IEEE Std 1149.1-1990
(Includes IEEE Std1149.1a-1993)

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

ISBN 1-55937-350-4

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 Pl149.1 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	Patrick F. McHugh	Gordon D. Robinson
Vassilios Gerousis	Johann Maierhofer	Robert J. Russell
Grady L. Giles	Math N. M. Muris	William H. Smith
Charles L. Hudson	Dieter Ohnesorge	Michael Tchou
Curtis Jensen	Stig Oresjo	Lee Whetsel
Dirk van de Lagemaat		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	Matthias Kaufmann	David L. Rutledge
Paul Bardell	William McAnney	Wim Sauerwald
Frans P. M. Beenker	Bruno Mueller	Erwin Trischler
Harry Bleeker	Paul Ocampo	Jon L. Turino
William C. Bruce	Anwar Osseyran	Chantal Vivier
Ray Chapman	Kenneth P. Parker	R.G. Walther
Peter Fleming	Michel Parot	H. Whittemore
Walter Ghisler	David Richards	Chi W. Yau
Najmi Jarwala	Derek Roskell	Mark Zagorin

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

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:

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:

The following persons were on the balloting committee:

Gerald F. Adams
Gordon Adshead
Robert Aitken
John Andrews
William Atwell
Chris Bagge
Behrooz Bandall
Paul Bardell
James Beausang
Elizabeth Benedict
R. G. Bennetts
Dilip K. Bhavsar
Sol Black
Harry Bleeker
Federico Bonzanigo
Melvin Breuer
Alan Bridges
Bill Bruce
Barry C. Bucher
Michael L. Bushnell
Terence Chan
R. Chandramouli
Kedong Chao
Tony Cheng
Wilton Chiles
John Chin
Michael Collins
R. Dandapani
Michael Donahue
David G. Edwards
Leo G. Egan
Chris Ellingham
Peter Fleming
Vassilios Gerousis
N. Gianbiasi
Grady L. Giles
Steven Gonet
A. J. van de Goor
Dong S. Ha
Peter L. Hansen
Vance Harwood
Abu Hassan
Kazumi Hatayama
Tord Haulin
Wei-Cheng Her
David Hickey
Hezi Himelfarb
Chuck Hudson
Axel Hunger
Masaaki Inadachi
Neal Jaarsma
Kamran Jamal
Najmi Jarwala
Curtis Jensen
Carl Kagy

B. Kaminska
Nick Kanopoulos
Jake Karrfalt
David C. Keezer
Satish Keshava
Russ Kneupper
Kevin Kornegay
Roger Kozlowski
Gunter Krampl
Thomas Kropf
Gerd Kruger
Thomas M. Kurihara
Steven K. Ladd
Dirk van de Lagemaat
David Landis
Bjorn B. Larsen
Robert Ledbetter
Wha-Joon Lee
Raymond Lee
Yann-Hang Lee
Jasopin Lee
Regis Leveugle
Marc Levitt
Adam W. Ley
Chih-Jen Lin
Antonio Lioy
Dan Loocke
G. Lousberg
Hede Ma
Rafic Makki
R. A. Mansfield
David Marquart
Colin M. Maunder
Patrick F. McHugh
David E. Medina
Earl Meiers
Michael Menovich
Donald Merlino
Joseph A. Mielke
Zeljko Miksic
Richard Miller
Steven D. Millman
Yinghua Min
John Moynihan
Tom Munns
Math N. M. Muris
Prawat Nagvajara
Zainalabedin Navabi
Roy H. Nesson
Arnold Nordsieck
Jim O'Reilly
Paul Ocampo
Dieter Ohnesorge
Folusho Oladeinde
Stig Oresjo

Calvin M. Osborne
Krishnan Palaniswami
Chris Papachristou
Kenneth P. Parker
Michel Parot
Michael Paulson
Bruce Peterson
Kenneth Posse
Paolo Prinetto
Lawrence Prucha
Jeffrey Quay
Rochit Rajsuman
Edward P. Ratazzi
Shishpal Rawat
Gordon D. Robinson
Derek Roskell
Ashim Roy
Robert J. Russell
Sergiu Samuel
Cayetano Sanchez
Dana Schow
Lori E. Schramm
Richard M. Sedmak
Ashok K. Sharma
Raj Sharma
Pradip K. Skimani
William H. Smith
Mani Soma
Arun Somani
Pradip K. Srimani
Bret A. Stewart
William Sullivan
Xiaoling Sun
Carlos Terrer
Jacques Tete
Cihan Tinaztepe
Mike Topsakal
Rodham E. Tulloss
Jon L. Turino
James Vanderwiele
Tom Volpe
Russell J. Wagner
K. D. Wagner
Malcolm Wallace
Gary Wang
Allen Warren
Thomas J. Wey
Lee Whetsel
Harry Whittemore
Brian Wilkins
David M. Wu
Cheng-Wen Wu
Stephen Yurash
Farzad Zarrinfar
Guenther Zwiehoff

CLAUSE PAGE

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.

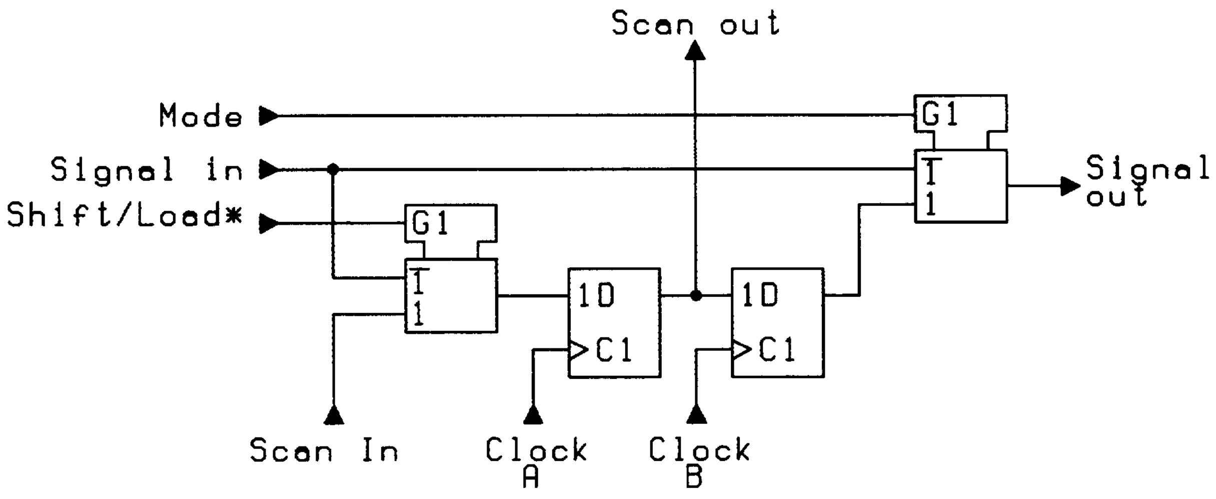


Figure 1-1—A Boundary-Scan Cell

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.

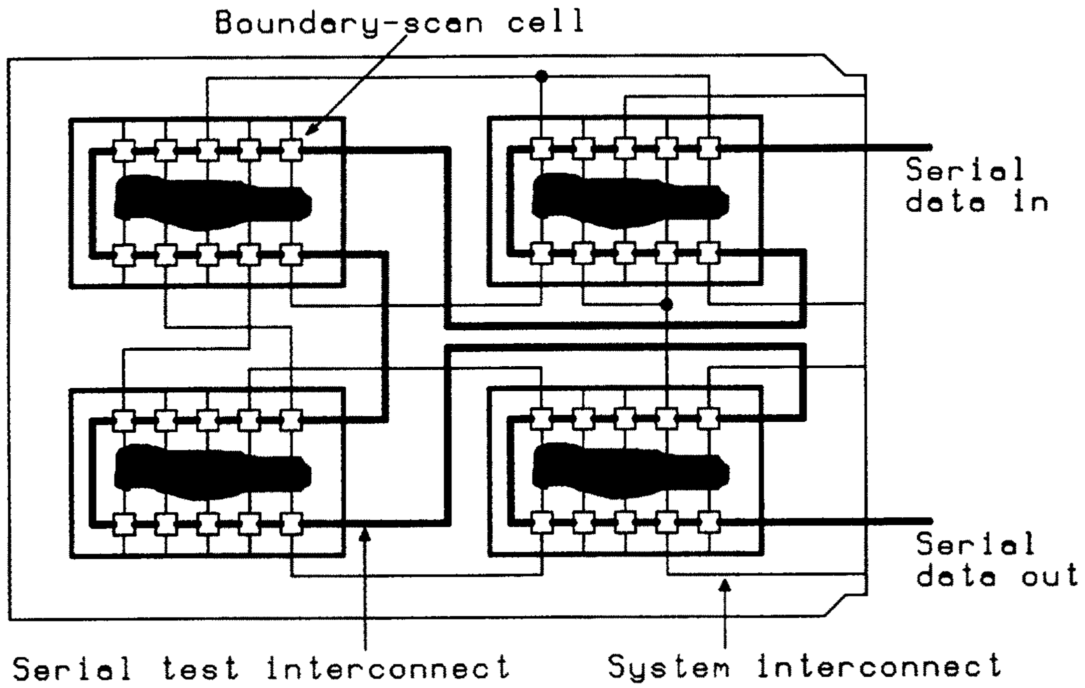


Figure 1-2—A Boundary-Scannable Board Design

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.

[1]Copies 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; and
b) A serial path formed by a daisy-chain connection of the serial test data pins (TDI and TDO).

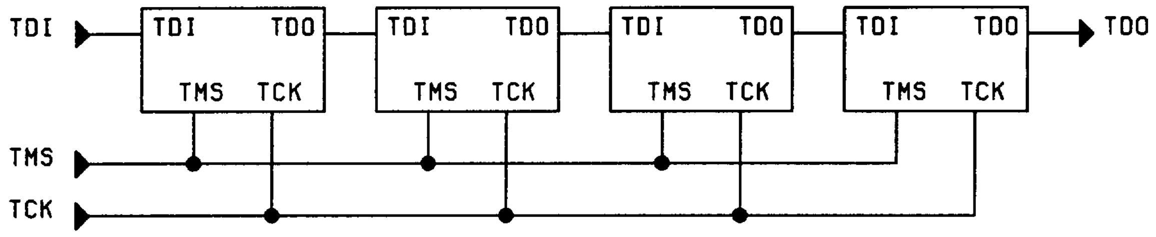


Figure 3-1—Serial Connection Using One TMS Signal

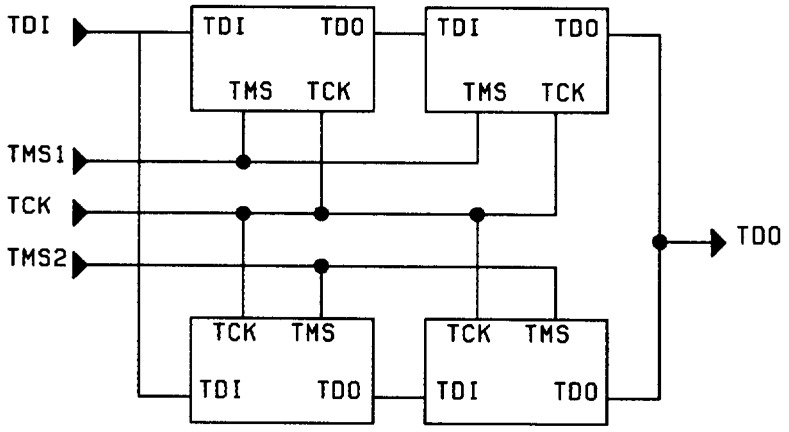


Figure 3-2—Connection in Two Paralleled Serial Chains

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.

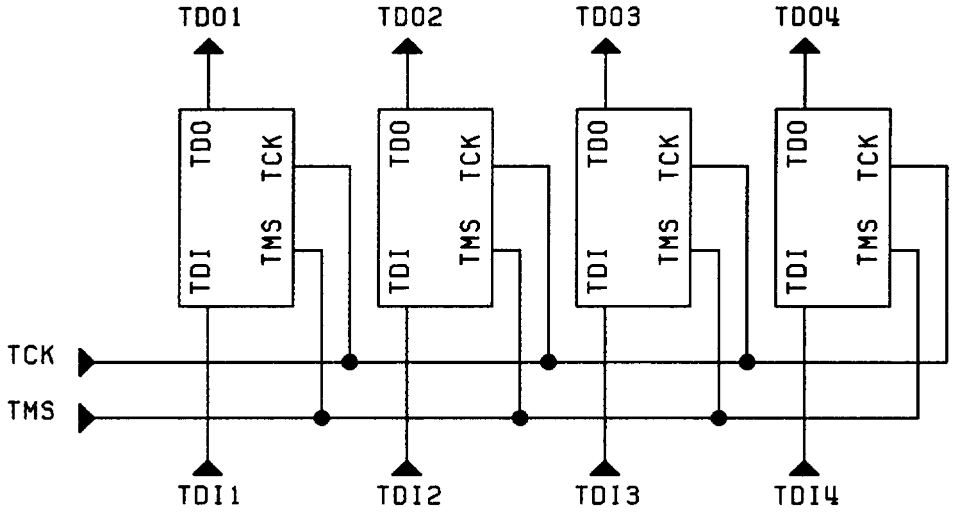


Figure 3-3—Multiple Independent Paths With Common TMS and TCK Signals

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 rally compliant with this standard.

4. Test Logic Architecture

This chapter defines the top-level design of the test logic accessed through the TAP. Detailed designrequirements 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.

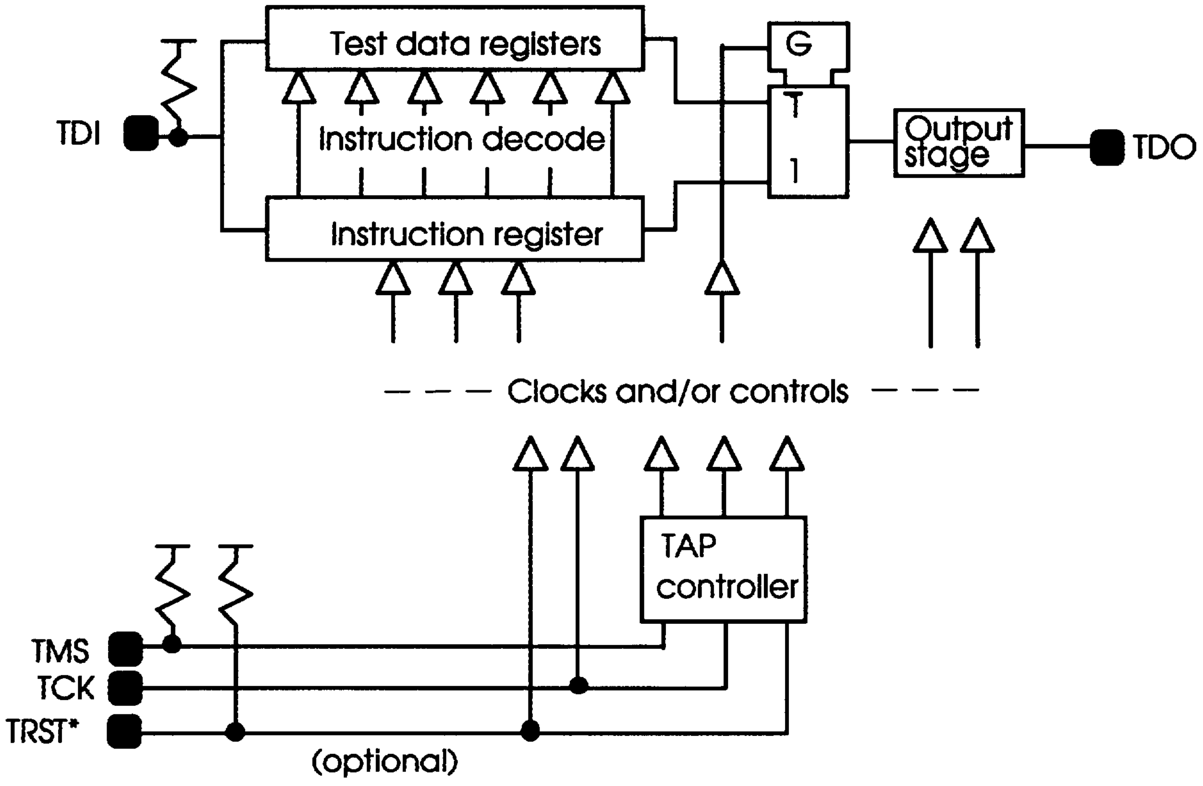


Figure 4-1—A Conceptual Schematic of the Test Logic

4.2 Test Logic Realization

4.2.1 Specifications

Rules

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, 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.

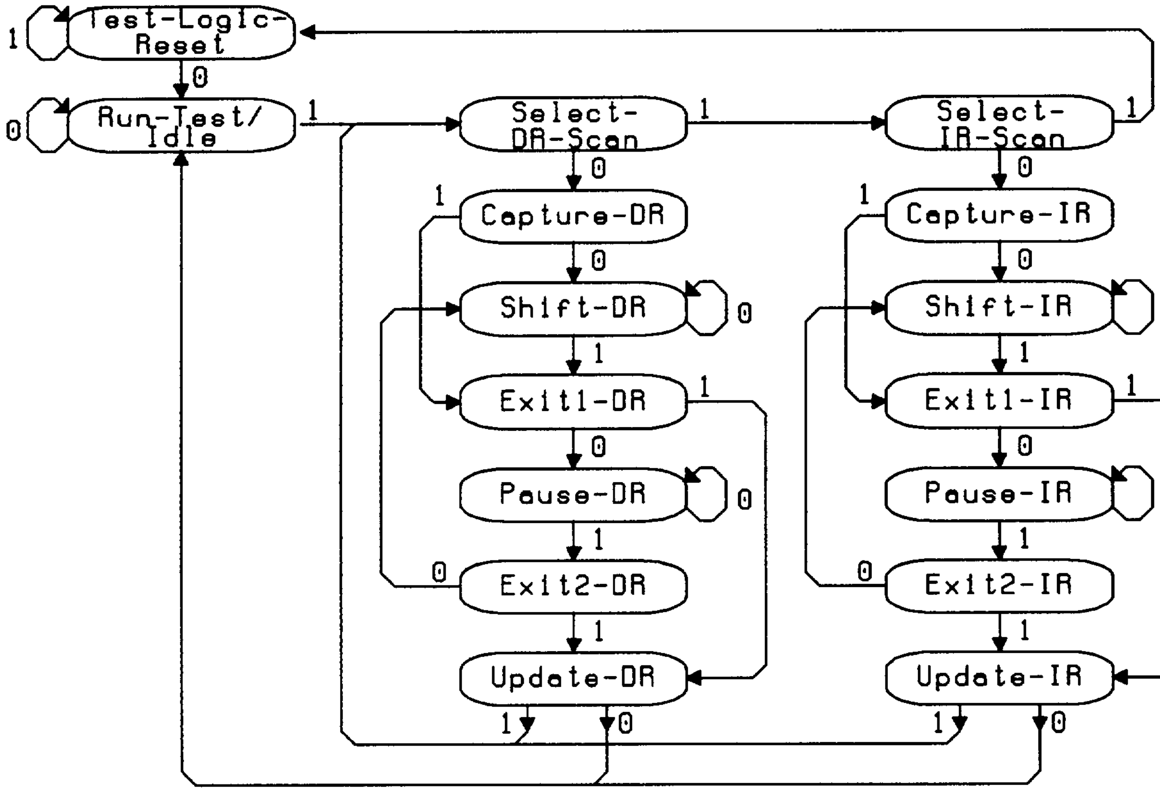


Figure 5-1—TAP Controller State Diagram

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.

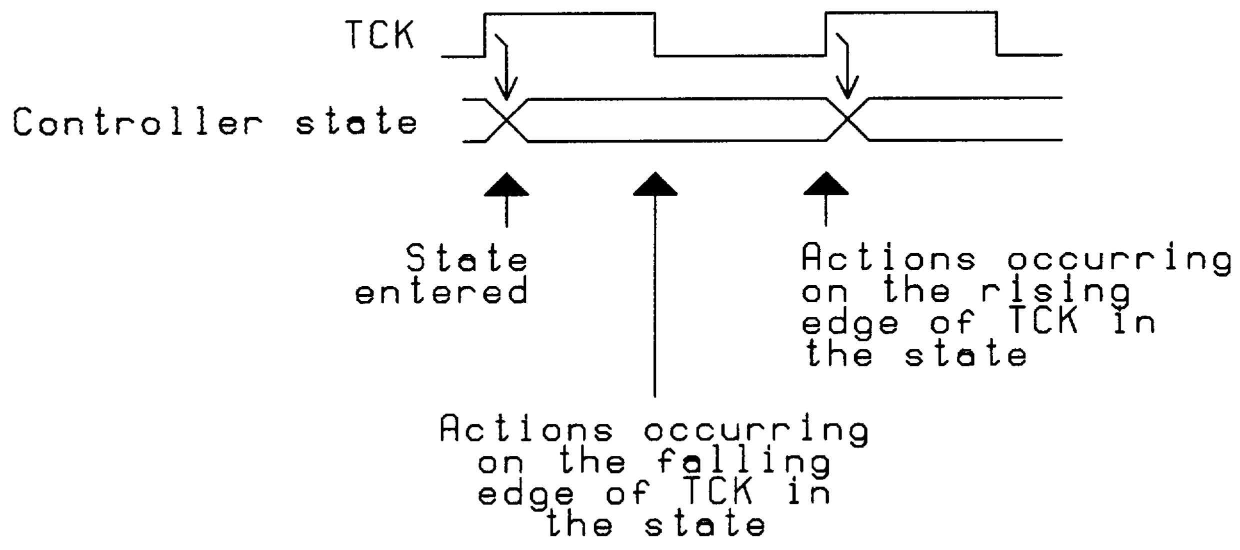

TCK
Controller state
State entered
Actions occurring on the rising edge of TCK in the state
Actions occurring on the falling edge of TCK in the state

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**		
	Update-DR	***Run-Test/Idle***	***Capture-DR***
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 *SAMPLE/ PRELOAD* 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.

