

IEEE Standard Test Access Port and Boundary-Scan Architecture

Sponsor
**Test Technology Standards Committee
of the
IEEE Computer Society**

Approved February 15, 1990
IEEE Standards Board

Approved June 17, 1993
IEEE 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.

Keywords: boundary-scan, boundary-scan register, circuit boards, circuitry, printed circuit boards, test ports

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA
Copyright © 1993 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1993. Printed in the United States of America
ISBN 1-55937-350-4

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 Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

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

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.

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 all 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 technical 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 on standards and requests for interpretations should be addressed to:

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

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Foreword

(This foreword is not a part of IEEE Std 1149.1-1990, IEEE 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.

Development of IEEE Standard Test Access Port and Boundary-Scan Architecture

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.

Between 1985 and approval on February 15, 1990, many individuals made valuable contributions to the development of this standard. At the time of approval of this standard, the members of the working group were:

Colin M. Maunder (*chair and editor*)
Rodham E. Tulloss, *Vice Chair*

Dilip K. Bhavsar
Vassilios Gerousis
Grady L. Giles
Charles L. Hudson
Curtis Jensen
Dirk van de Lagemaat

Patrick F. McHugh
Johann Maierhofer
Math N. M. Muris
Dieter Ohnesorge
Stig Oresjo

Gordon D. Robinson
Robert J. Russell
William H. Smith
Michael Tchou
Lee Whetsel
Thomas W. Williams

These people were supported by many other individuals from many different organizations who contributed time, administrative effort, and technical suggestions.

In particular, the working group wishes to acknowledge the contributions made by the following individuals:

LaNae Avra
Paul Bardell
Frans P. M. Beenker
Harry Bleeker
William C. Bruce
Ray Chapman
Peter Fleming
Walter Ghisler
Najmi Jarwala

Matthias Kaufmann
William McAnney
Bruno Mueller
Paul Ocampo
Anwar Osseyran
Kenneth P. Parker
Michel Parot
David Richards
Derek Roskell

David L. Rutledge
Wim Sauerwald
Erwin Trischler
Jon L. Turino
Chantal Vivier
R.G. Walther
H. Whittemore
Chi W. Yau
Mark Zagorin

The following people were on the balloting committee that approved this standard for submission to the IEEE Standards Board:

H. Gordon Adshead	Scott Davidson	Daniel Hu
Paul Akins	Vincent T. DeBuono	Andy Huang
Charles B. Albert	Bulent I. Dervisoglu	Charles Hudson
David Alley	Douglas Daskocil	Robert Hum
Anthony P. Ambler	Samuel Duncan	Axel Hunger
Allan H. Anderson	L. G. Egan	Phu Huynh
Kenneth Anderson	Samy El-Guebaly	Najmi Jarwala
Richard J. Andreano	Yacoub M. El-ziq	Curtis Jensen
Jack H. Arabian	Steven Elkind	William A. Johnson
Elmer Arment	Chris Ellingham	Edward M. Johnston
Dave Armstrong	Ado Erdal	Nick Kanopoulos
William D. Atwell, Jr.	John D. Evans	William L. Keiner
Chris B. Bagge	John M. Ewalt	Brian A. Kelley
William A. Baker	Patrick Fasang	Randy J. Kelsey
Paul H. Bardell	Erik L. Feldman	Javad Khakbaz
Robert W. Bassett	R. Scott Fetherston	Robert B. Kieffer
Frans P.M. Beenker	Kenneth W. Fisher	L. Kilroy
Steven Bennett	Peter Fleming	Charles R. Kime
R.G. Bennetts	Andrew Flint	Larry Kinney
Prashant Bhangui	Stephen Forde	William M. Klein
Christophe Bianchi	Ernest E. Forster	Frances Koo
Sol L. Black	Irving B. Frank	Roger Kozlowski
John Blecha	Andrew Fraser	Bruce P. Kraemer
Harry Bleeker	Vassilios Gerousis	Gunter T. Krampl
Paul Breedlove	Grady L. Giles	Gerd Kruger
Melvin Breuer	M. Gisborne	Steven Ladd
Alan L. Bridges	Prabhu Goel	Lak Ming Lam
David Brown	Festus Gail Gray	David L. Landis
William C. Bruce, Jr.	D. Greenwood	Thomas L. Langford II
Patrick H. Buffet	Terence Gregory	Bjorn Larsen
Philip Bullinger	David H. Grimm	Johnny LeBlanc
Mike Bullock	Edward B. Hakim	Robert Ledbetter
James C. Bussert	John S. Haluska	Ben H. Lee
Barry Caldwell	Marius Hancu	Raymond M. Lee
Gregory Callahan	Jeff Hansen	Wha-Joon Lee
Bruce Campbell	Peter L. Hansen	Duane G. Leet
John C. Caporal	Fred Harrison	Robert Lester
William Carney	Richard Hartman	Antonio Lioy
Ralph Cavin	Vance Harwood	Robert Lipp
Eduard Cerny	Wallace B. Harwood III	Gordon G. Liu
R. Chandramouli	Tord Haulin	Ulrich Ludemann
Hung Chang	John P. Hayes	Bruce H. Luhrs
Ray Chapman	Charles Hefner	Burt Magen
Wilton R. Chiles	Christian G. Heiter	Kenneth D. Mandl
Peter Claydon	Richard Hennessy	Michael Marhoefer
Melvin I. Cohen	Scott Herrington	James K. Mathewes, Jr.
Arthur M. Concouvis	William J. Hery	Colin M. Maunder
Nigel Crawley	Peter W. Hibson	Solomon Max
R.S. Crowley	Jack Hilibrand	Peter Maxwell
Les Crudele	Jim Homer	John C. McCann
Christo Da Costa	Lee F. Horney II	Patrick F. McHugh

Larry McNaughton
Earl J. Meiers
John Mick
Brent Miller
Jose M. Miranda
Maurice Mohr
Edward J. Moran
Larry Moran
Hussein T. Mouftah
John Moynihan
Shridhar K. Mukund
Math N. M. Muris
Brian T. Murray
Benoit Nadeau-Dostie
Joseph J. Nahas
Lori R. Nelson
Huan T. Nguyen
Van Minh Nguyen
Tom O'Rourke
Paul Ocampo
Dieter Ohnesorge
Wayne Olson
Calvin M. Osborne
A. Osseyran
Kenneth P. Parker
Michel Parot
Daniel Payne
James Pennell
Herman H. Plott
Kenneth E. Posse
Theo J. Powell
Paolo Prinetto
Lawrence Prucha
Jeffrey R. Quay

Edward Ramirez
Mark L. Rapiere
Shishpal Rawat
David J. Richards
Gordon D. Robinson
Lorin Rocks
Robert J. Rodio
Raul Rodriquez
Herman Roopchand
Barry C. Rosales
Derek Roskell
Craig G. Ross, Jr.
Ashim Roy
Robert J. Russell
William E. Russell, Jr.
Sergiu Samuel
Yvon Savaria
Robert S. Schamis
James E. Scharf
Bengt-Olaf Schneider
Teresa M. Schofield
Micaela Serra
Ravi Shankar
Simon J. Shaw
Jacob Shuker
David M. Siefert
Eric Skuldt
William J. Smerek
Jeffrey N. Smith
William H. Smith
Arun Somani
Fabio Somenzi
David Stannard
James H. Stewart
William Sullivan

Jacques Tazartes
Michael Tchou
Lluís Teres
Jacques Tete
David W. Thompson
Cihan Tinaztepe
Jack Trautman
Paul C. Tremoulet
Erwin Trischler
Joseph G. Tront
Li-Ching Tsai
Rod Tulloss
Jon Turino
Mark E. Turner
Dirk van de Lagemaat
James Van Derwiele
Ronald P. van Riessen
E. Velez
Rudolf E. Vogeli
Zvonko Vranesic
Ronald Wadsack
Malcolm R. Wallace
Bernath Walter
Mats Warne
J. Richard Weger
Lee Whetsel
Harry Whittemore
James Whitten
Brian R. Wilkins
Arthur Willging
T.W. Williams
William J. Wolf
Chi W. Yau
Guenther Zwiehoff

When the IEEE Standards Board approved this standard February 15, 1990, it had the following membership:

Marco W. Migliaro, *Chair*
James M. Daly, *Vice Chair*
Andrew G. Salem, *Secretary*

Dennis Bodson
Paul L. Borrill
Fletcher J. Buckley
Allen L. Clapp
Stephen R. Dillon
Donald C. Fleckenstein
Jay Forster *

Thomas L. Hannan
Kenneth D. Hendrix
John W. Horch
Joseph L. Koepfinger *
Michael A. Lawler
Donald J. Loughry
John E. May, Jr.
Lawrence V. McCall

L. Bruce McClung
Donald T. Michael *
Stig Nilsson
Roy T. Oishi
Gary S. Robinson
Terrance R. Whittemore
Donald W. Zipse

* Member Emeritus

Mary Lynne Nielsen
IEEE Standards Project Editor

Foreword to IEEE Std 1149.1a

(This foreword is not a part of IEEE Std 1149.1-1990 , IEEE Standard Test Access Port and Boundary-Scan Architecture.)

