to $\sqrt{-1}$ .The real portion of S is equal to σ and the imaginary component is equal to $j\omega$.

$$\frac{v_o}{v_i} = \frac{A_o}{1 + S\tau_{BW}} \tag{B.1}$$

Furthermore, assume that the test receiver has an input referred switching threshold of V_{TH}. The input voltage must exceed V_{TH} for a sufficient period of time to cause the test receiver to switch its output state. In the time domain (as designated by the upper case nomenclature) the output response of the test receiver will be described by Equation (B.2), time domain output response of the test receiver, which is shown in Figure B.2 for $T_{PW} \gg \tau_{BW}$.

NOTE—The propagation delay through the test receiver has been ignored.

$$\frac{V_o}{V_i} = A_o\left(1-\exp\left(-t/\tau_{BW}\right)\right) \tag{B.2}$$

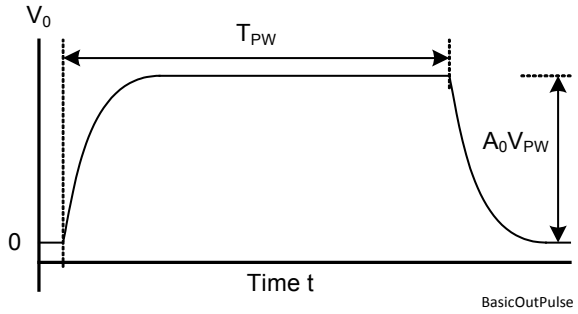


Figure B.2— Output response of the test receiver to the rectangular pulse

The output amplitude for an arbitrary input can be obtained by solving Equation (B.2) as shown in Equation (B.3), solution of Equation (B.2) for time t, as

$$\mathrm{t} = -\tau_{BW}\ln\left(1-\left(\frac{V_O}{V_i A_O}\right)\right) \tag{B.3}$$

In order for the receiver circuitry to switch the input amplitude will have to exceed the input referred switching threshold for a period of time long enough to allow the output level to reach the output referred switching threshold, $A_o V_{TH}$. Thus, the input pulse must persist for a minimum period of time, T_{TH}. This time can be calculated from Equation (B.3), as shown in Equation (B.4), minimum pulse width duration (for $V_{PW} > V_{TH}$), as

$$T_{TH} = -\tau_{BW}\ln\left(1-\left(\frac{V_{TH}}{V_{PW}}\right)\right) \tag{B.4}$$

If the pulse width is less than the minimum time, T_{TH}, the pulse will be rejected. Alternately, the pulse will be rejected if the bandwidth of the receiver circuit is as shown in Equation (B.5), test receiver bandwidth specification, as

$$f_{BW} < -\frac{1}{2\pi * \tau_{BW}}\ln\left(1-\left(\frac{V_{TH}}{V_{PW}}\right)\right) \tag{B.5}$$

B.2 Noise rejection by slew rate limitation

In most situations the switching of the test receiver will be limited more by its large signal and internal slew capability rather than its small signal bandwidth. This is manifested by the limited rate at which an arbitrary internal node of the receiver can transition due to its capacitive load and/or limited current drive capability. Hence, at this internal node of the test receiver circuitry, the voltage V_{INT} will be limited by its slew rate SR, as depicted in Figure B.3.

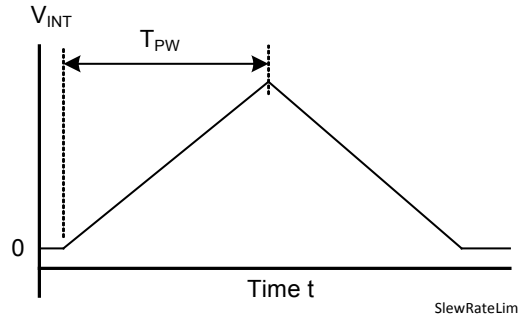


Figure B.3—Slew rate limited response of a control node internal to the test receiver

It is assumed that the duration of the input signal is not long enough to allow the internal node voltage to achieve its final value for the given level of input signal before the direction of the voltage transition reverses as the result of the reversal of the input signal. In order for the circuit to switch the voltage at the internal node, it will have to achieve a level equal to its internal switching threshold V_{THINT}. This situation can be expressed as the switching condition

$$V_{INT} = SR \times T_{PW} > V_{THINT} \tag{B.6}$$

or

$$T_{PW} > \frac{V_{THINT}}{SR} \tag{B.7}$$

If the duration of the input pulse does not meet the criteria set forth in this equation, it will be rejected.

Annex C

(informative)

Advanced I/O boundary-scan register cells

This standard defines new boundary-scan register ac/dc selection cells, and new boundary-scan register data cells on ac pins that have drive capability (output or bidirectional). The AC_1 and AC_2 cells are documented here by showing the original cell type from IEEE Std 1149.1 and the changes to it required by this standard. The rest of the adapted cells are shown with the added logic circled. The BSDL descriptions for these cells appear in 7.3. Generation of the various Mode signals used in the drawings are shown in C.10.

C.1 AC/DC selection cell AC_SelX

This cell is an adaptation of the highly general BC_1 cell from IEEE Std 1149.1. It is *only* used in *internal* contexts. This cell has no input multiplexer prior to the capture flip-flop, so by the definition (in BSDL) of what it captures in the *Capture-DR* TAP Controller state, this cell captures an unknown value X. This cell also has no mission signal to monitor, so the output multiplexer usually seen is also omitted. The cell is depicted in Figure C.1.

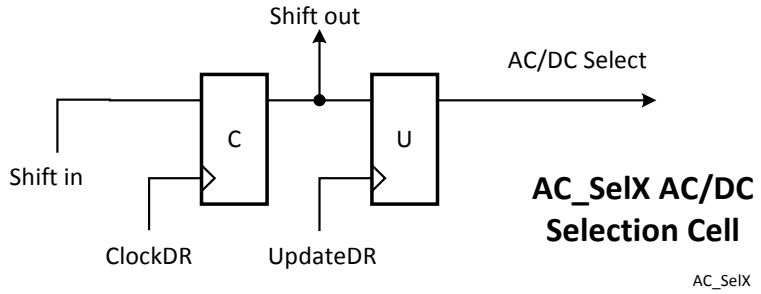


Figure C.1—AC_SelX internal cell design used for ac/dc selection

C.2 AC/DC selection cell AC_SelU

This cell is an adaptation of the BC_2 cell from IEEE Std 1149.1. It is *only* used in *internal* contexts. This cell has an input multiplexer prior to the capture flip-flop that selects the Update flip-flop content, so by the definition (in BSDL) of what it captures in the *Capture-DR* TAP Controller state, this cell captures the value UPD. This cell also has no mission signal to monitor, so the output multiplexer usually seen is also omitted. The cell is depicted in Figure C.2.

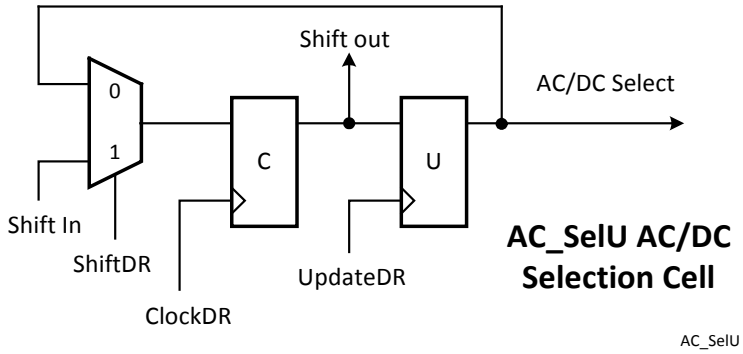


Figure C.2— AC_SelU internal cell design used for ac/dc selection

C.3 Output data cell AC_1 (supports INTEST)

The AC_1 cell in Figure C.3 is an adaptation of the highly general BC_1 cell from IEEE Std 1149.1-2013, Figure 11-31. This cell supports the INTEST instruction. Note that the BC_1 cell can be used in contexts other than supplying data to output drivers (input cells, control cells), but AC_1 cell only supports the *output2* and *output3* contexts.