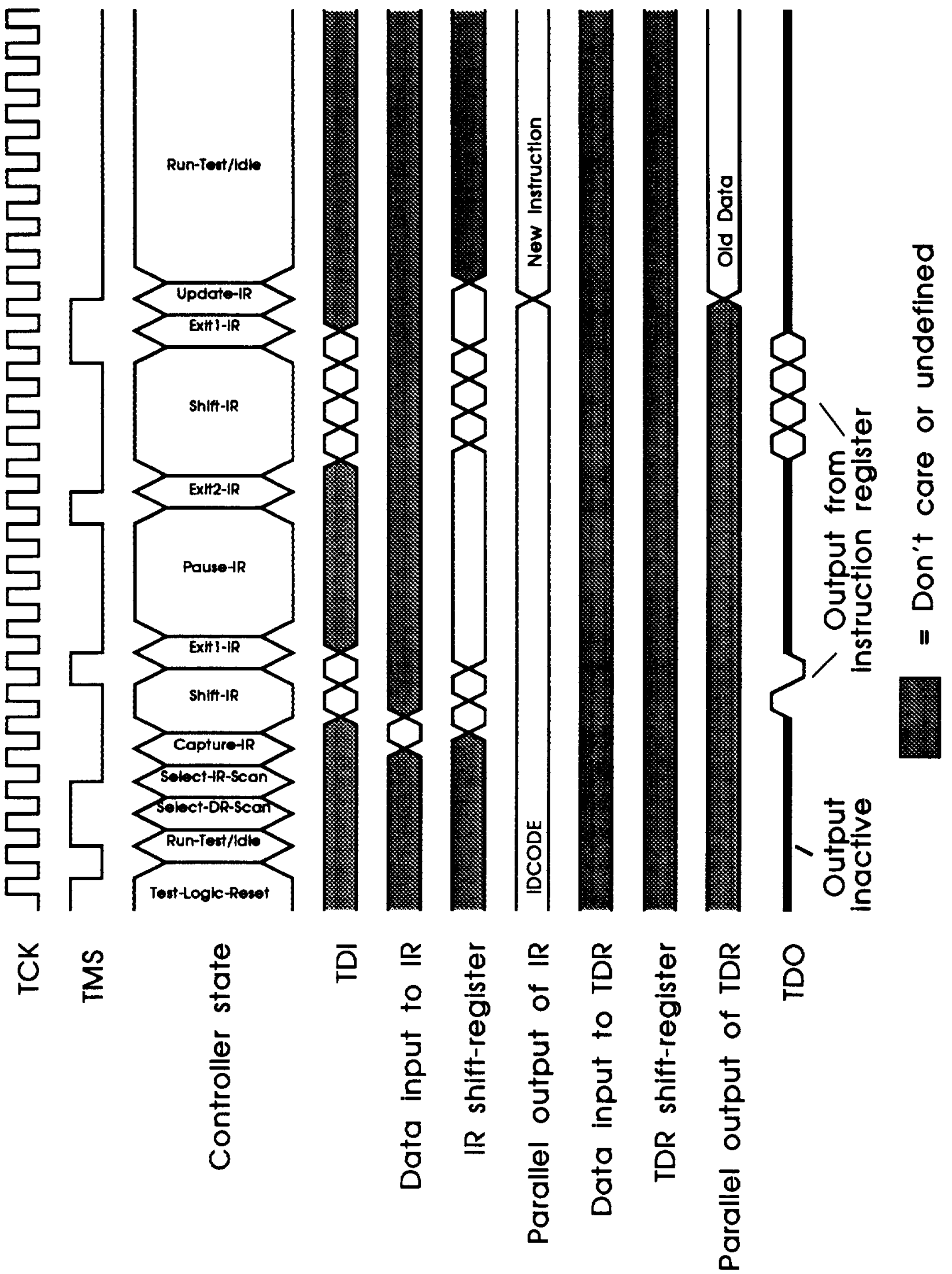


Figure 5-3—Test Logic Operation: Instruction Scan

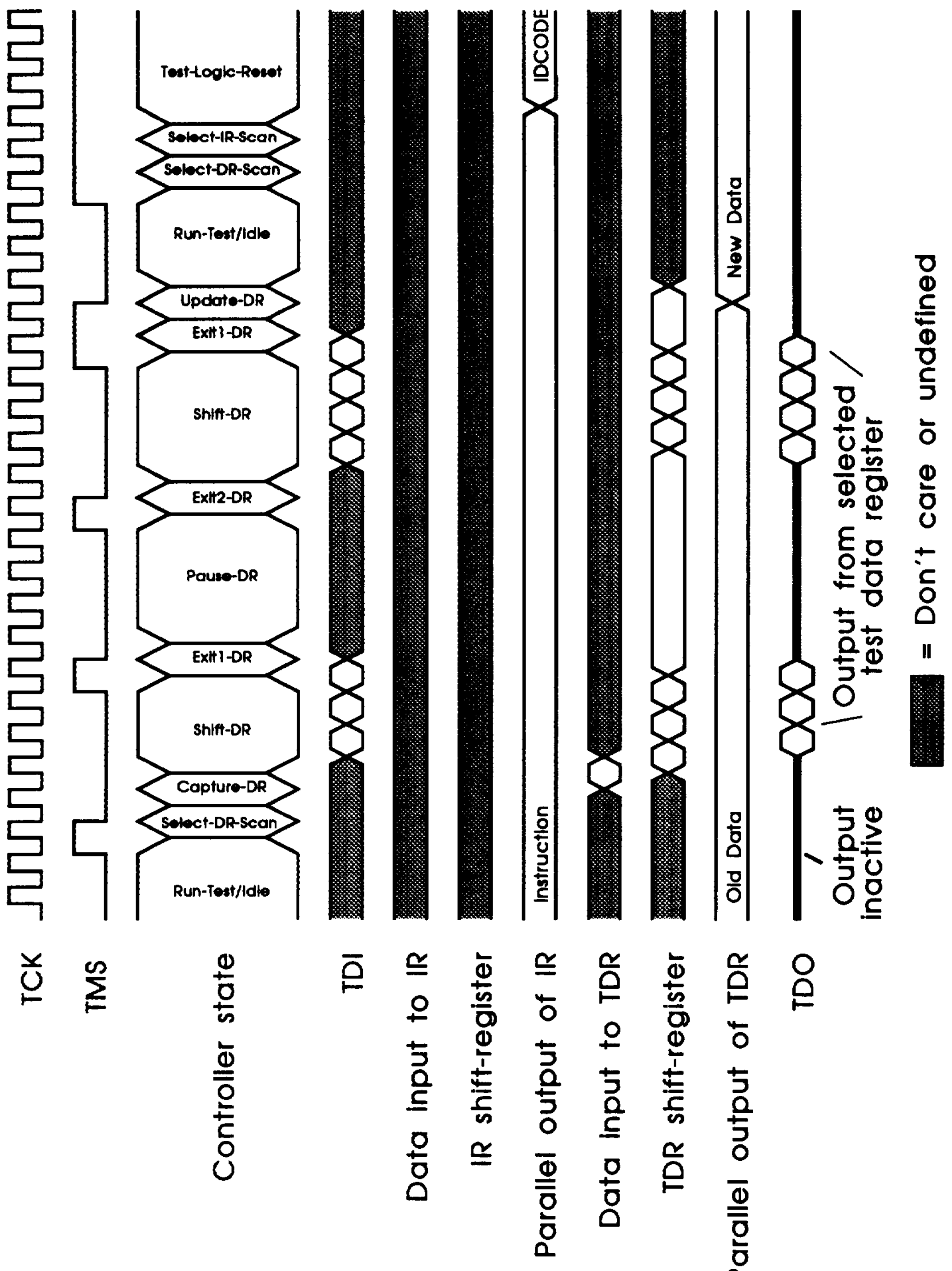


Figure 5-4—Test Logic Operation: Data Scan

Table 5-2—Test Logic Operation in Each Controller State

Controller State	Register Selected to Drive TDO	TDO Driver
Test-Logic-Reset	Undefined	Inactive
Run-Test/Idle	Undefined	Inactive
Select-DR-Scan	Undefined	Inactive
Select-IR-Scan	Undefined	Inactive
Capture-IR	Undefined	Inactive
Shift-IR	Instruction	Active
Exit1-IR	Undefined	Inactive
Pause-IR	Undefined	Inactive
Exit2-IR	Undefined	Inactive
Update-IR	Undefined	Inactive
Capture-DR	Undefined	Inactive
Shift-DR	Test data	Active
Exit1-DR	Undefined	Inactive
Pause-DR	Undefined	Inactive
Exit2-DR	Undefined	Inactive
Update-DR	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 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.

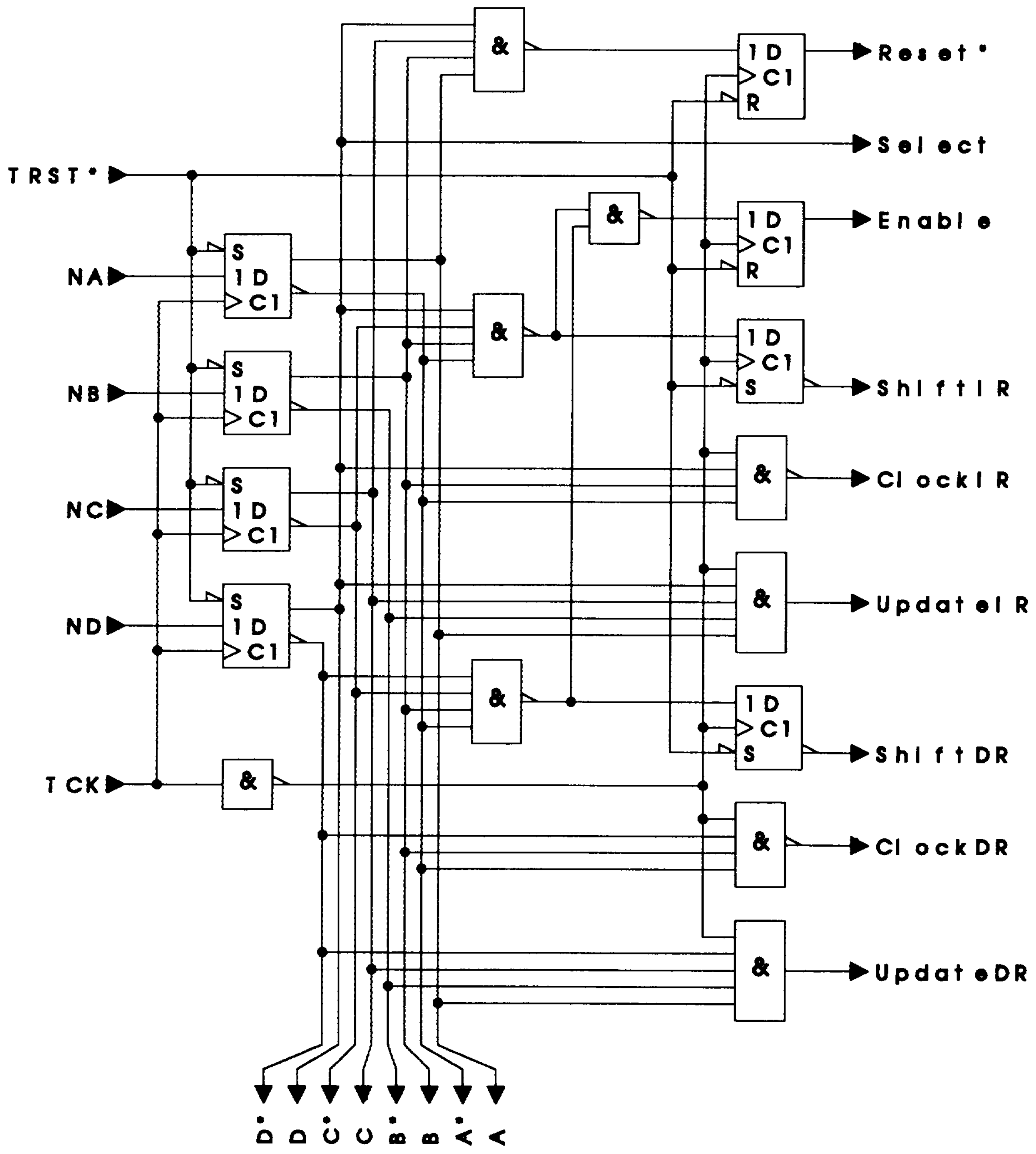


Figure 5-5—A TAP Controller Implementation—State Registers and Output Logic

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.

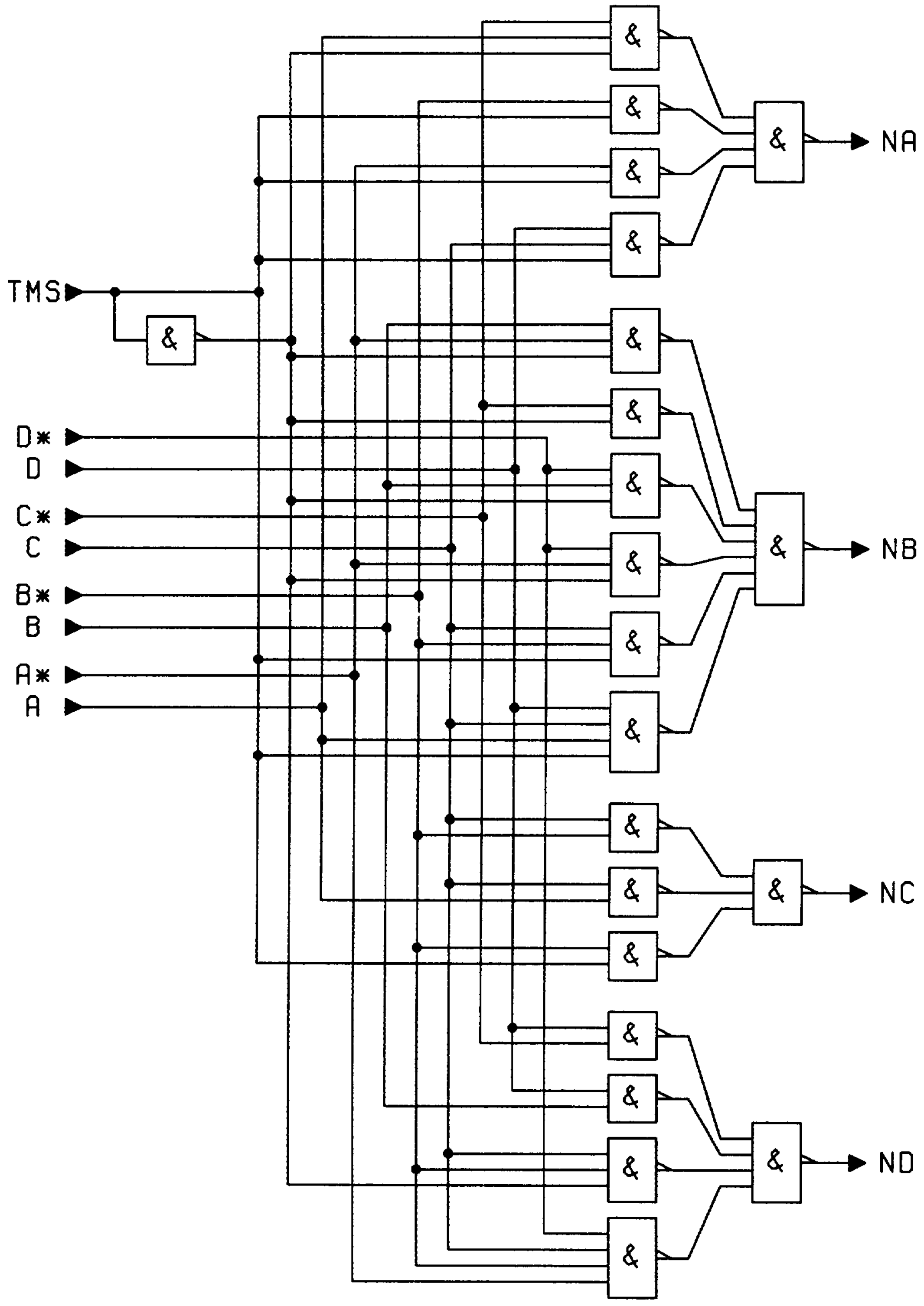


Figure 5-6—A TAP Controller Implementation—Next State Logic

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

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

```
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
```

where

T = value present at TMS

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

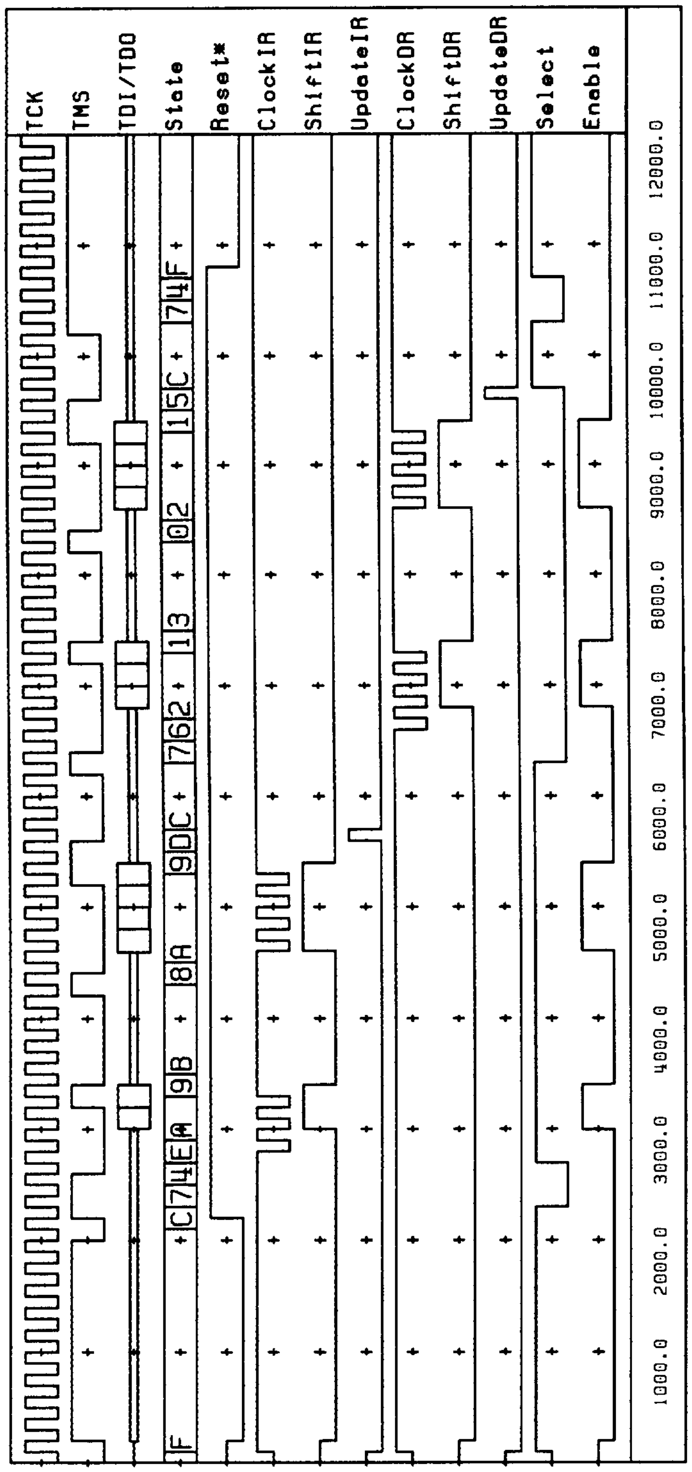


Figure 5-7—Operation of the Example TAP Controller

5.3 TAP Controller Initialization

5.3.1 Specifications

Rules

a) 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*, then 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* following enabling of compliance to this standard as described in 3.8, then the design of the assembled system has to ensure that a logic 0 is applied to TRST* when compliance is enabled.

b) The TAP controller shall not be initialized by operation of any system input, such as a system reset.

c) Where a dedicated reset pin (TRST*) is provided to allow initialization of the TAP controller, then 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, then the result shall be equivalent to that which would be achieved by application of a logic 0 to a TRST* input.

5.3.2 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 following 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 — Chapter 10 contains rules that ensure that boundary-scan circuitry at system pins does not interfere with system operation when the *Test-Logic-Reset* controller state is selected.

While the TAP controller will synchronously enter the *Test-Logic-Reset* controller state following five rising edges at TCK (provided 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 could also 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 may also be possible to use the independence of the system and test resets to allow sampling and examination of data following a system crash. This would require that the test logic be reset prior to 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 test logic, for example, as shown in figure 5-8.

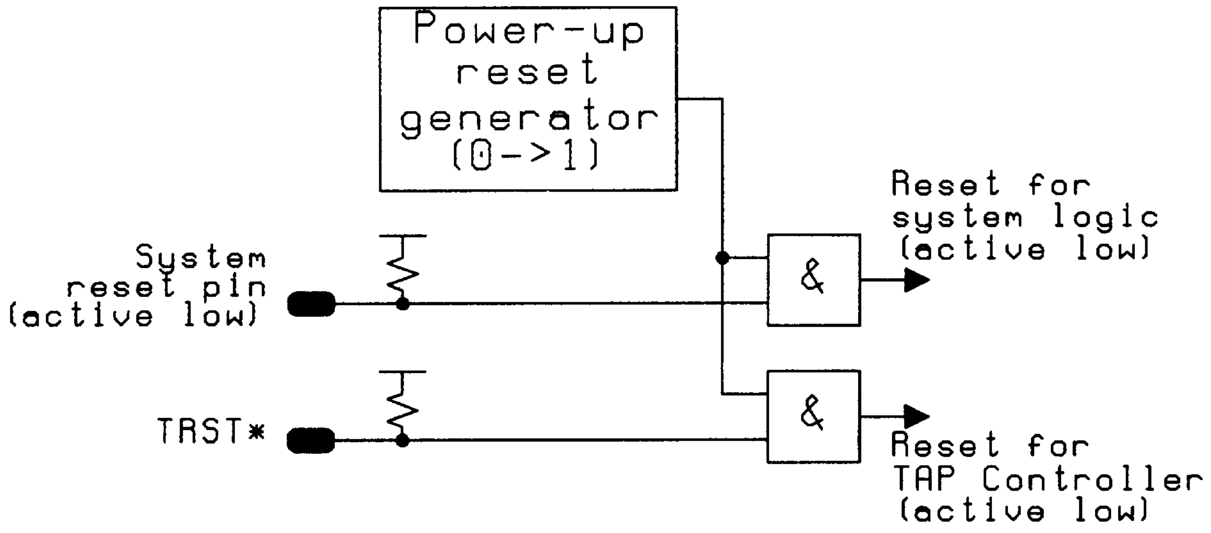

Power-up reset generator (0->1)
System reset pin (active low)
TRST*
&
&
Reset for system logic (active low)
Reset for TAP Controller (active low)

Figure 5-8—Use of Power-Up Reset for System and Test Logic

6. 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 Chapter 7, 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 chapter contains the design requirements for the instruction register.

6.1 Design and Construction of the Instruction Register

The instruction register is a shift-register-based design having 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.

6.1.1 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 6.2).
c) There shall be no inversion of data between the serial input and the aerial 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 6.2).
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.

6.1.2 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* or *Test-Logic-Reset* controller states. The timing and nature of these changes are discussed in detail in 6.2.

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 two configurations (*EXTEST* and *SAMPLE*/*PRELOAD*—see 7.2).

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 are designed to load a fixed logic value (0 or 1) during the *Capture-IR* controller state.

6.2 Instruction Register Operation

6.2.1 Specifications

Rules

a) The behavior of the instruction register in each TAP controller state shall be as defined in table 6-1.
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 following entry into a controller state.

Table 6-1—Instruction Register Operation in Each Controller State

Controller State	Shift-Register Stage	Parallel Output
Test-Logic-Reset	Undefined	Set to give the *IDCODE* (or *BYPASS*) instruction
Capture-IR	Load 01 into LSBs and, optionally, design-specific data or fixed values into MSBs	Retain last state
Shift-IR	Shift towards serial output	Retain last state
Exit1-IR Exit2-IR Pause-IR	Retain last state	Retain last state
Update-IR	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) Following 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*).

6.2.2 Description

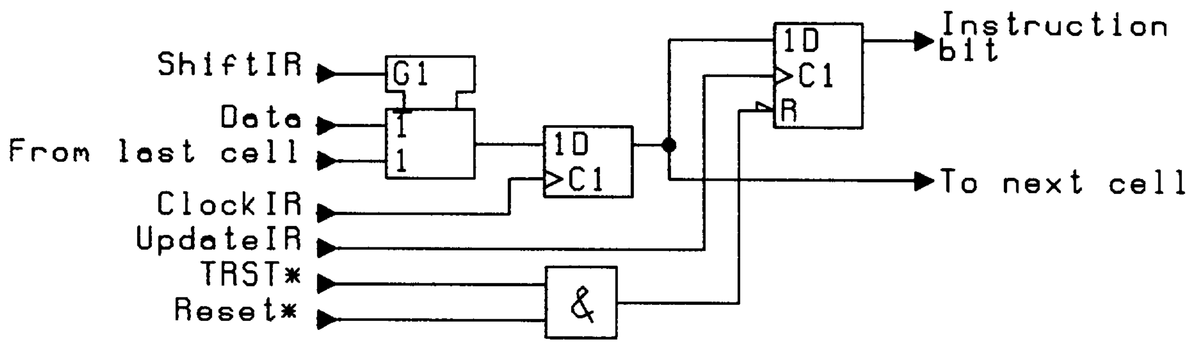


Figure 6-1—An Instruction Register Cell

NOTE — The parallel output flip-flop in this figure is provided with a reset input. To meet rules 6.2.1e and 6.2.1f, some or all instruction register cells will require use of a set, rather than a reset, input. A complete instruction register will contain at least one cell of each type, located as the two LSBs.

Figure 6-1 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 5.2:

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 Chapter 10. Note that in figure 6-1 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.
b) The clock input to the register in the serial path is only applied 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* or TRST* input of the cell. Referring to figures 5-5 and 5-6, 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*, 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 6.2.1e.
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 6.2.1b 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.

7. Instructions

The instruction register allows instructions to be serially entered into the test logic during an instruction-register scan cycle. This chapter 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.

7.1 Response of the Test Logic to Instructions

7.1.1 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 5-2).

d) Instruction codes that are not otherwise required to provide control of test logic shall be equivalent to the *BYPASS* instruction (see 7.4).

Permissions

e) 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.

7.1.2 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 8.2).

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 8.2).

Rule 7.1.1d 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 code.

7.2 Public Instructions

7.2.1 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 7.4, 7.6, and 7.7 respectively.)
c) If the optional device identification register is included in a component, then 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* and *EXTEST* instructions shall be as defined in 7.4 and 7.7.

Recommendations

f) It is recommended that products support either the *INTEST* or the *RUNBIST* instruction or both. (See 7.8 and 7.9.)

Permissions

g) A design may offer public instructions in addition to those defined in this standard to allow 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.

7.2.2 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, board interconnect test via the boundary-scan register, etc. 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 7.12).

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.

7.3 Private Instructions

7.3.1 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 codes that, if selected, would cause hazardous operation of the component.

7.3.2 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 codes (binary values) that may cause hazardous operation if used by the component purchaser.

7.4 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.

7.4.1 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 7.4.1b.

7.4.2 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 are consequently 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 following 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.

7.5 Boundary-Scan Register Instructions

As discussed in Chapter 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 clause 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 7.6 to 7.10. Requirements for the design of boundary-scan register cells are contained in Chapter 10.

7.5.1 An Overview of the Operation of the Boundary-Scan Register

The boundary-scan register is a shift-register based structure comprising 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 7-1.

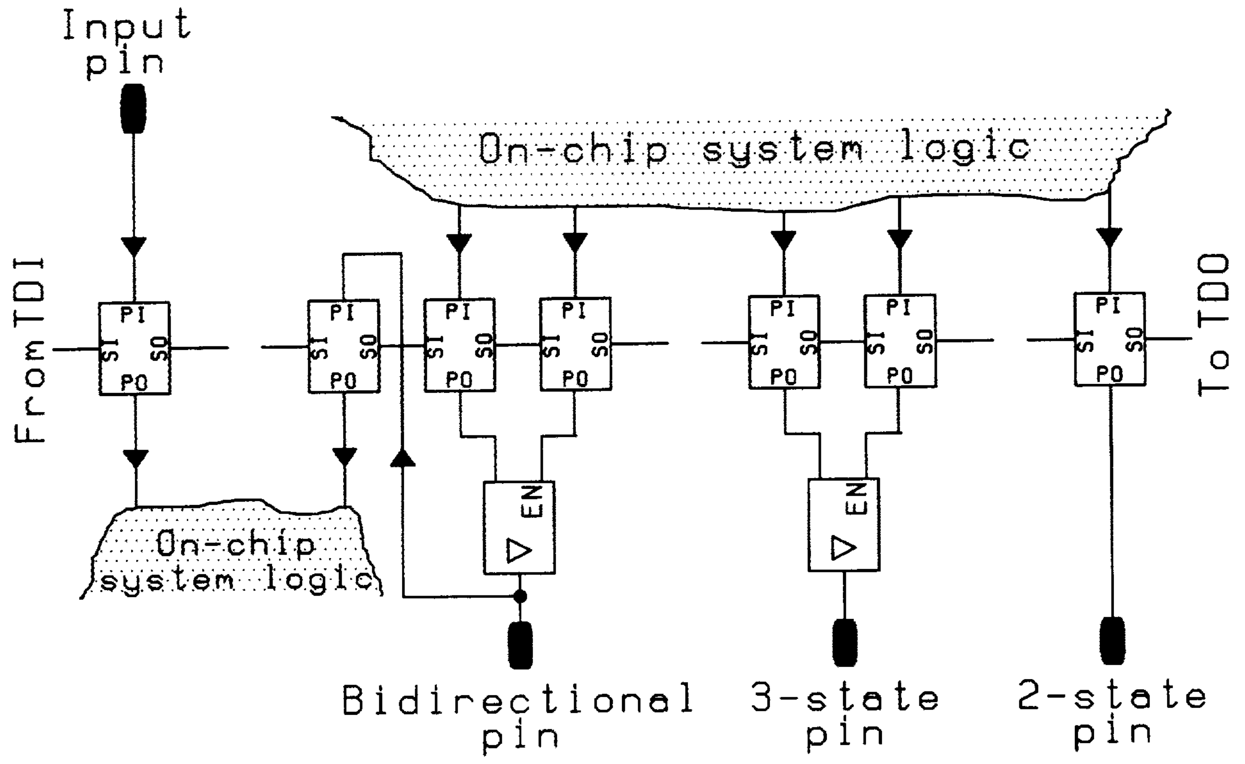


Figure 7-1—A Simplified View of the Boundary-ScanRegister

An example implementation for a cell that could be used in each of the locations shown in figure 7-1 is given in figure 7-2.