This supplement to IEEE Std 1149.1-1990 contains corrections, clarifications, and enhancements.

Corrections

Corrections are provided for errors located as a result of interaction between the developers and users of IEEE Std 1149.1-1990.

Clarifications

Interaction between developers and users of IEEE Std 1149.1-1990 highlighted parts of the original text (mostly, the descriptive material) that were ambiguous or insufficiently clear. This supplement provides clarifications that, it is hoped, will assist future users in implementing the standard.

Enhancements

Two enhancements have been made to the features defined by IEEE Std 1149.1-1990.

First, two new optional instructions have been defined—*CLAMP* and *HIGHZ*. These specifications standardize instructions that were often implemented as design-specific features on early commercial integrated circuits that claimed conformance to IEEE Std 1149.1-1990 . The provision of standard instruction names and corresponding specifications will assist in the development of design and test tools that can automatically exploit the existence of the relevant functionality.

Second, an option has been provided to switch a component from a mode in which it complies to IEEE Std 1149.1 into one in which it supports another design-for-test approach (e.g., level-sensitive scan-design for stand-alone component testing).

To assist readers already familiar with IEEE Std 1149.1-1990, paragraphs that contain changes are marked by margin bars in this supplement.

Development of This Supplement

The effort to develop this supplement was begun by the IEEE 1149.1 Working Group immediately following approval and publication of IEEE Std 1149.1-1990.

Between May 1990 and approval of this supplement on June 17, 1993, many individuals made valuable contributions. Particular thanks are due to:

Colin M. Maunder (*chair (1990) and vice-chair (1991)*)
Rodham E. Tulloss (*vice-chair (1990) and chair (1991, 1992)*)
Dirk van de Lagemaat (*vice-chair (Europe, 1992, 1993)*)

Several individuals joined the working group as development work progressed, reflecting the growing use of IEEE Std 1149.1-1990 by industry. At the time of approval of IEEE Std 1149.1a-1993, the members of the working group were:

Najmi Jarwala, *Chair*
John Andrews (*vice-chair (North America)*)
Math N. M. Muris (*vice-chair (Europe)*)

Dilip K. Bhavsar
Harry Bleeker
Terry Borroz
Bernhard Geisberger
Grady L. Giles
Peter L. Hansen
Adam W. Ley

Colin M. Maunder (editor)
Patrick F. McHugh
Dieter Ohnesorge
Stig Oresjo
Kenneth P. Parker
Michel Parot

Gordon D. Robinson
Markus Robinson
Robert J. Russell
Rodham E. Tulloss
Dirk van de Lagemaat
Lee Whetsel
T. W. Williams

The following persons were on the balloting committee:

Gerald F. Adams	B. Kaminska	Calvin M. Osborne
Gordon Adshead	Nick Kanopoulos	Krishnan Palaniswami
Robert Aitken	Jake Karrfalt	Chris Papachristou
John Andrews	David C. Keezer	Kenneth P. Parker
William Atwell	Satish Keshava	Michel Parot
Chris Bagge	Russ Kneupper	Michael Paulson
Behrooz Bandall	Kevin Kornegay	Bruce Peterson
Paul Bardell	Roger Kozlowski	Kenneth Posse
James Beausang	Gunter Krampl	Paolo Prinetto
Elizabeth Benedict	Thomas Kropf	Lawrence Prucha
R. G. Bennetts	Gerd Kruger	Jeffrey Quay
Dilip K. Bhavsar	Thomas M. Kurihara	Rochit Rajsuman
Sol Black	Steven K. Ladd	Edward P. Ratazzi
Harry Bleeker	Dirk van de Lagemaat	Shishpal Rawat
Federico Bonzanigo	David Landis	Gordon D. Robinson
Melvin Breuer	Bjorn B. Larsen	Derek Roskell
Alan Bridges	Robert Ledbetter	Ashim Roy
Bill Bruce	Wha-Joon Lee	Robert J. Russell
Barry C. Bucher	Raymond Lee	Sergiu Samuel
Michael L. Bushnell	Yann-Hang Lee	Cayetano Sanchez
Terence Chan	Jasopin Lee	Dana Schow
R. Chandramouli	Regis Leveugle	Lori E. Schramm
Kedong Chao	Marc Levitt	Richard M. Sedmak
Tony Cheng	Adam W. Ley	Ashok K. Sharma
Wilton Chiles	Chih-Jen Lin	Raj Sharma
John Chin	Antonio Lioy	Pradip K. Skimani
Michael Collins	Dan Loocke	William H. Smith
R. Dandapani	G. Lousberg	Mani Soma
Michael Donahue	Hede Ma	Arun Somani
David G. Edwards	Rafic Makki	Pradip K. Srimani
Leo G. Egan	R. A. Mansfield	Bret A. Stewart
Chris Ellingham	David Marquart	William Sullivan
Peter Fleming	Colin M. Maunder	Xiaoling Sun
Vassilios Gerousis	Patrick F. McHugh	Carlos Terrer
N. Gianbiasi	David E. Medina	Jacques Tete
Grady L. Giles	Earl Meiers	Cihan Tinaztepe
Steven Gonet	Michael Menovich	Mike Topsakal
A. J. van de Goor	Donald Merlino	Rodham E. Tulloss
Dong S. Ha	Joseph A. Mielke	Jon L. Turino
Peter L. Hansen	Zeljko Miksic	James Vanderwiele
Vance Harwood	Richard Miller	Tom Volpe
Abu Hassan	Steven D. Millman	Russell J. Wagner
Kazumi Hatayama	Yinghua Min	K. D. Wagner
Tord Haulin	John Moynihan	Malcolm Wallace
Wei-Cheng Her	Tom Munns	Gary Wang
David Hickey	Math N. M. Muris	Allen Warren
Hezi Himelfarb	Prawat Nagvajara	Thomas J. Wey
Chuck Hudson	Zainalabedin Navabi	Lee Whetsel
Axel Hunger	Roy H. Nesson	Harry Whittemore
Masaaki Inadachi	Arnold Nordsieck	Brian Wilkins
Neal Jaarsma	Jim O'Reilly	David M. Wu
Kamran Jamal	Paul Ocampo	Cheng-Wen Wu
Najmi Jarwala	Dieter Ohnesorge	Stephen Yurash
Curtis Jensen	Folusho Oladeinde	Farzad Zarrinfar
Carl Kagy	Stig Oresjo	Guenther Zwiehoff



Documents

Document Sharing Hub

When the IEEE Standards Board approved this standard on June 17, 1993, it had the following membership:

Wallace S. Read, *Chair*
Donald C. Loughry, *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Bari
Jose A. Berrios de la Paz
Clyde R. Camp
Donald C. Fleckenstein
Jay Forster *
David F. Franklin
Ramiro Garcia
Donald N. Heirman

Jim Isaak
Ben C. Johnson
Walter J. Karplus
Lorraine C. Kevra
E. G. "Al" Kiener
Ivor N. Knight
Joseph L. Koepfinger *
D. N. "Jim" Logothetis

Don T. Michael *
Marco W. Migliaro
L. John Rankine
Arthur K. Reilly
Ronald H. Reimer
Gary S. Robinson
Leonard L. Tripp
Donald W. Zipse

* Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall

Richard B. Engelman

David E. Soffrin
Stanley I. Warshaw

Mary Lynne Nielsen
IEEE Standards Project Editor

CLAUSE	PAGE
1. Introduction	1
1.1 Background Reading	1
1.2 An Overview of the Operation of IEEE Std 1149.1	1
1.3 The Use of IEEE Std 1149.1 to Test an Assembled Product	2
1.4 The Use of IEEE Std 1149.1 to Achieve Other Test Goals	5
2. General Information	5
2.1 Document Outline	5
2.2 Conventions	6
2.3 Definitions	6
2.4 References	9
3. The Test Access Port (TAP)	9
3.1 Connections That Form the Test Access Port (TAP)	9
3.2 The Test Clock Input—TCK	9
3.3 The Test Mode Select Input—TMS	10
3.4 The Test Data Input—TDI	11
3.5 The Test Data Output—TDO	11
3.6 The Test Reset Input—TRST*	12
3.7 Interconnection of Components Compatible With This Standard	13
3.8 Subordination of This Standard Within a Higher Level Test Strategy	14
4. Test Logic Architecture	15
4.1 Test Logic Design	16
4.2 Test Logic Realization	17
5. The TAP Controller	18
5.1 TAP Controller State Diagram	18
5.2 TAP Controller Operation	23
5.3 TAP Controller Initialization	31
6. The Instruction Register	32
6.1 Design and Construction of the Instruction Register	32
6.2 Instruction Register Operation	33
7. Instructions	35
7.1 Response of the Test Logic to Instructions	35
7.2 Public Instructions	36
7.3 Private Instructions	37
7.4 The <i>BYPASS</i> Instruction	37
7.5 Boundary-Scan Register Instructions	38
7.6 The <i>SAMPLE/PRELOAD</i> Instruction	42
7.7 The <i>EXTEST</i> Instruction	43
7.8 The <i>INTEST</i> Instruction	48
7.9 The <i>RUNBIST</i> Instruction	52