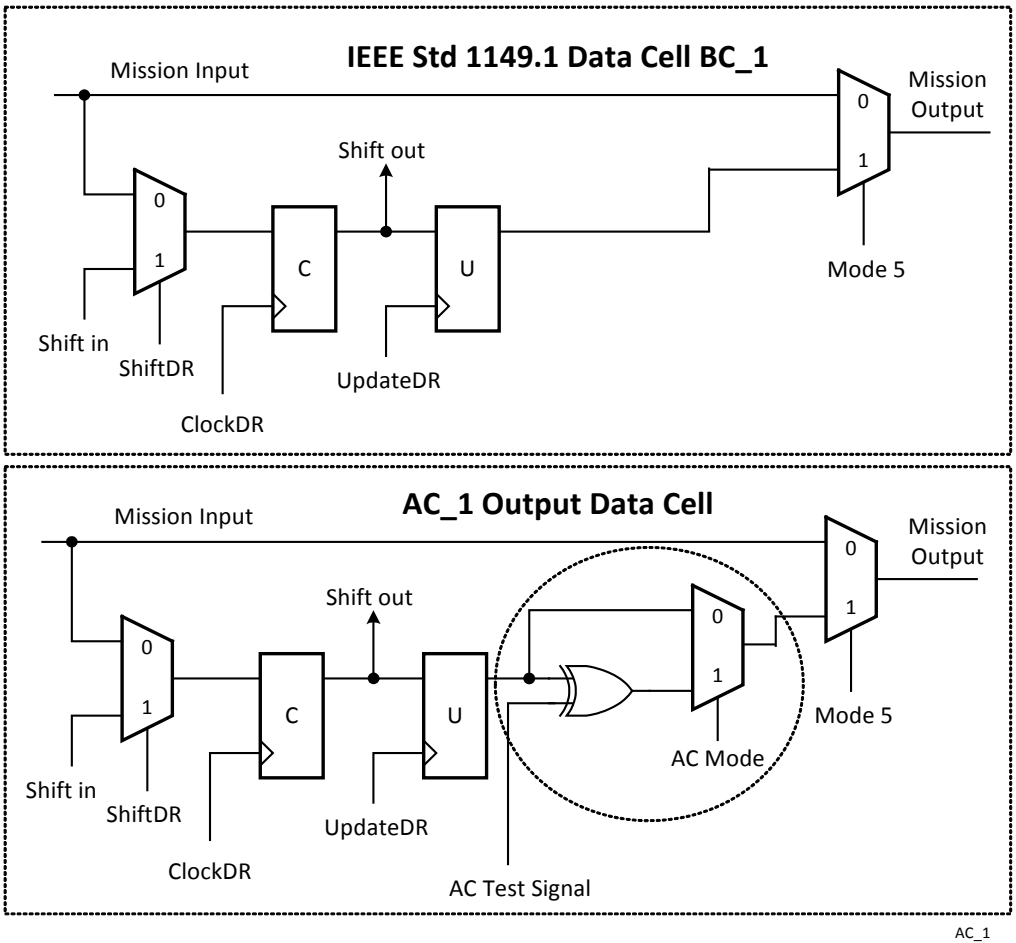


Figure C.3—AC_1 output data cell adapted from BC_1

The circuitry added for support of AC EXTEST test instructions is circled in Figure C.3. The generation of the Mode_5 signal is shown in C.10.

C.4 Output data cell AC_2

The AC_2 cell in Figure C.4 is an adaptation of the BC_2 cell from IEEE Std 1149.1-2013, Figure 11-32. Note that the BC_2 cell can be used in contexts other than supplying data to output drivers (input cells, control cells), but that the AC_2 cell only supports the output2 and output3 contexts. This cell does not support INTEST. The circuitry added for support of AC EXTEST test instructions is circled. The generation of the Mode_5 signal is shown in C.10.

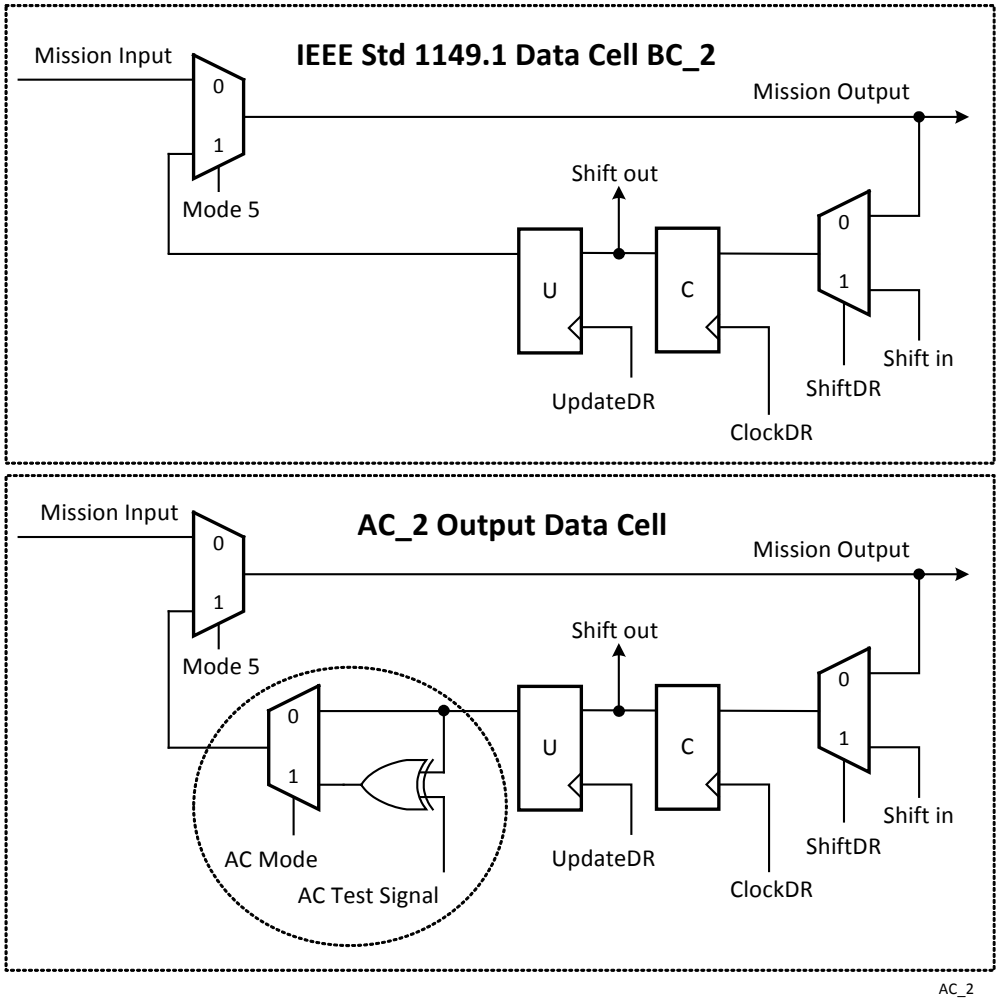


Figure C.4—AC_2 output data cell adapted from BC_2

C.5 Bidirectional output cell AC_7 (supports INTEST)

The AC_7 cell in Figure C.5 (shown with its companion BC_2 control cell) is an adaptation of the bidirectional BC_7 cell from IEEE Std 1149.1-2013, Figure 11-38. This cell supports the INTEST instruction. The circuitry added for support of AC EXTEST test instructions is circled. The generation of the Mode signals is shown in C.10.