The connections labeled PI, PO, SI, and SO in figure 7-2 are connected to adjacent cells, the on-chip system logic, and the system pins as shown in figure 7-1. Like all the cells shown in this standard, that shown in figure 7-2 is designed to respond to the ClockDR, ShiftDR, and UpdateDR signals generated by the example TAP controller implementation shown in figures 5-5 and 5-6. 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 Chapter 10, other cell designs are possible that meet the requirements of this standard for different sets of instructions. For example:

a) R2 may be either a flip-flop (as shown) or a latch.
b) 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.
c) If the *INTEST* instruction were not supported, R2 and M2 would be omitted from cells that feed data from a system pin to the on-chip system logic. The input labeled PI would then be directly connected to the output labeled PO.

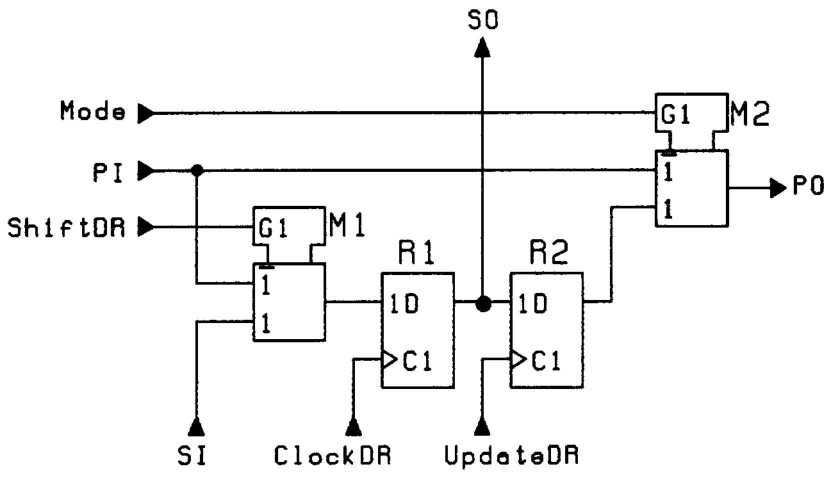


Figure 7-2—An Example Boundary-Scan Cell Design

7.5.2 Specifications for Boundary-Scan Register Instructions

The specifications for boundary-scan instructions given in the following clauses of this chapter define:

a) Whether the instruction is mandatory or optional;
b) Which test data registers can be connected in the serial path between TDI and TDO;
c) 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
d) The flow of data between the component's system pins, the boundary-scan register cells, and the on-chip system logic.

The specifications are supported by descriptive text that includes a version of figure 7-3 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.

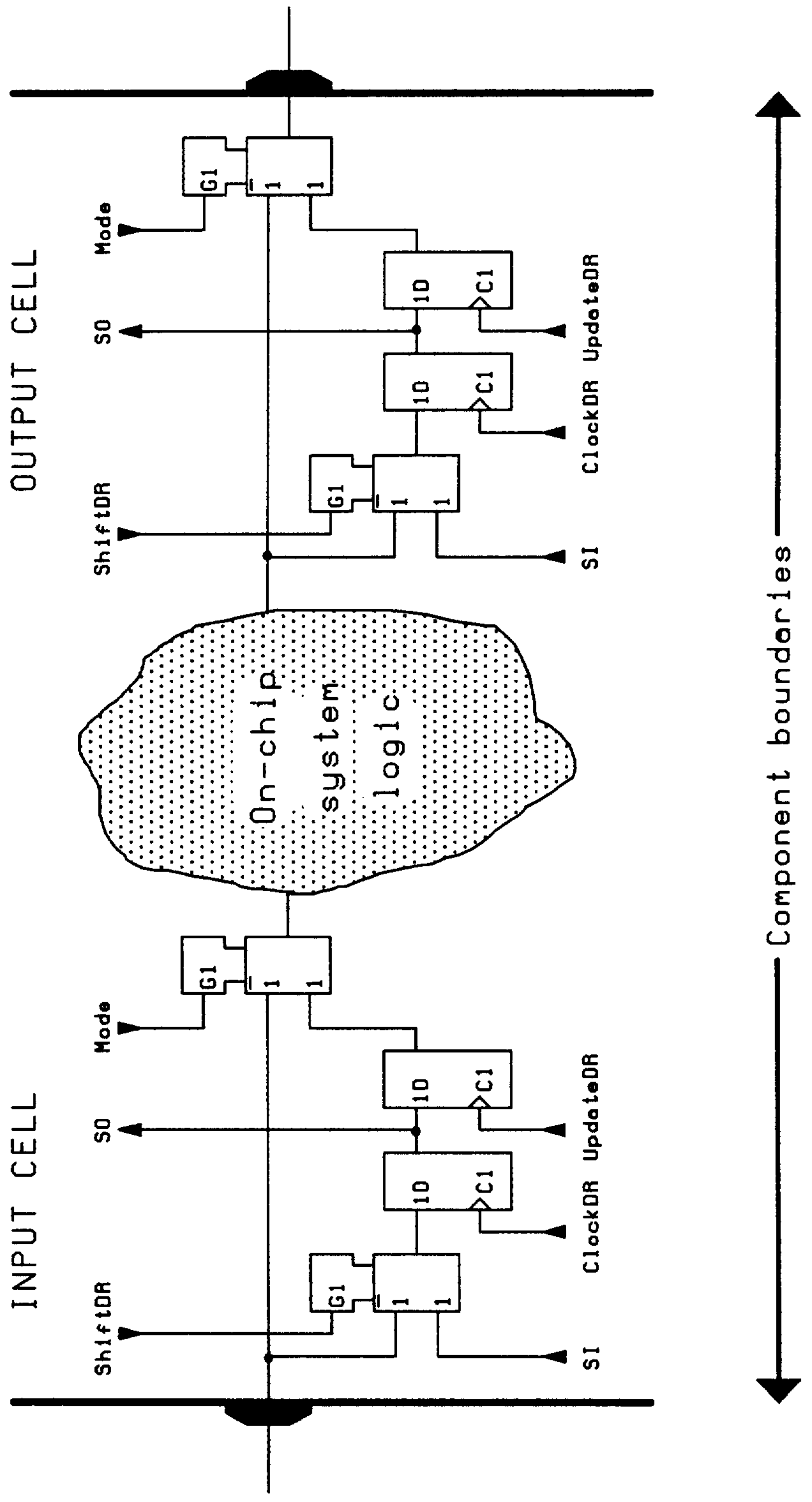


Figure 7-3—Circuit Used to Illustrate Boundary-Scan Instructions

7.6 The SAMPLE/PRELOAD Instruction

The mandatory *SAMPLE/PRELOAD* instruction allows a snapshot of the normal operation of the component to be taken and examined. It also allows data values to be loaded onto the latched parallel outputs of the boundary-scan shift register prior to selection of the other boundary-scan test instructions.

7.6.1 Specifications

Rules

a) Each component shall provide a *SAMPLE/PRELOAD* instruction.
b) The *SAMPLE/PRELOAD* 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).
c) When the *SAMPLE/PRELOAD* instruction is selected, the operation of the testlogic 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.
d) When the *SAMPLE/PRELOAD* instruction is selected, the states of all signals flowing 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.
e) When the *SAMPLE/PRELOAD* instruction is selected, parallel output registers/latches included in boundary-scan register cells shall load the data held in the associated shift-register stage on the falling edge of TCK in the *Update-DR* controller state.

Permissions

f) The binary value(s) for the *SAMPLE/PRELOAD* instruction may be selected by the component designer.

7.6.2 Description

The *SAMPLE/PRELOAD* instruction is used to allow scanning of the boundary-scan register without causing interference to the normal operation of the on-chip system logic. Data received at system input pins is supplied without modification to the on-chip system logic; data from the on-chip system logic is driven without modification through the system output pins; etc. For the example boundary-scan cell design given in figure 7-2, this is achieved by holding the Mode input at 0 when the *SAMPLE/PRELOAD* instruction is selected. As suggested by the instruction's name, two functions can be performed by use of the *SAMPLE/PRELOAD* instruction:

a) *SAMPLE* allows a snapshot to be taken of the data flowing from the system pins to the on-chip system logic or vice versa, without interfering with the normal operation of the assembled board. The snapshot is taken on the rising edge of TCK in the *Capture-DR* controller state, and the data can then be viewed by shifting through the component's TDO output. Example applications of the *SAMPLE* capability are:
   i) To provide an analog to the guided-probing process performed on an assembled board during functional test diagnosis, but without the need for physical contact; and
   ii) Verification of the interaction between components during normal functional operation.

   The flow of data for the *SAMPLE* phase of the *SAMPLE/PRELOAD* instruction is shown in figure 7-4.
b) *PRELOAD* allows an initial data pattern to be placed at the latched parallel outputs of boundary-scan register cells (e.g., as provided in the cells connected to system output pins) prior to selection of another boundary-scan test operation. For example, prior to selection of the *EXTEST* instruction, data can be loaded onto the latched parallel outputs using *PRELOAD*. As soon as the *EXTEST* instruction has been transferred to the parallel output of the instruction register, the preloaded data is driven through the system output pins. This ensures that known data, consistent at the board level, is driven immediately when the *EXTEST* instruction is entered; without *PRELOAD* indeterminate data would be driven until the first scan sequence had been completed. The flow of data for the *PRELOAD* phase of the *SAMPLE/PRELOAD* instruction is shown in figure 7-5.

The shifting of data for the *SAMPLE* and *PRELOAD* phases can occur concurrently when required—that is, while data captured is shifted out, the preload data can be shifted in.

Note that by moving the TAP controller through the sequence *Capture-DR* → *Exit1-Dr* → *Update-DR* while the *SAMPLE/PRELOAD* instruction is selected, the state of the signals flowing into and out of the on-chip system logic at the time of sampling can be loaded onto the latched parallel output of the boundary-scan shift register.

7.7 The EXTEST Instruction

The mandatory *EXTEST* instruction allows testing of off-chip circuitry and board level interconnections. Data would typically be loaded onto the latched parallel outputs of boundary-scan shift-register stages using the *SAMPLE/PRELOAD* instruction prior to selection of the *EXTEST* instruction.

NOTE — Following use of the *EXTEST* instruction, the on-chip system logic may be in an indeterminate state that will persist until a system reset is applied. Therefore, the on-chip system logic may need to be reset on return to normal (i.e., nontest) operation.

7.7.1 Specifications

Rules

a) Each component shall provide an *EXTEST* instruction.

b) A binary code for the *EXTEST* instruction shall be {000...0} (i.e., a logic 0 is loaded into every instruction register cell).

c) The *EXTEST* 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).

d) While the *EXTEST* instruction is selected, the on-chip system logic shall be controlled such that it cannot be damaged as a result of signals received at the system input or system clock input pins.

NOTE — This might be achieved by placing the on-chip system logic in a reset or "hold" state while the *EXTEST* instruction is selected.

e) When the *EXTEST* instruction is selected, the state of all signals driven from system output pins shall be completely defined by the data held in the boundary-scan register and change only on the falling edge of TCK in the *Update-DR* controller state.

f) When the *EXTEST* instruction is selected, the state of all signals received at system input pins shall be loaded into the boundary-scan register on the rising edge of TCK in the *Capture-DR* controller state.

Recommendations

g) The data loaded into boundary-scan register cells located at system output pins (2-state, 3-state, or bidirectional) in the *Capture-DR* controller state when the *EXTEST* instruction is selected should be independent of the operation of the on-chip system logic.

h) A value should be defined for each boundary-scan register cell that, when the *EXTEST* instruction is selected, will permit all component outputs to be overdriven simultaneously for an indefinite period without risk of damage to the component.

NOTE — This is easily achieved if all outputs can be set to an inactive drive state by previous use of the *SAMPLE/PRELOAD* instruction.

Permissions

i) The *EXTEST* instruction may have binary codes in addition to that defined in rule 7.7.1b.

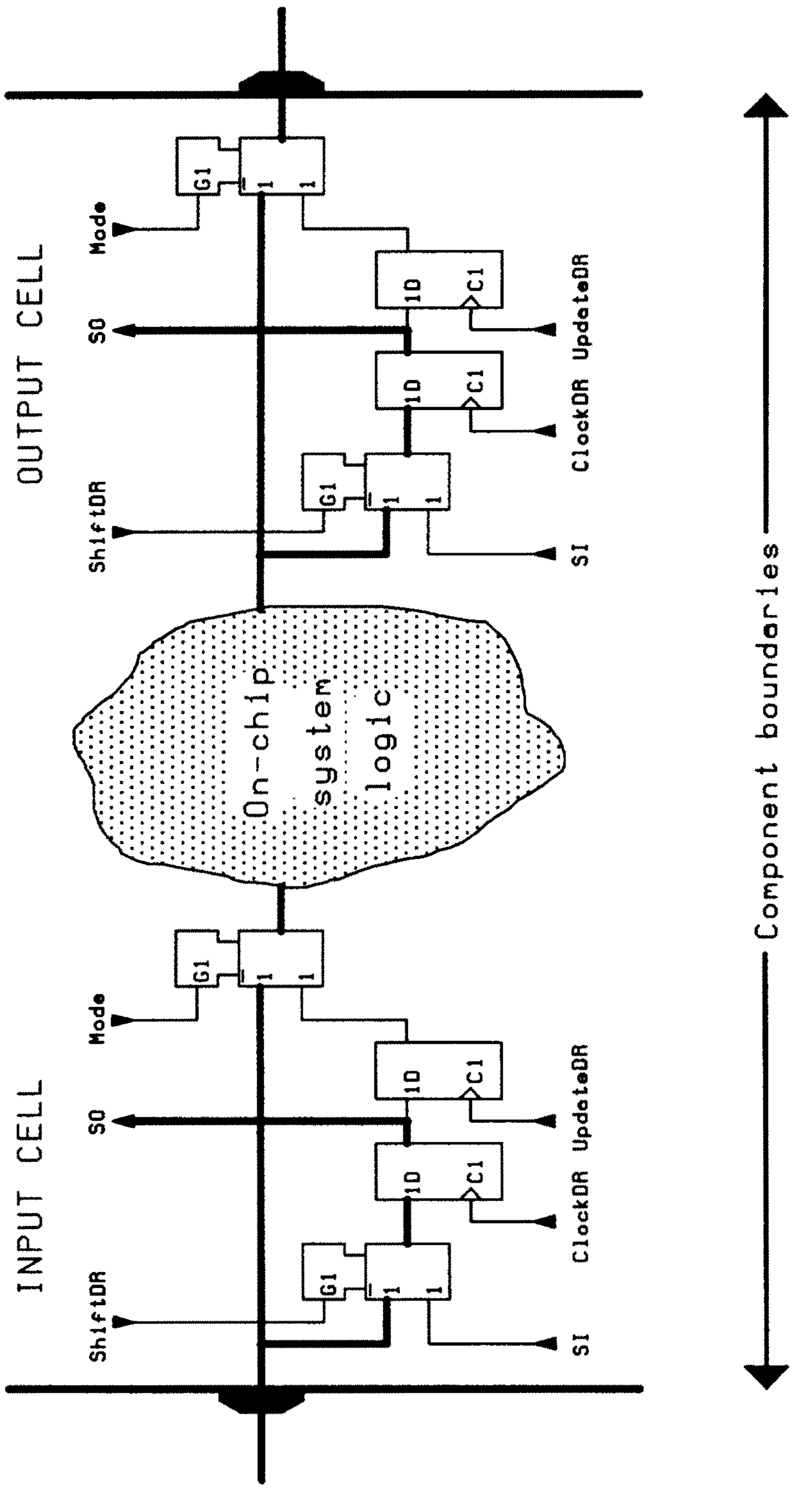


Figure 7-4—Data Flow for the SAMPLE Phase of the SAMPLE/PRELOAD Instruction

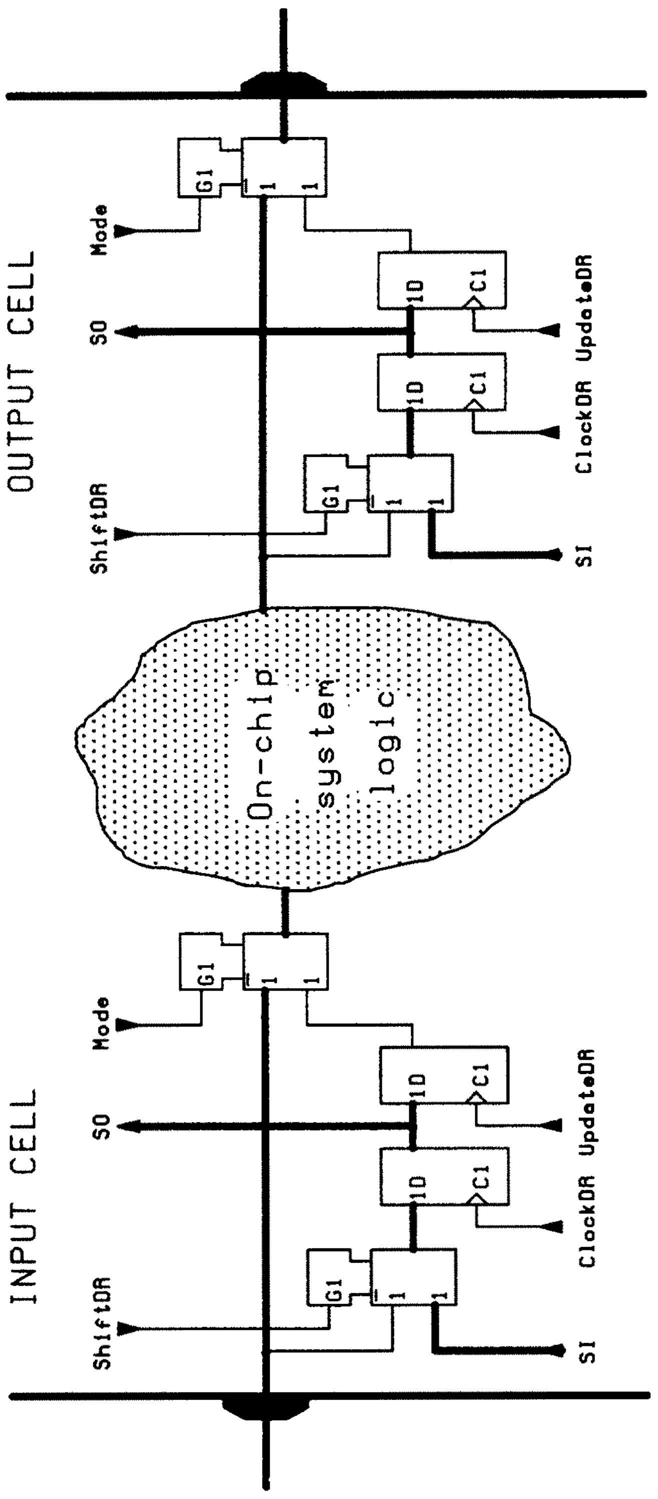


Figure 7-5—Data Flow for the PRELOAD Phase of the SAMPLE/PRELOAD Instruction

7.7.2 Description

The *EXTEST* instruction allows circuitry external to the component package—typically the board interconnect—to be tested. Boundary-scan register cells at output pins are used to apply test stimuli, while those at input pins capture test results. This instruction also allows testing of blocks of components that do not themselves incorporate boundary-scan registers. The flow of data through the boundary-scan register cells in this configuration is shown in figure 7-6. For example, at input pins data is first captured into the shift-register path and then shifted out of the component for examination; at output pins, data shifted into the component is applied to the external interconnection.

Typically, the first test stimulus to be applied using the *EXTEST* instruction will be shifted into the boundary-scan register using the *SAMPLE/PRELOAD* instruction. Thus, when the change to the *EXTEST* instruction takes place in the *Update-IR* controller state, known data will be driven immediately from the component onto its external connections. Where a total of *N* tests are to be applied using the *EXTEST* instruction, stimuli for tests 2 to *N* will be shifted in while the results from tests 1 to *N*-1 are shifted out. Note that, while the results from the final test—test *N*—are shifted out, a determinate set of data shall be shifted in that will leave the board in a consistent state at the end of the shifting process. This can be achieved by shifting the stimuli for test *N* (or indeed any other test) into the boundary-scan register again.

The *EXTEST* instruction also allows component outputs to be set to a state that minimizes the risk of damage when overdriven during in-circuit testing (see recommendation 7.7.1h). Such testing may be used where not all components on an assembled board are testable via boundary-scan.

Note that the boundary-scan register cells located at input pins may optionally be designed to allow signals to be driven into the on-chip system logic when the *EXTEST* instruction is selected. This allows user-defined values to be established at the system logic inputs, preventing misoperation in response to noise signals arriving from the board-level interconnect. The values driven may either be constant for the duration that *EXTEST* is selected (e.g., by including a blocking gate at the input to the system logic) or they may be loaded serially through the boundary-scan register, as shown in figure 7-6.

The *EXTEST* instruction can be entered by holding TDI at a constant low value and completing an instruction-scan cycle of sufficient duration to fill each instruction register on the board-level serial data path. The demands on the host test system are consequently reduced.

Recommendation 7.7.1g, where followed, ensures that data shifted out of the component in response to the *EXTEST* instruction is not altered by the presence of faults in the on-chip system logic. This simplifies diagnosis since any errors in the output bit stream can only be caused by faults in off-chip circuitry, in board-level interconnections, or in the boundary-scan registers used to apply the test.

While the *EXTEST* instruction is selected, the on-chip system logic may receive input signals that differ significantly from those expected during normal (nontest) operation. Rule 7.7.1d places the responsibility for correct handling of this situation on the component designer. If the on-chip system logic can tolerate any permutation of input signals that is received, then no specific design changes are required to meet this rule. (An example here would be the case where the on-chip system logic is entirely combinational.) However, for some components there may be input sequences that could place the on-chip system logic in a state where damage may result. In these cases, it is the responsibility of the designer to prevent the on-chip system logic from processing the "illegal" inputs while the *EXTEST* instruction is selected. As noted, this may be achieved by placing the on-chip system logic into a reset or "hold" state.

Alternatively, the data held in the boundary-scan register may be presented to the on-chip system logic while the *EXTEST* instruction is selected. Note that, where this is the case, rule 10.3.1e prohibits the imposition of any restriction on the logic values that may be driven to the on-chip system logic.

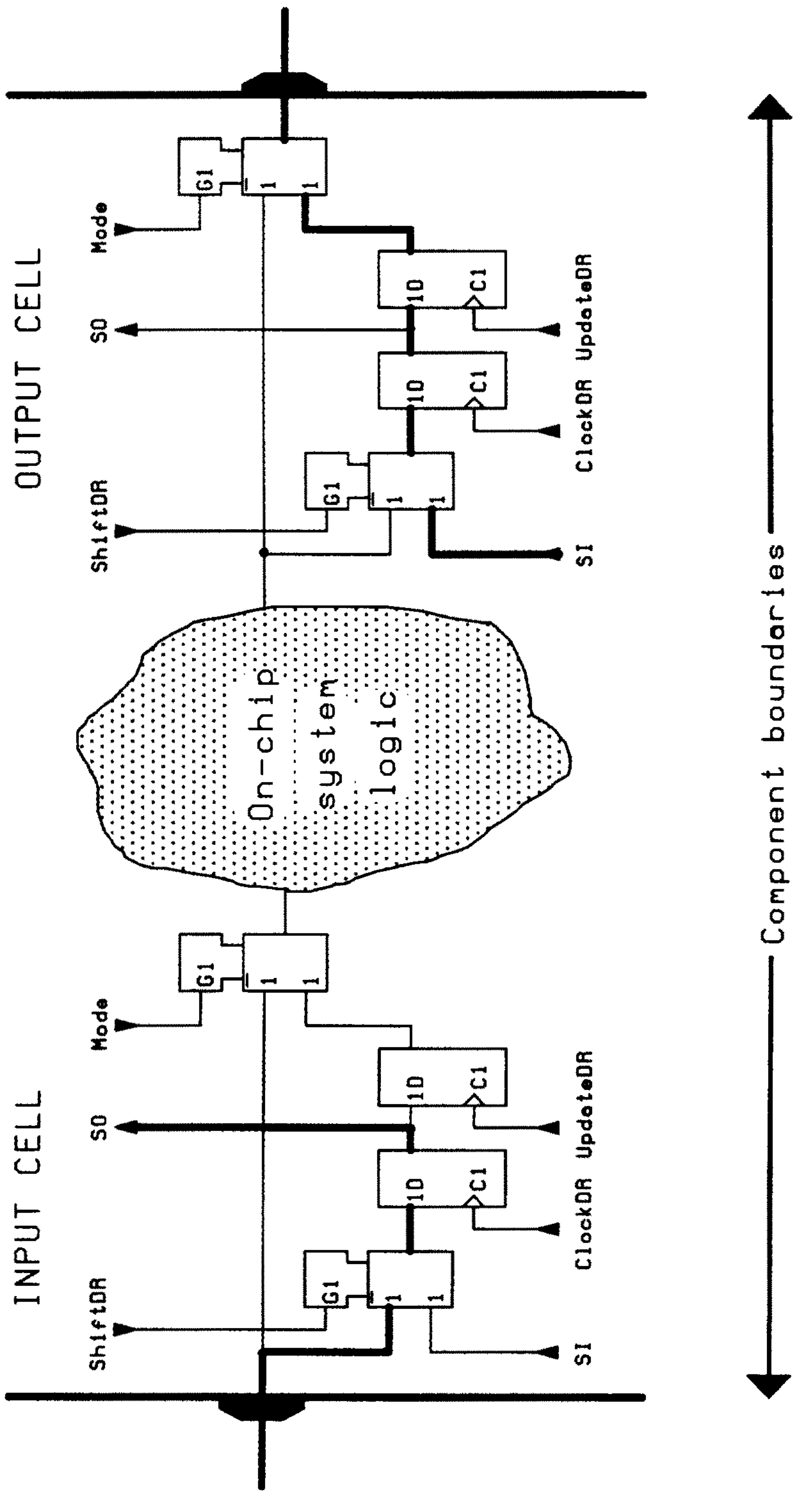


Figure 7-6—Test Data Flow While the EXTEST Instruction Is Selected

7.8 The INTEST Instruction

The optional *INTEST* instruction is one of two instructions defined by this standard that allow testing of the on-chip system logic while the component is assembled on the board. Using the *INTEST* instruction, test stimuli are shifted in one at a time and applied to the on-chip system logic. The test results are captured into the boundary-scan register and are examined by subsequent shifting. Data would typically be loaded onto the latched parallel outputs of boundary-scan shift-register stages using the *SAMPLE*/*PRELOAD* instruction prior to selection of the *INTEST* instruction.