CLAUSE	PAGE
7.10 The <i>CLAMP</i> Instruction	54
7.11 Device Identification Register Instructions.....	55
7.12 The <i>IDCODE</i> Instruction	55
7.13 The <i>USERCODE</i> Instruction	56
7.14 The <i>HIGHZ</i> Instruction	57
8. Test Data Registers	58
8.1 Provision of Test Data Registers.....	59
8.2 Design and Construction of Test Data Registers	61
8.3 Test Data Register Operation	62
9. The Bypass Register.....	64
9.1 Design and Operation of the Bypass Register.....	64
10. The Boundary-Scan Register	65
10.1 Introduction to This Chapter	66
10.2 Register Design	70
10.3 Register Operation	72
10.4 General Rules Regarding Cell Provision	73
10.5 Provision and Operation of Cells at System Logic Inputs	76
10.6 Provision and Operation of Cells at System Logic Outputs	84
10.7 Bidirectional Signals	99
10.8 Redundant Cells	104
10.9 Special Cases.....	105
11. The Device Identification Register	107
11.1 Design and Operation of the Device Identification Register	108
11.2 Manufacturer Identity Code	110
11.3 Part-Number Code	111
11.4 Version Code.....	111
12. Conformance and Documentation Requirements	111
12.1 Claiming Conformance to This Standard.....	111
12.2 Prime and Second Source Components	112
12.3 Documentation Requirements	113
Annex A An Example Implementation Using Level-Sensitive Design Techniques (Informative).....	116

IEEE Standard Test Access Port and Boundary-Scan Architecture

1. Introduction

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).

1.1 Background Reading

Persons who are not familiar with scan test and self-test techniques for digital electronic circuits may find it helpful to consult the following publications prior to reading this standard:

- AGRAWAL, V.D. and SETH, S.C., *Test generation for VLSI chips*, IEEE Computer Society Press, 1988.
- BENNETTS, R.G., *Design of testable logic circuits*, Addison-Wesley, 1984.
- EICHELBERGER, E.B. and WILLIAMS, T.W., A logic design structure for LSI testability, *Journal of Design Automation and Fault-Tolerant Computing*, vol. 2, no. 2, May 1978, pp. 165–178.
- KONEMANN, B. et al., Built-in logic block observation techniques, *Proceedings of the IEEE Test Conference*, IEEE Computer Society Press, 1979, pp. 37–41.
- MICZO, A., *Digital logic testing and simulation*, Harper & Row, 1986.

1.2 An Overview of the Operation of IEEE Std 1149.1

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

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 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 are 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.

Following 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.

1.3 The Use of IEEE Std 1149.1 to Test an Assembled Product

This clause 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.

1.3.1 Board Test Goals

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 is hierarchic in that it 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 will henceforth concentrate on the case of an assembled printed circuit board constructed from a collection of digital integrated circuits.

At the board level, goals (a) and (b) are typically achieved 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 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, these 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 pre-existing 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.

1.3.2 What Is Boundary-Scan?

The boundary-scan technique involves the inclusion of a shift-register stage (contained in a boundary-scan 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 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 either be 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 Chapter 10, 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.

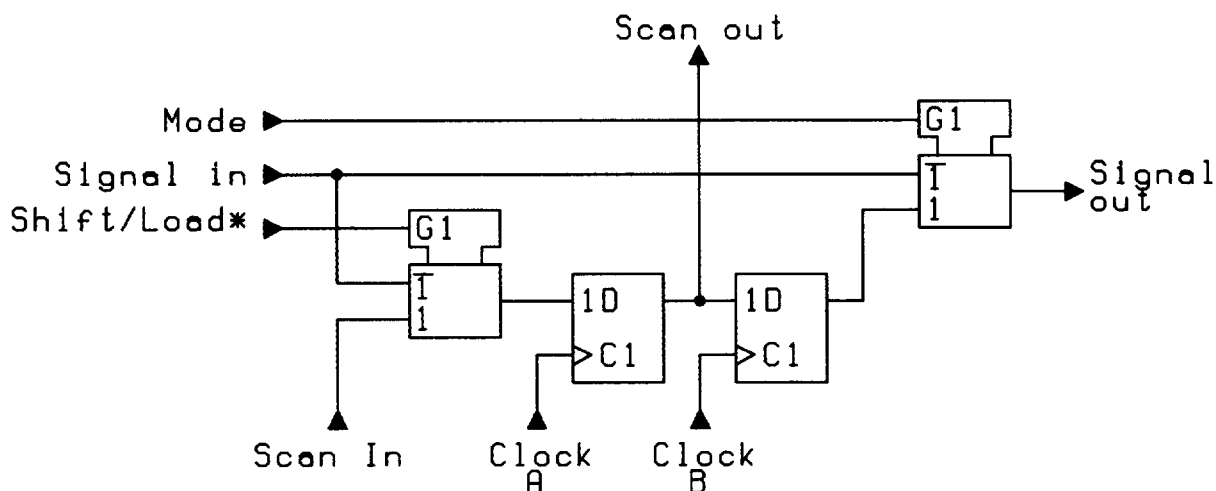


Figure 1-1—A Boundary-Scan Cell

The boundary-scan cells for the pins of a component are interconnected so as 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.

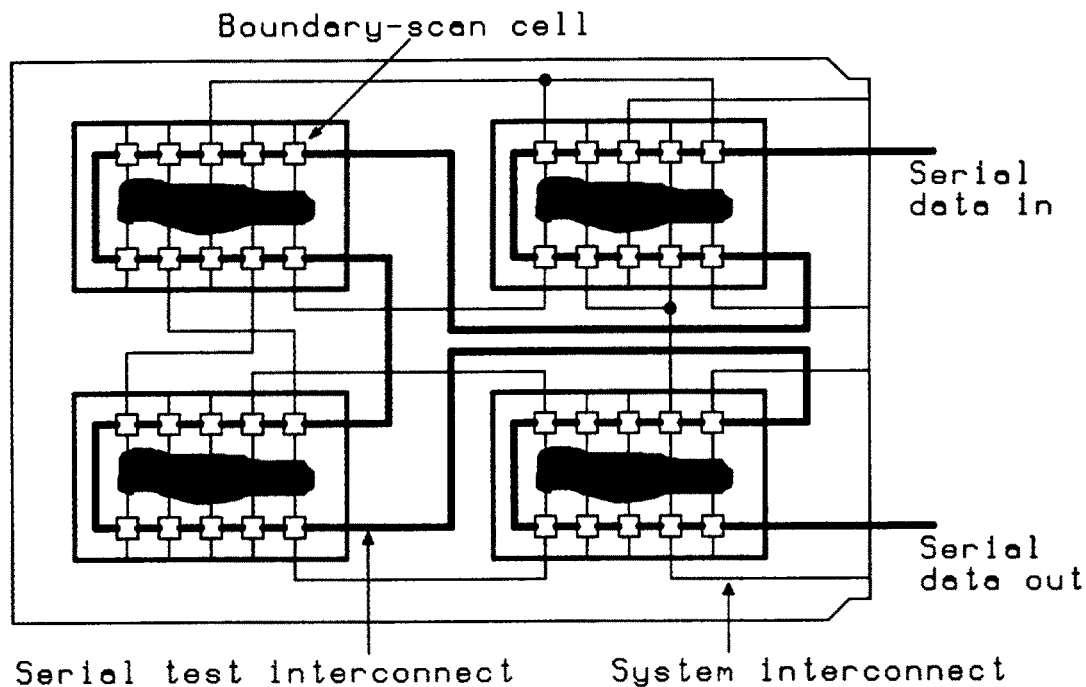


Figure 1-2—A Boundary-Scannable Board Design

If all the components used to construct a circuit have a boundary-scan register, then 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 using a functional (through the pins) ATE system or 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.

