

IEEE Standard Test Access Port and Boundary-Scan Architecture

Sponsor

Test Technology Standards Committee
of the
IEEE Computer Society

Reaffirmed 27 March 2008

Approved 14 June 2001

IEEE-SA Standards Board

Abstract: Circuitry that may be built into an integrated circuit to assist in the test, maintenance, and support of assembled printed circuit boards is defined. The circuitry includes a standard interface through which instructions and test data are communicated. A set of test features is defined, including a boundary-scan register, such that the component is able to respond to a minimum set of instructions designed to assist with testing of assembled printed circuit boards. Also, a language is defined that allows rigorous description of the component-specific aspects of such testability features.

Keywords: boundary scan, boundary-scan architecture, Boundary-Scan Description Language, boundary-scan register, BSDL, circuit boards, circuitry, integrated circuit, printed circuit boards, TAP, test, test access port, VHDL, VHSIC Hardware Description Language

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

Copyright © 2001 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 23 July 2001. Printed in the United States of America.

Print: ISBN 0-7381-2944-5 SH94949
PDF: ISBN 0-7381-2945-3 SS94949

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

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

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied “**AS IS.**”

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. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, 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 publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received 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 interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: 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 with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

IEEE is the sole entity that may authorize the use of certification marks, trademarks, or other designations to indicate compliance with the materials set forth herein.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (978) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

(This introduction is not part of IEEE Std 1149.1-2001, Standard Test Access Port and Boundary-Scan Architecture.)

This standard defines a test access port and boundary-scan architecture for digital integrated circuits and for the digital portions of mixed analog/digital integrated circuits. The facilities defined by the standard seek to provide a solution to the problem of testing assembled printed circuit boards and other products based on highly complex digital integrated circuits and high-density surface-mounting assembly techniques. They also provide a means of accessing and controlling design-for-test features built into the digital integrated circuits themselves. Such features might, for example, include internal scan paths and self-test functions as well as other features intended to support service applications in the assembled product.

History of the development of this standard

The process of developing this standard began in 1985, when the Joint European Test Action Group (JETAG) was formed in Europe. During 1986, this group expanded to include members from both Europe and North America and, as a result, was renamed the Joint Test Action Group (JTAG). Between 1986 and 1988, the JTAG Technical Subcommittee developed and published a series of proposals for a standardized form of boundary scan. In 1988, the last of these proposals—JTAG Version 2.0—was offered to the IEEE Testability Bus Standards Committee (P1149) for inclusion in the standard then under development. The Testability Bus Standards Committee accepted this approach. It decided that the JTAG proposal should become the basis of a standard within the Testability Bus family, with the result that the P1149.1 project was initiated. Following these decisions, the JTAG Technical Subcommittee became the core of the IEEE Working Group that developed this standard.

After the initial approval of this standard in February 1990 and its subsequent publication, the Working Group immediately began efforts to develop a supplement for the purpose of correction, clarification, and enhancement. This effort, spurred and guided by interaction between developers and users of the original standard, culminated in IEEE Std 1149.1a-1993, which was approved in June 1993.

The major changes to this standard introduced by IEEE Std 1149.1a-1993 were

- The addition of two optional instructions, *CLAMP* and *HIGHZ*, which standardized the names and specifications of features often implemented as design-specific features
- The addition of an optional facility to switch a component from a mode in which it complies to this standard into one in which it supports another design-for-test approach

Further, starting with a proposal made by Kenneth P. Parker and Stig Oresjo in 1990, an effort was undertaken to develop a language to describe components that conform to this standard. This effort concluded in the approval of IEEE Std 1149.1b-1994 in September 1994.

The major change introduced to this standard by IEEE Std 1149.1b-1994 was the addition of Annex B, which defines the Boundary-Scan Description Language. All other changes were minor and were strictly for clarification.

Changes introduced by this revision

This revision is primarily a housekeeping update, designed to consolidate learning from the first 10 years of the standard's use into the standard document.

The principal changes introduced are

- To reduce the risk of accidental entry into test mode, the requirement that a binary code for the *EXTTEST* instruction be {000...0} has been removed and use of this binary code for other instructions that result in entry to test mode has been deprecated [see 8.1.1 e), 8.8.1 h), B.8.11.3 d), and B.8.11.3 e)].
- To increase the flexibility with which instructions may be implemented and merged, the implicitly merged *SAMPLE/PRELOAD* instruction has been redefined as two separate instructions: *SAMPLE* and *PRELOAD*. *These instructions can continue to share a single binary code*, effectively resulting in a merged *SAMPLE/PRELOAD* instruction, but alternatively, they may now share binary codes with other instructions, provided that no rules for any of the merged instructions are violated. [see 8.1.1 g), 8.2.1 b), 8.6, 8.7, B.5.1.2, B.6.3 b), B.8.11.3 f), B.8.11.3 g), B.8.11.3 h), B.8.11.3 i), B.8.11.3 j), and B.8.13.3 a)].
- To enable more efficient implementation of boundary-scan register cells provided at system logic outputs, the source of data to be captured in such cells in response to the *SAMPLE* instruction is now allowed to be at the connected system pin [see 8.6.1 d), 11.6.1 a), 11.6.1 b), B.8.14.4 l), and B.10.2.4.1 b)]. Additionally, three new cell types based on this implementation (**BC_8**, **BC_9**, and **BC_10**) have been added to the Standard VHDL Package (see B.5.1.2, B.9, and Table B.3).
- To permit more flexible boundary-scan register cell implementations, sharing of circuitry between the boundary-scan register and other elements of the test and/or system logic has been allowed in limited cases [see 11.2.1 j) and 11.2.1 k)].
- To support more complete description of IC pin drivers with bus keeper circuits, a new value for <disable result> has been defined (**KEEPER**, see B.5.1.2, B.8.14.1, and B.8.14.3.7).
- To track the widespread acceptance of BSDL, the language has been made a normative part of the standard and its use for documentation has been mandated (see 13.3.1 c) and Annex B).

Additionally, a number of minor changes were made to correct and clarify the language of the standard [of special note, see 4.8.1 c), 11.7.1 b), B.8.14.4 k), and B.8.14.4 n)].

Participants

At the time of the issue of IEEE Std 1149.1-2001, the members of the working group were:

Christopher J. Clark, *Chair*

Adam W. Ley, *Editor*

John Andrews
Bill Aronson
Carl F. Barnhart
Dilip K. Bhavsar
Terry Borroz
John Braden
Bill Bruce
Tapan J. Chakraborty
Sung Chung
Jim Coleman
Frans De Jong
Bulent Dervisoglu

Ted Eaton
Bill Eklow
Dean Geerdes
Grady L. Giles
Peter Hansen
Neil Jacobson
Najmi Jarwala
London Jin
Wuudiann Ke
Arthur Khu
Ramesh Krishnamurthy
Tom Langford
Larry Lee

Colin Maunder
Benoit Nadeau-Dostie
Jay Nejedlo
Kenneth P. Parker
Michael Ricchetti
Gordon D. Robinson
Robert J. Russell
Adam Sheppard
Steve Stark
Ron Walther
Lee Whetsel
Thomas W. Williams

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

John Andrews
Carl Barnhart
Roger Bennetts
Terry Borroz
John Braden
Keith Chow
Sung S. Chung
Chris J. Clark
Frans de Jong
Ted Eaton
William Eklow
Dean Geerdes

Grady Giles
Peter Hansen
Lee F. Horney
Mitsuaki Ishikawa
Neil Jacobson
Jake Karrfalt
Thomas L. Langford
Adam W. Ley
Gregory A. Maston
Colin Maunder
Patrick McHugh

Earl J. Meiers
Yinghua Min
James A. Monzel
Benoit Nadeau-Dostie
Jay Nejedlo
Bruce E. Peterson
Mike Ricchetti
Gordon Robinson
Jaideep Roy
Robert J. Russell
Scott A. Yalcourt
T. W. Williams

When the IEEE-SA Standards Board approved this standard on 14 June 2001, it had the following membership:

Donald N. Heirman, *Chair*
James T. Carlo, *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Mark D. Bowman
Gary R. Engmann
Harold E. Epstein
H. Landis Floyd
Jay Forster*
Howard M. Frazier
Ruben D. Garzon

James H. Gurney
Richard J. Holleman
Lowell G. Johnson
Robert J. Kennelly
Joseph L. Koepfinger*
Peter H. Lips
L. Bruce McClung
Daleep C. Mohla

James W. Moore
Robert F. Munzner
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Akio Tojo
Donald W. Zipse

*Member Emeritus

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

Alan Cookson, *NIST Representative*
Donald R. Volzka, *TAB Representative*

Don Messina
IEEE Standards Project Editor

Contents

1.	Overview.....	1
1.1	Scope.....	1
1.2	Purpose.....	1
1.3	Document outline.....	5
1.4	Conventions	6
2.	References.....	6
3.	Definitions and Acronyms	6
3.1	Definitions	6
3.2	Acronyms.....	9
4.	Test access port (TAP).....	9
4.1	Connections that form the test access port (TAP)	9
4.2	Test clock input (TCK)	9
4.3	Test mode select input (TMS)	11
4.4	Test data input (TDI)	11
4.5	Test data output (TDO).....	12
4.6	Test reset input (TRST*)	12
4.7	Interconnection of components compatible with this standard	13
4.8	Subordination of this standard within a higher-level test strategy.....	15
5.	Test logic architecture.....	16
5.1	Test logic design	17
5.2	Test logic realization.....	17
6.	The TAP controller	18
6.1	TAP controller state diagram	18
6.2	TAP controller operation	24
6.3	TAP controller initialization	30
7.	The instruction register	33
7.1	Design and construction of the instruction register	33
7.2	Instruction register operation	34
8.	Instructions.....	36
8.1	Response of the test logic to instructions.....	36
8.2	Public instructions.....	37
8.3	Private instructions	38
8.4	The <i>BYPASS</i> instruction.....	38
8.5	Boundary-scan register instructions.....	39
8.6	The <i>SAMPLE</i> instruction	41
8.7	The <i>PRELOAD</i> instruction	43
8.8	The <i>EXTEST</i> instruction	45

8.9	The <i>INTEST</i> instruction	48
8.10	The <i>RUNBIST</i> instruction	53
8.11	The <i>CLAMP</i> instruction	56
8.12	Device identification register instructions	57
8.13	The <i>IDCODE</i> instruction	58
8.14	The <i>USERCODE</i> instruction	58
8.15	The <i>HIGHZ</i> instruction	59
9.	Test data registers	61
9.1	Provision of test data registers	61
9.2	Design and construction of test data registers	63
9.3	Test data register operation	64
10.	The bypass register	66
10.1	Design and operation of the bypass register	66
11.	The boundary-scan register	67
11.1	Introduction	68
11.2	Register design	72
11.3	Register operation	74
11.4	General rules regarding cell provision	76
11.5	Provision and operation of cells at system logic inputs	79
11.6	Provision and operation of cells at system logic outputs	87
11.7	Provision and operation of cells at bidirectional system logic pins	102
11.8	Redundant cells	108
11.9	Special cases	109
12.	The device identification register	111
12.1	Design and operation of the device identification register	112
12.2	Manufacturer identity code	114
12.3	Part-number code	115
12.4	Version code	115
13.	Conformance and documentation requirements	115
13.1	Claiming conformance to this standard	115
13.2	Prime and second source components	117
13.3	Documentation requirements	117
Annex A (informative)	An example implementation using level-sensitive design techniques	120
Annex B (normative)	Boundary-scan description language	129

IEEE Standard Test Access Port and Boundary-Scan Architecture

1. Overview

1.1 Scope

This standard defines test logic that can be included in an integrated circuit to provide standardized approaches to

- testing the interconnections between integrated circuits once they have been assembled onto a printed circuit board or other substrate;
- testing the integrated circuit itself; and
- observing or modifying circuit activity during the component's normal operation.

The test logic consists of a boundary-scan register and other building blocks and is accessed through a Test Access Port (TAP).

1.2 Purpose

1.2.1 An overview of the operation of IEEE Std 1149.1

This subclause provides a general overview of the operation of a component compatible with this standard and provides a background to the detailed discussion in later subclauses.

The circuitry defined by this standard allows test instructions and associated test data to be fed into a component and, subsequently, allows the results of execution of such instructions to be read out. All information (instructions, test data, and test results) is communicated in a serial format.

The sequence of operations would be controlled by a bus master, which could be either an automatic test equipment (ATE) or a component that interfaces to a higher-level test bus as a part of a complete system maintenance architecture. Control is achieved through signals applied to the Test Mode Select (TMS) and Test Clock (TCK) inputs of the various components connected to the bus master. Starting from an initial state in which the test circuitry defined by this standard is inactive, a typical sequence of operations would be as follows.

The first steps would be, in general, to load serially into the component the instruction binary code for the particular operation to be performed. The test logic defined by this standard is designed such that the serial movement of instruction information is not apparent to those circuit blocks whose operation is controlled by the instruction. The instruction applied to these blocks changes only on completion of the shifting (instruction load) process.

Once the instruction has been loaded, the selected test circuitry is configured to respond. In some cases, however, it is necessary to load data into the selected test circuitry before a meaningful response can be made. Such data is loaded into the component serially in a manner analogous to the process used previously to load the instruction. Note that the movement of test data has no effect on the instruction present in the test circuitry.

After execution of the test instruction, based where necessary on supplied data, the results of the test can be examined by shifting data out of the component to or through the bus master.

Note that in cases where the same test operation is to be repeated but with different data, new test data can be shifted into the component while the test results are shifted out. There is no need for the instruction to be reloaded.

Operation of the test circuitry may proceed by loading and executing several further instructions in a manner similar to that described and would conclude by returning the test circuitry and, where required, on-chip system circuitry to its initial state.

1.2.2 The use of IEEE Std 1149.1 to test an assembled product

This subclause outlines the use of the boundary-scan circuitry defined by this standard during the process of testing an assembled product such as a printed circuit board.

The test problem for any product constructed from a collection of components can be decomposed into three goals:

- a) To confirm that each component performs its required function;
- b) To confirm that the components are interconnected in the correct manner; and
- c) To confirm that the components in the product interact correctly and that the product performs its intended function.

This approach can be applied to a board constructed from integrated circuits, to a system constructed from printed circuit boards, or to a complex integrated circuit constructed from a set of simpler functional modules. To simplify the discussion, this description henceforth will concentrate on the case of an assembled printed circuit board constructed from a collection of digital integrated circuits.

At the board level, goal a) and goal b) typically are achieved by using in-circuit test techniques; for goal c), a functional test is required. However, in-circuit test techniques have significant limitations when viewed against evolving surface-mount interconnection technology, for example, the difficulty of making reliable contact to miniaturized features of the printed circuit board using a bed-of-nails fixture. How, then, might the above three test goals be achieved if test access becomes limited to the normal circuit connections, plus a relatively small number of special-purpose test connections?