The following rules apply where the *INTEST* instruction is provided.

NOTE — Following use of the *INTEST* instruction, the on-chip system logic may be in an indeterminate state that will persist until a system reset is applied. Therefore, the on-chip system logic may need to be reset on return to normal (i.e., nontest) operation.

7.8.1 Specifications

Rules

a) The *INTEST* 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).
b) The on-chip system logic shall be capable of single-step operation while the *INTEST* instruction is selected.
c) When the *INTEST* instruction is selected, all system outputs from the component shall be defined as follows:
   i) All signals driven out of the component shall be defined by data held in the boundary-scan register and shall change only on the falling edge of TCK in the *Update-DR* controller state or on selection of the *INTEST* instruction; or
   ii) All outputs from the component (including those that are 2-state nontest signals) shall be placed in an inactive drive state (e.g., high-impedance) on selection of the *INTEST* instruction.
d) When the *INTEST* instruction is selected, the state of all nonclock signals driven into the system logic from the boundary-scan register shall be completely defined by the data held in the register.
e) When the *INTEST* instruction is selected, the state of all signals output from the system logic to the boundary-scan register shall be loaded into the register on the rising edge of TCK in the *Capture-DR* controller state.

Recommendations

f) For boundary-scan register cells located at system input pins (clock or nonclock) or at bidirectional pins configured as inputs, the data loaded in the *Capture-DR* controller state when the *INTEST* instruction is selected should be independent of the operation of off-chip circuitry or board-level interconnections.

Permissions

g) The binary value(s) for the *INTEST* instruction may be selected by the component designer.

7.8.2 Description

The *INTEST* instruction allows static (slow-speed) testing of the on-chip system logic, with each test pattern and response being shifted through the boundary-scan register. The *INTEST* instruction requires that the on-chip system logic can be operated in a single-step mode, where the circuitry moves one step forward in its operation each time shifting of the boundary-scan register is completed.

The flow of data through the boundary-scan register cells while the instruction is selected is shown by the bold paths in figure 7-7. The topmost bold path through the cell at the output pin is that taken by the results of the test of the on-chip system logic; the lowermost path is that taken by the data to be held at the pin while the test is applied. Note that, for each test, the latched parallel output of the boundary-scan cell at the system output pin is updated from data shifted in before the state of the shift-register is overwritten with the test response.

While the *INTEST* instruction is selected, the boundary-scan register assumes the role of the ATE system used for stand-alone component testing. Cells at nonclock system input pins are used to apply the test stimulus, while those at system output pins capture the response. Stimuli and responses are moved into and out of the circuit by shifting the boundary-scan register. Note that this requires that the boundary-scan register cells located at system input pins are able to drive signals into the on-chip system logic.

Typically, the on-chip system logic will receive a sequence of clock events between application of the stimulus and capture of the response such that single-step operation is achieved. The specification of boundary-scan cells for system clock input pins allows the clocks for the on-chip system logic to be obtained in several ways while the *INTEST* instruction is selected. The following are offered as examples:

a) The signals received at system clock pins can be fed directly to the on-chip system logic as during normal operation of the component. Where this option is selected, the component design shall guarantee that precisely one single step of operation of the on-chip system logic occurs while, at least, a specified minimum number of TCK cycles are applied during the *Run-Test*/*Idle* controller state. The component shall be designed so that only one single step of operation is performed whether or not more than the specified minimum number of TCK cycles is applied while the TAP controller is in the *Run-Test*/*Idle* controller state. This may, for example, require that clock signals incoming to the component are gated before application to the on-chip system logic. In this way, operation of the on-chip system logic can be inhibited while test data is shifted through the boundary-scan register. Figure 7-8 illustrates how the system clock applied to the component should be controlled during testing of the on-chip system logic using the *INTEST* instruction.

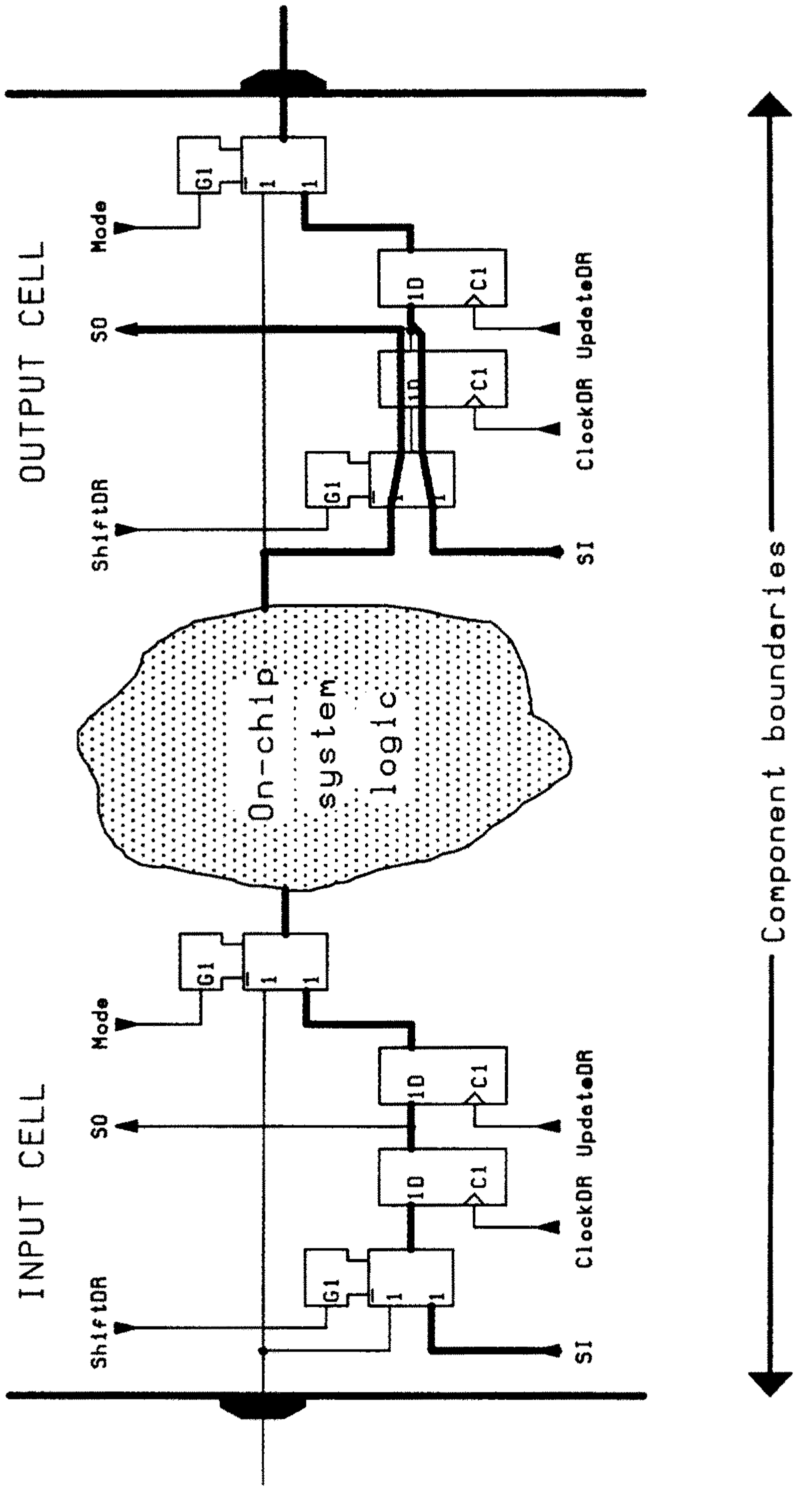


Figure 7-7—Test Data Flow While the INTEST Instruction Is Selected

While figure 7-8 illustrates a situation in which the system clock is a single positive-going pulse, rule 7.8.1b can be generalized to apply to components that employ multiple clock cycles for each step of operation or that have several clock input pins at which multiphase clock signals are received. Note that, while figure 7-8 shows entry into the *Run-Test/Idle* controller state from the *Update-DR* controller state, clock pulses would also be applied to the on-chip system logic were the *Run-Test/Idle* controller state entered from the *Update-IR* controller state.

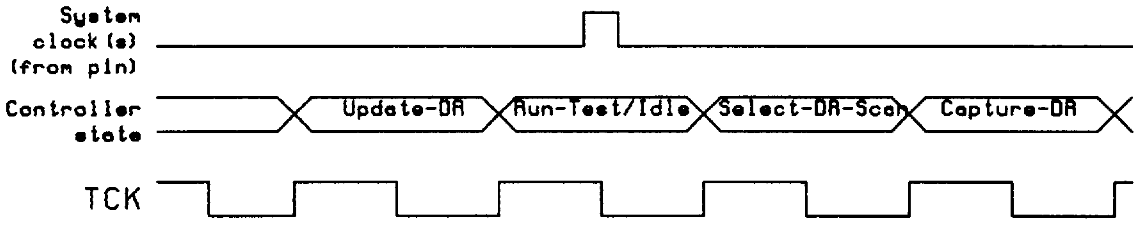


Figure 7-8—Control of Applied System Clock During INTEST

b) The on-chip system logic can be supplied with clock signals derived from TCK in the *Run-Test/Idle* controller state. In all other controller states, the clocks should not change state. Figure 7-9 shows a derived clock signal where the on-chip system logic responds to rising clock edges, for example.

c) Circuitry may be built into the component that, on entry into the *Run-Test/Idle* controller state, allows the on-chip system logic to complete one step of operation. For example, if the component were a microprocessor, it would be permitted to complete a single processing cycle, for example, by internal generation of a pulse on the hold signal. In this case, the clock(s) applied at the system clock pin(s) during the test could be free-running.

d) Clock signals can be shifted in via the boundary-scan path in the same manner that nonclock signals for the on-chip system logic are supplied. Note that this will require the boundary-scan register to be shifted for each distinct clock signal state (e.g., twice for a single-phase clock).

NOTE — This may be a hazard-prone operation for certain circuit designs.

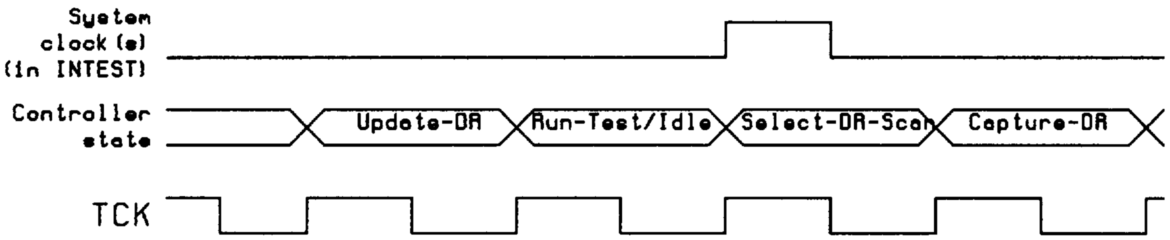


Figure 7-9—Use of TCK as Clock for On-Chip System Logic During INTEST

While the *INTEST* instruction is selected, the state of all system output pins is determined by the test logic. There are two options. First, the pin state may be determined by the data held in the boundary-scan register, shifted into the latched parallel outputs of the register during each pass through the scan sequence for the register. Second, every system output pin may be forced to an inactive drive state (e.g., high-impedance). This ensures that surrounding components on an assembled board are supplied known signal levels while the on-chip system logic test is in progress. Typically, a consistent set of data values would be shifted into the appropriate stages of the boundary-scan register using the *SAMPLE/PRELOAD* instruction prior to selection of the *INTEST* instruction. This data pattern is then reloaded each time a new *INTEST* test pattern is shifted into the boundary-scan register.

Recommendation 7.8.1f, where followed, ensures that data shifted out of the component in response to the *INTEST* instruction are not altered by the presence of faults in off-chip system logic, board-level interconnections, etc. This simplifies diagnosis, since any errors in the output bit stream can only be caused by faults in the on-chip system logic or in the boundary-scan register.

7.9 The RUNBIST Instruction

The optional *RUNBIST* instruction causes execution of a self-contained self-test of the component. Use of the instruction allows a component user to determine the health of the component without the need to load complex data patterns and without the need for single-step operation (as required for the *INTEST* instruction). While the *RUNBIST* instruction is selected, the state of all system output pins is determined by the test logic. There are two options. First, the pin state may be determined by the data held in the boundary-scan register, shifted into the latched parallel outputs of the register during each pass through the scan sequence for the register. Second, every system output pin may be forced to an inactive drive state (e.g., high-impedance).

The following rules apply where the *RUNBIST* instruction is provided.

NOTE — Following use of the *RUNBIST* instruction, the on-chip system logic may be in an indeterminate state that will persist until a system reset is applied. Therefore, the on-chip system logic may need to be reset on return to normal (i.e., nontest) operation.

7.9.1 Specifications

Rules

a) When the *RUNBIST* instruction is selected, the test data register into which the results of the self-test(s) will be loaded shall be connected for serial access between TDI and TDO in the *Shift-DR* controller state.
b) Self-test mode(s) of operation accessed through the *RUNBIST* instruction shall execute only in the *Run-Test/Idle* controller state.
c) Where a test data register (other than the boundary-scan register) shall be initialized prior to execution of the self-test, this shall occur at the start of the self-test without any requirement to shift data into the component (i.e., there shall be no requirement to enter seed values into any test data register other than the boundary-scan register).
d) A duration shall be specified for the test executed in response to the *RUNBIST* instruction (e.g., a number of rising edges of TCK or the system clock).
e) The result of the self-test(s) executed in response to the *RUNBIST* instruction shall be loaded into the test data register connected between TDI and TDO no later than the rising edge of TCK in the *Capture-DR* controller state.
f) Following the specified minimum duration, the test result observed by loading and shifting of the test data register selected by the *RUNBIST* instruction shall be constant regardless of when the *Capture-DR* controller state is entered.
g) Use of the *RUNBIST* instruction shall give the same result in all versions of a component.
h) Data shifted out of a component following completion of execution of a self-test accessed using the *RUNBIST* instruction shall be independent of the operation of off-chip circuitry or board-level interconnections.
i) All stages of the test data register selected by the *RUNBIST* instruction shall be set to determinate logic states (0 or 1) in the *Capture-DR* controller state.
j) The design of the component shall ensure that results of self-tests executed in response to the *RUNBIST* instruction are not affected by signals received at nonclock system input pins.
k) When the *RUNBIST* instruction is selected, all system outputs from the component shall be defined as follows:
    i) All signals driven out of the component shall be defined by data held in the boundary-scan register and shall change only on the falling edge of TCK in the *Update-DR* controller state or on selection of the *RUNBIST* instruction; or

 ii) All outputs from the component (including those that are 2-state nontest signals) shall be placed in an inactive drive state (e.g., high-impedance) on selection of the *RUNBIST* instruction.

l) The states of the parallel output registers or latches in boundary-scan register cells located at system output pins (2-state, 3-state, or bidirectional) shall not change while the *RUNBIST* instruction is selected, unless the associated pin has been placed in an inactive drive state (e.g., high-impedance) as defined in rule 7.9.1k(ii).

Recommendations

m) Where possible, components compatible with this standard should support the *RUNBIST* instruction.

Permissions

n) The binary value(s) for the *RUNBIST* instruction may be selected by the component designer.
o) Where a component includes multiple self-test functions, these may be executed either concurrently or in a sequence determined by the component manufacturer in response to the *RUNBIST* instruction. In the latter case, all sequencing should be taken care of within the component itself without requiring the alteration of the instruction register contents.
p) Additional public instructions may be provided to give user access to individual self-test functions within a component.
q) The test data register connected between TDI and TDO when the *RUNBIST* instruction is selected may be the boundary-scan register.
r) While the *RUNBIST* instruction is selected, the boundary-scan register may act as a pattern generator or signature compactor in the *Run-Test/Idle* controller state provided rule 7.9.11 is met.

7.9.2 Description

The *RUNBIST* instruction provides the component purchaser with a means of running a user-accessible self-test function within the component as a result of a single instruction. This permits all components on a board that offer the *RUNBIST* instruction to execute their self-tests concurrently, providing a rapid health check for the assembled board. Note, however, that the component manufacturer can include further private or public instructions to give access to individual self-test functions one at a time or to self-test functions that are not invoked by the *RUNBIST* instruction.

While the test is proceeding, the test logic defines the outputs from the component. As for the *INTEST* instruction, two options are available:

 i) The pin state may be determined by the data held in the boundary-scan register.
 ii) Every system output pin may be forced to an inactive drive state (e.g., high-impedance).

Where the former option is selected, the data values driven through the system output pins are fixed at the time the *RUNBIST* instruction is selected, based on data held in the boundary-scan register at that time. (This data may have been preloaded using the *SAMPLE/PRELOAD* instruction.) The boundary-scan register is controlled such that the data held in the latched parallel outputs of cells that feed system output pins do not change while the *RUNBIST* instruction is selected. Referring to figure 7-2, this might, for example, be achieved by holding the UpdateDR signal at 0 while the *RUNBIST* instruction is selected. The Mode signal would be held at 1.

Boundary-scan register cells may also be used to hold programmed signal values at inputs to the on-chip system logic while the self-test is executing (again, as shown in figure 7-7). Alternatively, boundary-scan register cells located at nonclock system logic inputs can be designed to act as a source of self-test data for the on-chip system logic. Similarly, boundary-scan register cells located at system logic outputs can act as compactors for the results of the self-test.

The specification of boundary-scan cells for system clock input pins allows the clocks for the on-chip system logic to be obtained in one of two ways while the *RUNBIST* instruction is selected:

a) The signals received at system clock pins can be fed directly to the on-chip system logic as during normal operation of the component. Where this is done, the design of the component shall ensure that the self-test executes only in the *Run-Test/Idle* controller state. The clock may, however, be active in other controller states.

b) The on-chip system logic can be supplied with clock signals derived from TCK in the *Run-Test/Idle* controller state. In all other controller states, the clocks should not change state.

The rules relating to the duration of a self-test executed in response to the *RUNBIST* instruction (rules 7.9.1d and 7.9.1f) ensure that sufficient clock edges can be applied to allow completion of self-tests executed concurrently in different components on an assembled board. Thus in a product containing components with self-test lengths of 1000, 5000, 10 000, and 50 000 rising clock edges on TCK, the complete board shall be left in the *Run-Test/Idle* controller state for at least 50 000 rising clock edges to ensure that all tests complete satisfactorily. Tests that complete before 50 000 clock edges have been applied will hold their results until they are accessed.

Rule 7.9.1g is included to ensure that the test for an assembled board is independent of the versions of the components mounted on it. This is an important consideration when working in a maintenance or repair environment, where the versions of the components used on a board may not be known. The rule can be met by forming the exclusive-OR of the result from execution of the *RUNBIST* instruction with a fixed (version-dependent) pattern. The output from this function would become the result loaded into the boundary-scan register or the other test data register connected between TDI and TDO.

Rule 7.9.1h ensures that data shifted out of the component in response to the *RUNBIST* instruction is not altered by the presence of faults in off-chip system logic, board-level interconnections, etc. This simplifies diagnosis, since any errors in the output bit stream can only be caused by faults in the on-chip system logic or in the test data register connected in the path between TDI and TDO.

7.10 The CLAMP Instruction

The optional *CLAMP* instruction allows the state of the signals driven from component pins to be determined from the boundary-scan register while the bypass register is selected as the serial path between TDI and TDO. The signals driven from the component pins will not change while the *CLAMP* instruction is selected.

The following rules apply where the CLAMP instruction is provided.

NOTE — Following use of the *CLAMP* instruction, the on-chip system logic may be in an indeterminate state that will persist until a system reset is applied. Therefore, the on-chip system logic may need to be reset on return to normal (i.e., nontest) operation.

7.10.1 Specifications

Rules

a) The *CLAMP* instruction shall select the bypass register to be connected for serial access between TDI and TDO in the *Shift-DR* controller state.

NOTE — The bypass register will behave fully as defined in Chapter 9 while the *CLAMP* instruction is selected. Therefore, it will load a logic 0 during the *Capture-DR* controller state and shift data during the *Shift-DR* controller state.

b) When the *CLAMP* instruction is selected, the state of all signals driven from system output pins shall be completely defined by the data held in the boundary-scan register. (For example, this data may be shifted into the boundary-scan register by previous use of the *SAMPLE/PRELOAD* instruction.)

c) The states of the parallel output registers or latches in boundary-scan register cells located at system output pins (2-state, 3-state, or bidirectional) shall not change while the *CLAMP* instruction is selected.

d) When the *CLAMP* instruction is selected, the on-chip system logic shall be controlled such that it cannot be damaged as a result of signals received at the system input or system clock input pins.

NOTE — This might be achieved by placing the on-chip system logic in a reset or "hold" state while the *CLAMP* instruction is selected.

Permissions

e) The binary value(s) for the *CLAMP* instruction may be selected by the component designer.

7.10.2 Description

During testing of a particular IC or a cluster of ICs on a loaded printed circuit board, it may be necessary to place static "guarding" values on signals that control operation of logic not involved in the test—for example, to place it in a state where it cannot respond to signals received from the logic under test. Such instances will undoubtedly occur during the change from in-circuit testing to testing that is based extensively on boundary-scan. In such cases, the "guarding" signal values would be maintained during application of the test.

The *EXTEST* instruction could be used for this purpose. This instruction would be loaded serially into the ICs that drive the signals on which "guarding" values are required. The required signal values would be loaded as a part of the complete serial data stream shifted into the board-level path, both at the start of the test and each time a new test pattern is entered. A limitation of this approach is that the length of the data pattern to be shifted for each test is increased by inclusion of the boundary-scan registers in the ICs involved in the "guarding" process. As a result, the test application rate is reduced.

The optional *CLAMP* instruction allows "guarding" values to be applied using the boundary-scan registers of the appropriate ICs, but does not retain these registers in the serial path during test application. In a case in which the *CLAMP* instruction is used to create "guarding," the following process would be used:

NOTE — It is presumed in the following description that every component implements the optional CLAMP instruction.

a) Prior to the test, the *SAMPLE/PRELOAD* instruction would be loaded into all ICs that will provide "guarding" signals during the upcoming test. Call this group of ICs *G*. If test set-up data are required in ICs not in *G* (i.e., in those ICs that will participate actively in the upcoming test), the *SAMPLE/PRELOAD* instruction may also be loaded into these ICs at this time.
b) Shift the "guarding" pattern into all relevant boundary-scan cells of the ICs in *G*. Any test set-up data required for the ICs to be tested are also loaded.
c) From this point on, until the test is concluded, every time instructions are to be scanned into devices on the board, enter the *CLAMP* instruction into the ICs in *G*. As long as the *CLAMP* instruction is maintained as the active instruction in the ICs of *G*, the output signal values of these ICs will be determined by the "guarding" data in their boundary-scan registers. Also, as a consequence of the use of the *CLAMP* instruction, the ICs in *G* all have their bypass registers selected throughout the test; thus, they contribute very little to the overall test time.

7.11 Device Identification Register Instructions

Use of the optional device identification register allows a code to be serially read from the component that shows:

a) The manufacturer's identity;
b) The part number; and
c) The version number for the part.

Two instructions are defined by this standard that use the device identification register: *IDCODE* and *USERCODE*. These instructions are defined in 7.11 and 7.12. Use of the *IDCODE* instruction will provide information on the base component while use of the *USERCODE* instruction will provide information on the particular programming of an off-board programmable component (e.g., a fuse-programmable logic device).

7.12 The IDCODE Instruction

7.12.1 Specifications

<u>Rules</u>

a) Where a device identification register is included in the design, the component shall provide an *IDCODE* instruction.

b) The *IDCODE* instruction shall select only the device identification 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 device identification register).
c) When the *IDCODE* instruction is selected, the vendor identification code shall be loaded into the device identification register on the rising edge of TCK following entry into the *Capture-DR* controller state.
d) When the *IDCODE* instruction is selected, all test data registers that can operate in either system or test modes shall perform their system function.
e) When the *IDCODE* 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 binary value(s) for the *IDCODE* instruction may be selected by the component designer.

7.12.2 Description

Where a device identification register is included in a component design, the *IDCODE* instruction is forced into the instruction register's parallel output latches during the *Test-Logic-Reset* controller state. This allows the device identification register to be selected by manipulation of the broadcast TMS and TCK signals, as well as by a conventional instruction register scan operation.

The importance of this means of selecting access to the device identification register is that it permits blind interrogation of the components assembled onto a printed circuit board, etc. Thus, in circumstances where the component population may vary (e.g., due to different programming of programmable parts) it is possible to determine what components exist in a product.

7.13 The USERCODE Instruction

7.13.1 Specifications

Rules

a) Where a device identification register is included in the design and the component is user-programmable such that the programming cannot otherwise be determined by the test logic, the component shall provide an *USERCODE* instruction.
b) The *USERCODE* instruction shall select the device identification 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 device identification register).
c) When the *USERCODE* instruction is selected, the user-programmable identification code shall be loaded into the device identification register on the rising edge of TCK following entry into the *Capture-DR* controller state.
d) When the *USERCODE* instruction is selected, all test data registers that can operate in either system or test modes shall perform their system function.
e) When the *USERCODE* 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 binary value(s) for the *USERCODE* instruction may be selected by the component designer.