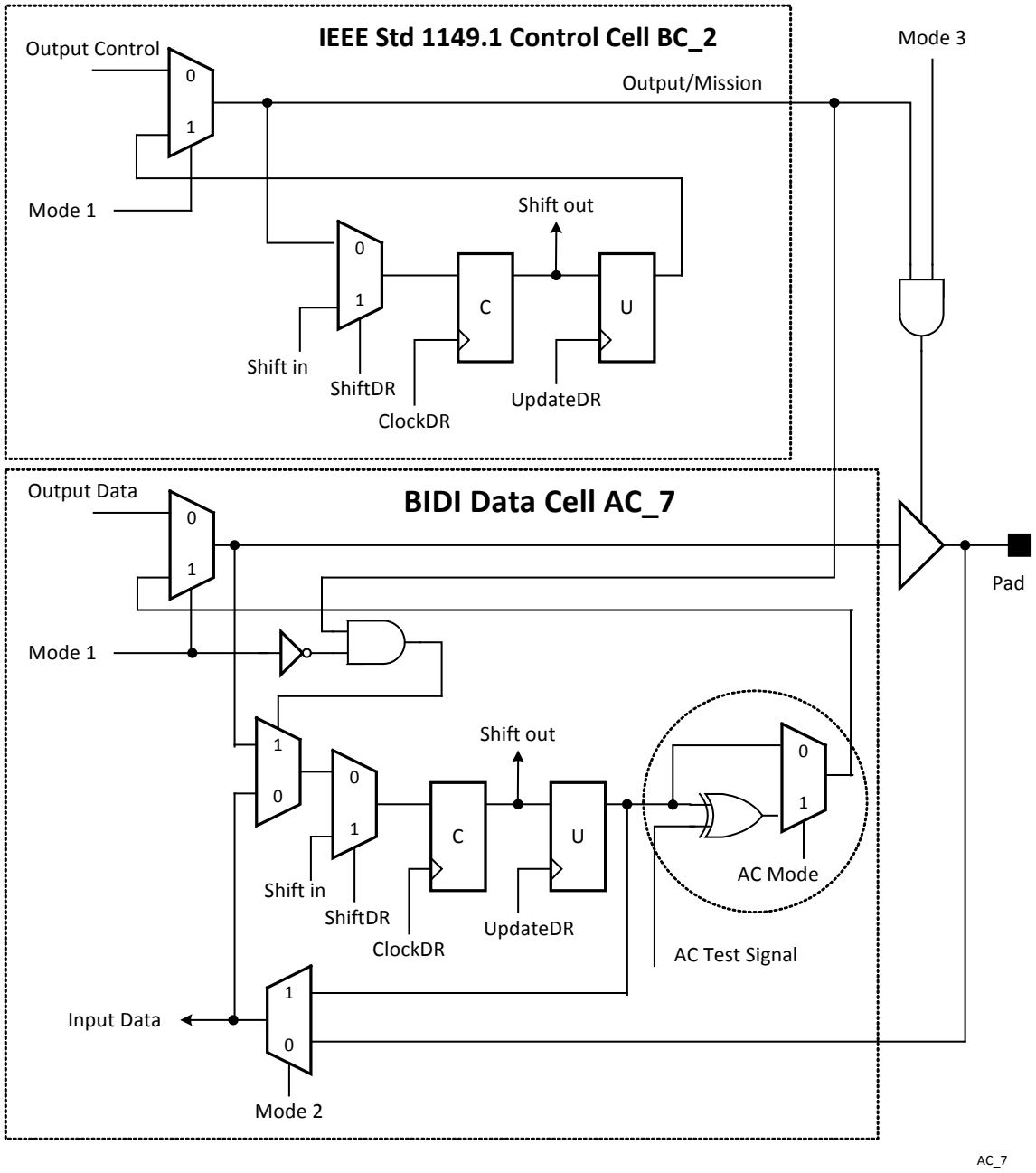


Figure C.5—AC_7 Output Cell adapted from BC_7

C.6 Bidirectional output cell AC_8

The AC_8 cell in Figure C.6 (shown with its companion BC_2 control cell) is an adaptation of the bidirectional BC_8 cell from IEEE Std 1149.1-2013, Figure 11-41. This cell does not support INTEST. The circuitry added for support of AC EXTEST test instructions is circled. The generation of the Mode signals is shown in C.10.

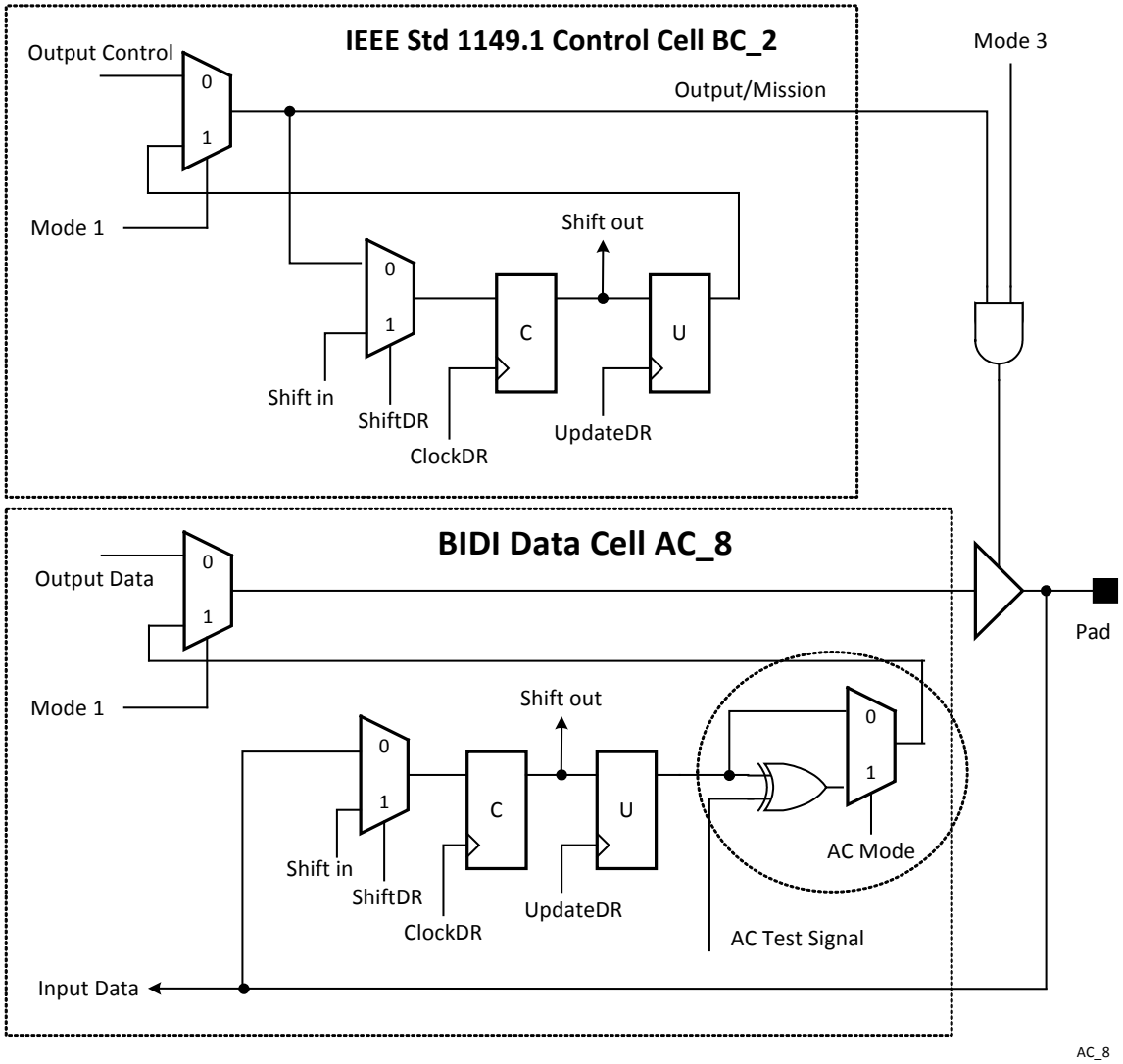


Figure C.6—AC_8 Output Cell adapted from BC_8

C.7 Self-monitoring output cell AC_9 (supports INTEST)

The AC_9 cell in Figure C.7 is an adaptation of the self-monitoring BC_9 cell from IEEE Std 1149.1-2013, Figure 11-33. This cell supports the INTEST instruction. The circuitry added for support of AC EXTEST test instructions is circled. The generation of the Mode signals is shown in C.10.

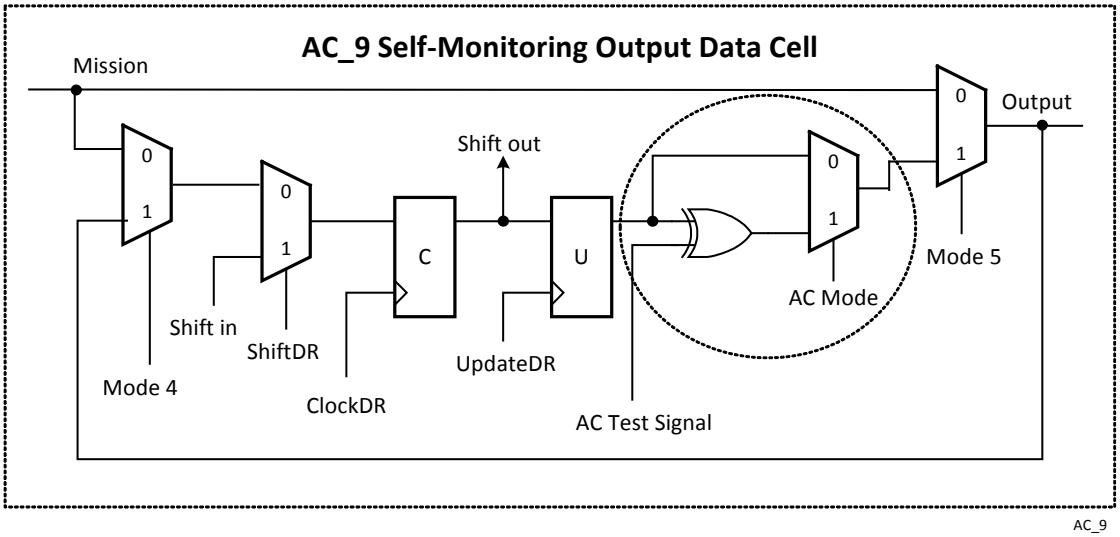


Figure C.7—AC_9 output cell adapted from BC_9

C.8 Self-monitoring output cell AC_10

The AC_10 cell in Figure C.8 is an adaptation of the self-monitoring BC_10 cell from IEEE Std 1149.1-2013, Figure 11-34. This cell does not support INTEST. The circuitry added for support of AC EXTEST test instructions is circled. The generation of the Mode signals is shown in C.10.