Considering goal a), it is clear that the vendor of an integrated circuit used in the board-level design will have an established test methodology for that component. The components could be tested on a proprietary ATE system or by using a self-test procedure embedded in the design. Information on the test methodology adopted is typically not available to the component purchaser. Even where self-test modes of operation are known to exist, they may not be documented and therefore are not available to the component user. Alternative sources of test data for the board test engineer may be the component test libraries supplied with

in-circuit test systems or the test programs developed by component users for incoming inspection of delivered devices.

Wherever the test data for a component originates, the next step is to use it once the component has been assembled onto the printed circuit board. If access is limited to the normal connections of the assembled circuit, this task may be far from simple. This is particularly true if the surrounding components are complex or if the board designer has tied some of the components' connections to fixed logic levels or has left component pins unconnected. Normally, it will not be possible to test the component in the same way that it was tested in isolation unless an in-circuit test is achievable.

To ensure that built-in test facilities can be used or that preexisting test patterns can be applied, a framework is needed that can be used to convey test data to or from the boundaries of individual components so that they can be tested as if they were freestanding. This framework will also allow access to and control of built-in test facilities of components. Boundary scan coupled with a test access bus provides such a framework.

The objective of this standard is to define a boundary-scan architecture that can be adopted as a standard feature of integrated circuit designs, thus allowing the required test framework to be created on assembled printed circuit boards and other products.

1.2.3 What is boundary scan?

The boundary-scan technique involves the inclusion of a shift-register stage (contained in a boundary-scan register cell) adjacent to each component pin so that signals at component boundaries can be controlled and observed using scan testing principles.

Figure 1-1 illustrates an example implementation for a boundary-scan register cell that could be used for an input or output connection to an integrated circuit. Dependent on the control signals applied to the multiplexers, data can be either loaded into the scan register from the Signal-in port (e.g., the input pin) or driven from the register through the Signal-out port of the cell (e.g., into the core of the component design). As will be discussed in detail in Clause 11, the second flip-flop (controlled by input Clock B) is provided to ensure that the signals driven out of the cell in the latter case are held while new data is shifted into the cell using input Clock A. This flip-flop is not required in all cases but is included in Figure 1-1 to simplify the discussion.

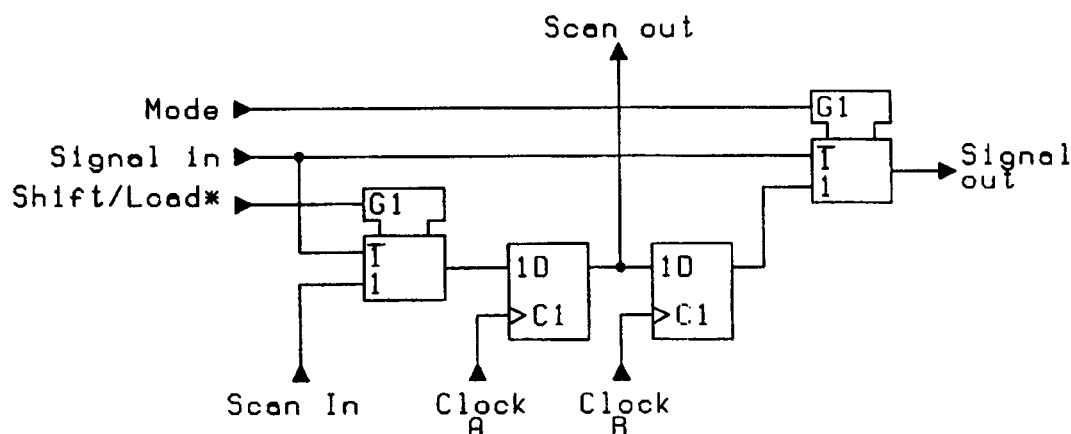


Figure 1-1—A boundary-scan register cell

The boundary-scan register cells for the pins of a component are interconnected to form a shift-register chain around the border of the design, and this path is provided with serial input and output connections and

appropriate clock and control signals. Within a product assembled from several integrated circuits the boundary-scan registers for the individual components could be connected in series to form a single path through the complete design, as illustrated in Figure 1-2. Alternatively, a board design could contain several independent boundary-scan paths.

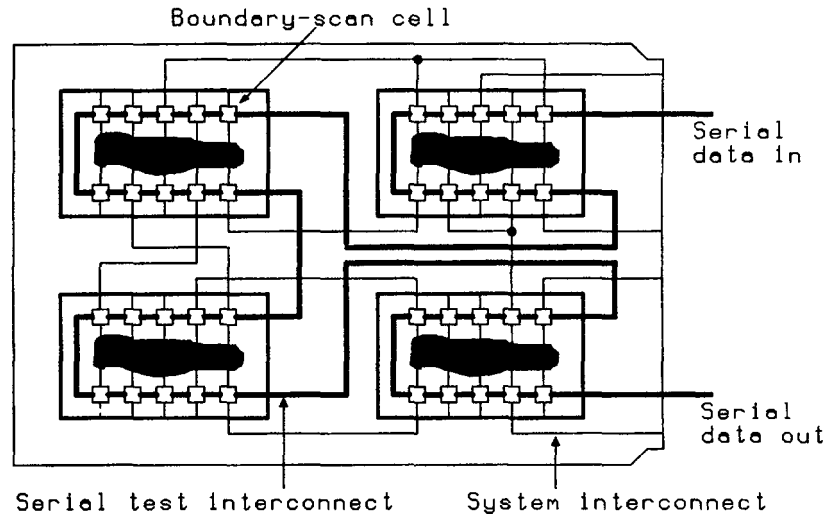


Figure 1-2—A boundary-scannable board design

If all the components used to construct a circuit have a boundary-scan register, the resulting serial path through the complete design can be used in two ways:

- a) To allow the interconnections between the various components to be tested, test data can be shifted into all the boundary-scan register cells associated with component output pins and loaded in parallel through the component interconnections into those cells associated with input pins; and
- b) To allow the components on the board to be tested, the boundary-scan register can be used as a means of isolating on-chip system logic from stimuli received from surrounding components while an internal self-test is performed. Alternatively, if the boundary-scan register is suitably designed, it can permit a limited slow-speed static test of the on-chip system logic since it allows delivery of test data to the component and examination of the test results.

These tests allow the first two goals discussed earlier to be achieved through the use of the boundary-scan register. In effect, tests applied using the register can detect many of the faults that in-circuit testers currently address, but without the need for extensive bed-of-nails access. The third goal—to functionally test the operation of the complete product—remains and can be achieved either by using a functional (through the pins) ATE system or by using a system-level self-test, for example.

Note also that by parallel loading the cells at both the inputs and outputs of a component and shifting out the results, the boundary-scan register provides a means of “sampling” the data flowing through a component without interfering with the behavior of the component or the assembled board. This mode of operation is valuable for design debugging and fault diagnosis since it permits examination of connections not normally accessible to the test system.

1.2.4 The use of IEEE Std 1149.1 to achieve other test goals

In addition to its application in testing printed circuit assemblies and other products containing multiple components, the test logic defined by this standard can be used to provide access to a wide range of design-

for-test features built into the components themselves. Such features might include internal scan paths, self-test functions [e.g., using built-in logic block observer (BILBO) elements], or other support functions.

Design-for-test features such as these can be accessed and controlled using the data path between the serial test data pins of the TAP defined by this standard. Instructions that cause internal reconfiguration of the component's system logic such that the test operation is enabled may be shifted into the component through the TAP.

1.3 Document outline

Circuit designs such as that defined by this standard are more easily understood if their specifications are accompanied by general descriptive material that places the details of the various parts of the design in perspective and provides examples of implementation. Clause 1 therefore contains an overview of the application of this standard to the testing of the digital portions of an electronic product consisting of many integrated circuits.

Subsequent clauses of the document contain the specifications for particular features of this standard. Two classes of material are contained in these clauses:

1.3.1 Specifications

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

- a) *Rules* specify the mandatory aspects of this standard. Subclauses that are rules contain the word *shall*.
- b) *Recommendations* indicate preferred practice for designs that seek to conform to this standard. Subclauses that are recommendations contain the word *should*.
- c) *Permissions* show how optional features may be introduced into a design that seeks to conform to this standard. These features will extend the application of the test circuitry defined by this standard. Subclauses that are permissions contain the word *may*.

1.3.2 Descriptions

Material not contained in subclauses entitled Specifications is descriptive material that illustrates the need for the features being specified or their application. This material includes schematics that illustrate a possible implementation of the specifications in this standard. Annex A to this standard contains an alternative implementation example. The descriptive material also discusses design decisions made during the development of this standard.

CAUTION: The descriptive material contained in this standard is for illustrative purposes only and does not define a preferred implementation. Examples are provided throughout this standard to illustrate possible circuit implementations. Where discrepancies between examples and specifications may occur, the specifications always take precedence. Readers should exercise caution when using these examples to ensure full compliance in their specific applications. In particular, it is emphasized that the examples are designed to effectively communicate the meaning of this standard. As such, they are logically correct; however, as always, a particular implementation may not operate properly with respect to timing and other parametric characteristics. One example of this concern is that the sample TAP controller implementation depicted in Figure 6-5 reasonably assumes a significantly greater delay in the flip-flop sourcing A and A* than in the inverter sourcing TCK*. It is possible to design a circuit where this assumption is violated, causing a critical race to occur that would invalidate the behavior of the TAP controller. Therefore, it is recommended that implementations be fully verified to ensure compliance under required operating conditions.

1.4 Conventions

The following conventions are used in this standard:

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

	15.1.1	c)	2)
Subclause number			
Index			
Option (if any)			

- b) Instruction and state names defined in this standard are shown in *italic* type in the text of this standard.
- c) Names of states and signals that pertain to the test data registers defined by this standard contain the characters DR, while those that pertain to the instruction register contain the characters IR.
- d) Names for signals that are active in their low state have an asterisk as the final character, e.g., TRST*.
- e) A positive logic convention is used; i.e., a logic 1 signal is conveyed as the more positive of the two voltages used for logic signals.

2. References

This standard shall be used in conjunction with the following standards. When the following standards are superseded by an approved revision, the revision shall apply.

EIA/JEP106, JEDEC Publication 106, Standard Manufacturer's Identification Code¹

IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition²

IEEE Std 1076-1993, IEEE Standard VHDL Language Reference Manual

IEEE Std 1149.4-1999, IEEE Standard for a Mixed Signal Test Bus

3. Definitions and Acronyms

For the purposes of this standard, the following terms and definitions apply. IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition should be referenced for terms not defined in this clause.

3.1 Definitions

3.1.1 active: When associated with a logic level (e.g., active-low), this term identifies the logic level to which a signal shall be set to cause the defined action to occur. When referring to an output driver (e.g., an active drive), this term describes the state in which the driver is capable of determining the voltage of the network to which it is connected.

¹Copies can be obtained from EIA/JEDEC, 2500 Wilson Blvd., Arlington, VA 22201-3834, USA (www.jedec.org).

²IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (<http://standards.ieee.org/>).

3.1.2 bidirectional pin: A component pin that can either drive or receive signals from external connections.

3.1.3 blind interrogation: Access to a facility (e.g., the device identification register) without prior knowledge of the test logic operation of the specific component being accessed.

3.1.4 built-in logic block observer (BILBO): A shift-register based structure used in some forms of self-testing circuit design.

3.1.5 capture: Load a value into a data register or the instruction register as a consequence of entry into the *Capture-DR* or *Capture-IR* controller state, respectively.

3.1.6 chip-on-board testing: A test of a component after it has been assembled onto a printed circuit board or other substrate.

3.1.7 clock: A signal where transitions between the low and high logic levels (or vice versa) are used to indicate when a stored-state device, such as a flip-flop or latch, may perform an operation.

3.1.8 falling edge: A transition from a high to a low logic level. In positive logic, a change from logic 1 to logic 0. Events that are specified to occur on the rising (falling) edge of a signal should be completed within a fixed (frequency-independent) delay specified by the component supplier.

3.1.9 high: The higher of the two voltages used to convey a single bit of information. For positive logic, a logic 1.

3.1.10 inactive: When referring to an output driver (e.g., an inactive drive), this term describes the state in which the driver is not capable of determining the voltage of the network to which it is connected.

3.1.11 input pin: A component pin that receives signals from an external connection.

3.1.12 instruction: A binary data word shifted serially into the test logic in order to define its subsequent operation.

3.1.13 least significant bit (LSB): The digit in a binary number representing the lowest numerical value. For shift-registers, the bit located nearest to the serial output, or the first bit to be shifted out. The least significant bit of a binary word or shift-register is numbered 0.

3.1.14 level-sensitive scan design (LSSD): A variant of the scan design technique that results in race-free, testable digital electronic circuits.

3.1.15 low: The lower of the two voltages used to convey a single bit of information. For positive logic, a logic 0.

3.1.16 most significant bit (MSB): The digit in a binary number representing the greatest numerical value. For shift-registers, the bit farthest from the serial output, or the last bit to be shifted out. Logic values expressed in binary form are shown with the most significant bit on the left.

3.1.17 nonclock: A signal where the transitions between the low and high logic levels do not themselves cause operation of stored-state devices. The logic level is important only at the time of a transition on a clock signal.

3.1.18 output pin: A component pin that drives signals onto external connections.

3.1.19 pin: The point at which connection is made between the integrated circuit and the substrate on which it is mounted (e.g., the printed circuit board). For packaged components, this would be the package pin; for components mounted directly on the substrate, this would be the bonding pad.

3.1.20 prime source: In the event that several vendors offer pin-for-pin compatible components, the prime source is the vendor that introduced the component type.

3.1.21 private: A design feature intended solely for use by the component manufacturer.

3.1.22 public: A design feature, documented in the component data sheet, that may be used by purchasers of the component.

3.1.23 reset: The establishment of an initial logic condition that can be either logic 0 or logic 1, as determined by the context.

3.1.24 rising edge: A transition from a low to a high logic level. In positive logic, a change from logic 0 to logic 1. Events that are specified to occur on the rising (falling) edge of a signal should be completed within a fixed (frequency-independent) delay specified by the component supplier.

3.1.25 scan design: A design technique that introduces shift-register paths into digital electronic circuits and thereby improves their testability.

3.1.26 scan path: The shift-register path through a circuit designed using the scan design technique.

3.1.27 second source: In the event that several vendors offer pin-for-pin compatible components, second-source suppliers are vendors of the component other than the prime source.

3.1.28 selected test data register: A test data register is selected when it is required to operate by an instruction supplied to the test logic.

3.1.29 signature analysis: A technique for compressing a sequence of logic values output from a circuit under test into a small number of bits of data (signature) that, when compared to stored data, will indicate the presence or absence of faults in the circuit.