1.4 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.

2. General Information

2.1 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. Chapter 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 chapters of the document contain the specifications for particular features of this standard. Two classes of material are contained in these chapters:

Specifications

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

Rules specify the mandatory aspects of this standard. Clauses that are rules contain the word **shall**.

Recommendations indicate preferred practice for designs that seek to conform to this standard. Clauses that are recommendations contain the word **should**.

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 the standard. Clauses that are permissions contain the word **may**.

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. The Appendix to this standard contains an alternative implementation example. The descriptive material also discusses design decisions made during the development of this standard.

NOTE — The descriptive material contained in this standard is for illustrative purposes only and does not define a preferred implementation.

2.2 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:

	15.1.1c (ii)
Clause 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.

2.3 Definitions

The following terms are used within this standard.

2.3.1 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.

2.3.2 ATE: Automatic Test Equipment.

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

2.3.4 BILBO: Built In Logic Block Observer. A shift-register based structure used in some forms of self-testing circuit design.

2.3.5 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.

2.3.6 BYPASS:: A defined instruction for the test logic defined by this standard (see 7.4).

2.3.7 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.

2.3.8 chip-on-board testing: A test of a component after it has been assembled onto a printed circuit board or other substrate, for example, using the facilities defined by this standard.

2.3.9 CLAMP: A defined instruction for the test logic defined by this standard (see 7.10).

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

2.3.11 EXTEST: External test—a defined instruction for the test logic defined by this standard (see 7.7).

2.3.12 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.

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

2.3.14 HIGHZ: High impedance. A defined instruction for the test logic defined by this standard (see 7.14).

2.3.15 IDCODE: Identity code—a defined instruction for the test logic defined by this standard (see 7.12).

2.3.16 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.

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

2.3.18 instruction: A binary data word shifted serially into the test logic defined by this standard in order to define its subsequent operation.

2.3.19 INTEST: Internal test—a defined instruction for the test logic defined by this standard (see 7.8).

2.3.20 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.

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

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

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

2.3.24 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.

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

2.3.26 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.

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

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

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

2.3.30 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.

2.3.31 RUNBIST: Run Built-In Self-Test—a defined instruction for the test logic defined by this standard (see 7.9).

2.3.32 SAMPLE/PRELOAD: A defined instruction for the test logic defined by this standard (see 7.6).

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

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

2.3.35 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.

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

2.3.37 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.

2.3.38 stand-alone testing: A test of a component performed before it is assembled onto a board or other substrate, for example, using ATE.

2.3.39 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.

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

2.3.41 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.

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

2.3.43 TAP: The Test Access Port defined by this standard (see Chapter 3).

2.3.44 TCK: The Test Clock input pin contained in the TAP defined by this standard (see 5.3.2).

2.3.45 TDI: The Test Data Input pin contained in the TAP defined by this standard (see 3.4).

2.3.46 TDO: The Test Data Output pin contained in the TAP defined by this standard (see 3.5).

2.3.47 test logic: Any item of logic that is a dedicated part of the test logic architecture defined by this standard or is at the time of interest configured as a part of the test logic architecture defined by this standard.

2.3.48 TMS: The Test Mode Select input pin contained in the TAP defined by this standard (see 3.3).

2.3.49 TRST*: The Test Reset input pin contained in the TAP defined by this standard (see 3.6).

2.3.50 TTL: Transistor Transistor Logic.

2.3.51 update: Transfer a logic value from the shift-register stage of a data register cell or an instruction register cell into that the latched parallel output stage of the cell as a consequence of entry into the *Update-DR* or *Update-IR* controller state, respectively.

2.3.52 USERCODE:: User identity code—a defined instruction for the test logic defined by this standard (see 7.13).

2.3.53 3-state pin: A component output pin where the drive may be either active or inactive (for example, at high impedance).

2.4 References

The following publications shall be used in conjunction with this standard. When standards in this document are referred to, the latest revision shall apply.

[1] *JEDEC Publication 106-A*, Standard Manufacturer's Identification Code, The Joint Electron Device Engineering Council, July 1986.¹

3. The 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 or the test logic defined by this standard.

3.1 Connections That Form the Test Access Port (TAP)

3.1.1 Specifications

Rules

- a) The TAP shall include the following connections (defined in 3.3, 3.5, 3.6.2, and 3.7.2): TCK, TMS, TDI, and TDO.
- b) 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 3.8.2 (see also 5.3).
- c) 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).

3.1.2 Description

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

3.2 The Test Clock Input—TCK

TCK provides the clock for the test logic defined by this standard.

3.2.1 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.

¹Copies can be obtained from JEDEC, 2001 I Street NW, Washington D.C. 20006, USA..

3.2.2 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 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 may also 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, indicated by use of the phrase “on the rising (falling) edge of TCK.” These operations have to be completed within a fixed (frequency-independent) delay following 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 following 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). It is expected that all propagation delays will be such that correct operation is achieved under these circumstances, particularly when data is being transferred between TDO of one chip and TDI of another.

3.3 The Test Mode Select Input—TMS

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

3.3.1 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 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.

3.3.2 Description

Rule 3.3.1b 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 6.2). 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.

3.4 The Test Data Input—TDI

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

3.4.1 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 response identical to the application of a logic 1.
- c) When data is being shifted from TDI towards TDO, test data received at TDI shall appear without inversion at TDO following a number of rising and falling edges of TCK determined by the length of the instruction or test data register selected.

3.4.2 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 3.4.1b 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 7.4). For TTL-compatible designs, this rule may be met by inclusion of a pull-up resistor in the component's TDI input circuitry.

3.5 The Test Data Output—TDO

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

3.5.1 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 5.2).

3.5.2 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 8.3).

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 are generally 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 non-overlapping clocks, the retiming required by rule 3.5.1a is an inherent feature of the design.

The capability 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.

3.6 The Test Reset Input—TRST*

The optional TRST* input provides for asynchronous initialization of the TAP controller (see 5.3).

3.6.1 Specifications

Rules

- a) If TRST* is included in the TAP, then the TAP controller shall be asynchronously reset to the *Test-Logic-Reset* controller state when a logic 0 is applied to TRST* (see 5.3).
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, then the design of the circuitry fed from that input shall be such that an undriven input produces a 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.

3.6.2 Description

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

Rule 3.6.1b 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 3.6.1c 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 3.6.1d 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.

3.7 Interconnection of Components Compatible With This Standard

3.7.1 Specifications

Permissions

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

3.7.2 Description

Figures 3-1, 3-2, and 3-3 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 3-1) contains:

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

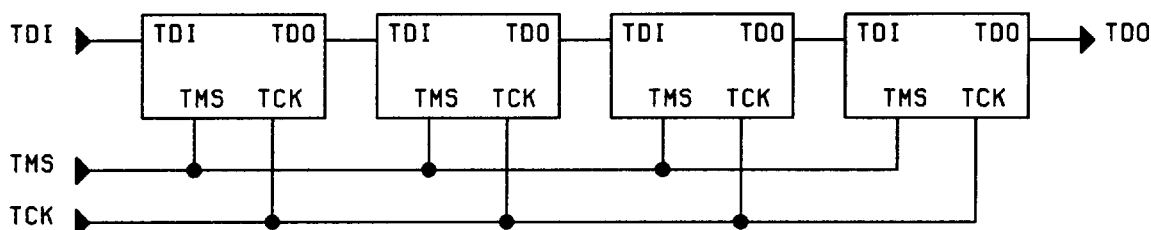


Figure 3-1—Serial Connection Using One TMS Signal

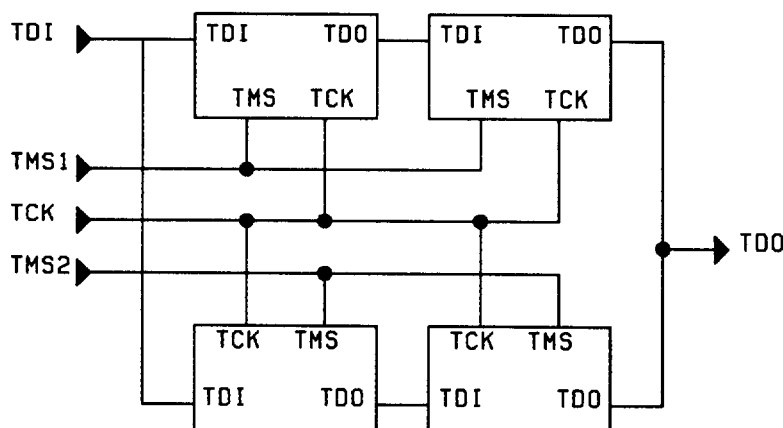


Figure 3-2—Connection in Two Paralleled Serial Chains

The hybrid serial/parallel connection shown in figure 3-2 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 3-3 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.