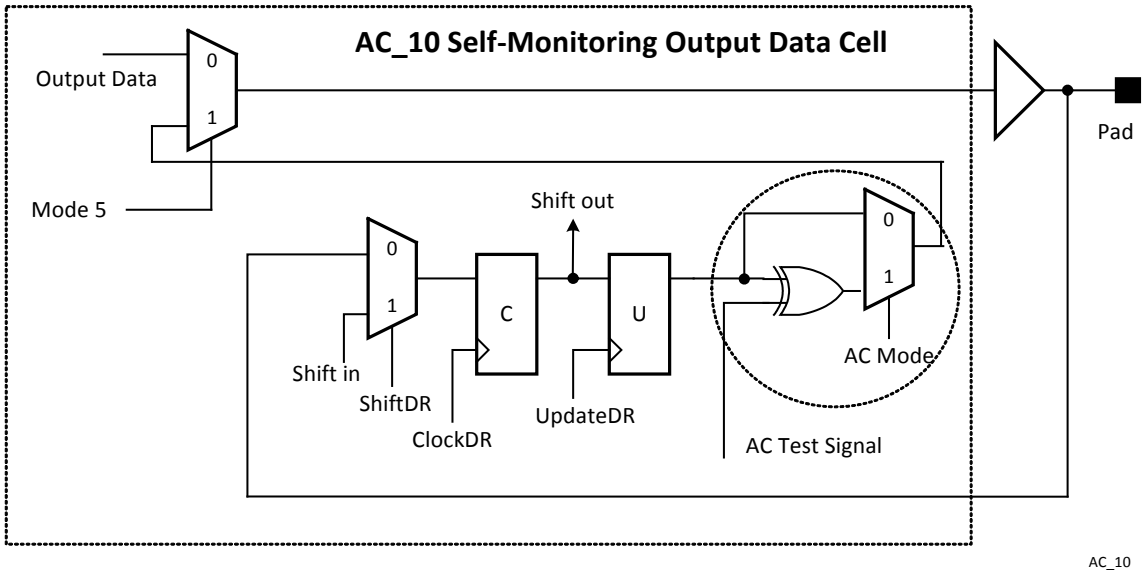


Figure C.8—AC_10 output cell adapted from BC_10

C.9 AC_40 and AC_41 cells

Cells AC_40 and AC_41 are adapted from the 1149.1 BC_4 cell, but capture a ‘0’ or ‘1’ (respectively) when the SAMPLE instruction is in effect.

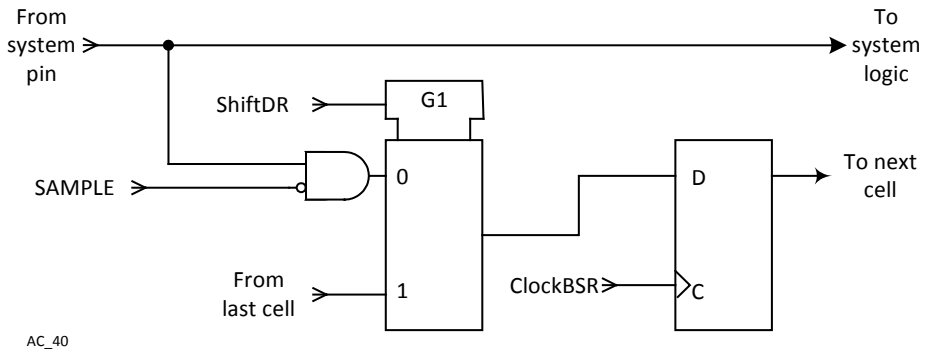


Figure C.9—AC_40 input cell adapted from BC_4, capture '0' during SAMPLE

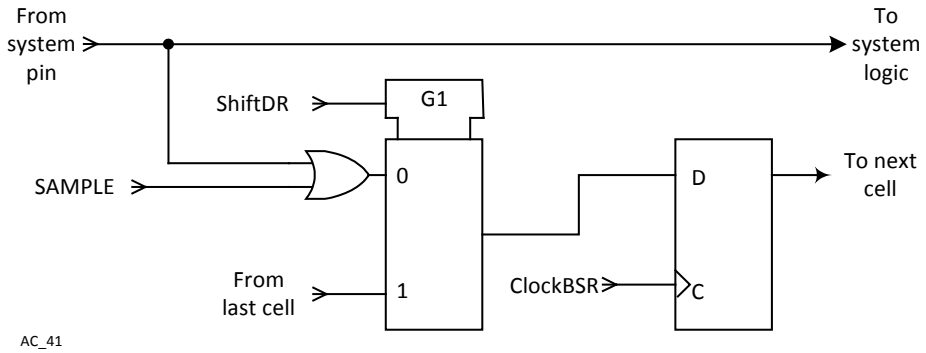


Figure C.10—AC_41 input cell adapted from BC_4, captures '1' during SAMPLE

C.10 AC cell mode controls

The Mode signal generation is defined in tables in IEEE Std 1149.1-2013, 11.6 and 11.7. Note that in those tables, where EXTEST is listed, the entry should be for EXTEST, EXTEST_PULSE, and EXTEST_TRAIN together. Rows for the INTEST instruction should be ignored for cells AC_2, AC_8 and AC_10.

Annex D

(informative)

Test receiver design examples

For these initial calculations, assume that the test receiver can be ac- or dc-coupled on the board. All calculations assume that the coupling and the termination are close to the receiver pads or even on-chip, resulting in negligible transmission line effects from the coupling to the receivers. As with all inputs that can be ac-coupled, the pads of the differential receivers must be biased by a matched pair of resistors to the desired reference voltage (usually the same as the common-mode voltage for drivers of the same technology and protocol).

D.1 LVDS with normal board coupling

Figure D.1 shows an LVDS channel that is ac-coupled, with the coupling capacitors in each leg before the load termination. In addition, to maintain a dc current path, there is a source termination at the driver pins. The terminations could be a single resistor between the legs, as shown here, or separate resistors from each leg to an appropriate voltage source (usually the same as the common-mode voltage.) The internal, high-impedance bias resistors to the termination voltage, V_{tt}, through the bias resistors, sets the static (DC) operating point for the mission and test receivers, when ac-coupled. To take advantage of this mode, the specification sheet would have to strongly require this type of coupling for all ac-coupled use.

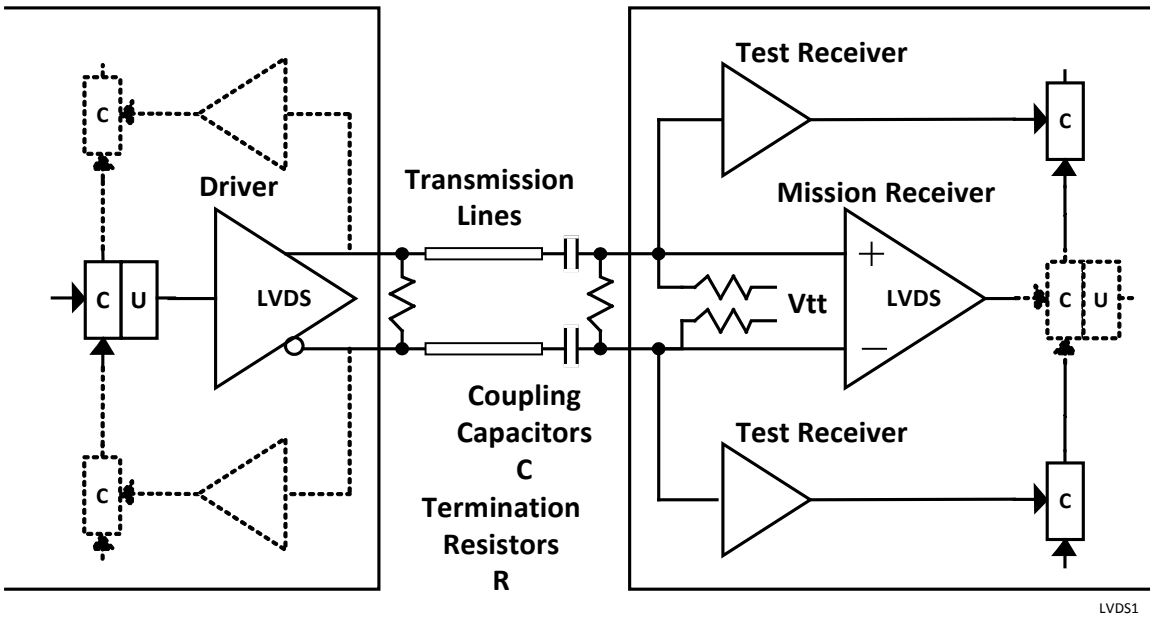


Figure D.1—LVDS driver/receiver with on-board source/load terminations and ac coupling

NOTE—In all figures in this annex, the boundary-scan register structures shown with dashed lines are optional.

D.1.1 Calculations

The first step is to examine the specifications of the driver and receiver to be implemented. For a specific LVDS driver and receiver, in a specific technology, the specifications might be:

— $V_{dd} = 2.5$ V

— $V(\text{common-mode}) = 1.25$ V

— $\Delta V_{\min}(\text{driver}) = 338$ mV， $\Delta V_{\max}(\text{driver}) = 480$ mV

— $f(\max) = 600$ MHz， $T_{Trans} = 400$ ps

The common mode and peak-to-peak voltages of the driver must be documented in the BSDL attributes.