3.1.30 stand-alone testing: A test of a component performed before it is assembled onto a board or other substrate, for example, using Automatic Test Equipment (ATE).

3.1.31 stuck-at fault: A failure in a logic circuit that causes a signal connection to be fixed at 0 or 1 regardless of the operation of the circuitry that drives it.

3.1.32 system: Pertaining to the nontest function of the circuit.

3.1.33 system logic: Any item of logic that is dedicated to realizing the nontest function of the component or is at the time of interest configured to achieve some aspect of the nontest function.

3.1.34 system pin: A component pin that feeds, or is fed from, the on-chip system logic.

3.1.35 test logic: Any item of logic that is a dedicated part of the test logic architecture or is at the time of interest configured as a part of the test logic architecture.

3.1.36 update: Transfer of a logic value from the shift-register stage of a data register cell or an instruction register cell into the latched parallel output stage of the cell as a consequence of the falling edge of the test clock input in the *Update-DR* or *Update-IR* controller state, respectively.

3.1.37 3-state pin: A component output pin where the drive may be either active or inactive (for example, at high impedance).

3.2 Acronyms

ATE	automatic test equipment
BILBO	built-in logic block observer
LSB	least significant bit
LSSD	level-sensitive scan design
MSB	most significant bit
TAP	test access port (see Clause 4)
TCK	test clock input (see 4.2)
TDI	test data input (see 4.4)
TDO	test data outputs (see 4.5)
TMS	test mode select (see 4.3)
TRST*	test reset (see 4.6)
TTL	transistor transistor logic

4. Test access port (TAP)

The TAP is a general-purpose port that can provide access to many test support functions built into a component, including the test logic defined by this standard. It is composed as a minimum of the three input connections and one output connection required by the test logic defined by this standard. An optional fourth input connection provides for asynchronous initialization of the test logic defined by this standard.

4.1 Connections that form the test access port (TAP)

4.1.1 Specifications

Rules

- The TAP shall include the following connections (defined in 4.2, 4.3, 4.4, and 4.5): TCK, TMS, TDI, and TDO.
- Where the TAP controller is not reset at power-up as a result of features built into the test logic, a TRST* input shall be provided as defined in 4.6 (see also 6.3).
- All TAP inputs and outputs shall be dedicated connections to the component (i.e., the pins used shall not be used for any other purpose).

4.1.2 Description

Dedicated TAP connections are required to allow access to the full range of mandatory features of this standard.

4.2 Test clock input (TCK)

The test clock input (TCK) provides the clock for the test logic defined by this standard.

4.2.1 Specifications

Rules

- a) Stored-state devices contained in the test logic shall retain their state indefinitely when the signal applied to TCK is stopped at 0.

Recommendations

- b) Since TCK inputs for many components may be controlled from a single driver, care should be taken to ensure that the load presented by TCK is as small as possible.

Permissions

- c) Stored-state devices contained in the test logic may retain their state indefinitely when the signal applied to TCK is stopped at 1.

4.2.2 Description

The dedicated TCK input is included so that the serial test data path between components can be used independently of component-specific system clocks, which may vary significantly in frequency from one component to the next. It also permits shifting of test data concurrently with normal system operation of the component. The latter facility is required to support the use of the TAP and test data registers in a design for on-line system monitoring. The provision of an independent clock ensures that test data can be moved to or from a chip without changing the state of the on-chip system logic. The independent clock is also essential if boundary-scan registers are to be usable for board interconnect testing in all circumstances—including cases where system clock signals are derived in one component for use in others.

While TCK will in many cases be driven by a free-running clock with a nominal 50% duty cycle, there may be situations where the clock needs to stop for a period. One example is when an ATE needs to fetch test data from backup memory (e.g., disc), since some test systems are unable to keep the clock running during such an operation. This standard requires that TCK can be stopped at 0 indefinitely without causing any change to the state of the test logic. While the TCK signal is stopped at 0, stored-state devices are required to retain their state so that the test logic may continue its operation when clock operation restarts. Optionally, a component also may allow TCK to be stopped at 1 for an indefinite period.

Many parts of the test logic perform operations in response to the rising or falling edge of TCK, as indicated by the use of the phrase “on the rising (falling) edge of TCK.” These operations have to be completed within a fixed (frequency-independent) delay after the occurrence of the relevant change at TCK, and this delay has to be specified by the component supplier. Therefore, the phrase “on the rising (falling) edge of TCK” should be interpreted as “within a specified delay after the rising (falling) edge of TCK.”

NOTE—In many applications, the TCK signal applied to components that conform to this standard will have a duty cycle close to 50% (i.e., the periods that the clock spends at 0 and 1 will be equal) at a given frequency. It is expected that all propagation delays will be such that correct operation is achieved under these circumstances (50% duty cycle at a given TCK frequency), particularly when data is being transferred between TDO of one chip and TDI of another.

4.3 Test mode select input (TMS)

The signal received at TMS is decoded by the TAP controller to control test operations.

4.3.1 Specifications

Rules

- a) The signal presented at TMS shall be sampled by the test logic on the rising edge of TCK.
- b) The design of the circuitry fed from TMS shall be such that an undriven input produces a logical response identical to the application of a logic 1.

Recommendations

- c) Since the TMS inputs for many components may be controlled from a single driver, care should be taken to ensure that the load presented by TMS is as small as possible.

4.3.2 Description

Rule 4.3.1 b) is included so that the TAP controller is forced into the *Test-Logic-Reset* controller state in the case of an undriven TMS pin. This ensures that normal operation of the complete design can continue without interference from the test logic (see 6.3). For TTL-compatible designs, the rule may be met by including a pull-up resistor in the component's TMS input circuitry.

Signal values presented at TMS are sampled by the test logic on the rising edge of TCK. It is expected that the bus master (ATE, bus controller, etc.) will change the signal driven to the TMS inputs of connected components on the falling edge of TCK. The waveforms shown elsewhere in this standard reflect this expectation.

4.4 Test data input (TDI)

Serial test instructions and data are received by the test logic at TDI.

4.4.1 Specifications

Rules

- a) The signal presented at TDI shall be sampled into the test logic on the rising edge of TCK.
- b) The design of the circuitry fed from TDI shall be such that an undriven input produces a logical response identical to the application of a logic 1.
- c) When data is being shifted from TDI toward TDO, test data received at TDI shall appear without inversion at TDO after a number of rising and falling edges of TCK determined by the length of the instruction or test data register selected.

4.4.2 Description

The data pins (TDI and TDO) provide for serial movement of test data through the circuit. The requirement for data to be propagated from TDI to TDO without inversion is included to simplify the operation of components compatible with this standard linked on a printed circuit board.

Values presented at TDI are clocked into the selected register (instruction or test data) on a rising edge of TCK. It is expected that the bus master (ATE, bus controller, etc.) will change the signal driven to the TDI input of the first component on a serial board-level path on the falling edge of TCK. The waveforms shown elsewhere in this standard reflect this expectation.

Rule 4.4.1 b) is included so that open-circuit faults in the board-level serial test data path cause a defined logic value to be shifted into the test logic. Note that when this constant value is shifted into the instruction register, the bypass register will be selected (as will be discussed further in 8.4). For TTL-compatible designs, this rule may be met by inclusion of a pull-up resistor in the component's TDI input circuitry.

4.5 Test data output (TDO)

TDO is the serial output for test instructions and data from the test logic defined in this standard.

4.5.1 Specifications

Rules

- a) Changes in the state of the signal driven through TDO shall occur only on the falling edge of TCK.
- b) The TDO driver shall be set to its inactive drive state except when the scanning of data is in progress (see 6.2).

4.5.2 Description

To ensure race-free operation, changes on TAP inputs (TMS and TDI) are clocked into the test logic defined by this standard on the rising edge of TCK while changes at the TAP output (TDO) occur on the falling edge of TCK. Similarly, for test logic able to drive or receive signals from system pins (e.g., the boundary-scan register), signals driven out of the component from the test logic change state on the falling edge of TCK, while those entering the test logic are clocked in on the rising edge (as will be discussed in 9.3).

The contents of the selected register (instruction or data) are shifted out of TDO on the falling edge of TCK. In the illustrations given in this document, edge-operated circuit designs generally are used. For an edge-operated implementation, note that the TDO output changes shall be delayed until the falling edge of TCK, which can be achieved by including a flip-flop clocked by the falling edge of TCK in the TDO output buffer. Where the registers are constructed from master and slave latches controlled by nonoverlapping clocks, the retiming required by Rule 4.5.1 a) is an inherent feature of the design.

The ability of TDO to switch between active and inactive drive is required to allow parallel, rather than serial, connection of board-level test data paths in cases where this is required. In TTL or CMOS technologies, for example, this requirement may be met through use of a 3-state output buffer.

4.6 Test reset input (TRST*)

The optional TRST* input provides for asynchronous initialization of the TAP controller (see 6.3).

4.6.1 Specifications

Rules

- a) If TRST* is included in the TAP, the TAP controller shall be asynchronously reset to the *Test-Logic-Reset* controller state when a logic 0 is applied to TRST* (see 6.3).

NOTE—As a result of this event, all other test logic in the component is asynchronously reset to the state required in the *Test-Logic-Reset* controller state.

- b) If TRST* is included in the TAP, the design of the circuitry fed from that input shall be such that an undriven input produces a logical response identical to the application of a logic 1.
- c) TRST* shall not be used to initialize any system logic within the component.

Recommendations

- d) To ensure deterministic operation of the test logic, TMS should be held at 1 while the signal applied at TRST* changes from 0 to 1.

4.6.2 Description

Initialization of the TAP controller in turn causes asynchronous initialization of other test logic included in the design, as discussed in the subsequent clauses of this standard.

Rule 4.6.1 b) is included to ensure that in the case of an unterminated TRST* input, test logic operation can proceed under control of signals applied at the TMS and TCK inputs. For TTL-compatible designs, this rule may be met by inclusion of a pull-up resistor in the TRST* input circuitry of the component.

Rule 4.6.1 c) ensures that the test logic can be reset independently of the on-chip system logic. This allows the test logic to be disabled by hard-wiring TRST* to logic 0.

Recommendation 4.6.1 d) is included to ensure that the test logic responds predictably when the signal applied to TRST* changes from 0 to 1. If rising edges occur simultaneously at TRST* and TCK when a logic 0 is applied to TMS, a race will occur, and the TAP controller may either remain in the *Test-Logic-Reset* controller state or enter the *Run-Test/Idle* controller state.

4.7 Interconnection of components compatible with this standard

4.7.1 Specifications

Permissions

- a) The TAP input and output connections may be interconnected at the board level in a manner appropriate to the assembled product.

4.7.2 Description

Figure 4-1, Figure 4-2, and Figure 4-3 illustrate three alternative board-level interconnections of components conforming to this standard.

In each example, the test bus may be controlled either by an ATE system or by a component that provides an interface to a test bus at the next level of product assembly (for example, at the board/backplane interface). In this standard, the device that controls the board-level test bus is referred to as the bus master.

Note that the minimum configuration (shown in Figure 4-1) contains

- Two broadcast signals (TMS and TCK) fed from the testability bus master to all slaves in parallel; and
- A serial path formed by a daisy-chain connection of the serial test data pins (TDI and TDO).

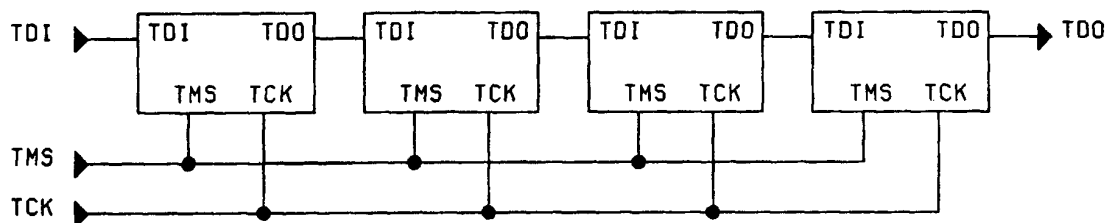


Figure 4-1—Serial connection using one TMS signal

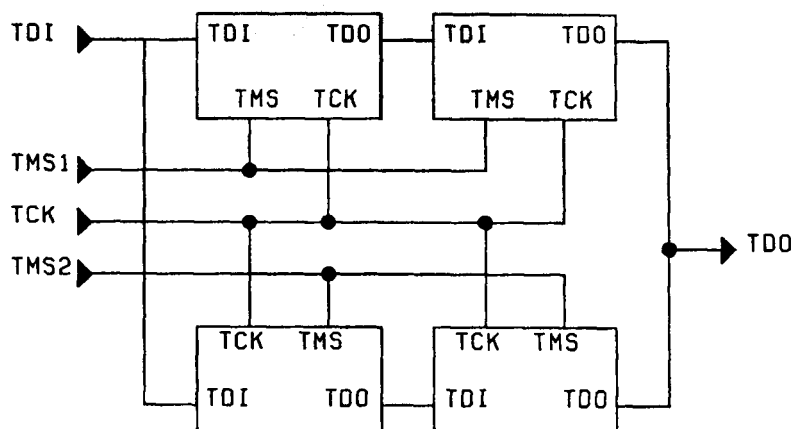


Figure 4-2—Connection in two paralleled serial chains

The hybrid serial/parallel connection shown in Figure 4-2 uses a pair of coordinated TMS signals (TMS1 and TMS2) to ensure that only one serial path is scanning data at a given time. This configuration makes use of the 3-state feature of the TDO output pin, which ensures that only the components that are scanning data have TDO in the active drive state.

Figure 4-3 shows the four components connected to give four separate serial paths through the complete board design. These paths have separate TDI and TDO signals but can be controlled from common TCK and TMS signals.

When choosing a configuration for the board-level interconnection of components conforming to this standard, it is necessary to consider the capability of test equipment and test pattern generators. It is fully expected that any test equipment and/or test pattern generators that intend to support a test methodology based on the boundary-scan architecture defined by this standard would be able to test the board-level configuration of Figure 4-1, since the degenerate form of this configuration is a single conformant component. On the other hand, some test equipment and/or test pattern generators may not be able to test the board-level configurations of Figure 4-2 and Figure 4-3.