7.13.2 Description

The *USERCODE* instruction allows a user-programmable identification code to be loaded and shifted out for examination. This instruction is required only for programmable components, where the programming cannot be determined through use of the test logic. The instruction allows the programmed function of the component to be determined.

7.14 The HIGHZ Instruction

Use of the optional *HIGHZ* instruction places the component in a state in which all of its system logic outputs are placed in an inactive drive state (e.g., high impedance). In this state, an in-circuit test system may drive signals onto the connections normally driven by a component output without incurring the risk of damage to the component.

The following rules apply where the *HIGHZ* instruction is provided.

NOTE — Following use of the *HIGHZ* instruction, the on-chip system logic may be in an indeterminate state that will persist until a system reset is applied. Therefore, the on-chip system logic may need to be reset on return to normal (i.e., nontest) operation.

7.14.1 Specifications

Rules

a) The *HIGHZ* instruction shall select the bypass register to be connected for serial access between TDI and TDO in the *Shift-DR* controller state.

NOTE — The bypass register will behave fully as defined in Chapter 9 while the *HIGHZ* instruction is selected. Therefore, it will load a logic 0 during the *Capture-DR* controller state and shift data during the *Shift-DR* controller state.

b) When the *HIGHZ* instruction is selected, all system logic outputs (including 2-state and 3-state outputs and bidirectional pins) of the component shall immediately be placed in an inactive-drive state (e.g., high impedance).

NOTE — According to the rules in 10.5, there shall be no consequential change in the states of the parallel output registers or latches in boundary-scan cells. For example, on leaving the *HIGHZ* instruction and selecting the *EXTEST* instruction, the data held in the boundary-scan register prior to selection of the *HIGHZ* instruction should be applied to the system output pins.

c) When the *HIGHZ* instruction is selected, the on-chip system logic shall be controlled such that it cannot be damaged as a result of signals received at the system input or system clock input pins.

NOTE — This might be achieved by placing the on-chip system logic in a reset or "hold" state while the *HIGHZ* instruction is selected.

Permissions

d) The binary value(s) for the *HIGHZ* instruction may be selected by the component designer.

7.14.2 Description

On boards where not all of the components are compatible with this standard, a need will continue to exist to use in-circuit test techniques in which test signals from an ATE system are driven into internal connections of the assembled board. To allow this to be done without risk of damage to the components that would normally control these connections, components should be designed such that their system logic output pins can be placed in an inactive-drive state while in-circuit testing proceeds. On a component compatible with this standard, provision of the *HIGHZ* instruction allows such a state to be entered by use of the TAP. (On components that do not comply with this standard, this would typically be achieved using a dedicated test-control pin.)

A further use of the *HIGHZ* instruction is to allow a source of test data to be connected to one or more signals internal to a loaded board in place of the normal driver(s). An example application is shown in figure 7-10.

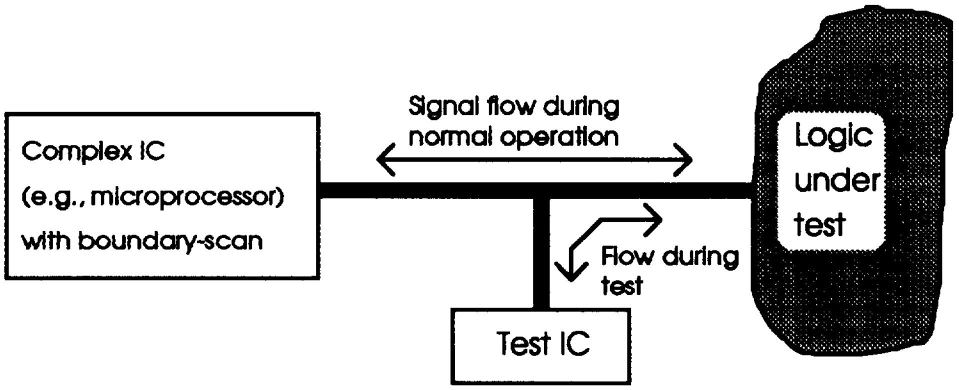


Figure 7-10—Use of the HIGHZ Instruction

During normal operation, the outputs of the test chip would be in an inactive drive state (e.g., high-impedance), while the outputs of the processor would be active. During testing, the *HIGHZ* instruction is entered into the processor chip with the result that its outputs enter the inactive drive state. The test chip can then be enabled to drive the connections into the logic under test (which might, for example, be an array of memory chips).

Note that, where the system requirement is for a 2-state output pin and both logic states are actively driven, a 3-state buffer will have to be provided purely to allow entry into the inactive state when the *HIGHZ* instruction is selected. The enable input to this buffer will be supplied directly from the instruction decoder, as illustrated in figure 7-11. (In figure 7-11, the signal from the instruction decoder would be logic 1 other than when the *HIGHZ* instruction is selected.) No boundary-scan cell is required in this signal path.

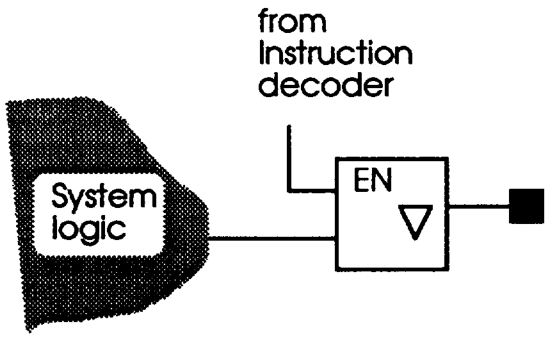


Figure 7-11—Provision of HIGHZ at a 2-State Pin

For a 2-state pin where only one state is driven actively, the output should be forced into the inactively driven state when the *HIGHZ* instruction is selected. For example, the output pulldown transistor for an open-collector output would be forced off.

8. Test Data Registers

The test logic architecture contains a minimum of two test data registers—the bypass and boundary-scan registers. In addition, the design of a third, optional, test data register is defined—the device identification register.

The architecture is extensible beyond the minimum requirements specified in this standard to allow access to any test-support features embedded in the design. These features might include scan-test, self-test registers, or access to key registers in the design (for example, via scannable shadow registers). Additional test data registers need not be intended for public access and use.

Each named test data register has a fixed length and can be accessed using one or more instructions. The registers can, where appropriate, share circuitry and can be concatenated to form further registers, provided that each distinct combination is given a new name (thus allowing it to meet the fixed length requirement).

This chapter defines the common design requirements for all test data registers incorporated in the test logic architecture defined by this standard. Specific design requirements for the bypass, boundary-scan and device identification registers are contained in Chapters 9, 10, and 11 respectively.

8.1 Provision of Test Data Registers

8.1.1 Specifications

Rules

a) The group of test data registers shall include, as a minimum, a bypass register and a boundary-scan register designed according to the requirements contained in this chapter and in Chapters 9 and 10 respectively.
b) Where a device identification register is included in the group of scannable test data registers, it shall be designed according to the requirements contained in this chapter and in Chapter 11.
c) All test data registers shall be designed according to the requirements contained in this chapter.

Permissions

d) Design-specific test data registers may be provided within the group of test data registers to give access to design-specific testability features.
e) Design-specific test data registers may (but need not) be publicly accessible.

8.1.2 Description

Figure 8-1 shows the bypass, boundary-scan, and optional test data registers realized as a set of shift-register based elements connected in parallel between a common serial input and a common serial output. Selection of the register that forms the serial path at a given time is controlled from the instruction register. In figure 8-1, this is shown to be achieved using a multiplexer; however, other implementations are possible.

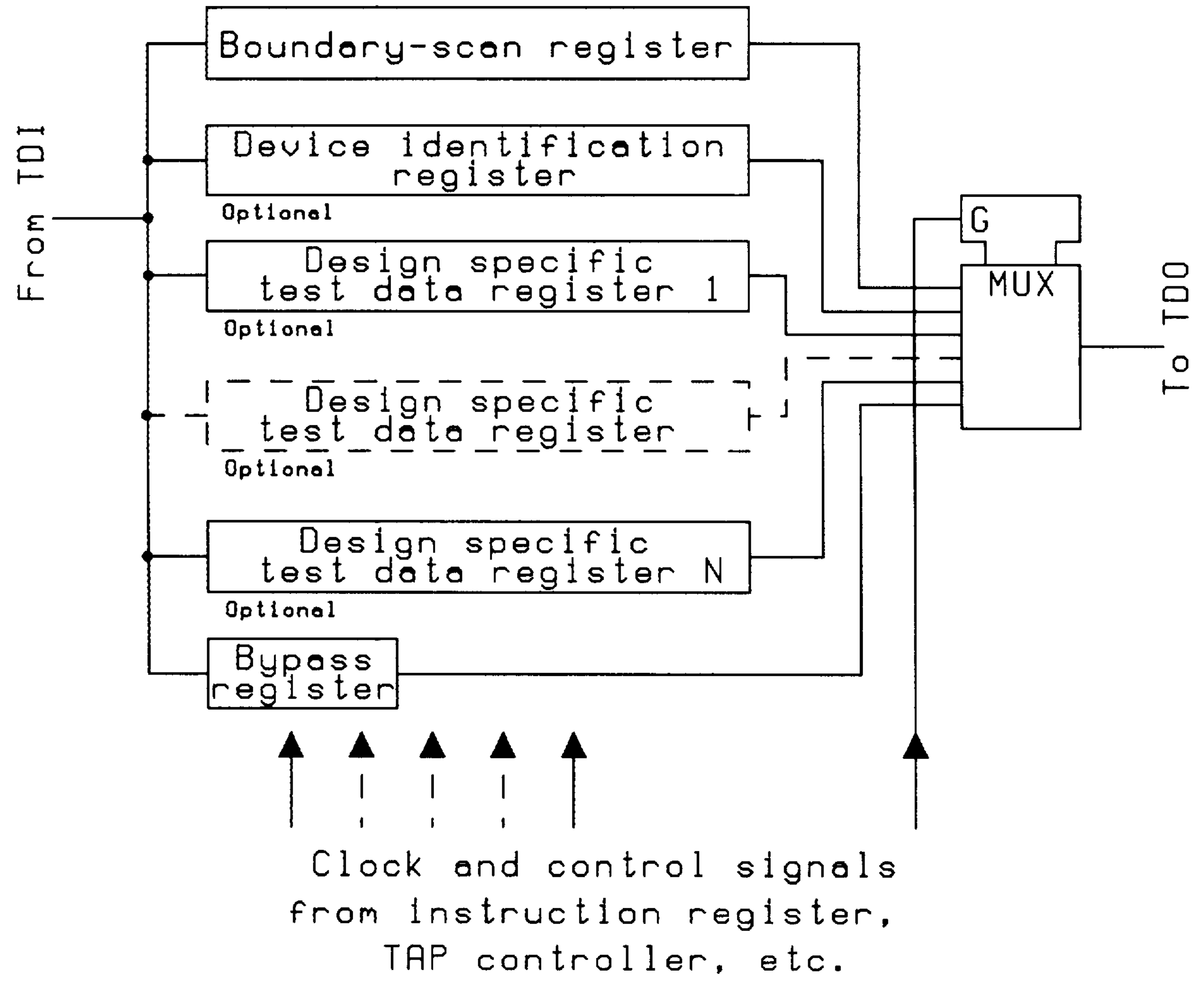

Boundary-scan register
Device identification register
Optional
From TDI
Design specific test data register 1
Optional
Design specific test data register
Optional
Design specific test data register N
Optional
Bypass register
G
MUX
To TDO
Clock and control signals from instruction register, TAP controller, etc.

Figure 8-1—An Implementation of the Group of Test Data Registers

The registers shown in figure 8-1 are briefly described as follows:

The Bypass Register

This provides a single-bit serial connection through the circuit when none of the other test data registers is selected. This register can, for example, be used to allow test data to flow through a particular device to other components in a product without affecting the normal operation of the particular component. The specification for the bypass register is contained in Chapter 9.

The Boundary-Scan Register

This allows testing of board interconnections, detecting typical production defects such as opens, shorts, etc. It also allows access to the inputs and outputs of components when testing their system logic or sampling of signals flowing through the system inputs and outputs. The specification for the boundary-scan register is contained in Chapter 10.

The Device Identification Register

This is an optional test data register that allows the manufacturer, part number, and variant of a component to be determined. If this register is included, then it should conform to the specification contained in Chapter 11.

The Design-Specific Test Data Registers

These optional registers may be provided to allow access to design-specific test support features in the integrated circuit such as self-tests, scan paths, etc. They need not be intended for public use and access, but may be made so if the component designer wishes.

8.2 Design and Construction of Test Data Registers

8.2.1 Specifications

Rules

a) Each test data register shall be given a unique name.
b) The design of each test data register shall be such that, when data is shifted through it, data applied to TDI appears without inversion at TDO following an appropriate number of TCK transitions when the TAP controller is in the *Shift-DR* state.
c) The length of each test data register shall be fixed, independent of the instruction by which it is accessed.
d) For programmable components, the length of each test data register shall be independent of the way that the component is programmed.

Permissions

e) A test data register may be constructed from segments or circuitry also used in one or more other registers provided the resulting design complies fully with the rules in this standard.
NOTE — The resulting combination shall be given a name distinct from those of the registers from which it is constructed.
f) Circuitry (including the shift-register paths) in the various test data registers included in a design may be shared between test data registers provided that the rules contained in this standard are met.
g) Unless specifically prohibited by this standard, circuitry contained in test data registers may be used to perform system functions when test operation is not required.

8.2.2 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. Except where identified specifically, the test data registers may also perform system functions, and thus be a part of the on-chip system logic, when they are not required to perform test functions.

Rule 8.2.1c requires that the length of any test data register is fixed, i.e., a test data register named FRED shall always contain, say, 20 stages no matter how or when it is accessed. Note that virtual registers are allowed (i.e., a named test data register may be built from circuitry shared with other test data registers or with the system logic). Therefore, requirements may exist for different segments of a single physical register to be accessed for different tests. In these cases, rule 8.2.1c can be met by assigning a unique name to each distinguishable register configuration (see figure 8-2 and table 8-1). Rule 8.2.1d requires that the length of a test data register is also fixed independently of how a component is programmed. These restrictions are necessary to avoid unnecessary complication of the software used to generate and apply tests.

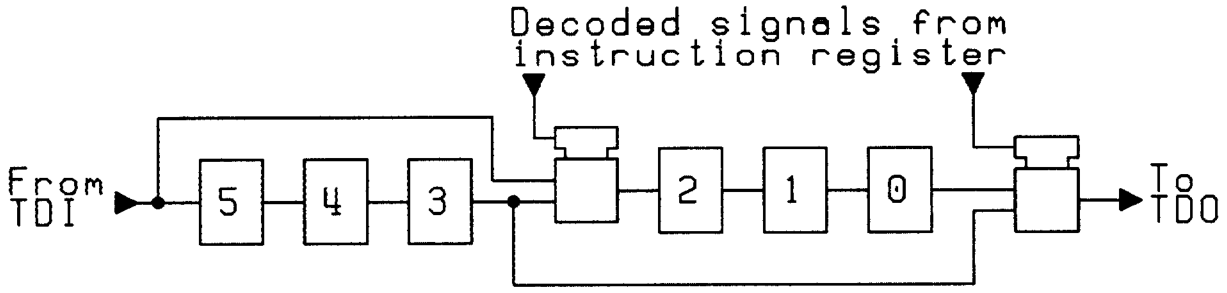


Figure 8-2—Construction of Test Data Registers From Shared Circuitry

Table 8-1—Naming of Test Data Registers That Share Circuitry

Test data register name	Stages that form the register
WHOLE_REG	5, 4, 3, 2, 1, 0
FRONT_REG	2, 1, 0
BACK_REG	5, 4, 3

8.3 Test Data Register Operation

8.3.1 Specifications

Rules

a) Each instruction shall identify a test data register that will be serially connected between TDI and TDO.
b) The test data register connected between TDI and TDO shall shift data one stage towards TDO following each rising edge of TCK in the *Shift-DR* controller state.
c) In the *Test-Logic-Reset* controller state, all test data registers shall be set so that either they perform their system function (if one exists) or they do not interfere with the operation of the on-chip system logic.
d) Where a test data register is required to load data from a parallel input in response to the current instruction, this data shall be loaded on the rising edge of TCK following entry into the *Capture-DR* controller state.
e) Test data registers enabled to drive data off-chip shall be designed such that component outputs change only
   i) On the falling edge of TCK following entry to the *Update-DR*, *Update-IR*, *Run-Test/Idle*, or *Test-Logic-Reset* controller state as a result of signals applied at TCK and TMS; or
   ii) Immediately on entry into the *Test-Logic-Reset* controller state as a result of a logic 0 being applied at TRST*.

   NOTE — This may require that the register be provided with a latched parallel output.
f) Where a test data register is required to operate in response to the *RUNBIST* instruction, the required operation shall occur in the *Run-Test/Idle* controller state.
g) Where no operation of a selected test data register is required in a given controller state in response to the current instruction, the register shall retain its last state unchanged.
h) Test data registers that are not selected by the current instruction shall be set so that either they perform their system function (if one exists) or they do not interfere with the operation of the on-chip system logic.

Permissions

i) In addition to the test data register enabled for shifting between TDI and TDO, an instruction may select further test data registers.

   NOTE — These should retain their last state in the *Shift-DR* controller state, but otherwise meet the rules set out above.

8.3.2 Description

These requirements ensure that test data registers operate correctly in conjunction with the TAP and TAP controller.

Note that, while an instruction may select test operation of more than one test data register, there can be only one test data register between TDI and TDO. All other selected test data registers retain their state in the *Shift-DR* controller state. One use of this capability is as follows.

Consider the case where several test data registers need to be accessed in sequence in order to establish starting conditions for a test or to examine test results. As an example, table 8-2 shows the sequence of events (starting from the *Test-Logic-Reset* controller state) that would be required if two design-specific test data registers and the boundary-scan register needed to be accessed in order to execute an instruction. Figure 8-3 shows the design of the group of test data registers for this example.

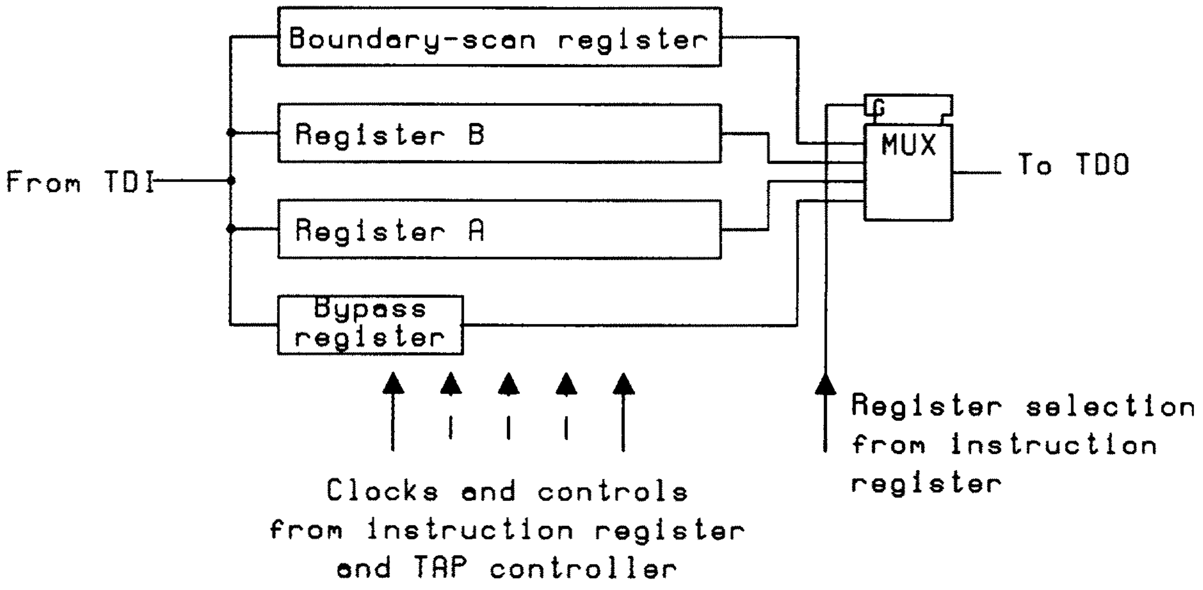

Boundary-scan register
Register B
Register A
Bypass register
From TDI
G
MUX
To TDO
Register selection from instruction register
Clocks and controls from instruction register and TAP controller

Figure 8-3—Example Design Containing Two Optional Test Data Registers

In step 4, serial access is required to register B in order to set its initial condition as required for execution of the test. However, register A was set to its required initial condition in step 2, so it is necessary to design the test logic such that register A can retain its state between steps 2 and 7 (when the self-test is executed) while register B and the boundary-scan register are accessed. Similarly, the test logic shall be designed such that register B retains the initial condition set during step 4 until step 7 and such that the reverse sequence of events can occur following completion of execution of the self-test.

Note that the design of the test logic may be such that test data registers shall be accessed in a fixed order in order to achieve the desired result. For example, the test logic may not allow register B to retain its state while register A is scanned.

Table 8-2—Sequential Access to Test Data Registers

Step	Action
0	Test logic inactive in the *Test-Logic-Reset* controller state.
1	Enter instruction that selects register A for connection between TDI and TDO.
2	Scan required initial values into register A.
3	Enter instruction that selects register B for connection between TDI and TDO and also keeps register A in its test mode of operation.
4	Scan required initial values into register B. Register A retains its state.
5	Enter the test instruction that selects test operation of registers A and B and connects the boundary-scan register between TDI and TDO.
6	Scan required values for the component inputs and outputs into the boundary-scan register. Registers A and B retain their state.
7	Execute the instruction by entering the *Run-Test/Idle* controller state.
8	Enter instruction that selects register B for connection between TDI and TDO and also keeps register A in its test mode of operation.
9	Scan test results out of register B. Register A retains its state.
10	Enter instruction that selects register A for connection between TDI and TDO.
11	Scan test results out of register A.

9. The Bypass Register

The bypass register provides a minimum length serial path for the movement of test data between TDI and TDO. This path can be selected when no other test data register needs to be accessed during a board-level test operation. Use of the bypass register in a component speeds access to test data registers in other components on a board-level test data path.

9.1 Design and Operation of the Bypass Register

9.1.1 Specifications

Rules

a) The bypass register shall consist of a single shift-register stage.
b) When the bypass register is selected for inclusion in the serial path between TDI and TDO by the current instruction, the shift-register stage shall be set to a logic zero on the rising edge of TCK following entry into the *Capture-DR* controller state.
c) The circuitry used to implement the shift-register stage in the bypass register shall not be used to perform any system function (i.e., it shall be a dedicated part of the test logic).
d) The operation of the bypass register shall have no effect on the operation of the on-chip system logic.

9.1.2 Description

The bypass register may be implemented as shown in figure 9-1.

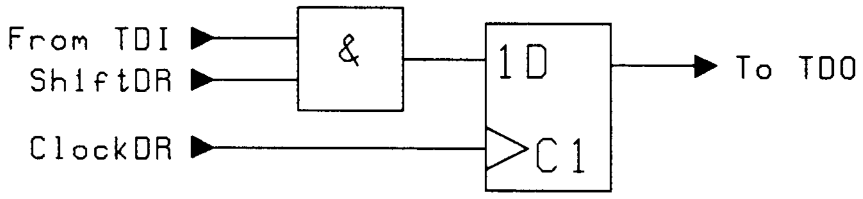


Figure 9-1—A Bypass Register Implementation

The provision of this register allows bypassing of segments of the board-level serial test data register that are not required for a specific test. Test access times to the segments of interest are reduced.

As an example, consider a circuit board containing 100 integrated circuits, each of which has 100 bits in its boundary-scan register. The boundary-scan path on the assembled board would include 10 000 shift-register stages if all the segments were connected in series simultaneously. This would give protracted test times, for example, when accessing just one of the integrated circuits on the path.

The ability to bypass segments of the shift-register path under control of the appropriate instruction register allows considerable shortening of the overall path in such circumstances. Continuing the example, 99 of the components could be set to shift only through their bypass register, with the integrated circuit under test having its full boundary-scan register in circuit. This would give a total serial path length of 199 stages—a considerable reduction compared to 10 000.

Rule 9.1.1b is included so that the presence or absence of a device identification register in the test logic can be determined by examination of the serial output data. The bypass register (which is selected in the absence of a device identification register) loads a logic 0 at the start of a scan cycle, whereas a device identification register loads a constant logic 1 into its LSB. When the *IDCODE* instruction is loaded into the instruction register, a subsequent data register scan cycle will allow the first bit of data shifted out of each component to be examined—a logic 1 showing that a device identification register is present. This allows blind interrogation of device identification registers by setting the *IDCODE* instruction as outlined in 11.1

A requirement of the *BYPASS* instruction is that, when it is selected, the on-chip system logic shall continue its normal operation undisturbed. Rule 9.1.1b is included so that this requirement can be met. Note, however, that provided rule 9.1.1b is met, the shift-register stage may be a shared resource used by several of the registers defined by this standard and also by any design-specific test data register.

10. The Boundary-Scan Register

NOTE — This rewritten version of Chapter 10 replaces the version that appeared in IEEE Std 1149.1-1990 . The objective of this rewriting was to improve the clarity of presentation of the rules and associated material while making a small number of corrections, highlighted by input from early adopters of this standard. Please note that the majority of changes made by this rewriting are cosmetic and that few technical changes have been made to the design of the boundary-scan register.

The boundary-scan register allows testing of circuitry external to a component—for example, board interconnect or external components that do not conform to this standard. The register also permits the system signals flowing into and out of the system logic to be sampled and examined without causing interference with the normal (nontest) operation of the on-chip system logic. Optionally, additional test functions may be supported—for example, testing of the on-chip system logic.

10.1 Introduction to This Chapter

This chapter specifies the design of the boundary-scan register in a component and the operation of the register in response to the various instructions defined by this standard.

Of the registers required by this standard, the boundary-scan register is the most complex. Its complexity lies neither in its shifting function nor in its architectural placement in parallel with the other required test data registers, both of which conform to the rules set out in Chapter 8. The complexity lies instead in the manner in which the register is connected around the on-chip system logic and in its operation in response to the instructions defined in Chapter 7. Design requirements for both connectivity and functional operation vary from cell to cell and are determined both by the type of signal (input to, or output from, the on-chip system logic) and by the set of instructions to be supported.