Typically, LVDS and other current-mode differential protocols are not ac-coupled. In order to support ac coupling with the coupling capacitors before the termination, the specification sheet for the circuit being designed requires source termination of 100 Ω across the driver outputs as well as the receiver inputs. This will reduce the voltage swings seen at the receiver by half. The expected maximum and minimum voltage swings at the receivers are calculated as follows:

— $\Delta V_{\max} = 0.5 \times \Delta V_{\max}(\text{driver}) = 0.5 \times 480 \text{ mV} = 240 \text{ mV}$

— $\Delta V_{\min} = 0.8 \times 0.5 \times \Delta V_{\min}(\text{driver}) = 0.8 \times 0.5 \times 338 \text{ mV} = 240 \text{ mV}$

The factor of 0.8 on the minimum voltage swing calculation is the designer's best estimate of the static signal attenuation that would be seen in most applications of this circuit. This factor should be adjusted using engineering judgment based on the expected use.

This is sufficient information to start calculating the design parameters for the test receiver. In this case, the transition time has been explicitly stated, so T_{Hyst} is set to a multiple of that

$$T_{Hyst} = 5 \times T_{Trans} = 5 \times 0.4 \text{ ns} = 2.0 \text{ ns}$$

The test receiver will be designed to reject pulses that are narrower than that duration.

The hysteresis voltages are calculated based on the receiver ΔV_{Min} calculated above. As long as the T_{Test} duration is long enough, it is assumed that the full ΔV amplitude is seen at each transition, above or below the receiver reference voltage. The hysteretic threshold for the EXTEST_PULSE and EXTEST_TRAIN instructions are set anywhere in the range of 50% to 90% of the minimum voltage swing. Picking the recommended middle of the range

$$V_{Hyst-Edge} = 0.7 \times \Delta V_{\min} = 0.7 \times 135 \text{ mV} = 95 \text{ mV}$$

The edge-detection hysteretic threshold value must be documented in the BSDL attributes.

For the EXTEST instruction, the test receiver must compare the input to a fixed threshold, reducing the amplitude to half of the ΔV above and below that threshold. Note that if the common-mode voltage of the driver did not match the internal reference voltage of the receiver, then either the hysteresis voltage should be reduced (for small differences), or this mode simply might not work, resulting in significantly increased difficulty in detecting shorted coupling capacitors.

$$V_{Hyst-Level} = 0.5 \times V_{Hyst-Edge} = 0.5 \times 95 \text{ mV} = 47.5 \text{ mV}$$

From Table 3, for a hysteresis level of 70% of $\Delta V_{\min}$:

— HP_Mult = 18

— LP_Mult = 9

In this case, the $\Delta V_{\max}$ to V_{Hyst_Edge} ratio is less than four, and no severe noise issues are know to exist in any of the test environments, so the HPLP_Ratio of two is acceptable based on Table 4.

The time constants for the self-referencing low-pass filter, the minimum time constant for the board high-pass coupling and the minimum time between test transitions can now be calculated:

— $T_{LP} = LP\_Mult \times T_{HYST} = 9 \times 2 \text{ ns} = 18 \text{ ns}$

— $T_{HP} = \text{HP\_LP\_Ratio} \times T_{LP} = 2 \times 18 \text{ ns} = 36 \text{ ns}$

— $T_{Test} > 3 \times T_{LP} = 3 \times 18 \text{ ns} = 54 \text{ ns}$

The values for the self-reference low-pass filter in the test receiver can now be calculated. Assuming an on-chip capacitor of 1 pF to ground, then

$$R = (18e^{-9}) / (e^{-12}) = 18 \text{ k}\Omega$$

For normal board ac coupling with the capacitors before termination, the minimum coupling capacitance can be calculated. Assuming a normal termination of 50 Ω per leg (100 Ω across the two legs)

$$C > (36e^{-9}) / 50 = 720 \text{ pF}$$

Finally, the bias voltage value for V_{tt} in Figure D.1 and the threshold voltage of the test receiver in dc test mode (EXTEST) must be selected. The common mode voltage described above for this technology is chosen: 1.25V. This also must be documented in the BSDL attributes.

D.1.2 BSDL

The BSDL AIO_Pin_Behavior entry for a pin using this LVDS driver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
  " LVDS_DRVR : AIO_VCM=1250 AIO_VPP=338  ; " &
    …
;
```

The BSDL AIO_Pin_Behavior entry for a pin using this test receiver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
   " LVDS_RCVR : LP_time=18.0e-9 HP_time=36.0e-9 " &
   "       AIO_VTH=1250 AIO_VHyst=95; "&
      …
;
```

In addition, there would need to be Port, Pin_Data, and Pin_Grouping entries for pins “LVDS_DRVR,” “LVDS_RCVR,” and their complements, and boundary-scan register entries for ports “LVDS_DRVR” and “LVDS_RCVR.”

D.1.3 Simulations

Figure D.2 and Figure D.3 are Hspice simulation results for an LVDS channel in a particular 0.18 micron CMOS technology and are used here for illustration. The driver common-mode voltage is 1.25 V, and the rest of the specifications are as stated in D.1. You should expect a simulation of a different technology to differ in many details from this example.

In Figure D.2, the first trace is the internal input to the driver. The second and third traces are the driver output pads, and the fourth and fifth are the receiver input pads, after the source termination resistor (100 Ω between legs), transmission lines, coupling capacitors, and load termination resistor (100 Ω between legs). The sixth and seventh traces are the outputs of the test receiver models attached to the positive and negative receiver pads. Hspice behavioral primitives were used to model the test receiver and the simultaneous switching noise.

Figure D.3 shows some of the internal signals of the test receiver behavioral model (see Figure 32 for the model equivalent circuit diagram). The first and fourth traces (for the positive and negative legs respectively) are the mathematical difference of the signal at the pad minus the low-pass filtered version of the signal at the pad, without adding/subtracting the hysteresis voltage V_{Hyst}. The other traces (second and third, fifth and sixth) are the respective set and clear pulses from the comparators to the memory flip-flops, after the V_{Hyst} and T_{Hyst} filtering.

In both figures, the equivalent of the falling edge of TCK in *Update-DR* TAP Controller state happens at time 100 ns. To make the effect of the simultaneous switching noise obvious, the driver input is not toggled at this time (it is common during testing that the value in the Update latch of the boundary-scan register cell would not change between two tests). The equivalent of the first falling edge of TCK in the *Run-Test/Idle* TAP Controller state occurs at 300 ns. Sampling the test receiver output to the Capture latch of the boundary cell would occur just before the transition at 500 ns, for the EXTEST_PULSE instruction.

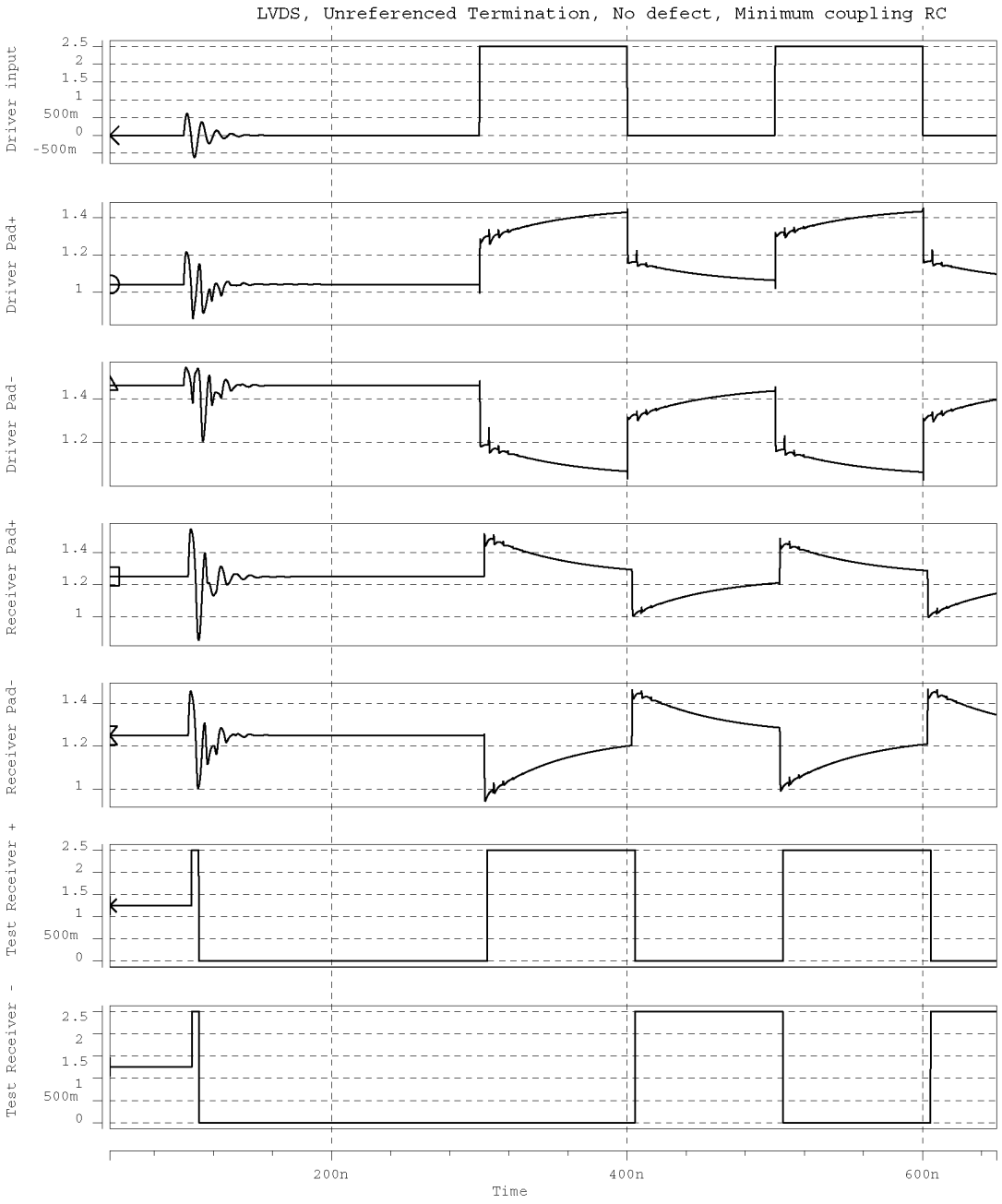


Figure D.2—Simulation of a differential LVDS channel with the parameters calculated in D.1