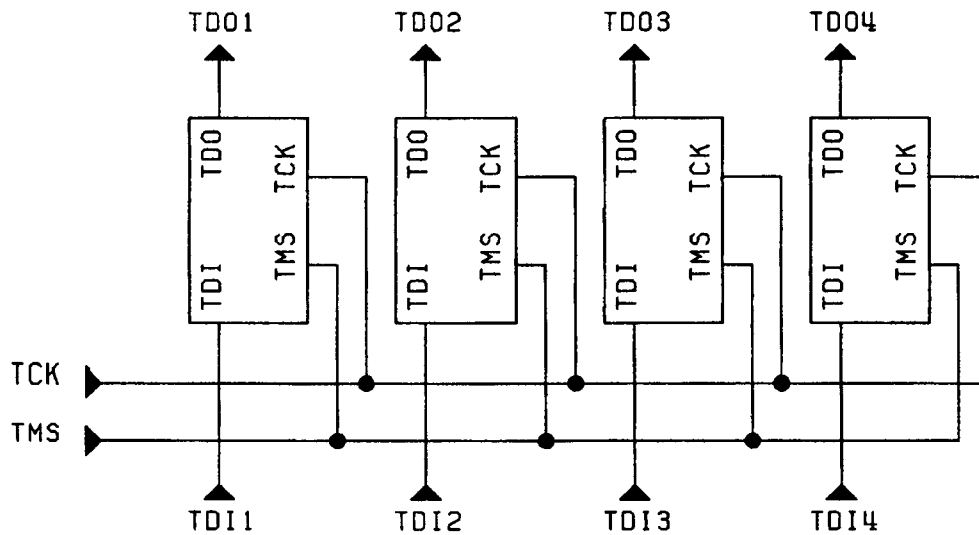


Figure 4-3—Multiple independent paths with common TMS and TCK signals

4.8 Subordination of this standard within a higher-level test strategy

While the test logic specified by this standard has been designed to be extensible to meet the particular needs of individual designers or companies (for example, by the flexibility of the instruction register), occasions may arise when it will be desirable to terminate compliance with this standard by a component temporarily and enable complementary test functionality. An example (illustrated in Annex A) involves a Level-Sensitive Scan Design (LSSD) infrastructure required for use during “stand-alone” component testing, which cannot be simultaneously operated with the test functionality defined by this standard (which is required to support testing of boards onto which the components implementing the two testing techniques will be assembled).

This subclause defines how compliance with this standard may be “switched on” or “switched off.” The rules require the change of test functionality to be under the control of signals applied at one or more component pins. Compliance has to be effected by a single logic pattern applied at these pins, not by a sequence of such patterns.

4.8.1 Specifications

Rules

- a) If a component is to be designed having both
 - 1) Test functionality compliant with this standard; and
 - 2) Other test functionality that is not to be controlled via the test circuitry and the means of control defined in this standard,

compliance with this standard shall be enabled/disabled by one or more steady-state logic patterns (called “compliance-enable” patterns) applied at a fixed set of component inputs, to be called “compliance-enable inputs.”

NOTE—The steady-state combinational logic pattern may be chosen from a set of such “compliance-enable” patterns, all of which have equivalent effect [see Permission 4.8.1 h)].

- b) Any one of the compliance-enable patterns, when applied to the compliance-enable inputs without regard to preceding patterns on these inputs, shall cause the component to be fully compliant with this standard.
- c) Once compliance with this standard is established by the application of a compliance-enable pattern at the compliance-enable inputs, compliance to this standard shall be maintained continuously until the logic pattern applied at the compliance-enable inputs ceases to be a compliance-enable pattern.

NOTES

1—This rule implies that transition between compliance-enable patterns must produce no untoward effects on compliance. Limiting the number of compliance-enable patterns is one way to prevent problems from arising.

2—The rules in other subclauses of this standard apply only when compliance is enabled. Therefore, where compliance-enable inputs are provided, each rule should be considered to be prefaced by “When compliance to this standard is enabled, ...” For example, Rule 4.1.1 c) should be read as stipulating that the TAP pins are dedicated connections and may not be used for any other purpose while compliance to this standard is enabled. When compliance is disabled, the TAP connections may be reused, for example, to provide controls for an alternative test mode of component operation.

3—The event of enabling compliance with this standard by changing the logic pattern applied at the compliance-enable inputs of a component need not have an effect on the component equivalent to that of power-up of the component.

- d) Compliance-enable inputs shall be dedicated inputs to the component and shall not be used for any other purpose.

Recommendations

- e) The number of compliance-enable inputs provided on a component should be minimized.

Permissions

- f) A component may have zero, one, or more compliance-enable inputs.
- g) If a component with compliance-enable input(s) has a TRST* line included in its TAP implementation, the design of the component may require that the TRST* input be driven low at the time of application of a compliance-enable pattern in order to achieve reset of the relevant test logic concurrent with the operation of that test logic.
- h) A component may have several compliance-enable patterns, all of which have an equivalent effect.

4.8.2 Description

If compliance-enable inputs are provided, there shall exist at least one logic pattern that, when applied at the compliance-enable inputs, will result in the component becoming fully compliant with this standard.

5. Test logic architecture

This clause defines the top-level design of the test logic accessed through the TAP. Detailed design requirements for the various blocks contained within the test logic design are contained in the subsequent clauses of this standard.

5.1 Test logic design

5.1.1 Specifications

Rules

- a) The following elements shall be contained in the test logic architecture:
 - 1) A TAP (see Clause 4);
 - 2) A TAP controller (see Clause 6);
 - 3) An instruction register (see Clause 7); and
 - 4) A group of test data registers (see Clause 9).
- b) The instruction and test data registers shall be separate shift-register based paths that are connected in parallel and have a common serial data input and a common serial data output connected to the TAP TDI and TDO signals, respectively.
- c) The selection between the alternative instruction and test data register paths between TDI and TDO shall be made under the control of the TAP controller, as defined in 6.2.

5.1.2 Description

A conceptual view of the top-level design of the test logic architecture defined by this standard is shown in Figure 5-1. This figure and the others included in the descriptive material contained in this standard are examples intended only to illustrate a possible embodiment of this standard. *These figures do not indicate a preferred implementation.*

Key features of the design are as follows:

- a) The TAP controller. This receives TCK and interprets the signals on TMS. The TAP controller generates clock or control signals or both as required for the instruction and test data registers and for other parts of the architecture. The specification for the TAP controller is contained in Clause 6.
- b) The instruction register. This allows the instruction to be shifted into the design. The instruction is used to select the test to be performed or the test data register to be accessed or both. The specification for the instruction register is contained in Clause 7.
- c) The group of test data registers. The group of test data registers shall include a bypass and a boundary-scan register. It also may include an optional device identification register and further optional test data registers. Further information on the structure of the group of test data registers is contained in Clause 9.

Note that, depending on the style of implementation of the test logic defined by this standard, circuitry may be required, in the output stage shown in Figure 5-1, to retime the signal passing through it to occur on the falling edge of TCK.

5.2 Test logic realization

5.2.1 Specifications

Rules

- a) The TAP controller, the instruction register, and the associated circuitry necessary for control of the instruction and test data registers shall be dedicated test logic (i.e., these test logic blocks shall not perform any system function).
- b) If test access is required to a test data register without causing any interference to the operation of the on-chip system logic, the circuitry used to construct that test data register shall be dedicated test logic.

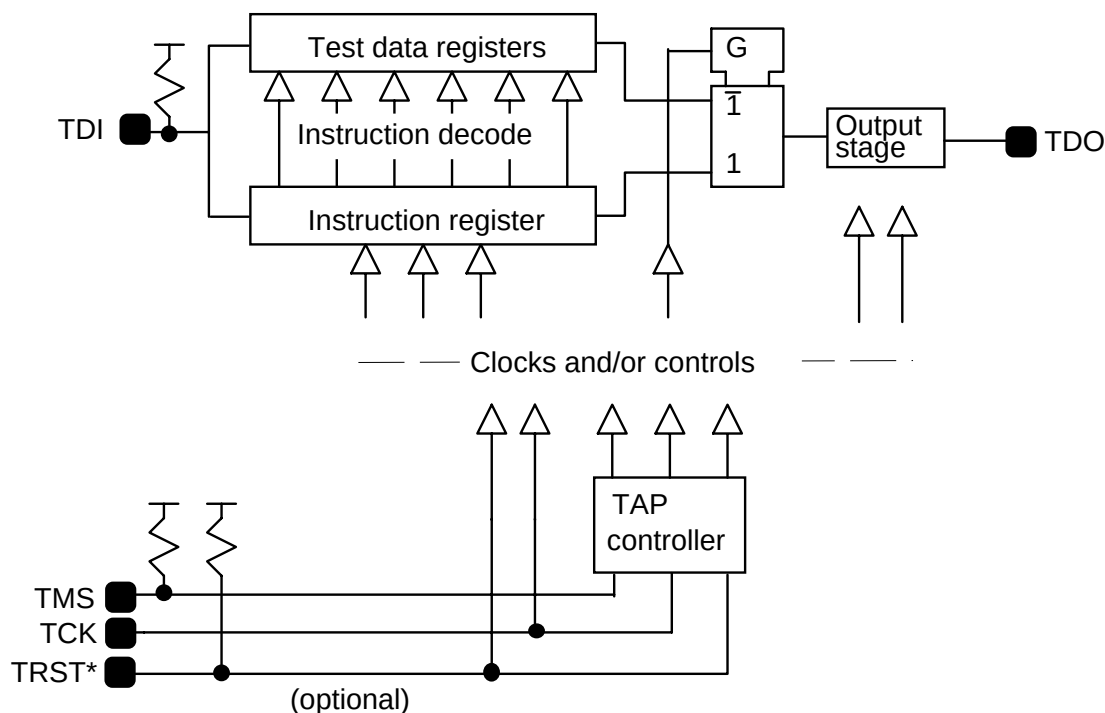


Figure 5-1—A conceptual schematic of the test logic

5.2.2 Description

While the example implementations contained in this standard show the various test data registers to be separate physical entities, circuitry may be shared between the test data registers provided that the rules contained in this standard are met. For example, this would allow the device identification register and the boundary-scan register to share shift-register stages, in which case the requirements of this standard would be met by operating the common circuitry in two different modes—the device identification register mode and the boundary-scan register mode.

6. The TAP controller

The TAP controller is a synchronous finite state machine that responds to changes at the TMS and TCK signals of the TAP and controls the sequence of operations of the circuitry defined by this standard.

6.1 TAP controller state diagram

6.1.1 Specifications

Rules

- The state diagram for the TAP controller shall be as shown in Figure 6-1.

NOTE—The value shown adjacent to each state transition in this figure represents the signal present at TMS at the time of a rising edge at TCK.

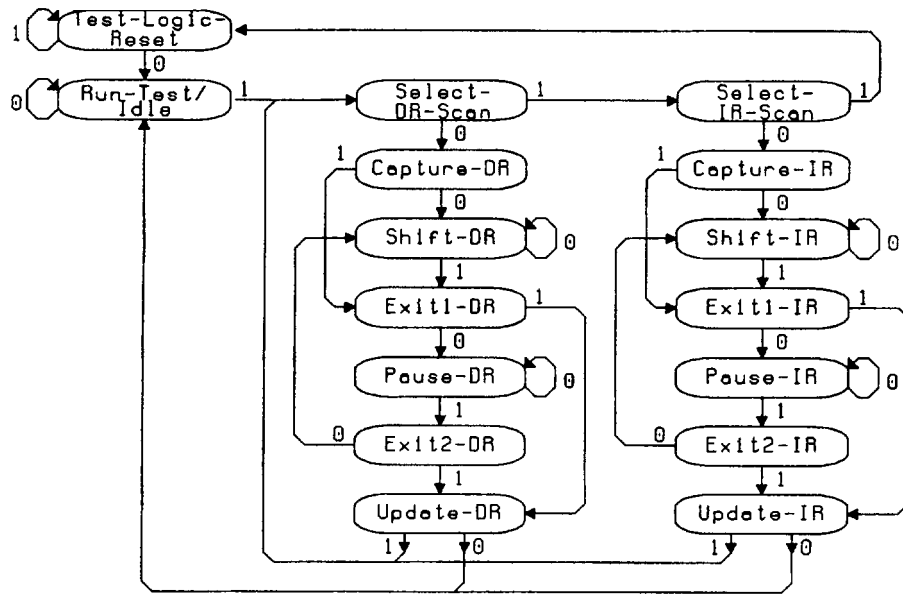


Figure 6-1—TAP controller state diagram

- b) All state transitions of the TAP controller shall occur based on the value of TMS at the time of a rising edge of TCK.
- c) Actions of the test logic (instruction register, test data registers, etc.) shall occur on either the rising or the falling edge of TCK in each controller state, as shown in Figure 6-2.

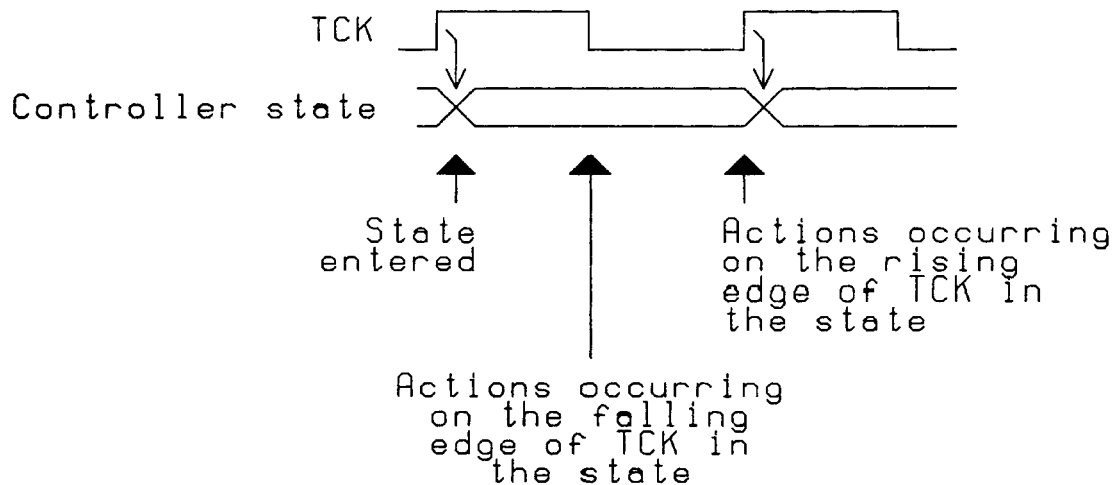


Figure 6-2—Timing of actions in a controller state

6.1.2 Description

The behavior of the TAP controller and other test logic in each of the controller states is briefly described as follows. Rules governing the behavior of the test logic defined by this standard in each controller state are contained in later clauses of this standard.

Test-Logic-Reset

The test logic is disabled so that normal operation of the on-chip system logic (i.e., in response to stimuli received through the system pins only) can continue unhindered. This is achieved by initializing the instruction register to contain the *IDCODE* instruction or, if the optional device identification register is not provided, the *BYPASS* instruction (see 7.2). No matter what the original state of the controller, it will enter *Test-Logic-Reset* when TMS is held high for at least five rising edges of TCK. The controller remains in this state while TMS is high.