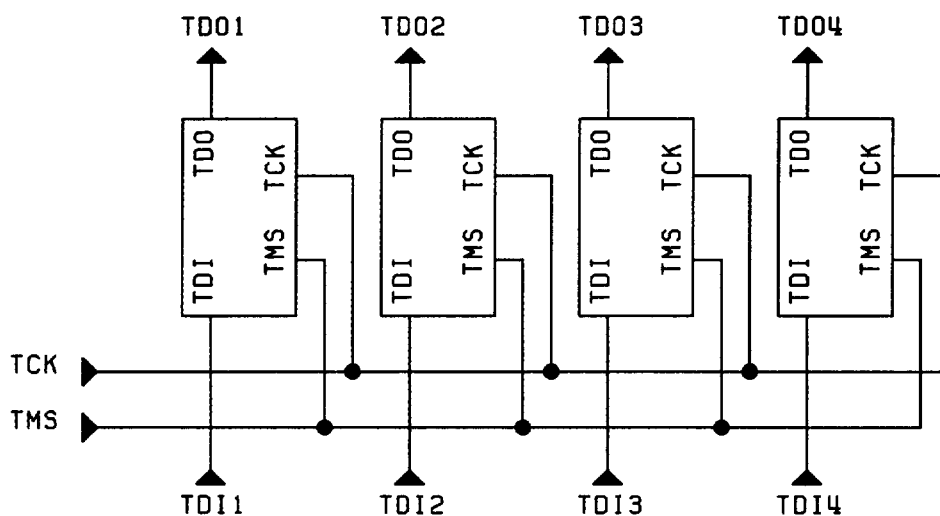


Figure 3-3—Multiple Independent Paths With Common TMS and TCK Signals

3.8 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 the Appendix) 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 clause 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, and not by a sequence of such patterns.

3.8.1 Specifications

Rules

- a) If a component is to be designed having both
 - i) Test functionality compliant with this standard; and
 - ii) Other test functionality that is not to be controlled via the test circuitry and the means of control defined in this standard,
 then 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 3.8.1i.)
- 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 of preventing problems from arising.
- 2 — The rules in other clauses 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 3.1.1c 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 alternate test mode of component operation.
- d) The event of enabling compliance with this standard by changing the logic pattern applied at the compliance-enable inputs of a component shall have an effect on the component equivalent to that of power-up of the component (see 6.2).

NOTE — Therefore, unless the optional TRST* input is provided, the transition at the compliance-enable inputs has to cause a reset of the test logic defined by this standard.
- e) Compliance-enable inputs shall be dedicated inputs to the component and shall not be used for any other purpose.

Recommendations

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

Permissions

- g) A component may have zero, one, or more compliance-enable inputs.
- h) 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.
- i) A component may have several compliance-enable patterns, all of which have equivalent effect.

3.8.2 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.

4. Test Logic Architecture

This chapter 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 chapters of this standard.

4.1 Test Logic Design

4.1.1 Specifications

Rules

- a) The following elements shall be contained in the test logic architecture:
 - i) A TAP (see Chapter 3);
 - ii) A TAP controller (see Chapter 5);
 - iii) An instruction register (see Chapter 6); and
 - iv) A group of test data registers (see Chapter 8).
- 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 5.2.

4.1.2 Description

A conceptual view of the top-level design of the test logic architecture defined by this standard is shown in figure 4-1. This figure, and the others included in the descriptive material contained in this standard, are examples intended only to illustrate a possible embodiment of the standard. These figures do not indicate a preferred implementation.

Key features of the design are:

- 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 Chapter 5.
- 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 Chapter 6.
- c) The group of test data registers. The group of test data registers shall include a bypass and a boundary-scan register. It may also 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 Chapter 8.

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 4-1, to retime the signal passing through it to occur on the falling edge of TCK.

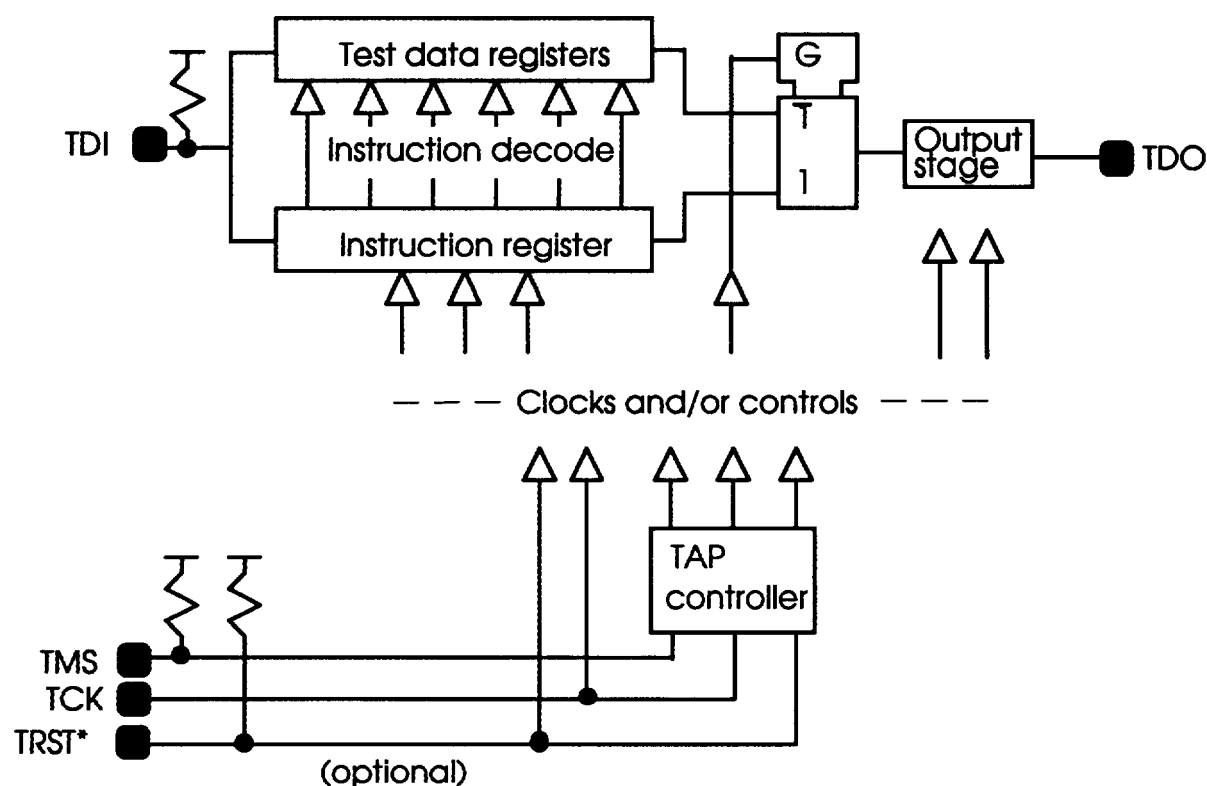


Figure 4-1 — A Conceptual Schematic of the Test Logic

4.2 Test Logic Realization

4.2.1 Specifications

Rules

- 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).
- If test access is required to a test data register without causing any interference to the operation of the on-chip system logic, then the circuitry used to construct that test data register shall be dedicated test logic.

4.2.2 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 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.

5. 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.

5.1 TAP Controller State Diagram

5.1.1 Specifications

Rules

- a) The state diagram for the TAP controller shall be as shown in figure 5-1.