Notice the decay of the signals at the receiver pads (Figure D.2, fourth and fifth traces) controlled by T_{HP}, and the even faster decay of the difference signals in the test receiver (Figure D.3, first and fourth traces) controlled by T_{LP}. Also note, on the difference signals (which, for clarity, do *not* include the hysteretic offset), the small over/undershoot that occurs due to the two time constants interacting (at approximately 350 and 450 nsec). Increasing the HPLP_Ratio reduces this over/undershoot, which subtracts from the test receiver noise margin.

Note also the effect of the simultaneous switching noise (imposed on the driver input at 100 nsec) on the test receiver. This has sufficient amplitude and duration to get past both the V_{Hyst} and T_{Hyst} filters in the test receivers. Both of the test receivers started at an indeterminate state (an artifact of the Hspice simulator) and both were toggled high and then low by the noise. Only after the transitions in the *Run-Test/Idle* TAP Controller state do the test receiver outputs assume the correct values.

Figure D.3—Simulation of test receiver internal signals (for both legs) for the example in D.1

D.2 LVDS with alternative board coupling

Figure D.4 shows an alternative topology for ac coupling on the board that has particular advantages for current-loop differential channels: place the coupling capacitors after the termination resistors. As the current loop is now intact, the source termination is no longer required, allowing full voltage swings of the transitions and possibly increasing noise margins. This topology has the disadvantage of making the input signals to the receiver relatively high-impedance, so that care must be taken to prevent noise coupling at that point. In addition, the coupling capacitance is dependent on the input impedance of the receiver and the bias resistor rather than the termination resistor. To take advantage of this mode, the specification sheet would have to strongly require this type of coupling for all ac-coupled use.

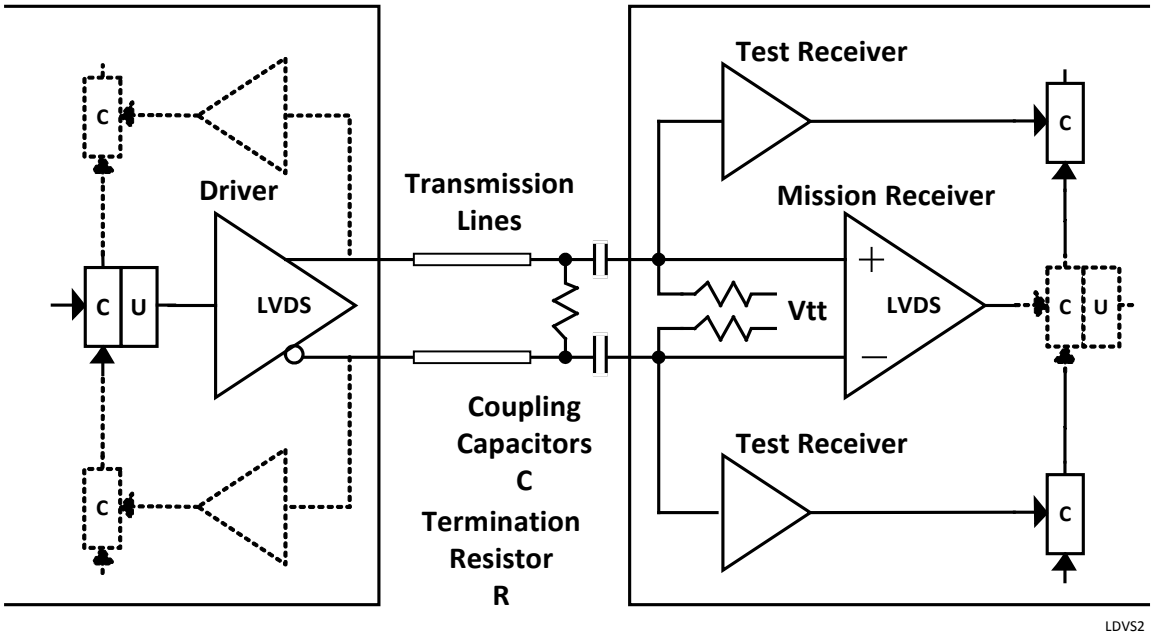


Figure D.4—LVDS with on-board load termination and ac coupling

D.2.1 Calculations

T_{Hyst} and the time constants T_{LP} and T_{HP} do not change from the previous example. The expected maximum and minimum voltage swings at the receiver and hysteretic thresholds are double the values in the previous example, and are calculated as follows:

— $\Delta V_{\max} = \Delta V_{\max}(\text{driver}) = 480 \text{ mV}$

— $\Delta V_{\min} = 0.8 \times \Delta V_{\min}(\text{driver}) = 0.8 \times 338 \text{ mV} = 270 \text{ mV}$

— $V_{Hyst-Edge} = 0.7 \times \Delta V_{\min} = 0.7 \times 270 \text{ mV} = 190 \text{ mV}$

— $V_{Hyst-Level} = 0.5 \times V_{Hyst-Edge} = 0.5 \times 190 \text{ mV} = 95 \text{ mV}$

The low-pass filter does not change. Values of 1 pF and 18 kΩ are still correct. The minimum high-pass filter capacitance calculation must now take into account the input impedance of the active receivers (assumed to be infinite, in this case), and the bias (assume 10 kΩ) and low-pass filter input resistance

$$R = 10 \text{ k}\Omega \| 18 \text{ k}\Omega = 6.43 \text{ k}\Omega, \; C > (36^{e-9}) / 6430 = 5.6 \text{ pF}$$

The lower limit of this value of C is small enough that there will be voltage division occurring between this capacitor and the sum of the mission and test receiver input capacitance, reducing the amplitude perceived by the test receiver. A more realistic minimum capacitance would be 20 pF, and the value probably used would typically be much higher than that. To verify that the coupling capacitor is greater than 20 pF, we recalculate the coupling time constant

$$T_{HP} > 6.43\ \text{k}\Omega \times 20\ \text{pF} = 6430 \times 20^{e-12} = 128.6\ \text{ns}\ ;\ \text{set}\ T_{HP} > 130\ \text{ns}$$