If the controller should leave the *Test-Logic-Reset* controller state as a result of an erroneous low signal on the TMS line at the time of a rising edge on TCK (for example, a glitch due to external interference), it will return to the *Test-Logic-Reset* state after three rising edges of TCK with the TMS line at the intended high logic level. The operation of the test logic is such that no disturbance is caused to on-chip system logic operation as the result of such an error. On leaving the *Test-Logic-Reset* controller state, the controller moves into the *Run-Test/Idle* controller state, where no action will occur because the current instruction has been set to select operation of the device identification or bypass register (see 7.2). The test logic is also inactive in the *Select-DR-Scan* and *Select-IR-Scan* controller states.

Note that the TAP controller will also be forced to the *Test-Logic-Reset* controller state by applying a low logic level at TRST*, if such is provided, or at power-up (see 6.3).

Run-Test/Idle

A controller state between scan operations. Once entered, the controller will remain in the *Run-Test/Idle* state as long as TMS is held low. When TMS is high and a rising edge is applied at TCK, the controller moves to the *Select-DR-Scan* state.

In the *Run-Test/Idle* controller state, activity in selected test logic occurs only when certain instructions are present. For example, the *RUNBIST* instruction causes a self-test of the on-chip system logic to execute in this state (see 8.10). Self-tests selected by instructions other than *RUNBIST* also may be designed to execute while the controller is in this state.

For instructions that do not cause functions to execute in the *Run-Test/Idle* controller state, all test data registers selected by the current instruction shall retain their previous state (i.e., Idle).

The instruction does not change while the TAP controller is in this state.

Select-DR-Scan

This is a temporary controller state in which all test data registers selected by the current instruction retain their previous state.

If TMS is held low and a rising edge is applied to TCK when the controller is in this state, the controller moves into the *Capture-DR* state and a scan sequence for the selected test data register is initiated. If TMS is held high and a rising edge is applied to TCK, the controller moves on to the *Select-IR-Scan* state.

The instruction does not change while the TAP controller is in this state.

Select-IR-Scan

This is a temporary controller state in which all test data registers selected by the current instruction retain their previous state.

If TMS is held low and a rising edge is applied to TCK when the controller is in this state, then the controller moves into the *Capture-IR* state and a scan sequence for the instruction register is initiated. If TMS is held high and a rising edge is applied to TCK, the controller returns to the *Test-Logic-Reset* state.

The instruction does not change while the TAP controller is in this state.

Capture-DR

In this controller state data may be parallel-loaded into test data registers selected by the current instruction on the rising edge of TCK. If a test data register selected by the current instruction does not have a parallel input, or if capturing is not required for the selected test, the register retains its previous state unchanged.

The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters either the *Exit1-DR* state if TMS is held at 1 or the *Shift-DR* state if TMS is held at 0.

Shift-DR

In this controller state, the test data register connected between TDI and TDO as a result of the current instruction shifts data one stage toward its serial output on each rising edge of TCK. Test data registers that are selected by the current instruction but are not placed in the serial path retain their previous state unchanged.

The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller either enters the *Exit1-DR* state if TMS is held at 1 or remains in the *Shift-DR* state if TMS is held at 0.

Exit1-DR

This is a temporary controller state. If TMS is held high, a rising edge applied to TCK while in this state causes the controller to enter the *Update-DR* state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the *Pause-DR* state.

All test data registers selected by the current instruction retain their previous state unchanged.

The instruction does not change while the TAP controller is in this state.

Pause-DR

This controller state allows shifting of the test data register in the serial path between TDI and TDO to be temporarily halted. All test data registers selected by the current instruction retain their previous state unchanged.

The controller remains in this state while TMS is low. When TMS goes high and a rising edge is applied to TCK, the controller moves on to the *Exit2-DR* state.

The instruction does not change while the TAP controller is in this state.

Exit2-DR

This is a temporary controller state. If TMS is held high and a rising edge is applied to TCK while in this state, the scanning process terminates and the TAP controller enters the *Update-DR* controller state. If TMS is held low and a rising edge is applied to TCK, the controller enters the *Shift-DR* state.

All test data registers selected by the current instruction retain their previous state unchanged.

The instruction does not change while the TAP controller is in this state.

Update-DR

Some test data registers may be provided with a latched parallel output to prevent changes at the parallel output while data is shifted in the associated shift-register path in response to certain instructions (e.g., *EXTEST*, *INTEST*, and *RUNBIST*). Data is latched onto the parallel output of these test data registers from the shift-register path on the falling edge of TCK in the *Update-DR* controller state. The data held at the latched parallel output should not change other than in this controller state unless operation during the execution of a self-test is required (e.g., during the *Run-Test/Idle* controller state in response to a design-specific public instruction).

All shift-register stages in test data registers selected by the current instruction retain their previous state unchanged.

The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters either the *Select-DR-Scan* state if TMS is held at 1 or the *Run-Test/Idle* state if TMS is held at 0.

Capture-IR

In this controller state the shift-register contained in the instruction register loads a pattern of fixed logic values on the rising edge of TCK. In addition, design-specific data may be loaded into shift-register stages that are not required to be set to fixed values (see Clause 7).

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters either the *Exit1-IR* state if TMS is held at 1 or the *Shift-IR* state if TMS is held at 0.

Shift-IR

In this controller state the shift-register contained in the instruction register is connected between TDI and TDO and shifts data one stage toward its serial output on each rising edge of TCK.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller either enters the *Exit1-IR* state if TMS is held at 1 or remains in the *Shift-IR* state if TMS is held at 0.

Exit1-IR

This is a temporary controller state. If TMS is held high, a rising edge applied to TCK while in this state causes the controller to enter the *Update-IR* state, which terminates the scanning process. If TMS is held low and a rising edge is applied to TCK, the controller enters the *Pause-IR* state.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

Pause-IR

This controller state allows shifting of the instruction register to be halted temporarily.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

The controller remains in this state while TMS is low. When TMS goes high and a rising edge is applied to TCK, the controller moves on to the *Exit2-IR* state.

Exit2-IR

This is a temporary controller state. If TMS is held high and a rising edge is applied to TCK while in this state, termination of the scanning process results, and the TAP controller enters the *Update-IR* controller state. If TMS is held low and a rising edge is applied to TCK, the controller enters the *Shift-IR* state.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

Update-IR

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of TCK in this controller state. Once the new instruction has been latched, it becomes the current instruction.

Test data registers selected by the current instruction retain their previous state.

When the TAP controller is in this state and a rising edge is applied to TCK, the controller enters the *Select-DR-Scan* state if TMS is held at 1 or the *Run-Test/Idle* state if TMS is held at 0.

The *Pause-DR* and *Pause-IR* controller states are included so that shifting of data through the test data or instruction register can be halted temporarily. For example, this might be necessary in order to allow an ATE system to reload its pin memory from disc during application of a long test sequence. Boundary-scan test sequences are likely to extend to the order of 10^7 test patterns for complex board designs.

The TAP controller states include the three basic actions required for testing: stimulus application (*Update-DR*), execution (*Run-Test/Idle*), and response capture (*Capture-DR*). However, not all these actions are required for every type of test. Table 6-1 lists the actions required for key types of test supported by this standard.

For scan testing, the stimulus is made available for use at the end of shifting or, if a parallel output latch is included, by updating the parallel output in the *Update-DR* controller state. The results of the test are captured into the test data register during the *Capture-DR* controller state.

Table 6-1—Use of controller states for different test types

Test type	Action required in this controller state		
	<i>Update-DR</i>	<i>Run-Test/Idle</i>	<i>Capture-DR</i>
Boundary-scan external test (e.g., <i>EXTEST</i>)	Yes	No	Yes
Boundary-scan internal scan test (e.g., <i>INTEST</i>)	Maybe	Maybe	Yes
Boundary-scan <i>PRELOAD</i>	Yes	No	Maybe
Boundary-scan <i>SAMPLE</i>	Maybe	No	Yes
Internally controlled built-in tests (e.g., <i>RUNBIST</i>)	No	Yes	No
Internal scan test (i.e., a specific user-defined PUBLIC instruction)	Maybe	No	Yes

For internally controlled self-testing circuit designs, the starting values of the registers are available at the end of shifting; there is no parallel output latch to update. The registers should operate under control of the internal test logic during *Run-Test/Idle*. Since the result is already contained in a test data register, no action is required during the *Capture-DR* controller state.

For an internal scan test, the target register consists of a serial concatenation of storage elements that support the normal system operation of the component. The construction of such a register is beyond the scope of this standard, and parallel output latches may or may not be present in a given implementation.

6.2 TAP controller operation

6.2.1 Specifications

Rules

- a) The TAP controller shall change state only in response to the following events:
 - 1) A rising edge of TCK;
 - 2) A transition to logic 0 at the TRST* input (if provided); or
 - 3) Power-up.
- b) The TAP controller shall generate signals to control the operation of the test data registers, instruction registers, and associated circuitry as defined in this standard (Figure 6-3 and Figure 6-4).

NOTE—In these figures, the assumption is made that the signals applied to TMS and TDI change state on the falling edge of TCK. The time at which these signals change state is not defined by this standard, but it should be such that the set-up and hold requirements of TMS and TDI are met. It is further assumed that the design includes the optional device identification register. Therefore, the figures show the *IDCODE* instruction being set onto the output of the instruction register in the *Test-Logic-Reset* controller state. If the device identification register is not included in the design, the output of the instruction register will be set to the *BYPASS* instruction in the *Test-Logic-Reset* controller state.

- c) The TDO output buffer and the circuitry that selects the register output fed to TDO shall be controlled as shown in Table 6-2.
- d) Changes at TDO defined in Table 6-2 shall occur on the falling edge of TCK after entry into the state.