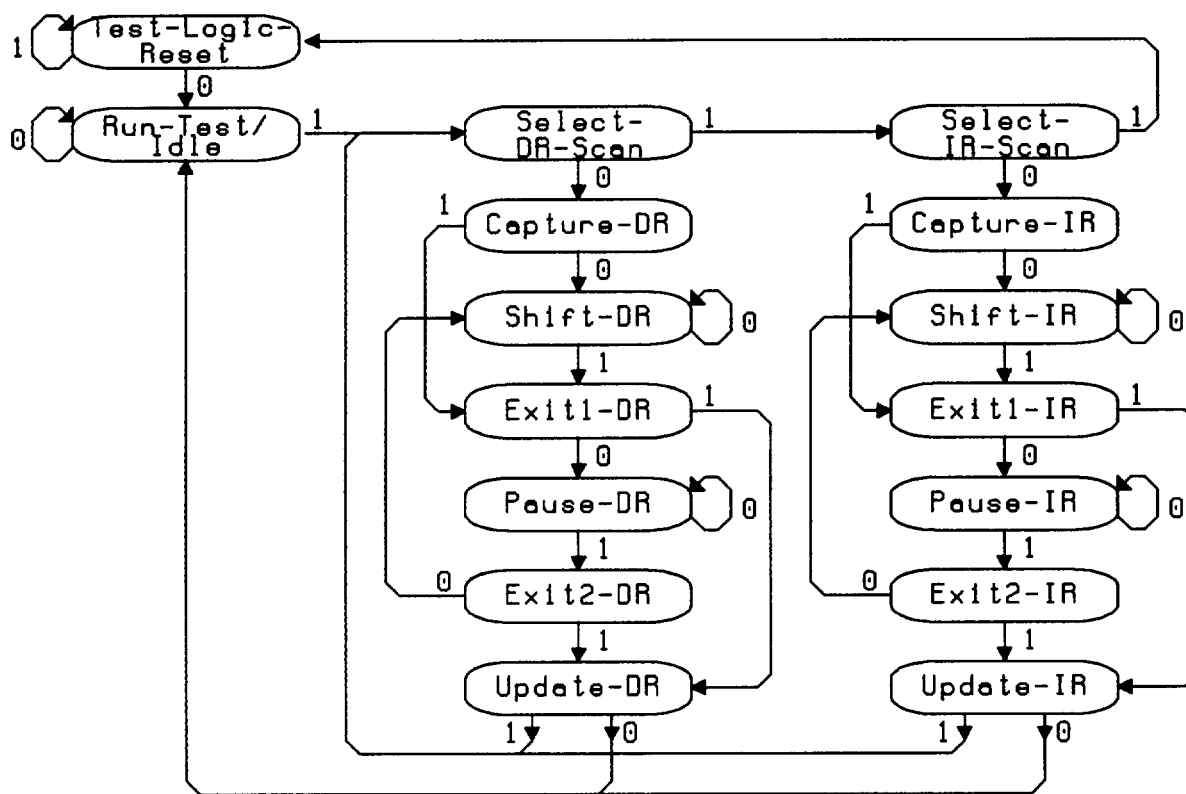


Figure 5-1 — TAP Controller State Diagram

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.

- 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 by figure 5-2.

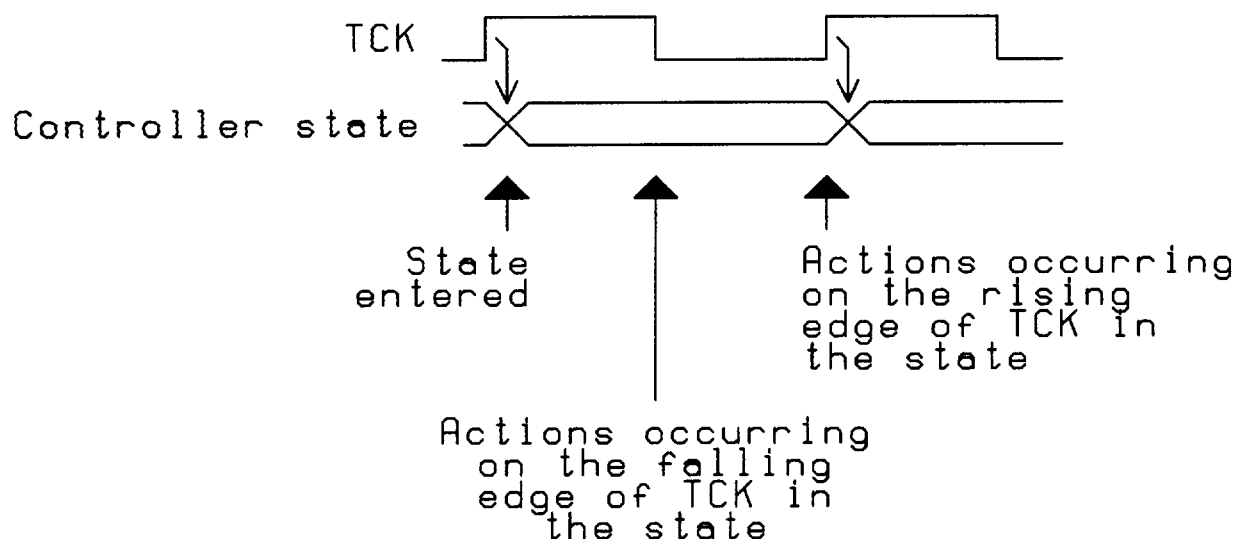


Figure 5-2—Timing of Actions in a Controller State

5.1.2 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 chapters 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 6.2). 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 following 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 7.2.1). 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 5.3).

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 7.9). Self-tests selected by instructions other than *RUNBIST* may also 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, then 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, then 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 towards 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 enters either 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 Chapter 6).

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 towards 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 enters either 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 testdata or instruction register can be temporarily halted. 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 5-1 lists the actions required for key types of test supported by this standard.

Table 5-1—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	Yes	No	Yes
Internal scan test	Maybe	No	Yes
Internal self-test using linear-feedback shift-registers, etc	No	Yes	No
Boundary-scan <i>SAMPLE/PRELOAD</i> test	Yes	No	Yes

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* state. The results of the test are captured into the test data register during the *Capture-DR* state.

For self-testing circuit designs based on linear-feedback shift-registers, 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 in their linear feedback shift-register modes during *Run-Test/Idle*. Since the result is already contained in a test data register, no action is required during *Capture-DR*.

5.2 TAP Controller Operation

Rules

- a) The TAP controller shall change state only in response to the following events:
 - i) A rising edge of TCK;
 - ii) A transition to logic 0 at the TRST* input (if provided); or
 - iii) 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 (figures 5-3 and 5-4).

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 the standard, but it should be such that the setup 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 5-2.
- d) Changes at TDO defined in Table 5-2 shall occur on the falling edge of TCK following entry into the state.