D.2.2 BSDL

The BSDL for the driver has not changed. The BSDL AIO_Pin_Behavior entry for a pin using this test receiver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
   " LVDS_Alt : LP_time=18.0e-9 HP_time=130.0e-9 " &
   "       AIO_VTH=1250 AIO_VHyst=190 ; " &
    …
;
```

In addition, there would need to be Port, Pin_Data, and Pin_Grouping entries for pin “LVDS_Alt” and its complement, and a boundary-scan register entry for port “LVDS_Alt.”

D.3 LVDS with on-chip coupling

On-chip coupling, shown in Figure D.5, is very similar to the alternative coupling case just described in that to minimize the size of the coupling capacitors, they are placed after the termination resistor (which could also be on-chip.) The current loop is intact so that the source termination is not required. In this case, however, ac coupling is guaranteed by the design and there is no requirement for the self-referencing low-pass filter in the test receiver.

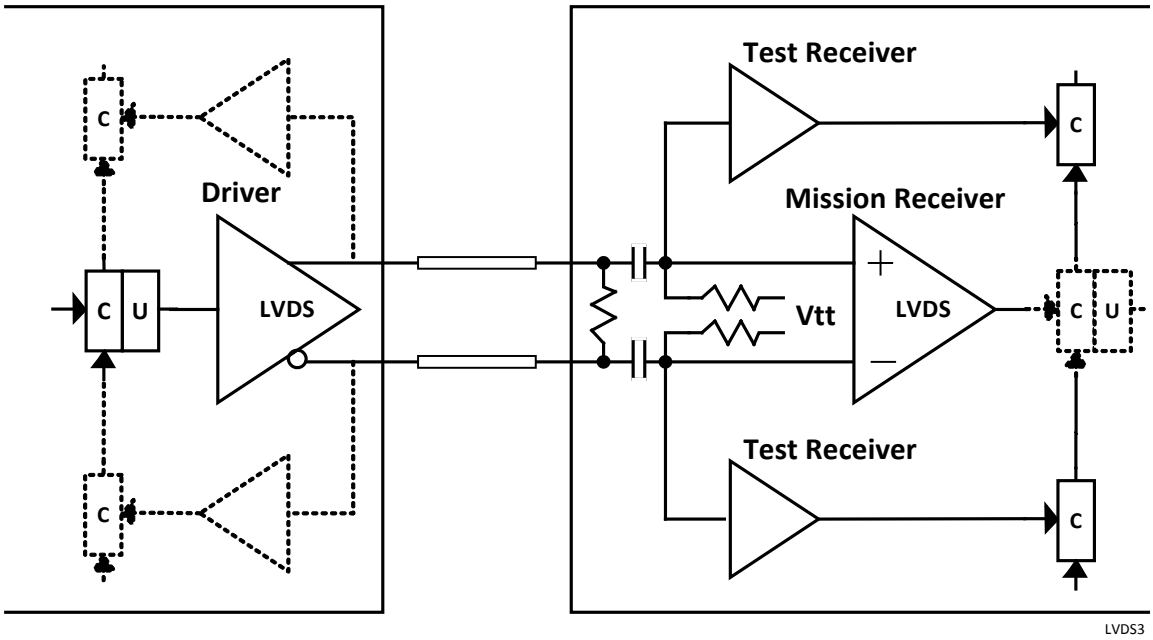


Figure D.5— LVDS with on-chip load termination and ac coupling

D.3.1 Calculations

Again, T_{Hyst} does not change. For a 70% threshold, Table 3, (using the "no LP" column), gives us an HP_Mult of 6, and the minimum T_{HP} is calculated as follows:

$$T_{HP} > \text{HP\_Mult} \times T_{Hyst} = 6 \times 2.0 \text{ ns} = 12 \text{ ns}$$

The expected maximum and minimum voltage swings at the receiver and the hysteretic thresholds are calculated as follows:

— $\Delta V_{\max} = \Delta V_{\max}(\text{driver}) = 480 \text{ mV}$

— $\Delta V_{\min} = 0.8 \times \Delta V_{\min}(\text{driver}) = 0.8 \times 338 \text{ mV} = 270 \text{ mV}$

— $V_{Hyst-Edge} = 0.7 \times \Delta V_{\min} = 0.7 \times 270 \text{ mV} = 190 \text{ mV}$

— $V_{Hyst-Level} = 0.5 \times V_{Hyst-Edge} = 0.5 \times 190 \text{ mV} = 95 \text{ mV}$

For the on-chip high-pass filter, assume an on-chip capacitor with a programmable bypass and a value of 1 pF, infinite input impedance for the active receivers, and then

$$R(\text{bias}) = (12e^{-9}) / (e^{-12}) = 12 \text{ k}\Omega$$

In this case, the coupling capacitor and the input capacitance of the mission and test receivers are very similar, leading to an apparent loss of amplitude at the receiver circuits of close to half. Clearly, the designer of this circuit would have to analyze and design the coupling, mission, and test receiver circuits as a single entity and verify its operation. The numbers presented here are illustrative.

D.3.2 BSDL

The BSDL for the driver has not changed. The BSDL AIO_Pin_Behavior entry for a pin using this test receiver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
   " LVDS_onchip : HP_time=12.0e-9 On_Chip_Programmable; " &
   "       AIO_VTH=1250 AIO_VHyst=190 ; " &
     …
;
```

In addition, there would need to be Port, Pin_Data, and Pin_Grouping entries for pin "LVDS_onchip" and its complement, and a boundary-scan register entry for port "LVDS_onchip."

D.4 LVPECL (low-voltage pseudo emitter-coupled logic)

LVPECL and other voltage-mode differential protocols do not have to maintain a current loop for operation, so there should be no concerns about requiring source termination. (This should be verified by analog simulation of the driver behavior when ac-coupled.) Figure D.6 shows an LVPECL channel with a normal board ac coupling topology.

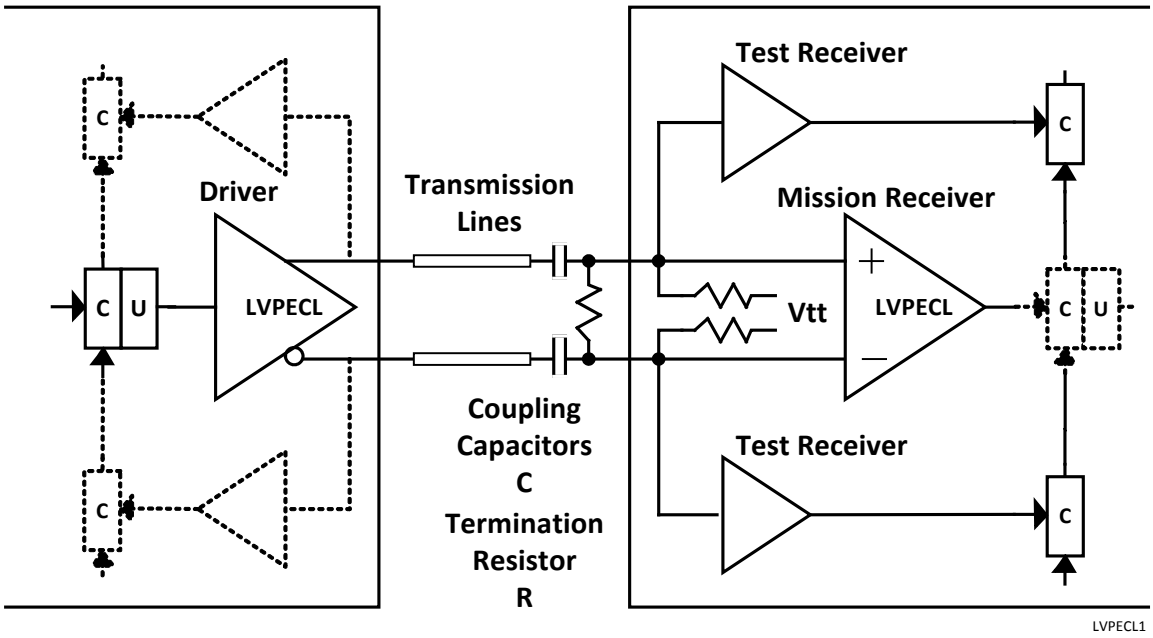


Figure D.6—LVPECL differential channel with on-board ac coupling before termination

D.4.1 Calculations

The first step is to examine the specifications of the driver and receiver to be implemented. For a specific LVPECL driver and receiver, in a specific technology, the specifications might be

— $Vdd = 3.3$ V

— V(common-mode) = 1.85 V to 2.15 V (2.00 V nominal)

— $\Delta V_{\min}(\text{driver}) = 600\ \text{mV}$，$\Delta V_{\max}(\text{driver}) = 1100\ \text{mV}$

— $f(\max) = 1000\ \text{MHz}$

The expected maximum and minimum voltage swings at the receivers are calculated* as follows:

— $\Delta V_{\max} = \Delta V_{\max}(\text{driver}) = 1100\ \text{mV}$

— $\Delta V_{\min} = 0.8 \times \Delta V_{\min}(\text{driver}) = 0.8 \times 600\ \text{mV} = 480\ \text{mV}$

The factor of 0.8 on the minimum voltage swing calculation is the designer's best estimate of the static signal attenuation that would be seen in most applications of this circuit. This factor should be adjusted using engineering judgment based on the expected use.

In this case, the transition time has not been explicitly stated, so T_{Trans} is set to one quarter the cycle time at f(max) and T_{Hyst} is set to a multiple of that

— $T_{Trans} = .25 \times 1^{e-9} = 250\ \text{ps}$

— $T_{Hyst} = 5 \times T_{Trans} = 5 \times 0.25\ \text{ns} = 1.25\ \text{ns}$

The hysteresis voltages are calculated based on the receiver ΔV_{Min} calculated above. As long as the T_{Test} duration is long enough, it is assumed that the full ΔV amplitude is seen at each transition, above or below the receiver reference voltage. The hysteretic threshold for the EXTEST_PULSE and EXTEST_TRAIN instructions are set anywhere in the range of 50% to 90% of the minimum voltage swing. Picking the middle of the range

$$V_{Hyst-Edge} = 0.7 \times \Delta V_{\min} = 0.7 \times 480\ \text{mV} = 336\ \text{mV}$$

For the EXTEST instruction, the test receiver must compare the input to a fixed threshold, reducing the amplitude to half of the ΔV above and below that threshold. Note that if the common-mode voltage of the driver did not match the internal reference voltage of the receiver, then either the hysteresis voltage should be reduced (for small differences), or this mode simply might not work, resulting in significantly increased difficulty in detecting shorted coupling capacitors.

$$V_{Hyst-Level} = 0.5 \times V_{Hyst-Edge} = 0.5 \times 336\ \text{mV} = 168\ \text{mV}$$

In this case, the specification sheet itself specifies that the common-mode voltage could vary 150 mV in either direction. This would leave very little noise margin compared to the calculated hysteresis voltage. Analog simulations of the EXTEST instruction with best and worst case drivers would be needed to determine whether and how to adjust this value.