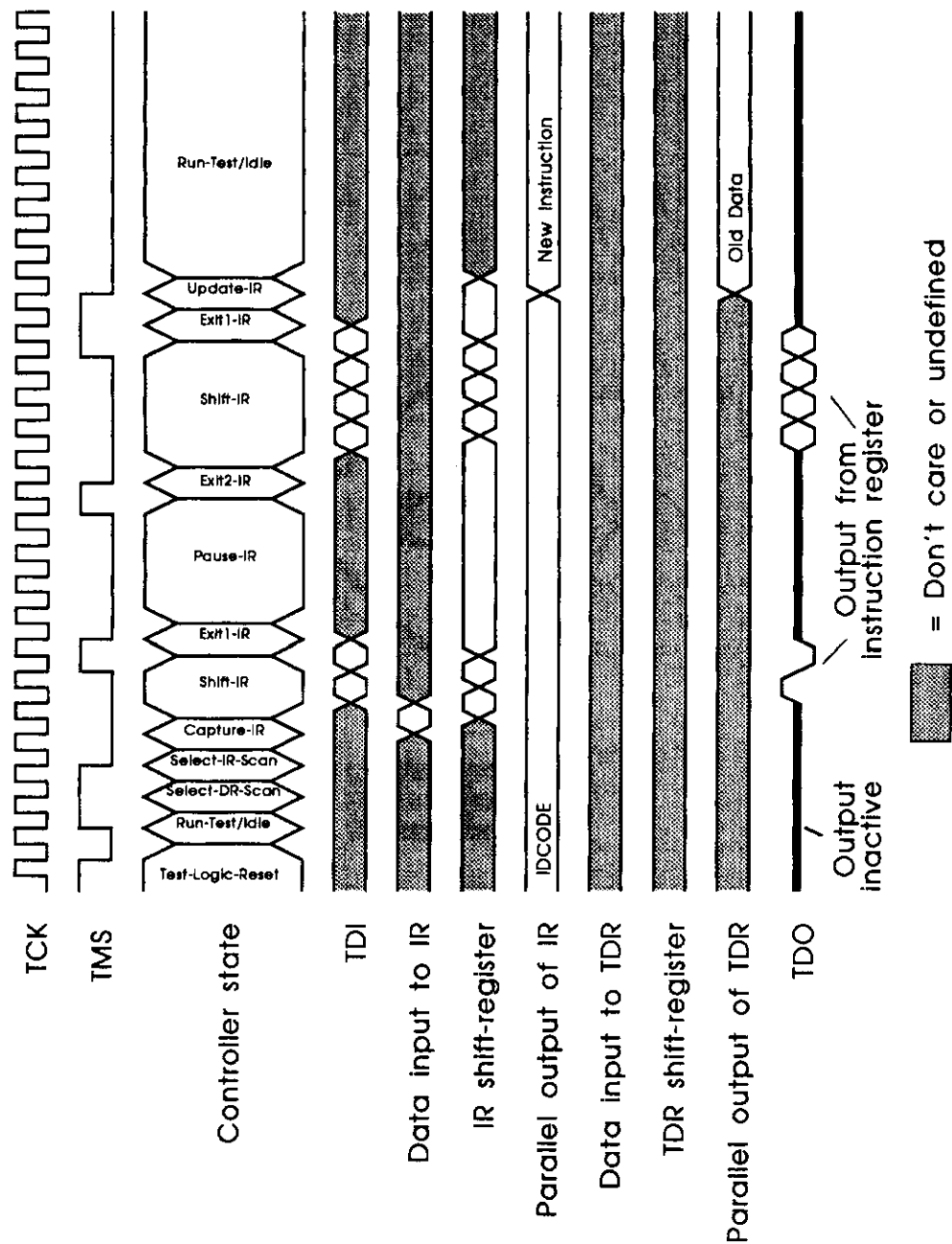


Figure 6-3—Test logic operation: instruction scan

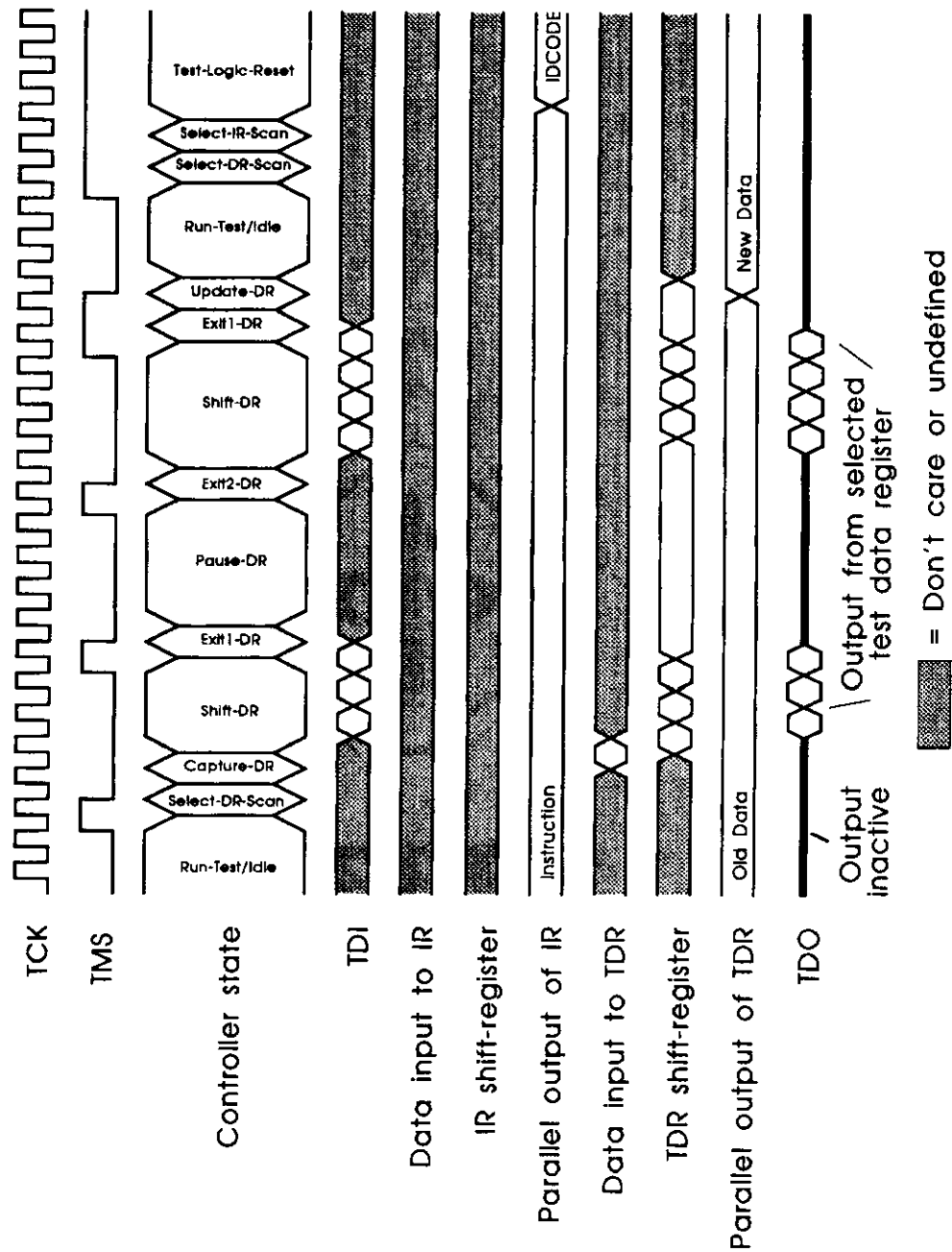


Figure 6-4—Test logic operation: data scan

Table 6-2—Test logic operation in each controller state

Controller state	Register selected to drive TDO	TDO driver
<i>Test-Logic-Reset</i>	Undefined	Inactive
<i>Run-Test/Idle</i>	Undefined	Inactive
<i>Select-DR-Scan</i>	Undefined	Inactive
<i>Select-IR-Scan</i>	Undefined	Inactive
<i>Capture-IR</i>	Undefined	Inactive
<i>Shift-IR</i>	Instruction	Active
<i>Exit1-IR</i>	Undefined	Inactive
<i>Pause-IR</i>	Undefined	Inactive
<i>Exit2-IR</i>	Undefined	Inactive
<i>Update-IR</i>	Undefined	Inactive
<i>Capture-DR</i>	Undefined	Inactive
<i>Shift-DR</i>	Test data	Active
<i>Exit1-DR</i>	Undefined	Inactive
<i>Pause-DR</i>	Undefined	Inactive
<i>Exit2-DR</i>	Undefined	Inactive
<i>Update-DR</i>	Undefined	Inactive

NOTE—Some components designed before publication of this standard may conform in every respect except that they have TDO active in the *Capture-IR*, *Pause-IR*, *Exit1-IR*, *Exit2-IR*, *Capture-DR*, *Pause-DR*, *Exit1-DR*, and *Exit2-DR* controller states, in addition to the *Shift-IR* and *Shift-DR* controller states. The functionality of these components is indistinguishable from that of components that fully conform to this standard except where the TDO output of such a component is connected to the TDO output of another (e.g., as shown in Figure 4-2).

6.2.2 Description

An example of a circuit that meets the requirements of 6.2.1 is shown in Figure 6-5 and Figure 6-6. This circuit generates a range of clock and control signals required not only to control the selection between the alternative instruction and test data register paths and the activity of TDO (as defined in Table 6-2) but also to control the example implementations of other items of test logic that are contained in this standard.

The assignment of controller states in the example implementation is given in Table 6-3.

The Boolean equations for the next state logic in Figure 6-5 and Figure 6-6 are as follows:

$$\begin{aligned}
 ND &:= DC^* + DB + T^*CB^* + D^*CB^*A^* \\
 NC &:= CB^* + CA + TB^* \\
 NB &:= T^*BA^* + T^*C^* + T^*D^*B + T^*D^*A^* + TCB^* + TDCA \\
 NA &:= T^*C^*A + TB^* + TA^* + TDC
 \end{aligned}$$

where

$$T = \text{value present at TMS}$$

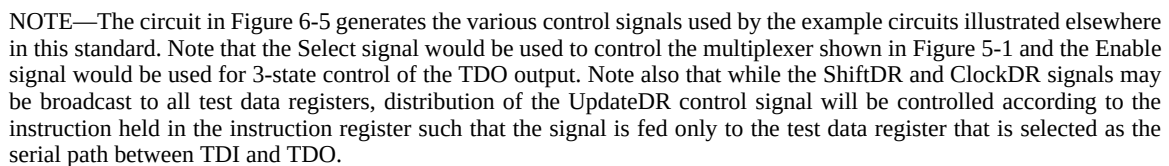


Figure 6-5—A TAP controller implementation—state registers and output logic

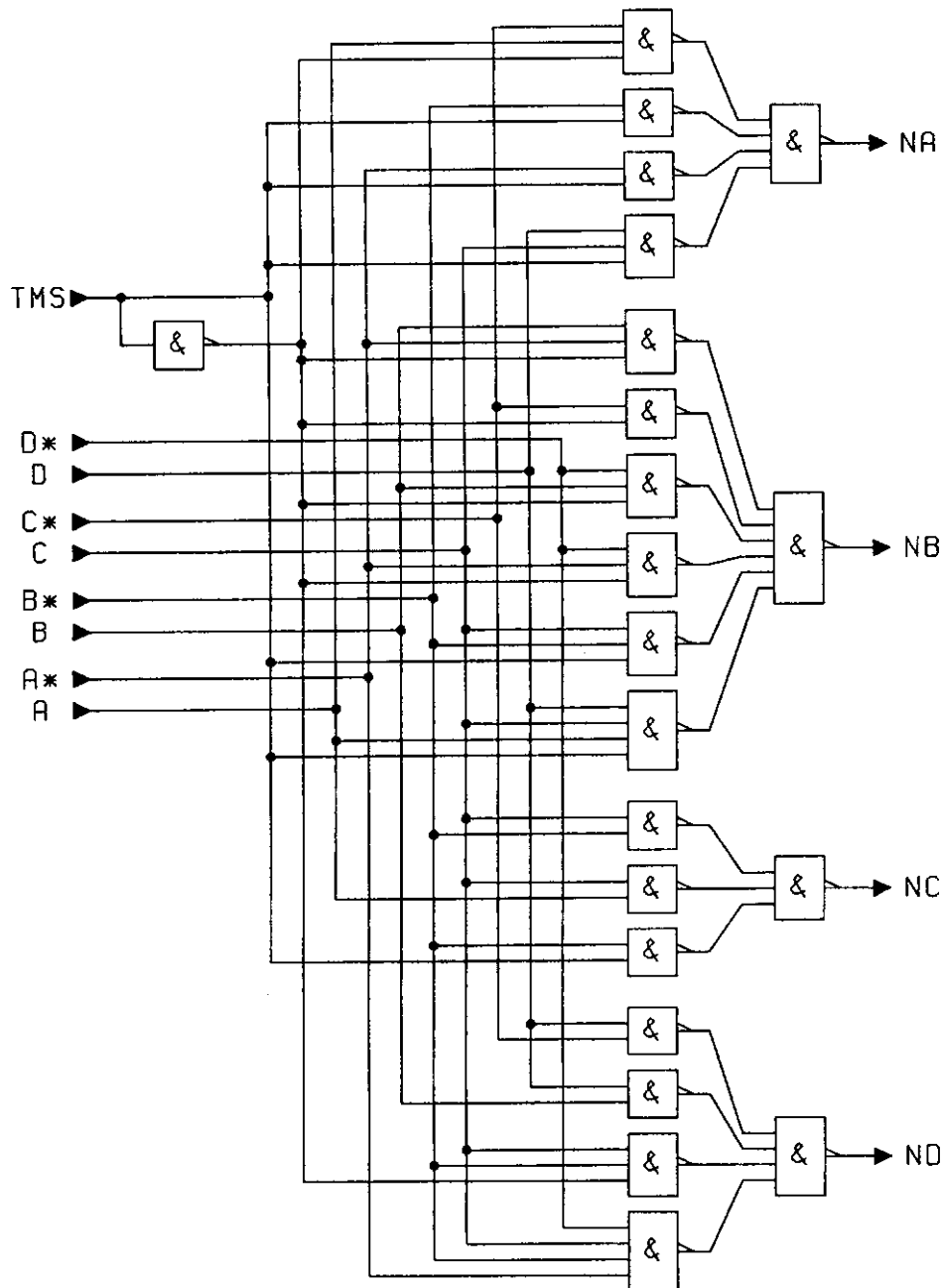


Figure 6-6—A TAP controller implementation—next state logic

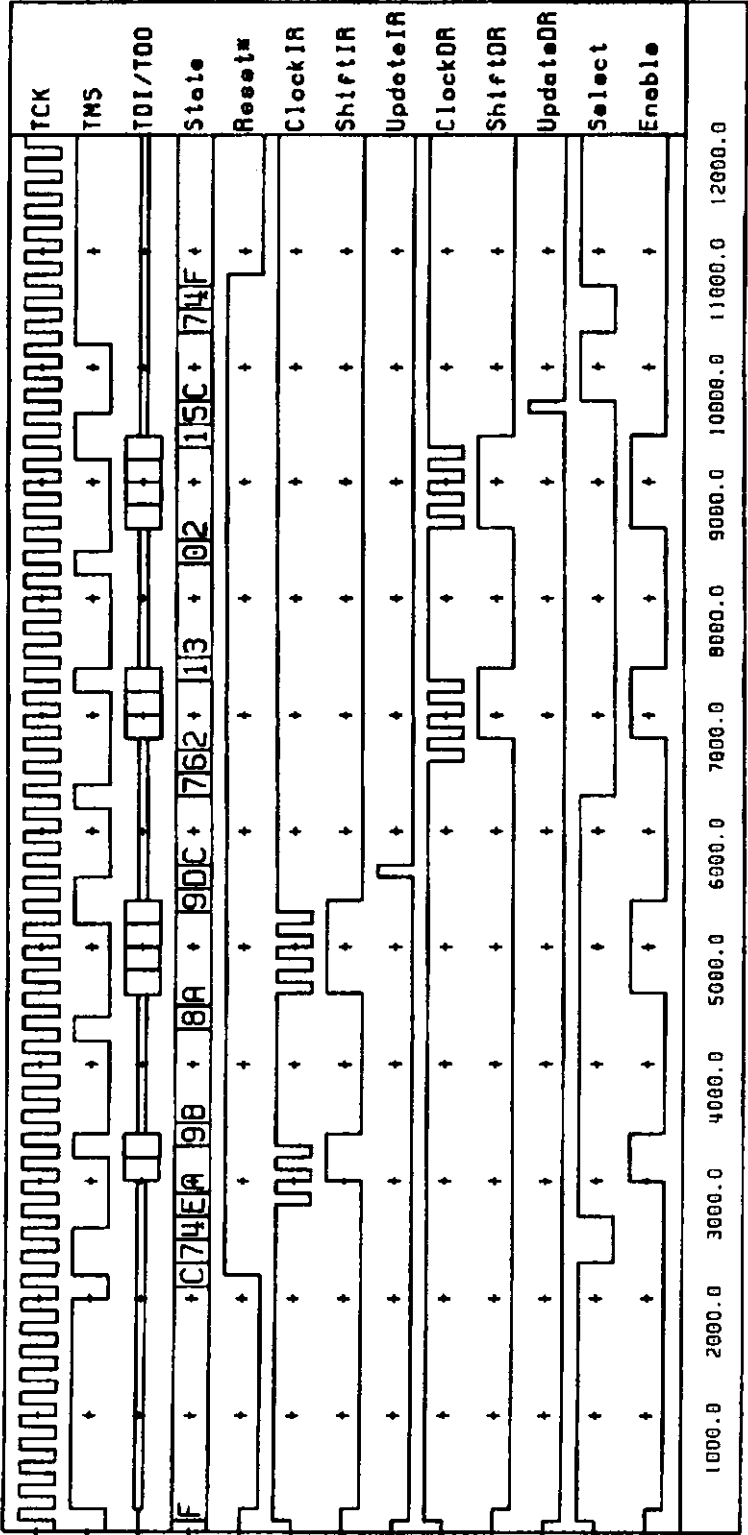


Figure 6-7—Operation of the example TAP controller

Figure 6-7 shows the operation of this controller implementation through instruction and test data register scan cycles.

Table 6-3—State assignments for example TAP controller

Controller State	DCBA (hex)
<i>Exit2-DR</i>	0
<i>Exit1-DR</i>	1
<i>Shift-DR</i>	2
<i>Pause-DR</i>	3
<i>Select-IR-Scan</i>	4
<i>Update-DR</i>	5
<i>Capture-DR</i>	6
<i>Select-DR-Scan</i>	7
<i>Exit2-IR</i>	8
<i>Exit1-IR</i>	9
<i>Shift-IR</i>	A
<i>Pause-IR</i>	B
<i>Run-Test/Idle</i>	C
<i>Update-IR</i>	D
<i>Capture-IR</i>	E
<i>Test-Logic-Reset</i>	F

6.3 TAP controller initialization

6.3.1 Specifications

Rules

- The TAP controller shall be forced into the *Test-Logic-Reset* controller state at power-up either by use of the TRST* signal or as a result of circuitry built into the test logic.

NOTE—If the TAP controller is to be reset at power-up using TRST*, the design of the assembled system has to ensure that a logic 0 is applied to TRST* when power is applied. Similarly, where the TAP controller is to be reset using TRST* after enabling of compliance to this standard as described in 4.8, the design of the assembled system has to ensure that a logic 0 is applied to TRST* when compliance is enabled.

- The TAP controller shall not be initialized by the operation of any system input, such as a system reset.
- Where a dedicated reset pin (TRST*) is provided to allow initialization of the TAP controller, initialization shall occur asynchronously when the TRST* input changes to the low logic level.