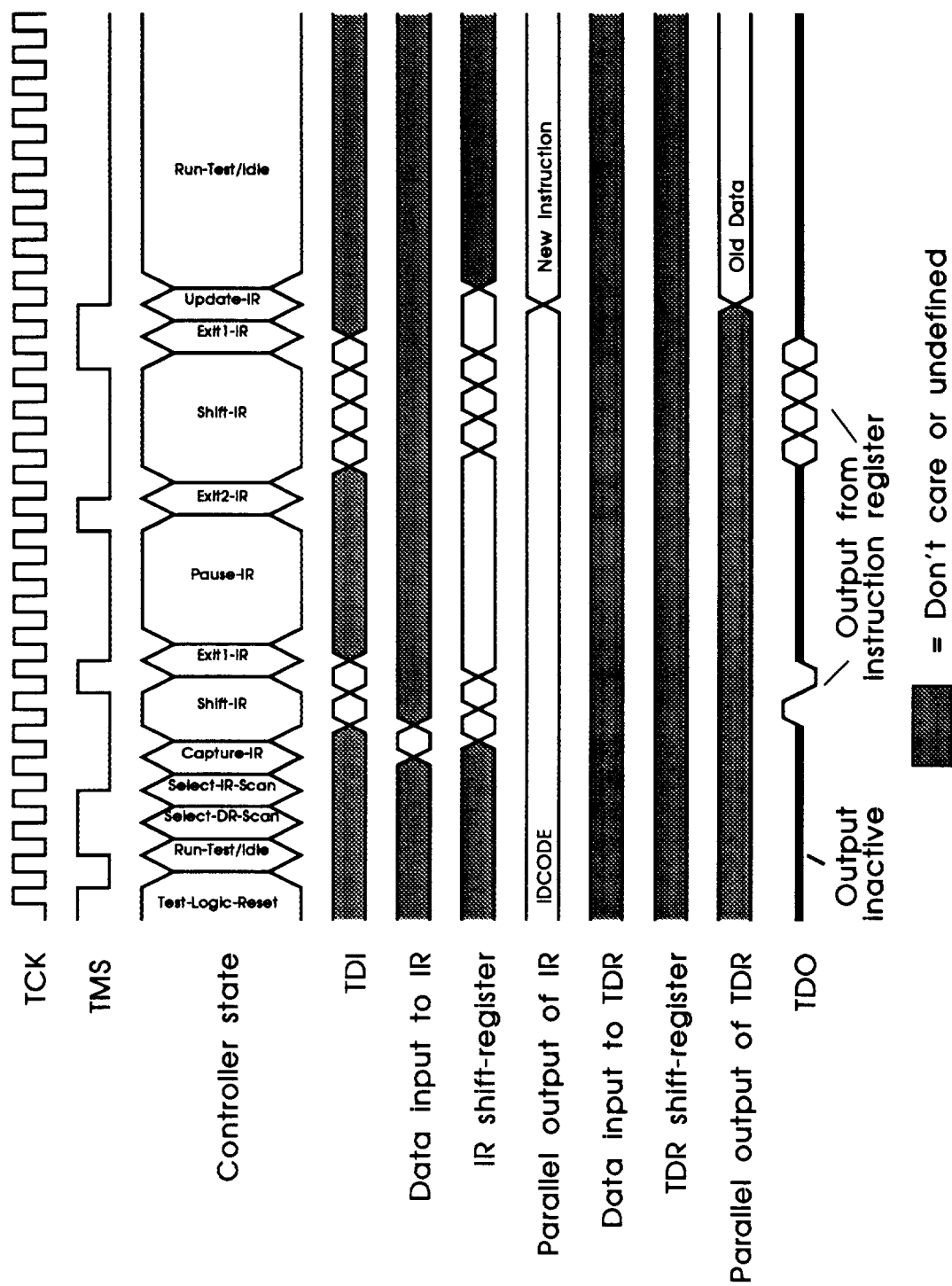


Figure 5-3—Test Logic Operation: Instruction Scan

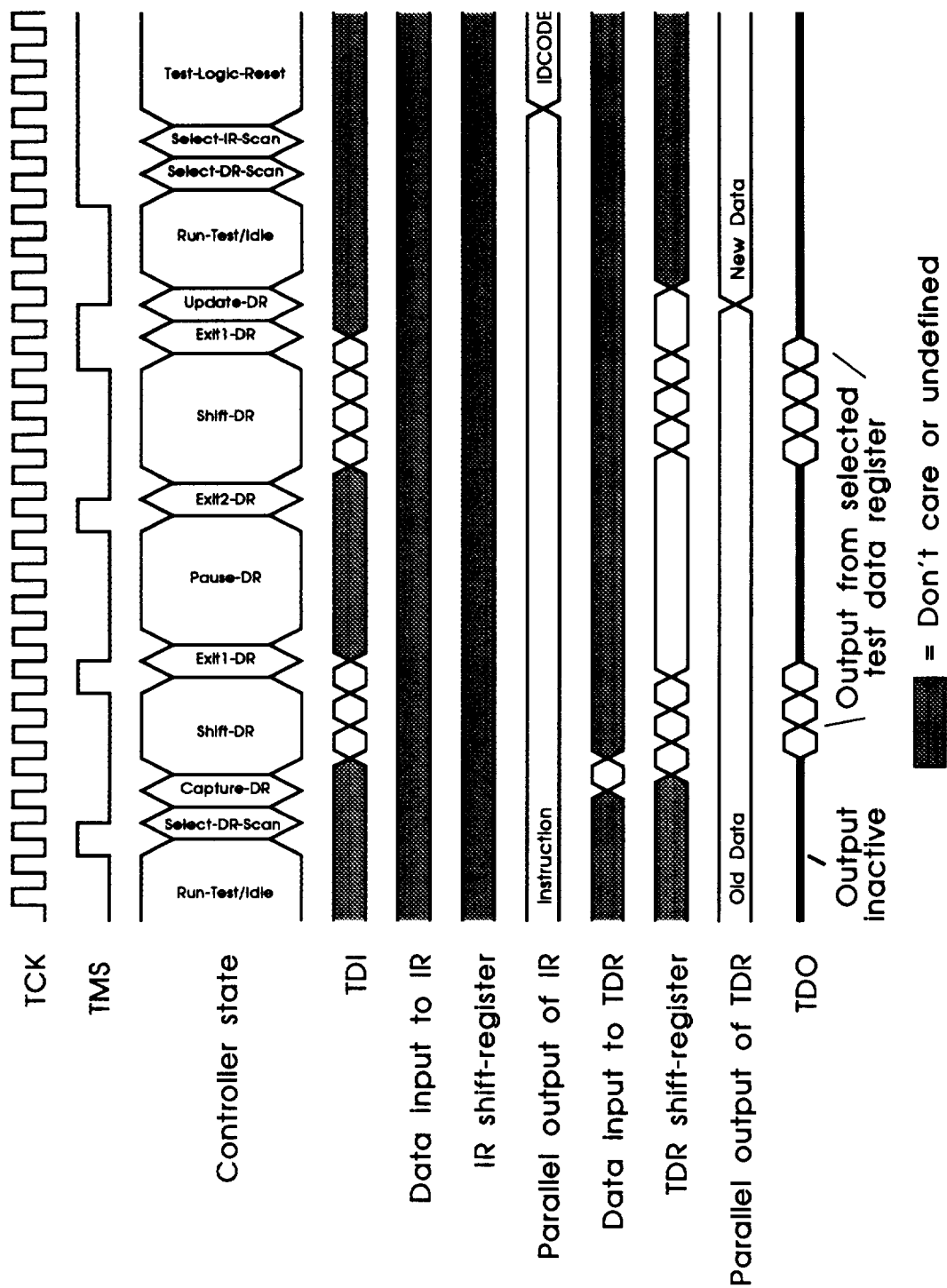


Figure 5-4— Test Logic Operation: Data Scan

Table 5-2—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 <i>Capture-IR</i> , <i>Pause-IR</i> , <i>Exit1-IR</i> , <i>Exit2-IR</i> , <i>Capture-DR</i> , <i>Pause-DR</i> , <i>Exit1-DR</i> , and <i>Exit2-DR</i> controller states, in addition to the <i>Shift-IR</i> and <i>Shift-DR</i> 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 3-2).		

5.2.1 Description

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

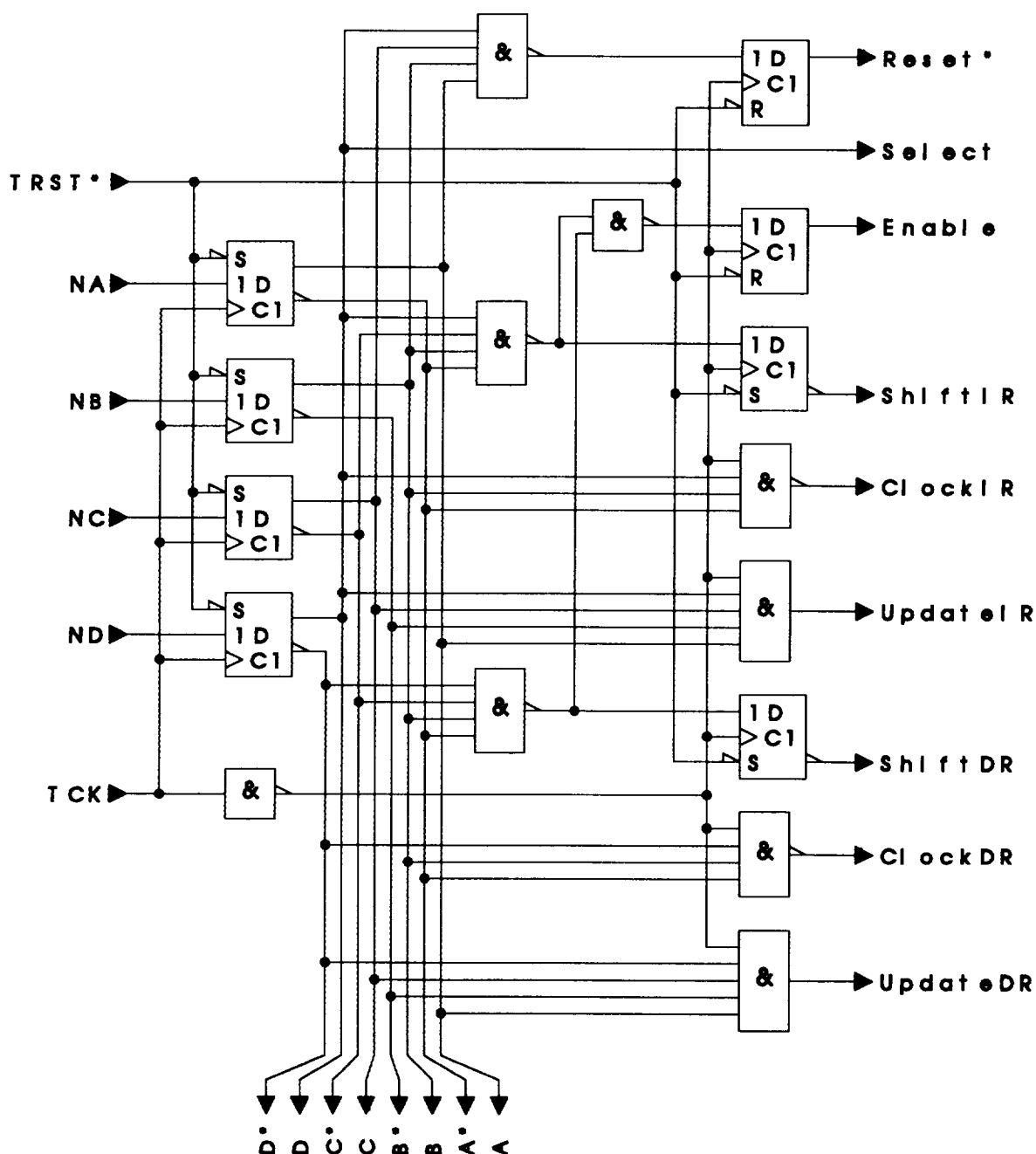


Figure 5-5—A TAP Controller Implementation—State Registers and Output Logic

NOTE — The circuit in figure 5-5 generates the various control signals used by the example circuits illustrated elsewhere in this standard. Note that the Select signal would be used to control the multiplexor shown in figure 4-1 and the Enable signal would be used for 3-state control of the TDO output. Note also that, while the ShiftDR and ClockDR signals may be broadcast to all test data registers, distribution of the UpdateDR control signal will be controlled according to the instruction held in the instruction register such that the signal is fed only to the test data register that is selected as the serial path between TDI and TDO.