The design specifications are presented in three groups:

a) *Register design and operation (10.2 and 10.3).* The structure of the boundary-scan register is specified. Also presented are specifications for the operation of the shift-register at the heart of the boundary-scan register and for the parallel output latches or flip-flops that are required for some shift-register stages.
b) *Cell provision and operation (10.4 through 10.8).* Rules are presented that specify where boundary-scan register cells must be provided and how they are to be connected between the system pins of a component and the system inputs and outputs of the on-chip system logic. Further rules are presented that define how signals are to be routed through boundary-scan register cells in response to selection of the various instructions defined by this standard.
c) *Cell merging (10.9).* Finally, permissible ways in which boundary-scan register cells required by the cell provisioning rules can be merged are specified. Application of these rules will reduce the cost of implementation of a boundary-scan register in certain special cases.

To simplify the presentation of the rules, this chapter uses the terminology and approach described in the following subclauses.

10.1.1 Approach

To simplify the presentation of the rules in this chapter, the boundary-scan register will be described as though it is being added to an already finished design—one that does not conform to this standard. Such a design may be thought of as shown in figure 10-1.

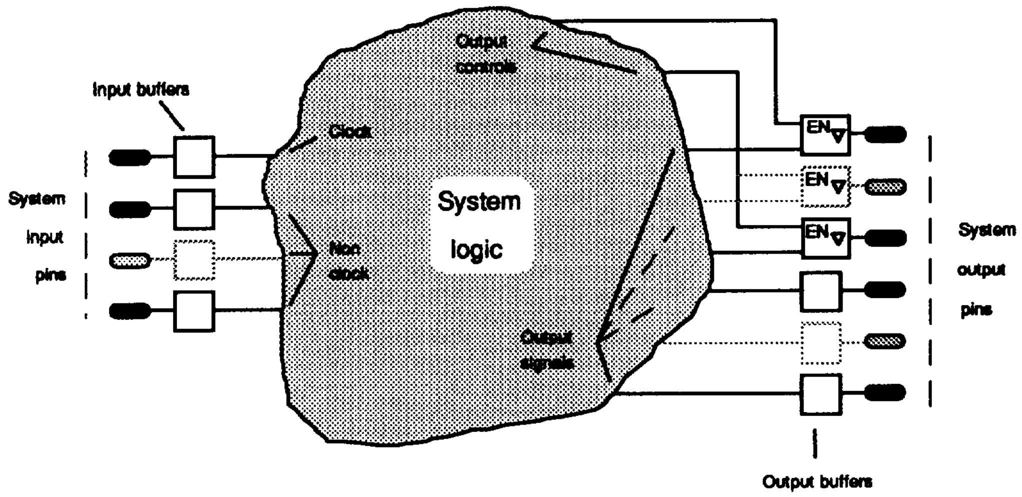


Figure 10-1—A Component Without Boundary-Scan

Note the following features in figure 10-1:

a) *The system logic*. This is the circuitry that realizes the nontest, digital function of the component.
b) *The system pins*. The term "system pin" is used throughout this chapter to refer to any system (i.e., nontest) terminal to which an external connection may be made. For packaged components, external connections are made to package pins, typically by means of a soldering process. However, in some cases an integrated circuit chip will be assembled directly onto a substrate without prior packaging. In such cases, the term "system pin" should be interpreted as the point to which the external connection is made, i.e., the bonding pad.
c) *The input/output buffers*. The buffers are connected between the system pins and the system logic. Note that output buffers may have control as well as data inputs from the system logic. The signals received at the control inputs determine the manner in which the output buffer operates. For example, figure 10-1 shows several 3-state output buffers at which the enable input (EN) is used to determine whether or not the output is driven. Other types of output buffer, at which control signals may be used to determine different characteristics of the signal driven off-chip, are possible.
d) *The inputs and outputs of the system logic*. Two types of input to the on-chip system logic should be distinguished:
   i) Clock inputs. Transitions at these inputs, from the low to 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. In an edge-triggered design, the edges (logic level transitions) received at clock inputs are used to trigger operation of all or part of the on-chip system logic, and steady-state logic values received at these signals have no significance. In a level-sensitive design, clock inputs are used to enable storage devices in the on-chip system logic to load data values. Note that the values loaded into stored-state devices are not determined by the values of clock inputs.
   ii) Nonclock inputs. This group includes all other inputs to the on-chip system logic. Typically, signals applied at these inputs are used to supply data or to select an operation to be performed.
   Outputs from the on-chip system logic drive output buffers (or, as will be discussed later, inputs to mixed analog/digital circuit blocks located external to the system logic). It is necessary to distinguish between the two types of signal that may be driven to an output buffer:
   iii) Output control signals. In a component without boundary-scan, such as that shown in figure 10-1, these signals would directly drive "enable" inputs of output buffers and hence determine either the direction of signal flow at bidirectional pins or whether 3-state output buffers were enabled.

iv) Data signals. In a component without boundary-scan, these signals would drive data inputs to output buffers (or, as will be discussed later, inputs to mixed analog/digital circuit blocks located external to the system logic).

NOTE — A single output from the on-chip system logic may drive an output control signal to one output buffer and a data signal to another.

10.1.2 Signal Paths to the On-Chip System Logic

Each signal path into or out of the on-chip system logic is considered to be a fanout tree with one or more branches. Signals enter the fanout tree at the trunk (e.g., from an input buffer) and leave through the branches (e.g., at the inputs of the on-chip system logic). For example, figure 10-2 shows signal paths between one system input pin and several inputs to the on-chip system logic. (In many cases, the fanout tree will be regarded as being contained within the system logic. In these cases, only a single point-to-point connection is considered to be present between the pin and the logic; that is, a connection with only one fanout branch.)

10.1.3 Boundary-Scan Register Cell

NOTE — The rules contained in this chapter describe the boundary-scan register as a logical device, not a particular physical implementation. Further, for clarity of presentation, the example boundary-scan register cell designs presented in this chapter show the circuitry to be separate from that used to construct the various other features defined in this standard. However, be aware that the rules of this standard permit parts of the circuitry used to construct boundary-scan register cells—notably the shift-register stages—to be used in the implementation of other features defined by this standard, such as the bypass and device identification registers. Where circuitry is shared between the boundary-scan register and other features defined by this standard, boundary-scan register cells may appear that are more complex than those described here.

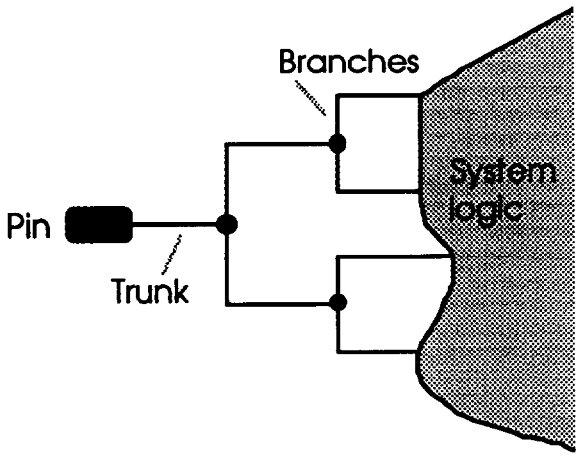


Figure 10-2—Input Connection

Every boundary-scan register cell is considered to have a number of data terminals (at least two) and a number of clock and control inputs, appropriate to the style of implementation. Contained within each cell is a single shift-register stage, often provided with a parallel input and a parallel output (which may be latched). This shift-register stage uses two of the data connections of the cell as a serial input and a serial output. By way of these connections, the cell is linked to those cells before and after it in the boundary-scan register.

Cells that have three data terminals allow signals entering or leaving the on-chip system logic to be observed, but not controlled. For such a cell, the third data terminal functions as a parallel input to a parallel-in, serial-out shift-register stage. When the boundary-scan register is selected as the serial path between TDI and TDO by an instruction, the data present at this terminal are loaded into the shift-register stage on the rising edge of TCK in the *Capture-DR* controller state (i.e., as required by the rules of Chapter 8). Cells of this type may be called "observe-only" cells. They will be connected to a signal path entering or leaving the on-chip system logic, as shown in Figure 10-3.

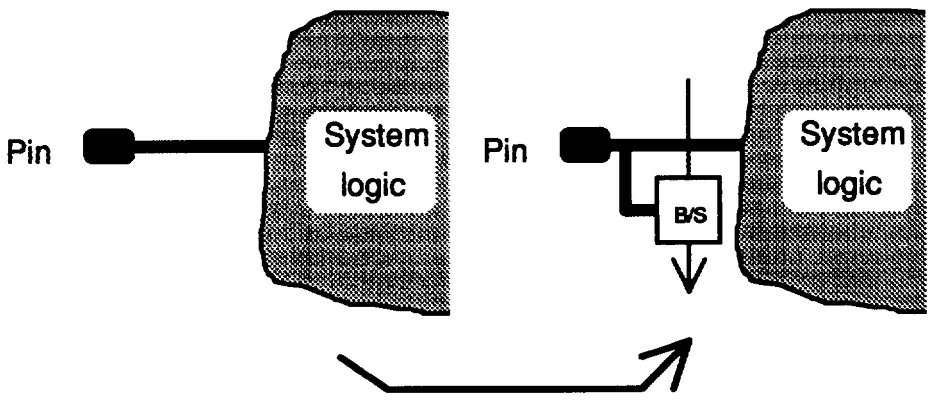


Figure 10-3—Connection of an Observe-Only Boundary-Scan Register Cell

Cells with four or more data terminals are inserted into signal paths entering or leaving the on-chip system logic, as shown for the case of an input in figure 10-4. Such cells may be called "control-and-observe" cells. The shift-register stage in a control-and-observe cell can load the value of the signal path into which they are inserted and, hence, allow observation of that signal. Also, when required, the value held in the shift-register stage can be driven to the wire in place of the normal (nontest) source. In some cases, a constant signal value may also be driven to the wire in place of the normal (nontest) source.

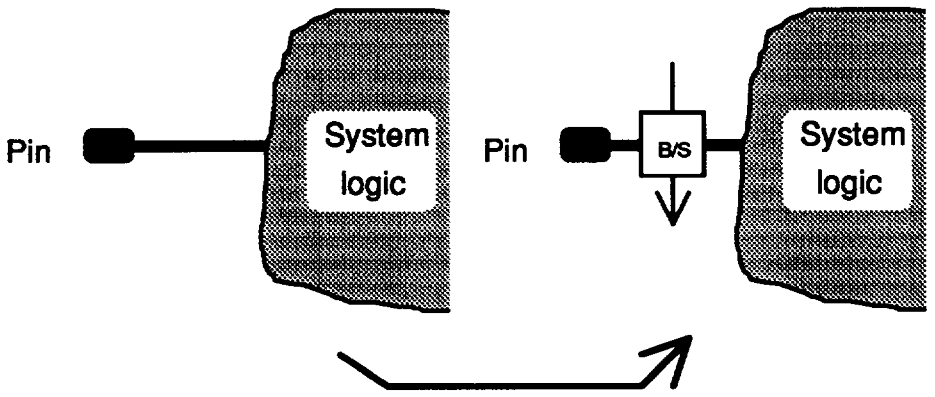


Figure 10-4—Insertion of a Control-and-Observe Boundary-Scan Register Cell

Cells that have only two data terminals are redundant in the sense that they could have been omitted from the design without jeopardizing compliance of the component to this standard. Such cells may be simple shift-register stages. Typically, they will exist in a component design as a result of programming or customization of boundary-scan register cells (see 10.8).

The rules presented later in this chapter define the manner of cell insertion or addition required at each type of connection into or out of the system logic.

A conceptual model of a control-and-observe boundary-scan register cell is shown in figure 10-5. Such cells contain a parallel-in, parallel-out shift-register stage. Signals flow into the cell through one terminal of the cell (termed the parallel input) and out through another (termed the parallel output). Logic (typically, one or more multiplexors) within a control-and-observe cell determines routing of the signal on the parallel input of the cell to the parallel output of the cell. The signal may be routed through the "parallel" terminals of the parallel-in, parallel-out shift-register stage or, in normal (nontest) operation of the component, the routing logic is set such that the signal is driven directly from the parallel input of the cell to the parallel output without any change in signal value.

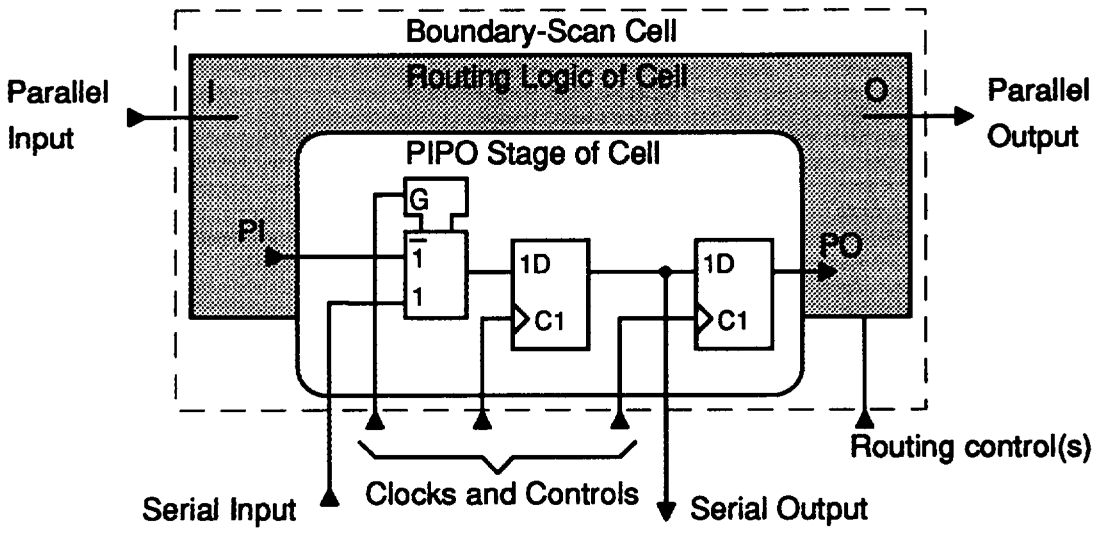


Figure 10-5—Conceptual View of a Control-and-Observe Boundary-Scan Register Cell

10.2 Register Design

10.2.1 Specifications

Rules

a) The boundary-scan register shall consist only of boundary-scan register cells as defined in this chapter.
b) Sufficient boundary-scan register cells shall be provided to meet the requirements for each system connection into or out of the on-chip system logic fully, as defined later in this chapter.
c) Each boundary-scan register cell shall contain a single shift-register stage and shall have a serial input terminal and a serial output terminal, by means of which the cell is linked to those cells before and after it in the boundary-scan register or, in the case of the cells at each end of the register, to the remainder of the test logic defined by this standard.
d) A boundary-scan register cell shall have two or more data terminals (including the serial terminals required by rule 10.2.1c).

NOTE — Each cell will also have several clock and control inputs, the number of which will be determined by the style of implementation.

e) Boundary-scan register cells that have three data terminals shall be designed such that one data terminal is a parallel input to the shift-register stage in the cell. (Such cells are named “observe-only” cells.)
f) For boundary-scan register cells that have four or more data terminals, data terminals other than those used for serial input and output shall be parallel inputs and parallel outputs of the cell and be connected by routing logic
   i) To each other; and
   ii) To the parallel input and parallel output of the shift-register stage.

NOTE — Often, but not always, shift-register stages will require latched parallel outputs.

g) For a given component, the ordering of cells in the boundary-scan register shall be fixed. In particular, the ordering shall not vary as a result of any operation of the on-chip system logic.
h) For a given component, the length of the boundary-scan register shall be fixed. In particular, the length shall not vary as a result of any operation of the on-chip system logic.
i) In the event that no boundary-scan register cells are required for a component, a register consisting of a single shift-register stage shall be provided.

NOTE — This situation will arise where a component only contains test logic as defined or permitted by this standard. Such a component could be described as being dedicated to testing; it will not contribute to the system function of an assembled board.

j) Circuitry used to implement boundary-scan register cells shall not be used to perform any system function (i.e., it shall be a dedicated part of the test logic).

NOTE — Subject to conformance with rule 10.2.1k, circuitry may be shared between the boundary-scan register and another part of the test logic. For example, the shift-register stages may also be used by another test data register.

k) Where a boundary-scan register cell has a latched parallel output, the circuitry that implements this functionality shall not be used for any other test or nontest purpose (that is, it shall be a dedicated part of the boundary-scan register and not, for example, the device identification register).

Permissions

l) Rule 10.2.1i may be met by selecting the bypass register whenever this standard requires selection of the boundary-scan register as the path between TDI and TDO.

m) Boundary-scan register cells may be connected in any order.

NOTE — See rule 10.2.1g.

10.2.2 Description

The boundary-scan register is a mandatory feature of this standard and shall be included in each component that claims conformance to this standard. It consists of a number of cells, equal to the number of shift-register stages contained in the register. These cells are positioned around the on-chip system logic of a component as specified in later clauses in this chapter. They are connected to form a single shift-register-based path that is connected between TDI and TDO in the *Shift-DR* controller state when an appropriate instruction is selected. (With regard to the instructions defined in this standard, the boundary-scan register is defined to be the serial path between TDI and TDO in the *Shift-DR* controller state for the *SAMPLE/PRELOAD*, *EXTEST*, *INTEST*, and, optionally, *RUNBIST* instructions.)

Figure 10-6 illustrates the design of the shift-register portion of the boundary-scan register. Note that not all stages are provided with latched parallel outputs.

For a given component, the minimum permissible length of the boundary-scan register is a function of the number and type of system (nontest) connections into and out of the on-chip system logic. (Typically, these will be off-chip connections.) The rules that determine the minimum set of cells that has to be provided are presented later in this chapter. Note, however, that every component claiming to be compliant to this standard shall have a boundary-scan register that contains at least one shift-register stage.

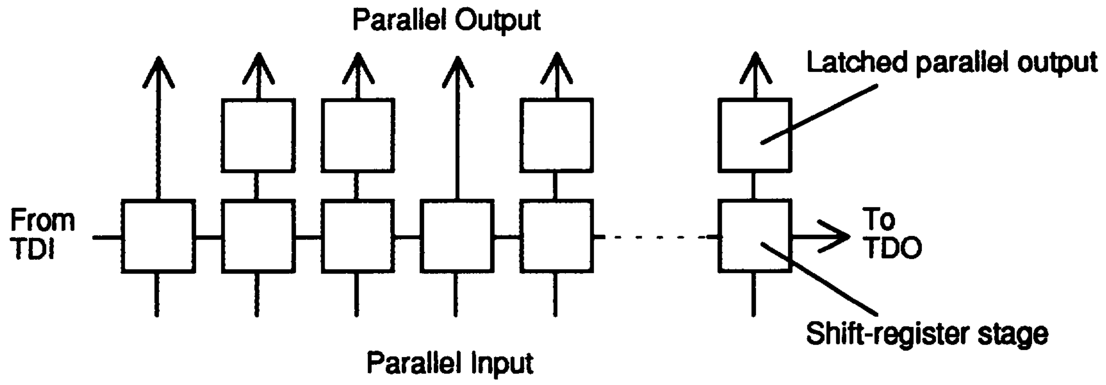


Figure 10-6—Boundary-Scan Shift-Register Design

10.3 Register Operation

10.3.1 Specifications

Rules

a) When the boundary-scan register is selected as the serial path between TDI and TDO in the *Shift-DR* controller state, data entered at TDI shall appear without inversion at TDO following application of a number of paired rising and falling edges at TCK equivalent to the length of the boundary-scan register.

b) When the *EXTEST*, *INTEST*, or *SAMPLE/PRELOAD* instruction is selected, latched parallel outputs of the boundary-scan shift-register shall change state only on the falling edge of TCK in the *Update-DR* controller state, at which time each shall be set to the state of its corresponding shift-register stage.

c) When the *CLAMP* or *RUNBIST* instruction is selected or when the boundary-scan register is not selected as the serial path between TDI and TDO, all latched parallel outputs shall retain their state at least until
   i) The *Test-Logic-Reset* controller state is entered as a result of application of a logic 0 at TRST*; or
   ii) The first falling edge of TCK occurs in the *Test-Logic-Reset* controller state when that state is entered as a result of signals applied at TCK and TMS.

   NOTE — Permission 10.3.1g allows the boundary-scan register to be reset following entry into the *Test-Logic-Reset* controller state; so, in a case in which such implementation details are not known, it should be assumed that the states of the latched parallel outputs of the boundary-scan register are unknown once this state has been entered.

d) No limit shall be imposed on the number of system output pins that may change state in a single *Update-DR*, *Update-IR*, or *Test-Logic-Reset* controller state as a result of the operation of the test logic; neither shall restrictions be placed on the data patterns that may be driven (e.g., a maximum limit on the number of system logic outputs that may drive a "1").

   NOTE — Designers should consider that, in test mode, a boundary-scan test may manipulate several buses in ways that could never occur during normal system operation. This may cause transient or steady-state power consumption to exceed that expected for normal system operation. Be especially aware of this when breadboarding, socketing, or fixturing components compatible with this standard because, in these environments, power distribution may be suboptimal.

e) No limit shall be placed on the combinations of logic values that may be shifted into the boundary-scan register.

   NOTE — During test operations, combinations of signals may be driven either to the on-chip system logic or off-chip that will not arise during normal operation of the component. This is particularly the case where multiple boundary-scan register cells are used to drive signals that are normally driven from a single source (see rules 10.5.1c, 10.5.1d, and 10.6.1d)

Permissions

f) The delay between the falling edge of TCK and consequent changes at system output pins may be deliberately skewed between system outputs, e.g., because of a need to avoid simultaneous switching at several or all system outputs.

   NOTE — In the case of the example implementations shown in this standard, this skew could be added by injecting small delays in the UpdateDR clock path and *Mode-n* control signals to each boundary-scan register cell. Such skew may be required both for boundary-scan register cells that feed data signals and for those that feed output control signals. In the latter case, the added skew will prevent excessive current demand due to simultaneous changes from "disable" to "active" or vice versa.

g) Where boundary-scan register cells are provided with shift-register stages with latched parallel outputs, these outputs may be reset to either logic state (0 or 1)
   i) When the *Test-Logic-Reset* controller state is entered as a result of application of a logic 0 at TRST*; or
   ii) On the first falling edge of TCK in the *Test-Logic-Reset* controller state when that state is entered as a result of signals applied at TCK and TMS.

10.3.2 Description

To meet the requirements of the *SAMPLE/PRELOAD* instruction, it has to be possible to move data through the boundary-scan register without interfering with the normal system operation of the component. This is achieved by making the shift-register stages used by the boundary-scan register a dedicated part of the test logic; that is, they cannot perform any system function. Note, however, that this rule is met when the shift-register stages may be a shared resource used by several of the registers defined by this standard and by any design-specific test data register.

In contrast, the circuitry used to provide a latched parallel output to some boundary-scan register stages shall only be used for the single stated purpose. Further, the latched output has to be controlled such that it retains its last state whenever the boundary-scan register is not selected by the current instruction. This latter feature allows a set of data values to be shifted into the register and placed on the various latched parallel outputs by use of the *SAMPLE/ PRELOAD* instruction. These values will then be retained until such time as they are modified at the end of a subsequent shifting process (i.e., in the *Update-DR* controller state). Thus, when an instruction is entered that selects the data held in the boundary-scan register to be driven out of the component (e.g., the *EXTEST* instruction), the previously loaded data is immediately available for use. (See the discussion in 7.6.)

10.4 General Rules Regarding Cell Provision

10.4.1 Specification

Rules

a) One or more boundary-scan register cells shall be provided at each system input or output of the on-chip system logic, as detailed in 10.5 and 10.6.
b) For components that contain analog circuitry external to the on-chip system logic (i.e., analog circuitry having off-chip connections as shown in figure 10-7), the connections between the on-chip analog circuits and the on-chip system logic shall be considered to be off-chip connections.
c) Boundary-scan register cells shall not be provided at:
   i) TAP pins (TCK, TDI, TDO, TMS, and TRST*);
   ii) Pins that enable compliance to this standard (see 3.8); or
   iii) Nondigital pins (e.g., power and analog pins).

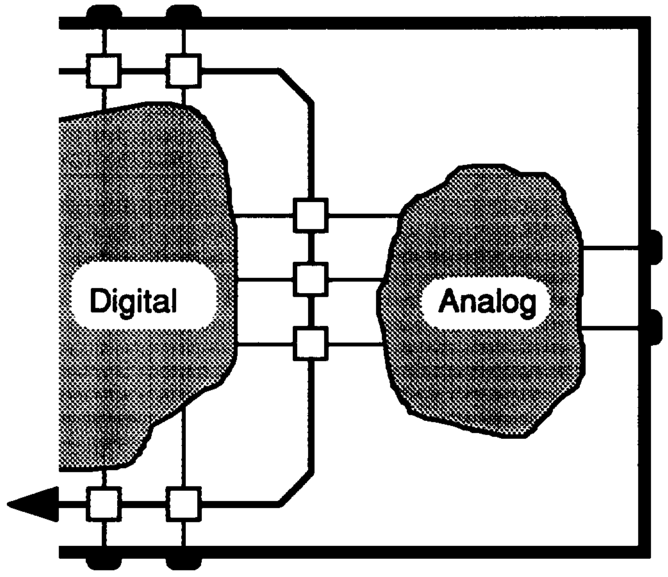


Figure 10-7—A Component That Contains Analog Circuitry

NOTE — Differential drivers and receivers that operate through detection of the direction of current flow are considered to be "analog" circuits. Therefore, in such cases, this rule requires a single boundary-scan register cell to be placed between each differential driver or receiver and the on-chip system logic, as illustrated in figure 10-8. See 10.4.2 for information on the application of these rules to other types of paired inputs and outputs, e.g., differential signals that operate using conventional logic voltages.