- d) Where the TAP controller is initialized at power-up by operation of circuitry built into the test logic, the result shall be equivalent to that which would be achieved by application of a logic 0 to a TRST* input.

6.3.2 Description

In a board design that contains wired junctions or buses, provision shall be made to ensure that at power-up any period of contention between drivers on the bus is kept within limits that ensure that no damage occurs to the components on the board.

When boundary-scan circuitry is inserted between the on-chip system logic and package pins, it becomes essential to ensure that shortly after power-up this circuitry enters a state where buses and wired junctions are controlled by the system circuitry, i.e., the *Test-Logic-Reset* controller state.

NOTE—Clause 11 contains rules that ensure that boundary-scan circuitry at system pins does not interfere with normal system operation when the *Test-Logic-Reset* controller state is selected.

While the TAP controller will synchronously enter the *Test-Logic-Reset* controller state after five rising edges at TCK (provided that TMS is held high), the worst-case time taken to reach this state may exceed that at which damage could occur. Further, it cannot be guaranteed that the clock will be running at the time at which power is applied to the board. Therefore, the “reset at power-up” requirement is included.

The requirement can be met in a variety of ways, for example, by inclusion of a power-up reset within the integrated circuit or by asymmetric design of the latches or registers used to construct the TAP controller. It also could be met by inclusion of a dedicated TRST* pin for the TAP controller. However, a system reset cannot also be used to initialize the TAP controller, since this would compromise the ability to test system interconnections at the board level using the boundary-scan circuitry. In some systems it also may be possible to use the independence of the system and test resets to allow sampling and examination of data after a system crash. This would require that the test logic be reset before reinitialization of the on-chip system logic.

Where a power-up reset facility is provided within the component, this can be used to initialize both the system and the test logic, for example, as shown in Figure 6-8.

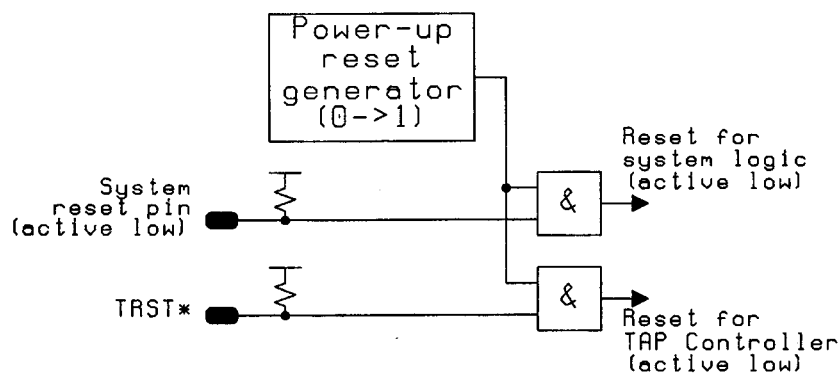


Figure 6-8—Use of power-up reset for system and test logic

7. The instruction register

The instruction register allows an instruction to be shifted into the design. The instruction is used to select the test to be performed or the test data register to be accessed or both. As will be discussed in Clause 8, a number of mandatory and optional instructions are defined by this standard. Further design-specific instructions can be added to allow the functionality of the test logic built into a component to be extended.

Optionally, the instruction register allows examination of design-specific information generated within the component.

This clause contains the design requirements for the instruction register.

7.1 Design and construction of the instruction register

The instruction register is a shift-register-based design that has an optional parallel input for register cells other than the two nearest to the serial output. The instruction shifted into the register is latched at the completion of the shifting process.

7.1.1 Specifications

Rules

- a) The instruction register shall include at least two shift-register-based cells capable of holding instruction data.
- b) The instruction shifted into the instruction register shall be latched such that changes in the effect of an instruction occur only in the *Update-IR* and *Test-Logic-Reset* controller states (see 7.2).
- c) There shall be no inversion of data between the serial input and the serial output of the instruction register.
- d) The two least significant instruction register cells (i.e., those nearest the serial output) shall load a fixed binary “01” pattern (the 1 into the least significant bit location) in the *Capture-IR* controller state (see 7.2).

Recommendations

- e) Where the parallel inputs of instruction register cells are not required to load design-specific information, then these cells should be designed to load fixed logic values (0 or 1) in the *Capture-IR* controller state.

Permissions

- f) Parallel inputs may be provided to instruction register cells (other than the two least significant cells) to permit capture of design-specific information in the *Capture-IR* controller state.

7.1.2 Description

The parallel output from the instruction register is latched to ensure that the test logic is protected from the transient data patterns that will occur in its shift-register stages as new instruction data is entered. The latched parallel output is controlled such that it can change state only in the *Update-IR* and *Test-Logic-Reset* controller states. The timing and nature of these changes are discussed in detail in 7.2.

The minimum size (two instruction register cells) is necessary to meet rules stated elsewhere in this standard:

- a) The instruction register shall allow selection of the bypass register.

- b) The instruction register shall allow access to the boundary-scan register in at least three configurations (*EXTEST*, *PRELOAD*, and *SAMPLE*—see 8.2).

It is permissible [see Permission 8.1.1 g)] for instructions to share binary codes provided that none of the rules that specify the merged instructions is violated. In earlier editions of this standard, *SAMPLE* and *PRELOAD* were specified as a merged instruction—*SAMPLE/PRELOAD*—such that the four instructions mandated by this standard—*BYPASS*, *EXTEST*, *PRELOAD*, and *SAMPLE*—could be implemented using three binary codes, leaving the fourth achievable with a minimum 2-bit instruction register available for implementation of an optional instruction.

In addition, fault isolation of the board-level serial test data path shall be supported. This is achieved by loading a constant binary “01” pattern into the least significant bits of the instruction register at the start of the instruction-scan cycle.

The inclusion of the optional design-specific data inputs to the instruction register allows key data signals within the device to be examined at the start of testing, with future test actions potentially depending on the design-specific information gathered. Where the parallel inputs to instruction register cells are not used for design-specific information, it is recommended that these cells be designed to load a fixed logic value (0 or 1) during the *Capture-IR* controller state.

7.2 Instruction register operation

7.2.1 Specifications

Rules

- a) The behavior of the instruction register in each TAP controller state shall be as defined in Table 7-1.
- b) All actions resulting from an instruction shall terminate when a different instruction is transferred to the parallel output of the instruction register (i.e., in the *Update-IR* or *Test-Logic-Reset* controller states).
- c) All operations of shift-register stages shall occur on the rising edge of TCK after entry into a controller state.

Table 7-1—Instruction register operation in each controller state

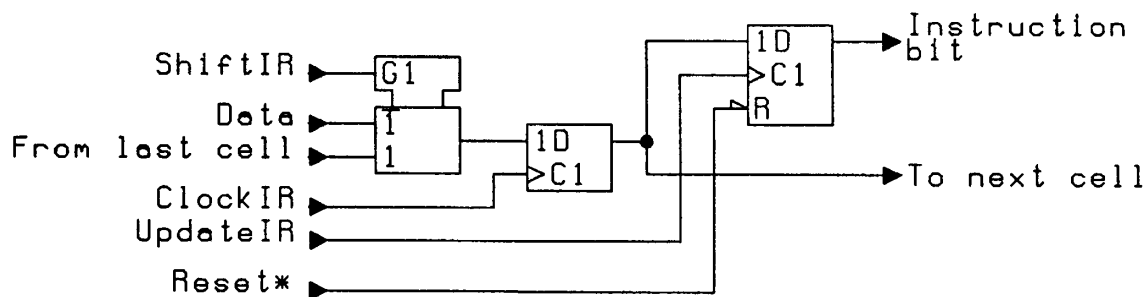
Controller state	Shift-register stage	Parallel output
<i>Test-Logic-Reset</i>	Undefined	Set to give the <i>IDCODE</i> (or <i>BYPASS</i>) instruction
<i>Capture-IR</i>	Load 01 into LSBs and, optionally, design-specific data or fixed values into MSBs	Retain last state
<i>Shift-IR</i>	Shift toward serial output	Retain last state
<i>Exit1-IR</i> <i>Exit2-IR</i> <i>Pause-IR</i>	Retain last state	Retain last state
<i>Update-IR</i>	Retain last state	Load from shift-register
All other states	Undefined	Retain last state

- d) The data present at the parallel output of the instruction register shall be latched from the shift-register stage on the falling edge of TCK in the *Update-IR* controller state.
- e) After entry into the *Test-Logic-Reset* controller state as a result of the clocked operation of the TAP controller, the *IDCODE* instruction (or, if there is no device identification register, the *BYPASS* instruction) shall be latched onto the instruction register output on the falling edge of TCK.
- f) If the TRST* input is provided, when a low signal is applied to the input, the latched instruction shall change asynchronously to *IDCODE* (or, if no device identification register is provided, to *BYPASS*).

7.2.2 Description

Figure 7-1 shows an implementation of an instruction register cell that satisfies these requirements and operates in response to the signals generated by the example TAP controller design contained in 6.2:

- a) The parallel output (labeled Instruction bit) is updated at the end of the instruction-scan cycle during the *Update-IR* controller state. This shall occur on the falling edge of TCK because a change in the latched instruction can result in a change at system output pins due to the operation of the boundary-scan register. Such changes shall occur on the falling edge of TCK as defined in Clause 11. Note that in Figure 7-1 an edge-triggered flip-flop is provided adjacent to the shift-register stage to meet this requirement. Alternative implementations, for example, where a level-operated latch is used or the storage element follows (rather than precedes) the instruction decoding logic, are permissible.



NOTE—The parallel output flip-flop in this figure is provided with a reset input. To meet Rules 7.2.1 e) and 7.2.1 f), some or all instruction register cells will require use of a set, rather than a reset, input.

Figure 7-1—An instruction register cell

- b) The clock input to the register in the serial path is applied only during the *Capture-IR* and *Shift-IR* controller states.
- c) The parallel output is reset in the *Test-Logic-Reset* controller state as a result of a logic 0 received at the Reset* input of the cell. Referring to Figure 6-5 and Figure 6-6, notice that a low Reset* signal will be generated on the falling edge of TCK after entry into the *Test-Logic-Reset* controller state under control of TMS and TCK (TRST* held at 1). The parallel output of the instruction register will change on the falling edge of TCK, as is the case in the *Update-IR* controller state. In contrast, when a logic 0 is applied to TRST*, Reset* consequently is asserted (low) (see Figure 6-5) and the change at the parallel output occurs immediately, irrespective of the state of TMS or TCK. Note that some cells will need to be designed such that the parallel output is set high during this controller state so that the value of the *IDCODE* (or *BYPASS*) instruction is loaded onto the complete register's outputs as required by Rule 7.2.1 e).

- d) Application of a 0 at TRST* causes the parallel output to be asynchronously set low. Again, some cells may need to be designed to be set high by TRST* such that the value of the *IDCODE* (or *BYPASS*) instruction is forced onto the register's outputs.

Note that the parallel data inputs to the two least significant stages (instruction register stages 0 and 1) shall be tied to fixed logic levels (1 for the least significant bit, 0 for the next-least significant bit).

Rule 7.2.1 b) ensures that the operation of the test data registers, etc., is determined only by the current instruction and that there is no possibility that actions resulting from any instruction (e.g., execution of an internal self-test) can continue once the instruction is removed. The circuit under test may not be in a known state if a new instruction is loaded before the previous one has run to completion.

8. Instructions

The instruction register allows instructions to be entered serially into the test logic during an instruction-register scan cycle. This clause defines the minimum range of instructions that shall be supplied and the operations that occur in response to those instructions. Optional instructions and the resulting operation of the test logic are also defined, together with the requirements for extensions to the instruction set defined in this standard.

8.1 Response of the test logic to instructions

8.1.1 Specifications

Rules

- a) Each instruction shall completely define the set of test data register(s) that may operate and (where required) interact with the on-chip system logic while the instruction is current.
- b) Test data registers that are not selected by the current instruction shall be controlled such that they do not interfere with the operation of the on-chip system logic or the selected test data registers.
- c) Each instruction shall cause a single serial test data register path to be enabled to shift data between TDI and TDO in the *Shift-DR* controller state (as defined in Table 6-2).
- d) Instruction binary codes that are not otherwise required to provide control of test logic shall be equivalent to the *BYPASS* instruction (see 8.4).

Recommendations

- e) Use of the binary code {000...0} should be avoided for instructions that disrupt normal (i.e., nontest) operation of the component.

NOTE—Earlier editions of this standard mandated that a binary code for the *EXTEST* instruction be {000...0} (i.e., a logic 0 is loaded into every instruction register cell). While use of this binary code is no longer mandated, nor is it prohibited, it should be noted that such use may be detrimental to the implementation of high-reliability systems, as an apparent stuck-at-zero fault condition at a component's TDI pin could result in unexpected selection of *EXTEST* and consequent removal of the component from normal service.

Permissions

- f) The mode of operation of a test data register may be defined by a combination of the current instruction and further control information contained in test data registers.
- g) Two or more instructions may share a single binary code provided that all the rules for the separate instructions are met.

8.1.2 Description

The instructions loaded into the instruction register are decoded in order to achieve two key functions.

First, each instruction defines the set of test data registers that may operate while the instruction is current. Other test data registers should be controlled such that they cannot interfere with the operation of the on-chip system logic or with the operation of the selected test data registers. Several registers may be set into test modes simultaneously (for an example, see 9.2).

Second, an instruction defines the serial test data register path that is used to shift data between TDI and TDO during data register scanning. Note that a particular instruction may result in a single test data register being connected between TDI and TDO or in several test data registers being serially interconnected between TDI and TDO (for an example, see 9.2).

Rule 8.1.1 d) ensures that every pattern of 1s and 0s that can be fed into the instruction register produces a defined response and, in particular, that a test data register is connected between TDI and TDO for every possible instruction binary code.

Permission 8.1.1 g) allows for the merging of instructions to operate under a single binary code when their respective behaviors are not mutually exclusive. A prime example of such merging would be a sharing of a single binary code between *SAMPLE* and *PRELOAD*. The resulting merged behavior, which could be called *SAMPLE/PRELOAD*, would be fully equivalent to that mandated in earlier editions of this standard.

8.2 Public instructions

8.2.1 Specifications

Rules

- a) Public instructions shall be available for use by purchasers of a component.
- b) The following public instructions shall be provided in all components claiming conformance to this standard: *BYPASS*, *SAMPLE*, *PRELOAD*, and *EXTEST* (see 8.4, 8.6, 8.7, and 8.8, respectively).
- c) If the optional device identification register is included in a component, the *IDCODE* instruction shall be provided.
- d) If the optional device identification register is included in a user-programmable component that does not allow the programming via the test logic defined by this standard, then the *USERCODE* instruction shall be provided.
- e) The binary codes for the *BYPASS* instruction shall be as defined in 8.4.