From Table 3, for a hysteresis level of 70% of ΔV_{Min}:

— HP_Mult = 18

— LP_Mult = 9

In this case, the ΔV_{Max} to $V_{Hyst\text{-}Edge}$ ratio is less than four, and no severe noise issues are know to exist in any of the test environments, so the HPLP_Ratio of two is acceptable based on Table 3.

The time constants for the self-referencing low-pass filter, the minimum time constant for the board high-pass coupling time constant, and the minimum time between test transitions can now be calculated

— $T_{LP} = \text{LP\_Mult} \times T_{HYST} = 9 \times 1.25\ \text{ns} = 11.25\ \text{ns}$

— $T_{HP} = \text{HP\_LP\_Ratio} \times T_{LP} = 2 \times 11.25\ \text{ns} = 22.5\ \text{ns}$

— $T_{Test} > 3 \times T_{LP} = 3 \times 11.25\ \text{ns} = 33.75\ \text{ns}$

The values for the self-reference low-pass filter in the test receiver can now be calculated. Assuming an on-chip capacitor of 1 pF to ground, then

$$R = (11.25e^{-9}) / (1e^{-12}) = 11.25 \text{ k}\Omega$$

For normal board ac coupling with the capacitors before termination, the minimum coupling capacitance can be calculated. Assuming a normal termination of 50 Ω per leg (100 Ω across the two legs)

$$C > (22.5e^{-9}) / 50 = 450 \text{ pF}$$

Finally, the bias voltage value for V_{tt} in Figure D.6 and the threshold voltage of the test receiver in dc test mode (EXTEST) must be selected. The common mode voltage described above for this technology is chosen: 2.0V. Given the possible range, however, it would be preferred that this value be made programmable so that a value more closely matched to the actual driver could be used during the shorted capacitor test using the EXTEST instruction.

D.4.2 BSDL

The BSDL AIO_Pin_Behavior entry for a pin using this LVPECL driver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
  " LVPECL_DRVR : AIO_VCM=2000 AIO_VPP=600; " &
    …
   ;
```

The BSDL AIO_Pin_Behavior entry for a pin using this test receiver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
   " LVPECL_RCVR : LP_time=11.25e-9 HP_time=22.5e-9; " &
   "       AIO_VTH=2000 AIO_VHyst=336 ; " &
     …
;
```

In addition, there would need to be Port, Pin_Data, and Pin_Grouping entries for pins "LVPECL_DRVR," "LVPECL_RCVR," and their complements, and boundary-scan register entries for ports "LVPECL_DRVR" and "LVPECL_RCVR."

D.5 LVPECL with guaranteed on-board ac coupling

In an application (such as an ASIC library) where the I/O designer can require ac coupling on the board (presumably, there is another version of the circuit for dc-coupled applications), then there is no requirement for the self-referencing low-pass filter in the test receiver. Figure D.6 does not change and still represents the board circuit.

D.5.1 Calculations

For the on-board high-pass filter, Table 3 (in 6.2.3.3, "no LP" column) gives an HP_Mult of 6. Calculate the minimum T_{HP} value

$$T_{HP} > \text{HP\_Mult} \times T_{Hyst} = 6 \times 1.25 \text{ ns} = 7.5 \text{ ns}$$

With an assumed termination resistance of 50 Ω, and infinite input impedance for the active receivers, then

$$C > 7.5 \text{ ns}/50\ \Omega = 7.5^{e-9}/50 = 150 \text{ pF}$$

This is large enough to not cause any capacitive voltage division. All other time and voltage calculations remain the same.

D.5.2 BSDL

The BSDL AIO_Pin_Behavior entry for a pin using this test receiver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
   " LVPECL_noLP : HP_time=7.5e-9; " &
   "       AIO_VTH=2000 AIO_VHyst=336 ; " &
     …
;
```

In addition, there would need to be Port, Pin_Data, and Pin_Grouping entries for pin "LVPECL_noLP" and its complement, and a boundary-scan register entry for port "LVPECL_noLP."

D.6 LVPECL with on-chip coupling

On-chip coupling places the coupling capacitors after the termination resistor (which could also be on-chip) in order to minimize the capacitor size as shown in Figure D.7. In this case, ac coupling is guaranteed by design and there is no requirement for the self-referencing low-pass filter in the test receiver. Again, for simplicity, we assume the on-chip capacitor is bypassed under the control of a TDR field.

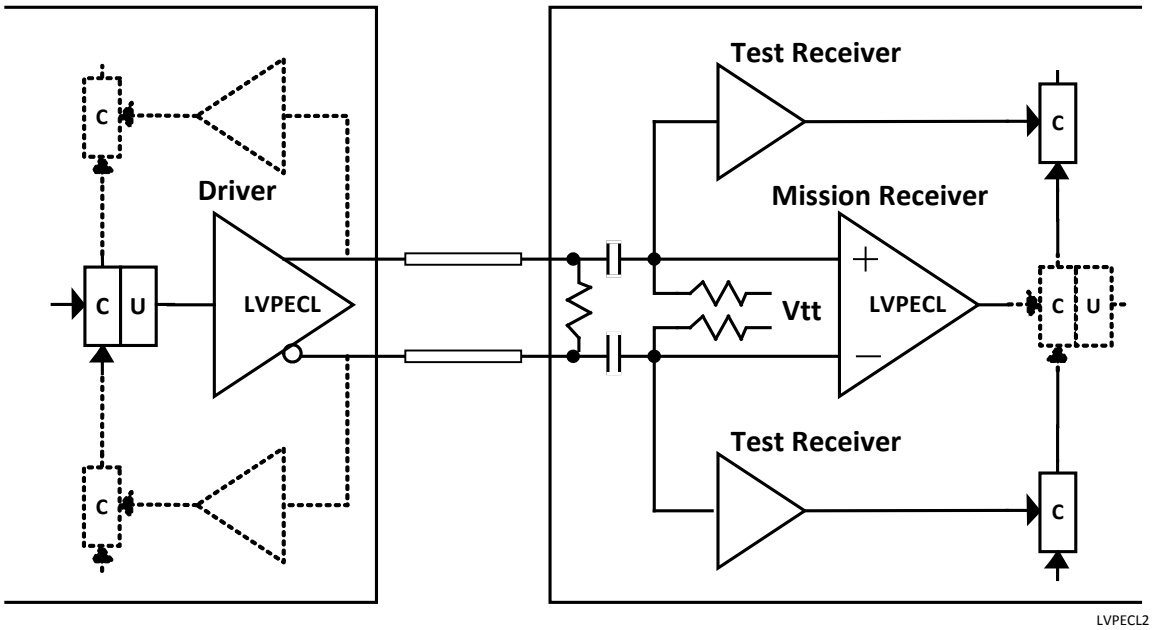


Figure D.7— LVPECL with on-chip ac coupling after termination

D.6.1 Calculations

For the on-chip high-pass filter, Table 3, the "no LP" column gives an HP_Mult of 6. Calculate the actual T_{HP} value

$$T_{HP} = \text{HP\_Mult} \times T_{Hyst} = 6 \times 1.25 \text{ ns} = 7.5 \text{ ns}$$

With an assumed an on-chip coupling capacitor of 1 pF, infinite input impedance for the active receivers, then

— $R(\text{bias}) = \left(7.5e^{-9}\right) \Big/ \left(e^{-12}\right) = 7500\ \Omega$

All other time and voltage calculations remain the same. Again, the coupling capacitor and the input capacitance of the mission and test receivers are very similar, leading to an apparent loss of amplitude at the receiver circuits of close to half. Clearly, the designer of this circuit would have to analyze and design the coupling, mission, and test receiver circuits as a single entity and verify its operation. The numbers presented here are illustrative.

D.6.2 BSDL

The BSDL AIO_Pin_Behavior entry for a pin using this test receiver would be:

```
attribute AIO_Pin_Behavior of ACDEV1 : entity is
   " LVPECL_onchip : HP_time=7.5e-9 On_Chip_Programmable " &
   "       AIO_VTH=2000 AIO_VHyst=336 ; " &
     …
;
```

In addition, there would need to be Port, Pin_Data, and Pin_Grouping entries for pin "LVPECL_onchip" and its complement, and a boundary-scan register entry for port "LVPECL_onchip."

Annex E

(informative)

A proposed “INITIALIZE” instruction

This annex is omitted from this revision of this standard. The topic of test data initialization has been included in the revised IEEE Std 1149.1-2013, Clause 14, Initialization data register.

Annex F

(informative)

Bibliography

[B1] IEEE Std 1149.8.1™, IEEE Standard for boundary-scan-Based Stimulus of Interconnections to Passive and/or Active Components.[7,8]

[7] The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc
[8] IEEE publications are available from The Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

Consensus

WE BUILD IT.

Connect with us on:

Facebook: https://www.facebook.com/ieeesa

Twitter: @ieeesa

LinkedIn: http://www.linkedin.com/groups/IEEESA-Official-IEEE-Standards-Association-1791118

IEEE-SA Standards Insight blog: http://standardsinsight.com

YouTube: IEEE-SA Channel

IEEE
standards.ieee.org
Phone: +1 732 981 0060 Fax: +1 732 562 1571