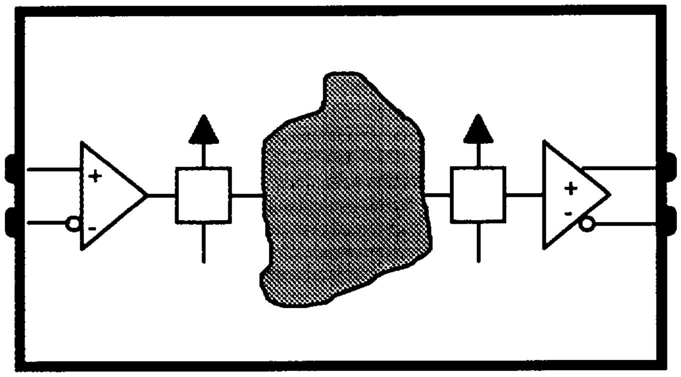


Figure 10-8—A Component With Differential Inputs and Outputs

d) The connection of boundary-scan register cells of the control-and-observe type shall be such that, if each cell were replaced by a short-circuit connection between its parallel input and parallel output, the normal (nontest) logical operation of the component would not be altered.

NOTE — There may, however, be changes in performance.

e) There shall be no logic between any boundary-scan register cell and the system pin to which that cell is connected.

NOTE — "Transparent" devices such as buffers and I/O buffers are not considered to be "logic" and may exist outside the boundary-scan register. Inverters may also exist outside the boundary-scan register subject to conformance to rules 10.5.1h, 10.5.1i, 10.6.1k, 10.6.11, 10.6.1m, and 10.6.1n. Devices that perform a logic operation (such as gates, flip-flops, or latches) are considered to be logic devices and shall not be placed outside the boundary-scan register.

Permissions

f) Where some inputs or outputs are not connected to package pins in a particular packaged configuration of a component, boundary-scan register cells may be provided for the unconnected signals.

10.4.2 Description

Figure 10-9 illustrates the placement of boundary-scan register cells.

Boundary-scan register cells are placed such that the state of each digital system pin (including clock pins) can be controlled or observed using the boundary-scan register. These cells may also allow the state of the system logic inputs and outputs to be controlled and observed respectively.

Extension of the design of the boundary-scan register to cover cases in which analog circuit blocks are located external to the on-chip system logic, between the logic and the pins, is straightforward. In such components, the signals that form the interface between the purely digital circuitry and the mixed analog-digital circuit block(s) are considered to be equivalent to system pins. Therefore, boundary-scan register cells are provided for connections that flow to or from the mixed analog-digital block(s) (see rule 10.4.1b and figure 10-7).

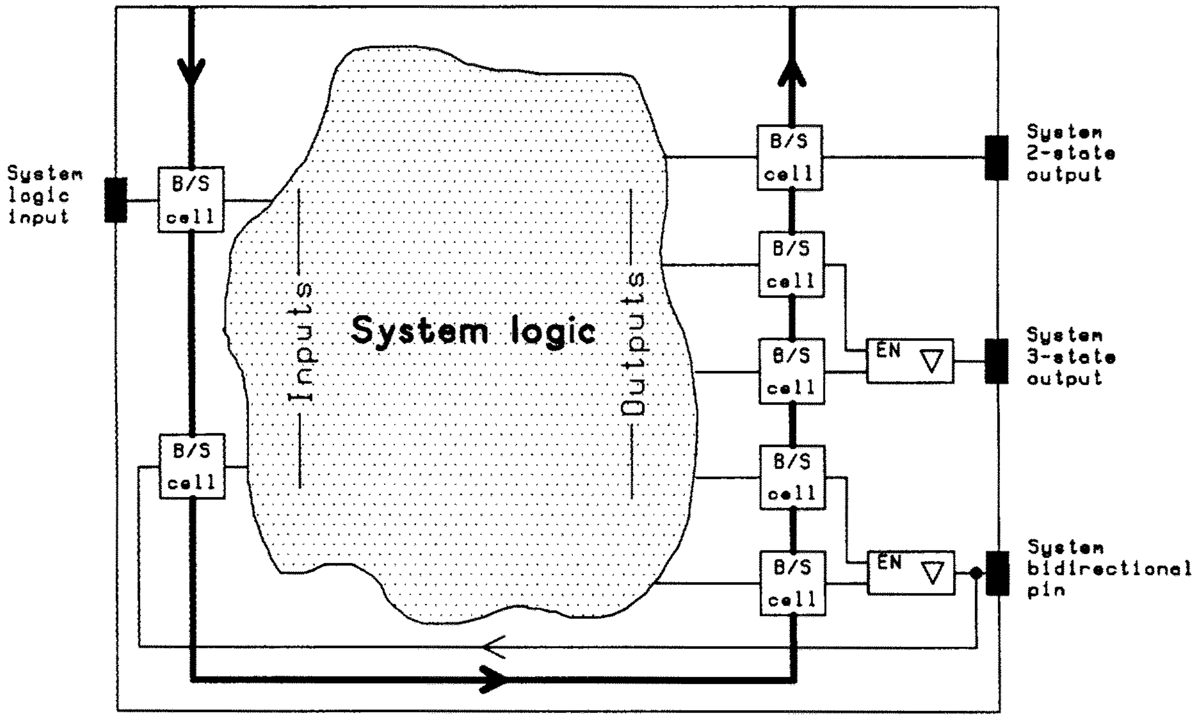


Figure 10-9—Placement of Boundary-Scan Register Cells

The specification of the boundary-scan register also addresses cases in which logic signals are communicated between components by nondigital or nonelectronic means. Examples would be using optical interconnect or capacitative coupling. In these cases, drivers and receivers are considered to be analog circuit blocks and are placed outside the boundary-scan register cells for the relevant logic signals.

The case in which a pair of system pins is used to carry a single logic signal into or out of a component (e.g., as at a differential input or output) is slightly more complex and merits further discussion.

Typically, “paired” I/O will be provided to enhance the performance of connections between components—for example, to enable reliable communication between components in a noisy environment or to reduce skew in high-performance systems. The characteristics that differentiate paired I/O from conventional digital signals are

a) The signals flowing through the pair of pins are driven from or received by a single buffer.
b) The signals should always be connected in pairs if the “enhanced” behavior (e.g., noise rejection) of the chip-to-chip connection is to be maintained.

Two types of paired I/O are commonplace:

— Those that work by altering or sensing the direction of current flow around the loop formed by the two connections.
— Those that appear in many respects to be “conventional” digital signals—for example, because they use TTL-compatible voltage levels.

For current-flow paired I/O signals, the differential input or output buffer should be considered to be an analog circuit. Therefore, one boundary-scan register cell shall be placed between each buffer and the on-chip system logic (figure 10-8).

For paired conventional signals, the location of the boundary-scan register cells at output pins will I depend on the precise characteristics of the link. The following is suggested:

- Where it is possible for the output signals to be used independently (losing the enhanced characteristics of the connection, but retaining the ability to convey the logic signal), two cells should be provided for each driver, one for each output pin. This ensures testability of board-level interconnections where one of the output signals from a pair is used to drive a "conventional" input pin on another component.
- Where the signals that comprise the pair cannot be used independently, then a single boundary-scan register cell should be provided between the system logic and the output buffer.

For a paired "conventional" input, a single boundary-scan register cell has to be placed between the buffer and the on-chip system logic, allowing observation of the logic signal transmitted across the pair.

10.5 Provision and Operation of Cells at System Logic Inputs

This clause provides rules for the provision and operation of boundary-scan register cells at inputs to the on-chip system logic. These inputs may be driven by buffers at system input pins or at bidirectional system pins when they are controlled to operate as system inputs. Alternatively, such inputs may be driven by mixed analog/digital circuit blocks located external to the on-chip system logic.

As discussed in 10.1.1, system logic inputs may be either clock or nonclock. Many of the rules presented in 10.5.1 are common to both types of input. However, some rules apply only to one input type. Where this is the case, the rule, recommendation, or permission is labeled "*For clock (nonclock) inputs only:*."

NOTE — This clause addresses provision of boundary-scan register cells at inputs to the on-chip system logic in eases in which these inputs are driven by system input pins during normal (nontest) operation. The case in which an input to the on-chip system logic is driven by a system bidirectional pin during normal (nontest) operation is discussed in 10.7.

10.5.1 Specifications

Rules

a) Each signal driven to a system logic input (e.g., from an input buffer) shall be capable of being observed by at least one boundary-scan register cell (figure 10-10).

NOTE — The cells may be observe-only or control-and-observe. Where more than one control-and-observe cell is provided to observe a single signal driving a system logic input, rule 10.5.1e applies. Note that such additional cells are redundant in the sense that they could be omitted from the design without jeopardizing compliance to this standard (see 10.8).

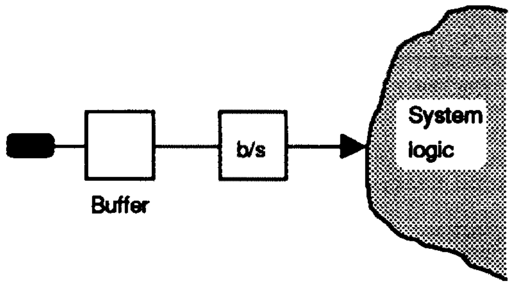


Figure 10-10—Provision of a Boundary-Scan Cell at a System Input

b) Each boundary-scan register cell that can observe a signal driven to a system logic input shall observe precisely one such signal.

c) *For nonclock inputs only*: If the *INTEST* instruction is supported, then each nonclock input *I* to the on-chip system logic shall be driven from precisely one boundary-scan register cell, and this cell shall be one of those that observes the system input signal that drives *I* during normal (nontest) operation.

NOTES:

1 — For clock inputs, provision of a control-and-observe cell is optional (see rule 10.5.1d and permission 10.5.11).

2 — It is permissible for a set of control-and-observe boundary-scan register cells to be distributed in a fanout network from, for instance, one system input buffer to the set of inputs to the on-chip system logic normally driven by that buffer (see figure 10-11). In such cases, rule 10.3.1e applies.

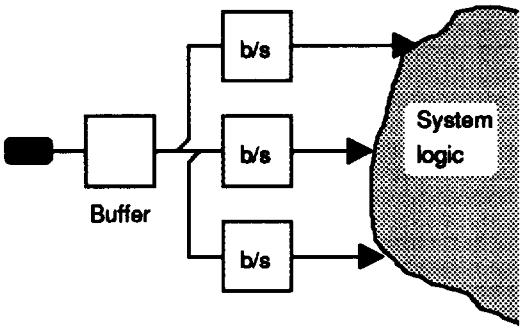


Figure 10-11—Provision of Multiple Boundary-Scan Cells at One Input

d) *For clock inputs only:* If the *INTEST* instruction is supported and one or more control-and-observe cells is provided that observe the system clock input signals of a component, then each system clock input *I* to the on-chip system logic shall be driven from precisely one control-and-observe cell, and this cell shall be one of those that observes the system clock input that drives *I* during normal (nontest) operation.

NOTES:

1 — This rule is activated when the option set out in permission 10.5.11 is exercised.

2 — It is permissible for a set of control-and-observe boundary-scan register cells to be distributed in a fanout network from, for instance, one system input buffer to the set of inputs to the on-chip system logic normally driven by that buffer (see figure 10-11). In such cases, rule 10.3.1e applies.

e) The parallel output of a control-and-observe cell that observes a signal driven to a system logic input shall drive only one of the following:

i) One or more system logic inputs;

ii) An output signal driven to a single system output pin or external analog/digital circuit block; or

iii) The signal driven to the output control of one or more output buffers.

NOTE — It is a consequence of this rule that, where a signal from, for instance, a system input pin would normally fan out to drive more than one of the above options, more than one control-and-observe cell is required. Refer to 10.6 for rules relating to output signals and output controls.

f) Each cell that observes a signal driven to a system logic input shall be designed to route signals as shown in table 10-1.

Table 10-1—Routing of Signals in Cells at System Logic Inputs

Instruction	The signal loaded into the shift-register stage of each cell on the rising edge of TCK in the *Capture-DR* controller state is:	The signal driven from the parallel output of control- and-observe cells while the instruction is selected is:
CLAMP	Not relevant	Not defined*
EXTEST	The signal driven to the system logic input from the external source	Not defined
HIGHZ	Not relevant	Not defined
INTEST	Not defined	*For non clock inputs only:* The parallel output of the shift- register stage *For clock inputs only:* See rule 10.5.1g
RUNBIST	Not defined. (Not relevant unless boundary- scan register is selected as the serial path between TDI and TDO)	*For nonclock input only:* Not defined† *For clock inputs only:* Defined by rule 10.5.1g
SAMPLE/PRELOAD	The signal driven to the system logic input from the external source	The signal received at the connected system pin
BYPASS, IDCODE, USERCODE	Not relevant	The signal received at the connected system pin

*See rule 10.5.1h and permissions 10.5.1n and 10.5.1o.
†See rules 10.5.1h and 10.5.1k and permissions 10.5.1p and 10.5.1q.

g) *For clock inputs only*: When the *INTEST* or *RUNBIST* instruction is selected, the signal driven to the on-chip system logic shall be one of the following:
   i) The signal received at the connected system pin;
   ii) The TCK signal, controlled such that the on-chip system logic changes state only in the *Run-Test/Idle* controller state; or
   iii) For *INTEST* only, the parallel output of the shift-register stage.

   1—Where option 10.5.1g(i) is selected, the component designer should assume that the signal applied to the clock input pin will be free-running and not externally controllable. Therefore, to meet rules 7.8.1b and 7.9.1b, circuitry shall be built into the component to ensure that only appropriate clock transitions cause operation of the on-chip system logic (for example, in the case of the *INTEST* instruction, a "hold" signal may be pulsed internally to ensure single stepping in the *Run-Test/Idle* controller state).

   2—Where a component has more than one clock input pin, the component designer should ensure correct operation of the *INTEST* and *RUNBIST* instructions, if provided, for all frequency and phase relationships between the clock input signals that are permissible for correct nontest operation of the component.

h) The component shall not be damaged as a result of signals fed to the on-chip system logic when the *CLAMP*, *EXTEST*, *HIGHZ*, or *RUNBIST* instructions are selected.

   NOTE — This may be achieved by disabling operation of the on-chip system logic or by designing the cell such that a constant logic signal is output when these instructions are selected.

i) The design of a boundary-scan register cell that observes the signal driven to a system logic input shall be such that, if a logic value *V* is present at the system input pin at the time when data are loaded from the signal into the shift-register stage of the boundary-scan register cell (in the *Capture-DR* controller state), then the value shifted out of the cell through TDO during the immediate subsequent shifting of the boundary-scan register shall be *V*.

   NOTE — For example, where a logic 0 is applied to a system input pin, a logic 0 has to be observed at TDO after loading of the cell that observes that pin. See figure 10-12.

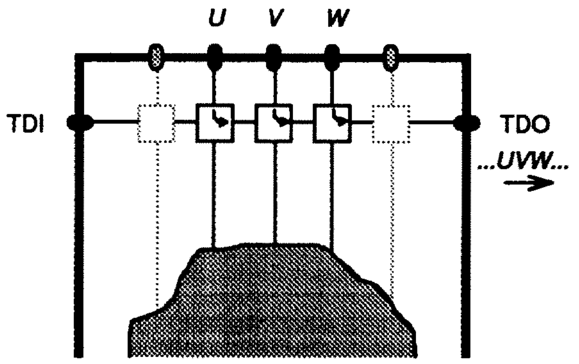


Figure 10-12—Noninversion of Data Between Pin and TDO

j) For each control-and-observe cell that observes a signal driven to the system logic from a system input pin *P*, a data value *D* shifted into the cell through TDI and subsequently driven to the on-chip system logic when the *INTEST* instruction is selected shall, cause the same result as the application of value *D* at *P* during normal (nontest) operation of the component (figure 10-13).

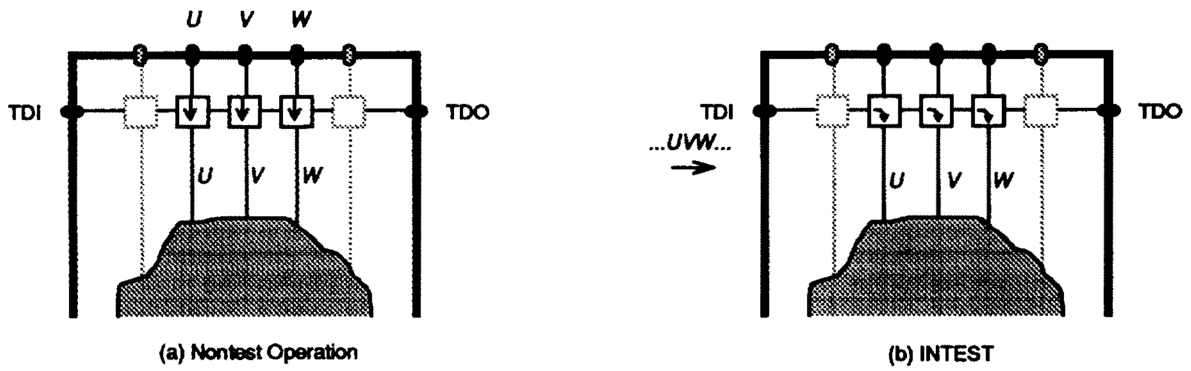


Figure 10-13—Noninversion of Data Between TDI and the System Logic

k) When the *RUNBIST* instruction is provided and selected, the design of boundary-scan register cells that observe signals driven to the on-chip system logic shall be such as to prevent interference with self-test execution as a result of data received from external sources (e.g., system input pins).

Permissions

l) *For clock inputs only*: If the *INTEST* instruction is supported, then precisely one of the provided boundary-scan register cells that observes the signal driven to a clock input to the on-chip system logic may also be able to control that input.

NOTE — See rule 10.5.1d.

m) The shift-register stage of a control-and-observe cell that drives an input to the on-chip system logic may be provided with a latched parallel output.

NOTES:

1 — Where the latched parallel output is omitted, transient data values will be driven to the on-chip system logic during shifting of the boundary-scan register when the *INTEST* instruction is selected. Where this would cause unwanted operation of the on-chip system logic, the parallel output of the shift-register stage can be latched such that the data driven to the system logic changes only on completion of shifting (in the *Update-DR* controller state).

2 — Where latched parallel outputs are provided, rule 10.3.1c and permission 10.3.1g apply.

n) Control-and-observe cells that drive inputs to the on-chip system logic may be designed such that, when the *CLAMP*, *EXTEST*, or *RUNBIST* instruction is selected, the signal driven to the system logic is the parallel output of the shift-register stage.
o) Control-and-observe cells that drive inputs to the on-chip system logic may be designed such that, when the *CLAMP*, *EXTEST*, or *HIGHZ* instruction is selected, a constant signal value is driven to the system logic.
p) Cells may be designed to act as generators of test patterns for the on-chip system logic when the *RUNBIST* instruction (or an alternative self-test instruction) is selected.
q) Cells may be controlled such that, during the execution of *RUNBIST* or an alternative self-test instruction, data may flow between the system input pins and the on-chip system logic without modification.

   NOTE — However, the results of executing the *RUNBIST* instruction will be independent of data received at system input pins (see rule 7.9.1h).
r) Inputs to the on-chip system logic may be observed by one or more observe-only boundary-scan register cells in addition to the cell required by rule 10.5.1a.

   NOTE — Such additional cells are redundant in the sense that they could be omitted from the design without jeopardizing compliance to this standard (see 10.8).

10.5.2 Description

In the example implementations for boundary-scan register cells contained in this chapter, the routing of data through each cell is controlled by one or more mode-control signals (labeled Mode or Mode-N). Different mode-control signals are used for cells at input and output pins of the component, and these are derived from the instruction present at the parallel output of the instruction register.

Example designs for boundary-scan register cells located at system input pins are given in figures 10-14 to 10-18. Note that rule 10.3.1c permits the input to an observe-only boundary-scan cell to be taken from any signal that is transparently driven from the system input via a noninverting path—for example, from a point in a signal distribution tree. Table 10-2 shows the value of the Mode signal for the cells illustrated in figures 10-14 and 10-15 for each of the boundary-scan register instructions defined in Chapter 7.

NOTE — When the *EXTEST* instruction is selected, the cell shown in figure 10-14 feeds data received at the system input pin to the on-chip system logic. In many cases, the on-chip system logic will be tolerant of such signals, which will usually not be representative of those received during normal (nontest) operation. However, in some cases it may be necessary to prevent flow-through of received data (e.g., by adding a logic gate at the output from the cell to the on-chip system logic as shown in figure 10-16). Alternatively, the cell shown in figure 10-18 may be used, which allows user control of the data presented to the on-chip system logic while the *EXTEST* instruction is selected.

Table 10-2—Mode Signal Generation for the Example Cells in Figures 10-14 and 10-15