Recommendations

- f) It is recommended that products support either the *INTEST* or the *RUNBIST* instruction or both (see 8.9 and 8.10).

Permissions

- g) A design may offer public instructions in addition to those defined in this standard to give the device purchaser access to design-specific features.
- h) Where binary codes for public instructions are not defined by this standard, they may be assigned as required for the particular design.

8.2.2 Description

Public instructions provide the component purchaser with access to test features that help in test tasks, e.g., go/no-go testing of the component via its self-test and board interconnect test via the boundary-scan register. The purchaser expects that the results of such tests will be independent of the variant of the component installed in a particular board, of the source of the component, etc. An exception, of course, is when the test results are intended to distinguish the variant, etc., as would be the case if the *IDCODE* instruction were used (see 8.13).

The binary code of an instruction is the sequence of data bits shifted serially into the instruction register from TDI during the *Shift-IR* controller state.

8.3 Private instructions

8.3.1 Specifications

Permissions

- a) The public instructions may be supplemented with private instructions intended solely for the use of the component manufacturer.
- b) The operation of private instructions need not be documented.
- c) If private instructions are utilized in a component, the vendor shall clearly identify any instruction binary codes that, if selected, would cause hazardous operation of the component.

8.3.2 Description

Private instructions allow the component manufacturer to use the TAP and test logic to gain access to test features embedded in the design for design verification, production testing, or fault diagnosis. The component manufacturer may require tests performed using these features to give results that differ between variants of the component, for example, that would render documentation and use by component purchasers difficult.

Note that some private instructions may cause a component to operate in a manner that could be hazardous. For example, if a private instruction causes component inputs to become outputs for test data, etc., then damage may result if the instruction is selected while the component is surrounded by other components on an assembled board. The vendor shall therefore clearly identify any instruction binary codes that may cause hazardous operation if used by the component purchaser.

8.4 The *BYPASS* instruction

The bypass register contains a single shift-register stage and is used to provide a minimum-length serial path between the TDI and the TDO pins of a component when no test operation of that component is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test operations.

8.4.1 Specifications

Rules

- a) Each component shall provide a *BYPASS* instruction.
- b) A binary code for the *BYPASS* instruction shall be {111...1} (i.e., a logic 1 entered into every instruction register cell).

- c) The *BYPASS* instruction shall select the bypass register to be connected for serial access between TDI and TDO in the *Shift-DR* controller state.
- d) When the *BYPASS* instruction is selected, all test data registers that can operate in either system or test modes shall perform their system function.
- e) When the *BYPASS* instruction is selected, the operation of the test logic shall have no effect on the operation of the on-chip system logic.

Permissions

- f) The *BYPASS* instruction may have binary codes in addition to that defined in Rule 8.4.1 b).

8.4.2 Description

The *BYPASS* instruction can be entered by holding TDI at a constant high value and completing an instruction-scan cycle. The demands on the host test system consequently are reduced in cases where access is required, say, to only chip 57 on a 100-chip board. In this case, the overall instruction pattern that shall be shifted into the design consists of a background of 1s with a small field of specific instruction data.

Note also that since the TDI input is designed such that when it is not terminated it behaves as though a high signal were being applied, an open circuit fault in the serial board-level test data path will cause the bypass register to be selected after an instruction-scan cycle. Therefore, no unwanted interference with the operation of the on-chip system logic can occur.

Where no device identification register is provided in a component, the *BYPASS* instruction is forced into the latches at the parallel outputs of the instruction register during the *Test-Logic-Reset* controller state. This ensures that a complete serial path through either bypass or device identification registers is established.

A consideration for usage is that if *BYPASS* is operated in some components while test-mode instructions (e.g., *EXTEST*) are operated in others, the normal system-logic operation of those components that are operating *BYPASS* may conflict with the test operation of the others. Therefore, careful analysis of interactions is necessary.

8.5 Boundary-scan register instructions

As discussed in Clause 1, the boundary-scan register is composed of cells connected between the on-chip system logic and the component's system input and output pins. This subclause is included to provide an overview of the structure and operation of the boundary-scan register that will assist the reader in understanding the specifications for the mandatory and optional instructions that make use of the boundary-scan register.

Design requirements for the boundary-scan register instructions are contained in 8.6 to 8.11. Requirements for the design of boundary-scan register cells are contained in Clause 11.

8.5.1 An overview of the operation of the boundary-scan register

The boundary-scan register is a shift-register-based structure that includes a variety of different cell designs matched onto the requirements of the particular component. Different cell designs are used according to the type of system pin concerned (input, output, 3-state, bidirectional) and according to the set of boundary-scan instructions supported.

A simplified view of a boundary-scan register is shown in Figure 8-1.

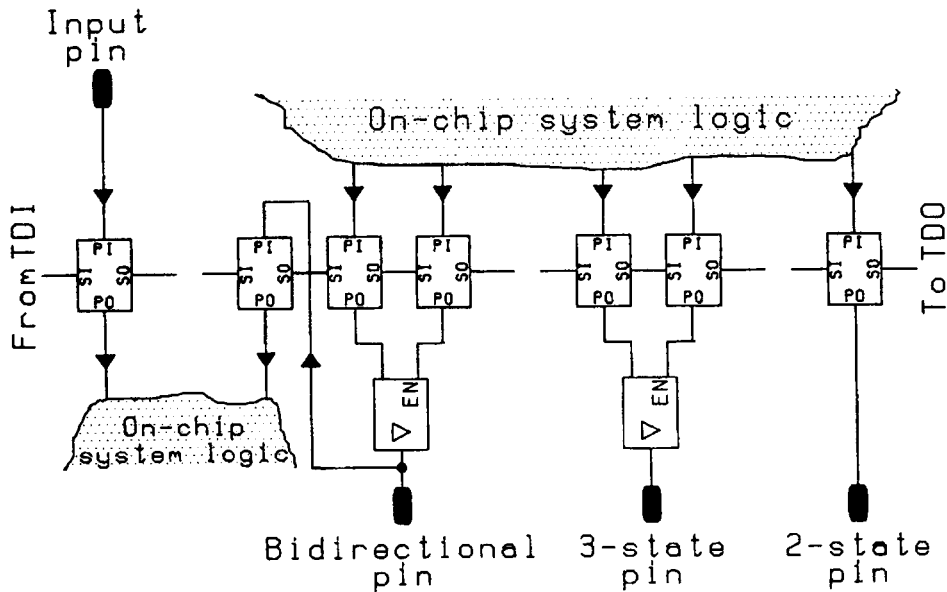


Figure 8-1—A simplified view of the boundary-scan register

An example implementation for a cell that could be used in each of the locations shown in Figure 8-1 is given in Figure 8-2.

The connections labeled PI, PO, SI, and SO in Figure 8-2 are connected to adjacent cells, the on-chip system logic, and the system pins, as shown in Figure 8-1. Like all the cells shown in this standard, that shown in Figure 8-2 is designed to respond to the ClockDR, ShiftDR, and UpdateDR signals generated by the example TAP controller implementation shown in Figure 6-5 and Figure 6-6. The Mode input shall be controlled according to the type of pin connected to the cell (input, output, etc.) and the specific instruction selected.

Use of this cell design, with appropriate signals supplied to the Mode input of each cell, will result in a component that supports the *SAMPLE*, *PRELOAD*, *EXTEST*, and *INTEST* instructions. As will be discussed in Clause 11, other cell designs are possible that meet the requirements of this standard for different sets of instructions. For example:

- R2 may be either a flip-flop (as shown) or a latch.
- R2 is optional for cells that feed data from a system pin to the on-chip system logic, e.g., the cells at system input pins. The lower input to M2 would in such cases be fed directly from the output of R1.
- If the *INTEST* instruction were not supported, R2 and M2 could be omitted from cells that feed data from a system pin to the on-chip system logic. The input labeled PI then would be connected directly to the output labeled PO.

8.5.2 Specifications for boundary-scan register instructions

The specifications for boundary-scan instructions given in the following subclauses of this clause define

- Whether the instruction is mandatory or optional;
- Which test data registers can be connected in the serial path between TDI and TDO;
- The restrictions (if any) on the choice of binary codes for each instruction (i.e., the patterns of 1s and 0s that, when shifted into the instruction register, cause the instruction to be selected); and
- The flow of data between the component's system pins, the boundary-scan register cells, and the on-chip system logic.

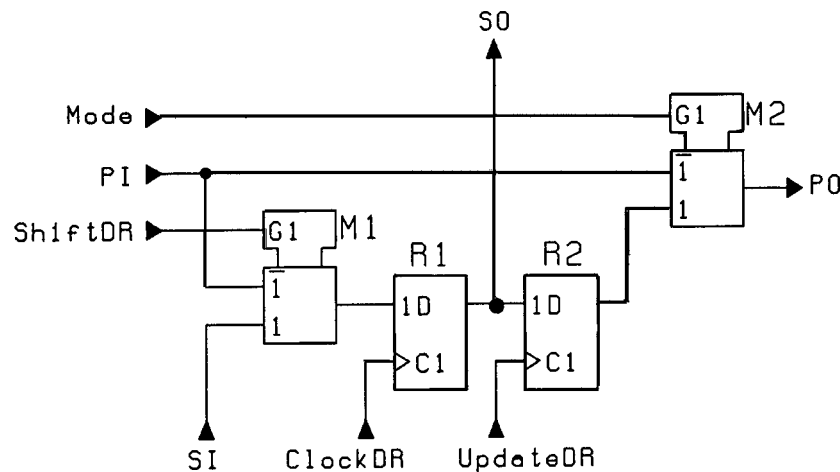


Figure 8-2—An example boundary-scan register cell design

The specifications are supported by descriptive text that includes a version of Figure 8-3 that shows one input and one output for a component. The solid bold lines in later copies of this figure show the mandatory data flows for each instruction.

8.6 The *SAMPLE* instruction

The mandatory *SAMPLE* instruction allows a snapshot of the normal operation of the component to be taken and examined.

8.6.1 Specifications

Rules

- Each component shall provide a *SAMPLE* instruction.
- The *SAMPLE* instruction shall select *only* the boundary-scan register to be connected for serial access between TDI and TDO in the *Shift-DR* controller state (i.e., no other test data register may be connected in series with the boundary-scan register).
- When the *SAMPLE* instruction is selected, the operation of the test logic shall have no effect on the operation of the on-chip system logic or on the flow of signals between the system pins and the on-chip system logic.
- When the *SAMPLE* instruction is selected, the states of all signals flowing from the on-chip system logic or through system pins (input or output) shall be loaded into the boundary-scan register on the rising edge of TCK in the *Capture-DR* controller state.

NOTE—The intent of this rule is to specify when the loading action should occur. Detailed specifications for the choices of signal values to be loaded are provided in Rules 11.5.1 f) and 11.6.1 h), respectively, for system logic inputs and system logic outputs.

Recommendations

- Where each of *SAMPLE* and *PRELOAD* implements the functionality of the other, they should share a common binary value(s).

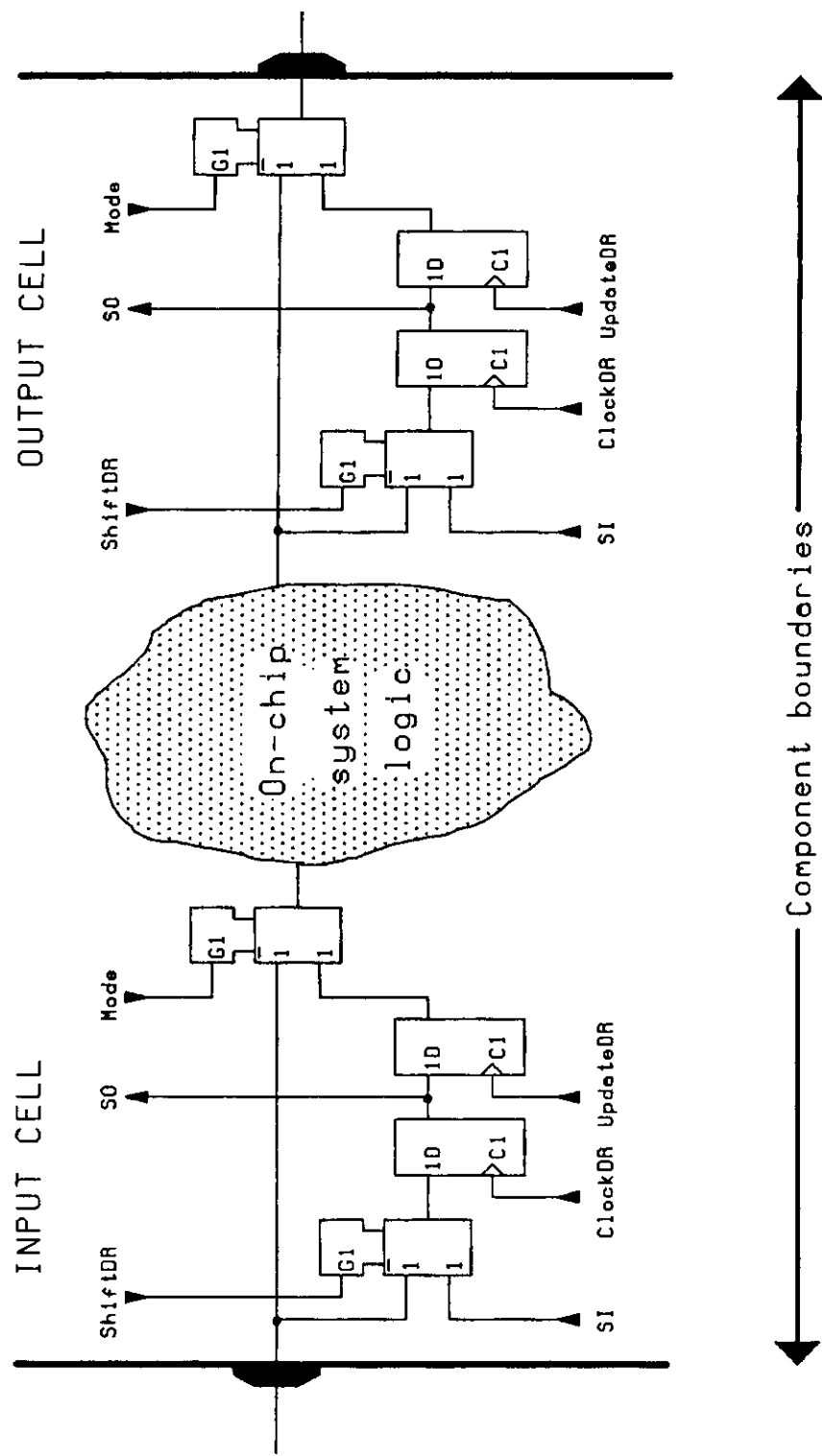


Figure 8-3—Circuit used to illustrate boundary-scan instructions