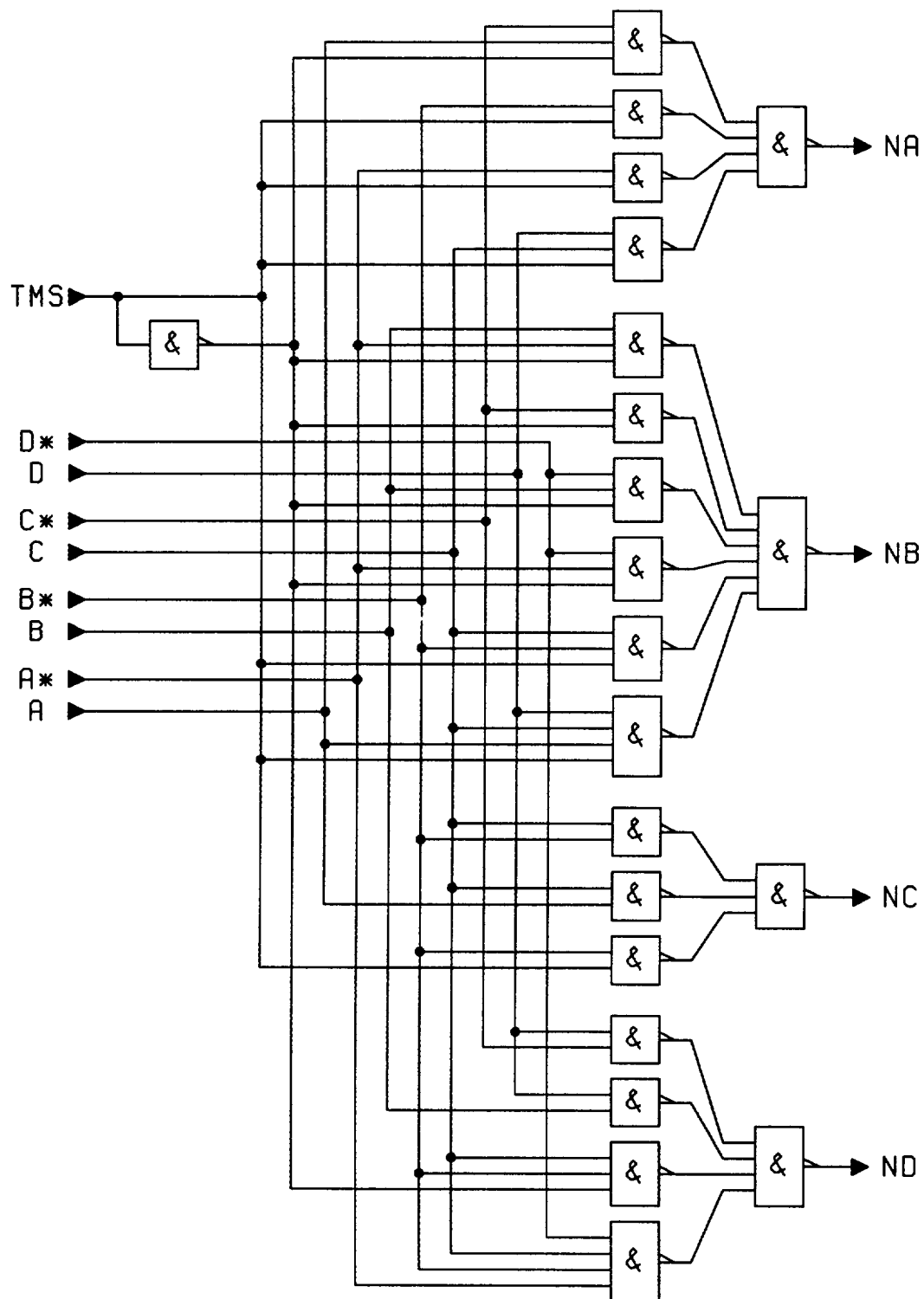


Figure 5-6—A TAP Controller Implementation—Next State Logic

The assignment of controller states in the example implementation is given in Table 5-3.

Table 5-3—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

The Boolean equations for the next state logic in figures 5-5 and 5-6 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 = value present at TMS

Figure 5-7 shows the operation of this controller implementation through instruction and test data register scan cycles.

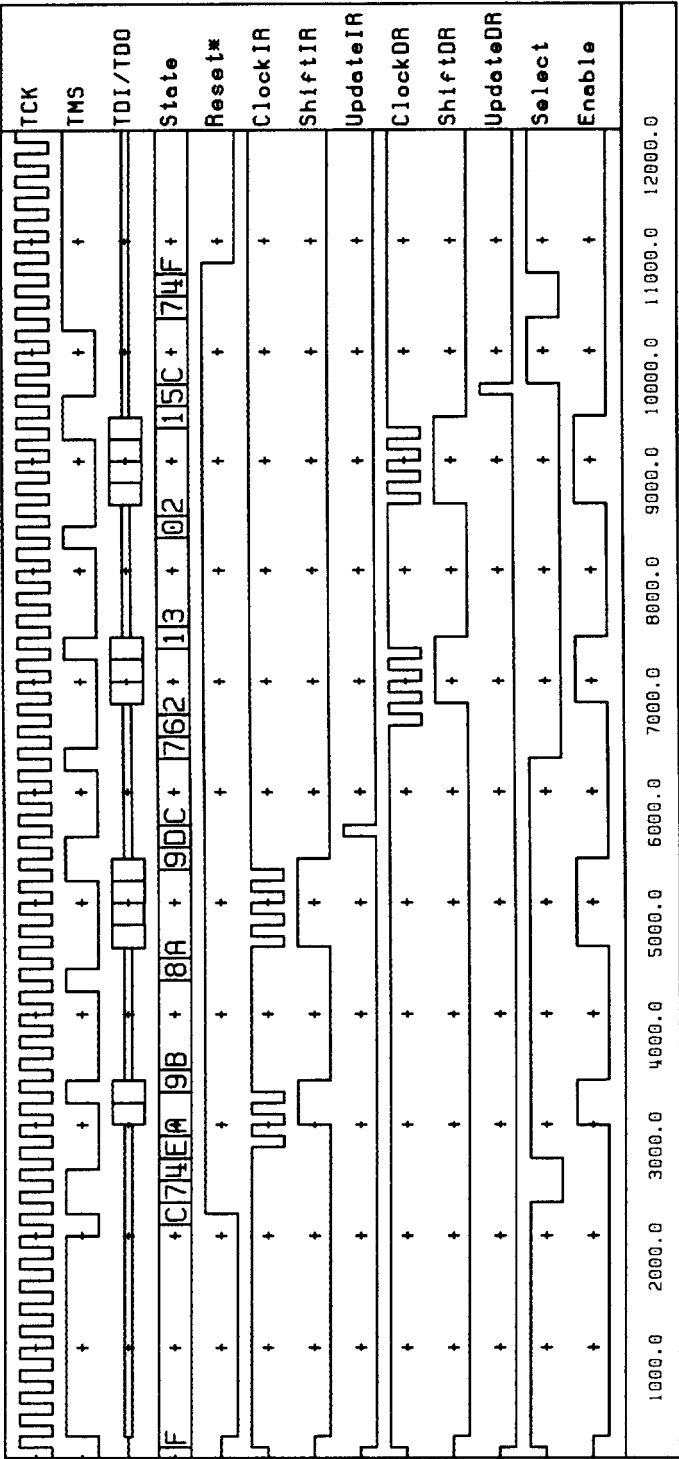


Figure 5-7 — Operation of the Example TAP Controller