Instruction	**Mode**
EXTEST	0
SAMPLE/PRELOAD	0
INTEST	1
RUNBIST	X
CLAMP	X

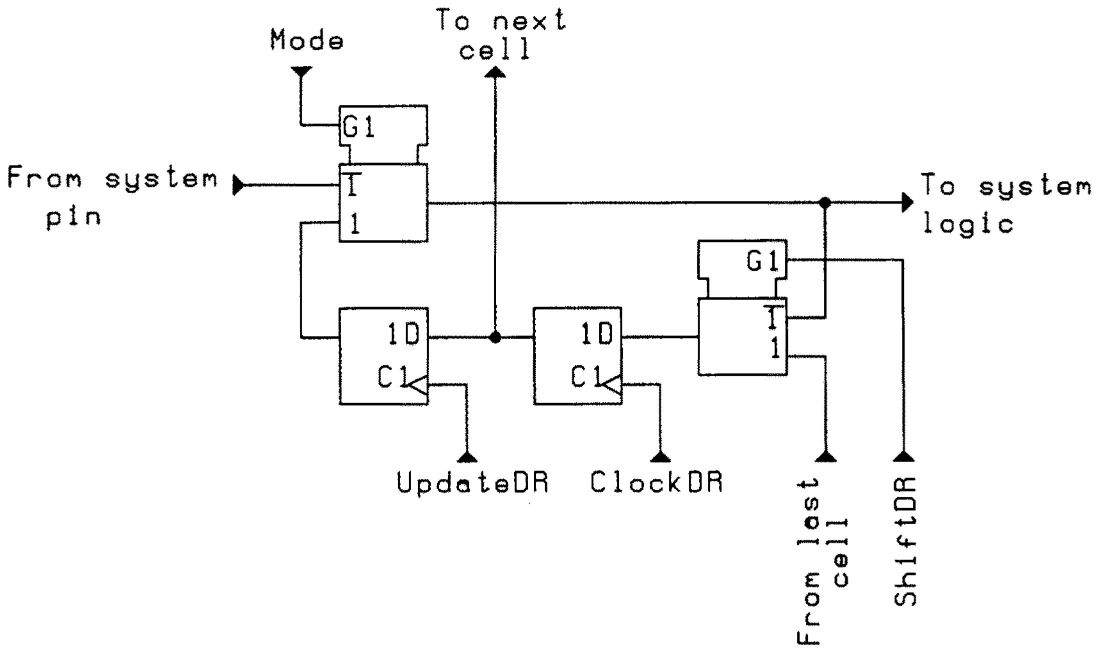


Figure 10-14—An Input Cell With Parallel Output Register

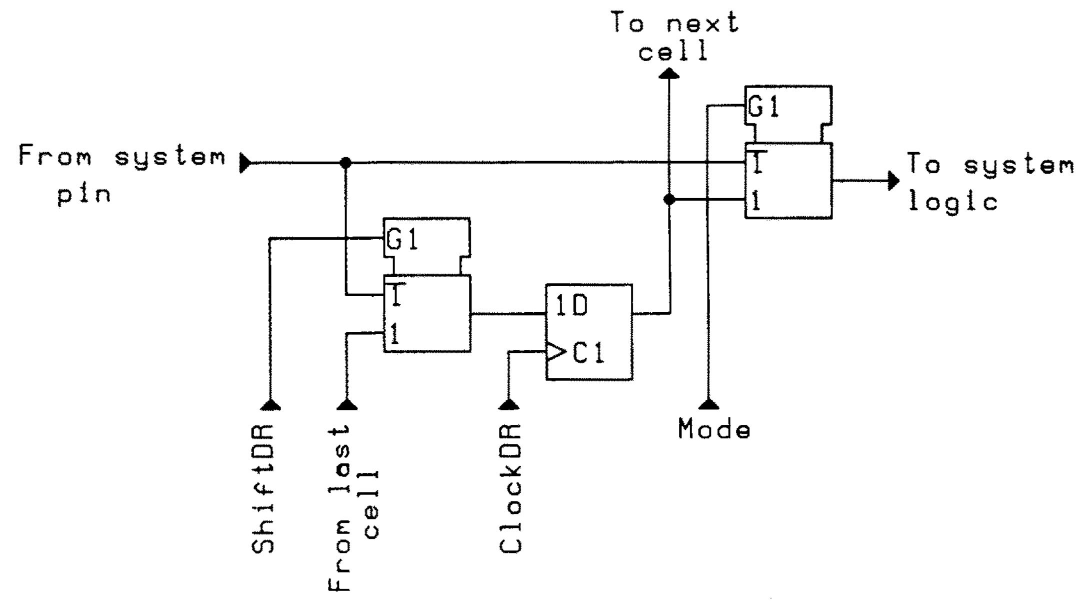


Figure 10-15—An Input Cell Without Parallel Output Register

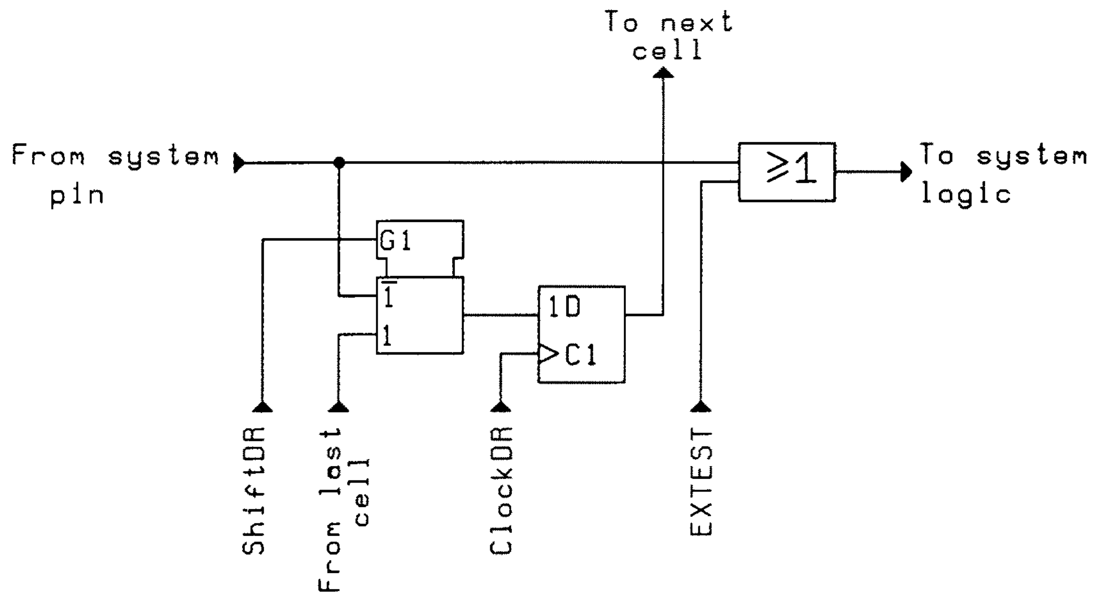


Figure 10-16—A Cell That Forces the System Logic Input to 1 During EXTEST

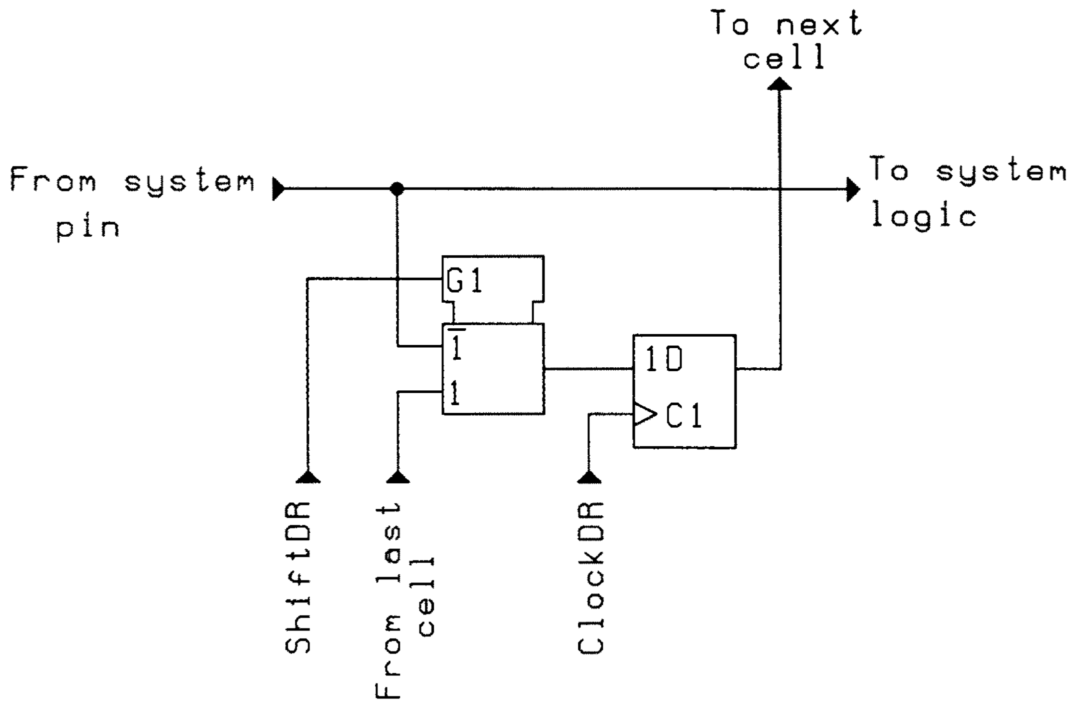


Figure 10-17—An Observe-Only Input Cell

The circuits in figures 10-14 and 10-15 allow the on-chip system logic to be driven from the boundary-scan register cell when the *INTEST* instruction is selected in accordance with rule 10.5.1f and, for clock inputs, rule 10.5.1g(iii). The circuit of figure 10-17 cannot drive signals into the system logic and may be used at a clock input in accordance with rule 10.5.1g(i). The latter design can be used in circumstances where the delay introduced in the signal path by the multiplexer would cause a design target to be exceeded. (An example would be a high-performance clock pin.)

The circuit in figure 10-16 implements permission 10.5.1o.

The design in figure 10-14 includes a parallel output register that is updated from the shift-register stage in the *Update-DR* controller state (permission 10.5.1m). This register is included to prevent the changes at the output of the shift-register stage during shifting from being applied to the on-chip system logic when the *INTEST* instruction is selected (which could cause unwanted operation). Note that the parallel output could alternatively be held in a level-operated latch, enabled by a logic 1 on the UpdateDR input from the example TAP controller (figures 5-5 and 5-6).

The design shown in figure 10-18 can be used for boundary-scan cells located at both system input and 2-state system output pins, although the Mode signal applied to the cell may need to be different in each case. When the cell is used at a system input pin, the Mode signal should be controlled as shown in table 10-3.

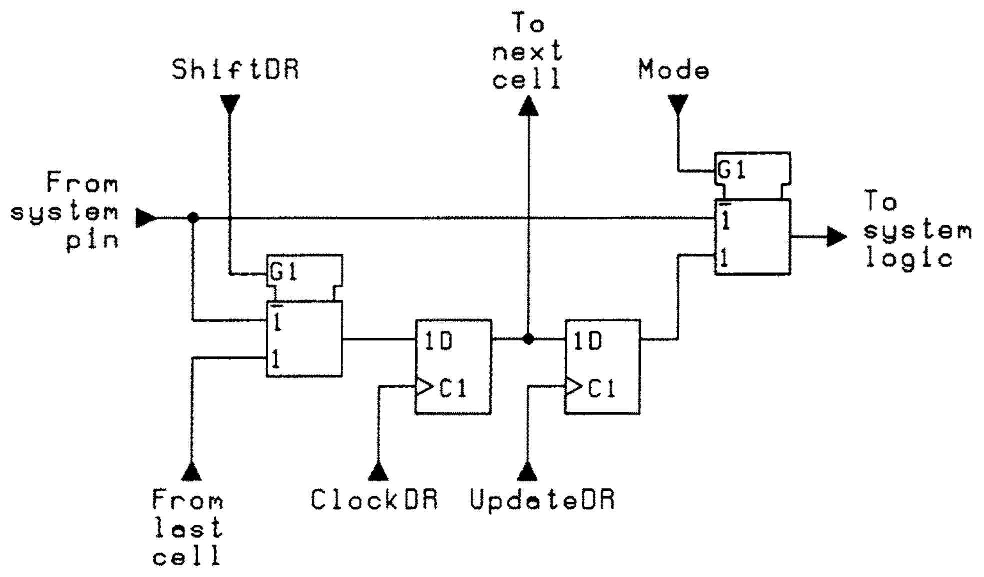


Figure 10-18—A Cell Design That Can Be Used for Both Input and Output Pins

Table 10-3—Mode Signal Generation for the Example Cell in Figure 10-18

Instruction	Mode
EXTEST	1
SAMPLE/PRELOAD	0
INTEST	1
RUNBIST	1
CLAMP	1

Note that the sole difference between the rules that apply to nonclock and clock system inputs is that the provision of control-and-observe cells is not mandatory at clock inputs when the *INTEST* instruction is supported. Thus, there is no requirement for circuitry to be inserted into the signal path between a clock input pin and the on-chip system logic.

10.6 Provision and Operation of Cells at System Logic Outputs

The rules in this clause apply to outputs from the on-chip system logic. These outputs may drive data inputs of buffers at system output pins or at bidirectional system pins when they are controlled to operate as system outputs. They may also drive activity controls of buffers located at such pins. Alternatively, on-chip system logic outputs may drive on-chip mixed-signal circuit blocks that are not a part of the on-chip system logic.

Many of the rules presented in 10.6.1 apply to both control and data signals output from the on-chip system logic. However, some rules apply only to on-chip system logic outputs that drive data inputs of buffers at system output pins, while others apply only to outputs that drive control inputs at such buffers. In cases in which a rule is intended to have limited application to one or the other use of a system logic output signal, the rule is prefaced by the phrase "*Control (Data) inputs to buffers only:*."

NOTE —

1) —Inputs to on-chip analog/digital circuit blocks are considered to be equivalent to data inputs to output buffers.
2) —Where the optional *HIGHZ* instruction is provided, selection of this instruction will place every output pin in an inactive drive state, including pins where there is no system requirement for 3-state capability. Where 3-state capability will be provided solely to allow implementation of the *HIGHZ* instruction, the pin should be treated as a 2-state pin for the purposes of this standard.
3) —This clause addresses provision of boundary-scan register cells at outputs from the on-chip system logic in cases where these outputs drive system output pins during normal (nontest) operation. The case in which an output from the on-chip system logic drives a system bidirectional pin during normal (nontest) operation is discussed in 10.7.

10.6.1 Specifications

Rules

a) Each output of the on-chip system logic (data or control) shall be observed by at least one control-and-observe boundary-scan register cell.
b) *Data inputs to buffers only*: For a given data input to an output buffer, precisely one of the boundary-scan register cells that satisfies rule 10.6.1a shall be capable of driving that data input (see figure 10-19).
NOTE — Rule 10.6.1a provides that at least one cell will monitor any given output from the on-chip system logic. There may be more than one such cell. If, for example, an output from the on-chip system logic fans out to several system output pins, rules 10.6.1b and 10.6.1d require precisely one cell to be capable of driving the connected pin to be placed in each fanout branch. Additional "redundant" observe-only cells may be included in the design (permission 10.6.1r) that can observe the output from the on-chip system logic but that cannot drive any system output pin (see 10.8).

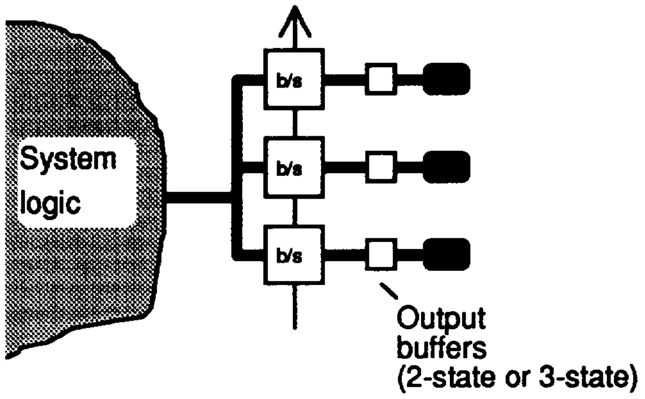


Figure 10-19—Provision of Boundary-Scan Register Cells at System Logic Outputs

c) *Control inputs to buffers only*: For a given control input to an output buffer, precisely one of the boundary-scan register cells that satisfies rule 10.6.1a shall be capable of driving that control input.
NOTE — See also rules 10.6.1d and 10.6.1e.
d) The parallel output of a control-and-observe boundary-scan register cell that observes a system output from the on-chip system logic shall drive only one of the following:
   i) A single data input to a system output buffer (2-state or 3-state, or the output signal for a bidirectional pin); or
   ii) The control inputs to a set of output buffers.

NOTE — In the latter case, see rule 10.6.1e. Note that where a single output from the on-chip system logic is used both as a control for the output buffers at a group of pins and as the data signal output at one or more other pins, then separate boundary-scan register cells are required for the control and non-control signal paths, as illustrated in figure 10-20.

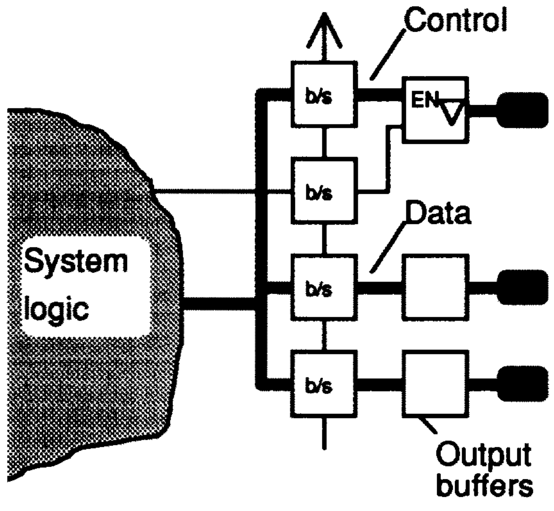


Figure 10-20—Provision of Cells When One Output Is Used Both as Control and Data

e) *Control inputs to buffers only*: Where one boundary-scan register cell drives control inputs to output buffers at several system output pins, a given data value (0 or 1) held in the cell shall cause the same operation at all connected system output pins (e.g., a 0 may be defined to place all connected pins in an inactive drive state).
NOTE — See also recommendation 10.6.1p.

f) In cases where a single control signal is driven to a set of system output pins that includes both 3-state and bidirectional pins, then, in interpreting rule 10.6.1e, enabling the 3-state system output(s) shall be considered to be the same as setting bidirectional system pin(s) to output mode; disabling 3-state system output(s) shall be considered to be the same as setting bidirectional pin(s) to input mode.

g) Each cell that observes a system logic output shall be designed to route signals as shown in table 10-4.

Table 10-4—Routing of Signals in Cells at System Logic Outputs

Instruction	The signal loaded Into the shift- register stage of each cell on the rising edge of TCK in the *Capture- DR* controller state is:	The signal driven from the parallel output of each control-and-observe cell while the instruction is selected is:
CLAMP	Not relevant	The latched parallel output of shift- register stage
EXTEST	Not defined	The latched parallel output of shift- register stage
HIGHZ	Not relevant	The value that disables connected output buffers*
INTEST	The signal output from the on-chip system logic	The latched parallel output of shift- register stage or The value that disables connected output buffers
RUNBIST	Not defined. (Not relevant unless the boundary-scan register is selected as the serial path between TDI and TDO)	The latched parallel output of shift- register stage or The value that disables connected output buffers
SAMPLE/PRELOAD	The signal output from the on-chip system logic	The signal output from the on-chip system logic
BYPASS, IDCODE, USERCODE	Not relevant	The signal output from the on-chip system logic

*This option is available only to cells that drive control inputs of output buffers. Where this option is selected for the control input to an output buffer, the data input to that buffer when the instruction is selected may be regarded as "Not defined".

h) *Control inputs to buffers only*: For each relevant instruction, the option in the righthand column of table 10-4 to drive a value that disables connected output buffers shall be selected either for all cells that drive control inputs of output buffers or for none of these cells.

NOTE — For example, if on selection of the *INTEST* instruction one 3-state output pin of the component was forced to the high-impedance state, then all other 3-state output pins would also be forced to the high-impedance state when the *INTEST* instruction is selected.

i) Every control-and-observe boundary-scan register cell that observes an output from the on-chip system logic (data or control) shall be provided with a latched parallel output.

j) When the *EXTEST*, *SAMPLE/PRELOAD*, or *INTEST* instruction is selected, the latched parallel outputs of control-and-observe boundary-scan register cells that observe outputs from the on-chip system logic shall latch the data held in the shift-register stage on the falling edge of TCK in the *Update-DR* controller state.

k) When the *CLAMP, HIGHZ*, or *RUNBIST* instruction is selected, the latched parallel outputs of control-and-observe boundary-scan register cells that observe outputs from the on-chip system logic shall retain their state unchanged in all controller states.

l) *Data inputs to buffers only*:

— If *C* is the control-and-observe boundary-scan register cell that drives data to an output buffer for system output pin *P*; and

— If *O* is the output of the on-chip system logic that is observed by *C* and that drives data to *P* during normal (nontest) operation; and

— If *C* is the *n*th cell of the boundary-scan register; and

— If *V* is the logic signal driven from *P* in normal (nontest) operation (i.e., when the signal driven from *P* is determined by the output from *O*)

then, when an instruction is selected that requires the output of *O* to be captured into *C* (e.g., the *SAMPLE/PRELOAD* instruction), the logic value observed as the *n*th bit output from the boundary-scan register at TDO in the scan operation immediately following capture shall be *V* (figure 10-21).

NOTES:

1 — For example, where the system logic would drive a logic 0 through the system output pin, a logic 0 has to be observed at TDO after loading of the cell that observes that pin. See figure 10-21.

2 — For outputs where only one logic value is actively driven (e.g., open-collector outputs), receipt of one data value (0 or 1) from the on-chip system logic will cause the the output to be inactive. In these cases, the data value observed at TDO will be the value that, when fed to the output buffer from the on-chip system logic, will cause the output to be inactive.

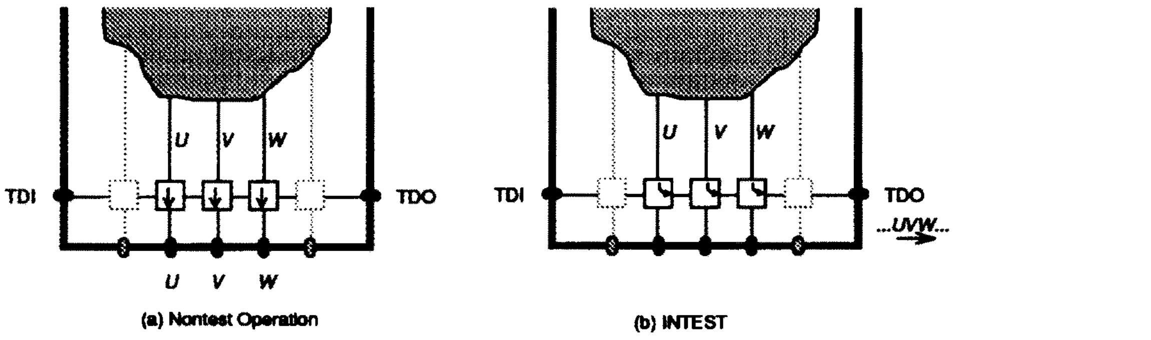


Figure 10-21—Noninversion of Data Between the System Logic and TDO

m) *Data inputs to buffers only*:
- If *C* is the control-and-observe boundary-scan register cell that drives data to an output buffer for system output pin *P*; and
- If *C* is the *n*th cell of the boundary-scan register; and
- If *V* is the value of the *n*th bit of a serial data stream *S* input to the boundary-scan register via TDI; and
- If the length of *S* is equal to the number of cells in the boundary-scan register

then, when an instruction is selected that causes the latched parallel output of *C* to determine the value of the data signal driven from *P* (e.g., *EXTEST*) and when the data stream *S* is shifted into the boundary-scan register and, immediately subsequent to the shifting operation, updated to the latched parallel outputs of the boundary-scan register, the value of the data signal output from *P* shall be *V* (figure 10-22).

NOTE — For outputs where only one logic value is actively driven (e.g., open-collector outputs), receipt of one data value (0 or 1) from the on-chip system logic will cause the the output to be inactive. In these cases, the data value input at TDI shall be the value that, when fed to the output buffer from the on-chip system logic, will cause the output to be inactive.

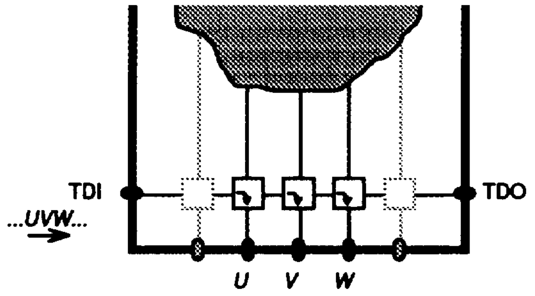


Figure 10-22—Noninversion of Data Between TDI and a System Output Pin

n) *Control inputs to buffers only*:
- If *C* is the control-and-observe boundary-scan register cell that drives the control input of the output buffer for a system output pin *P*; and
- If *O* is the output of the on-chip system logic that is observed by *C* and that controls the activity of *P* during normal (nontest) operation; and
- If C is the *n*th cell in the boundary-scan register; and

— If *V* (not *V*) is the value of the *n*th bit of *S* output from the boundary-scan register through TDO immediately following capture of *O* into *C* when *P* would be active (inactive) in normal (nontest) operation; and

— If the length of *S* is equal to the number of cells in the boundary-scan register

then, when an instruction is selected that causes the latched parallel output of *C* to determine the activity of *P* (e.g., *EXTEST*) and when a data pattern *S* containing *V* (not *V*) as the *n*th bit is shifted into the latched parallel outputs of the boundary-scan register, *P* shall be active (inactive).

NOTE — For example, where a logic 0 output from the system logic would normally disable a driven output buffer, then a logic 0 should be shifted out through TDO whenever the output would have been disabled. Further, a logic 0 shifted into the cell through TDI should, if driven to the output buffer, cause the output buffer to be disabled. The effect of this rule is similar to that of rules 10.6.11 and 10.6.1m (figure 10-23).

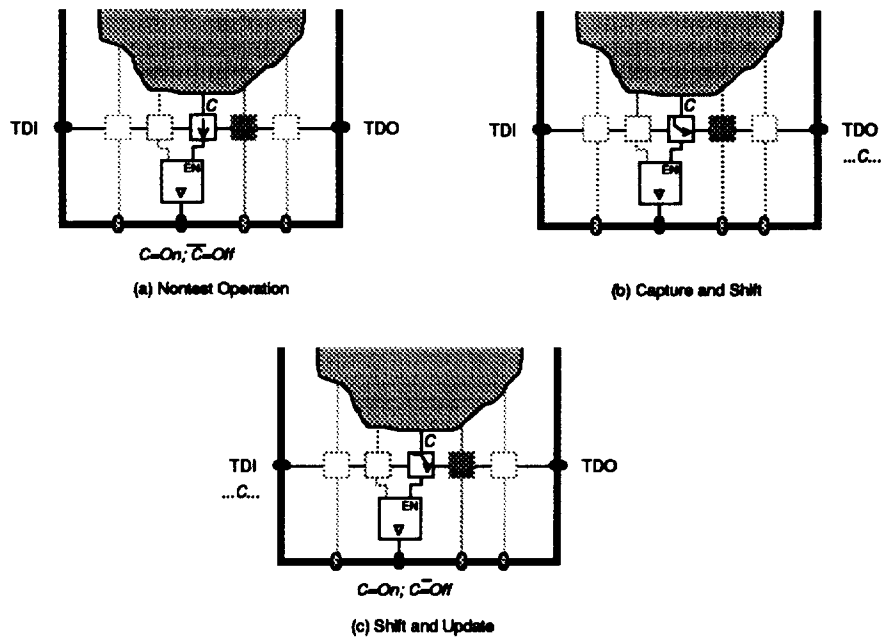


Figure 10-23—Noninversion of Control Signal Values Between the System Logic and TDO

o) *Control inputs to buffers only*: If the latched parallel output of a boundary-scan register cell that drives a control input to an output buffer is reset in the *Test-Logic-Reset* controller state, then it shall be reset to the state that will cause the connected output buffer(s) to be disabled.

NOTE — The timing of the reset is specified in rule 10.3.1c and permission 10.3.1g.

Recommendations

p) *Control inputs to buffers only*: The control signal for each functionally distinct group of system output pins (e.g., an address bus or a data bus) should be driven from a distinct boundary-scan cell dedicated to that purpose, even where the output from the on-chip system logic observed by that cell would normally drive a common control signal to more than one such group of pins.

NOTE — This reduces the complexity of the test generation task because, daring test, buffers can be enabled independently as required.

Permissions

q) Boundary-scan register cells that observe outputs from the on-chip system logic may be designed to act as a part of a signature computing register for test results when the *RUNBIST* instruction (or alternative self-test instruction) is selected.

r) Outputs from the on-chip system logic may be observed by one ar more observe-only boundary-scan register cells in addition to the control-and-observe cells required by the preceding rules.

NOTE — Such additional cells are redundant in the sense that they could be omitted from the design without jeopardizing compliance to this standard (see 10.8).

10.6.2 Description

For 2-state output pins, where signals can only be at the high or low logic level at any given instant, one boundary-scan register cell is sufficient to allow the state of the pin to be controlled or observed. However, for 3-state pins the capability exists for data to be driven actively or inactively, such that four states are possible. Data from a minimum of two boundary-scan register cells are therefore required to allow the state (signal value plus active/inactive) of a 3-state pin to be controlled or observed.

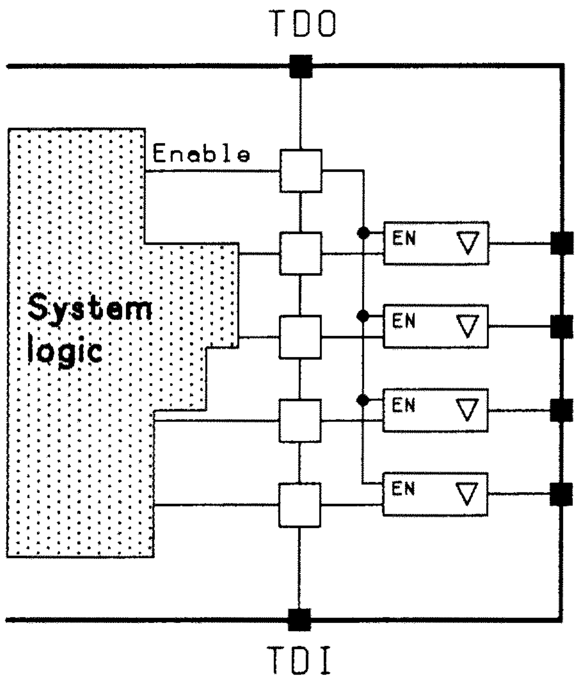


Figure 10-24—Control of Multiple 3-State Outputs From One Signal

Although it would appear that the additional cells might significantly increase the overhead needed to implement boundary-scan, it is only necessary to provide one additional cell for each 3-state enable signal generated in the circuit, although a judicious use of a few additional cells is recommended in 10.6.1p (see, for example, figure 10-24). Thus, where many 3-state output pins are controlled from a single source, as for example in a microprocessor address bus, only one additional cell is required to give the necessary control. Since, given the basic design of the circuit, it would be a design error if such 3-state pins were wired together, there is no need for the design of the boundary-scan register to account for this possibility.

The need for the additional cells at output enable signals is illustrated through figure 10-25. This shows a wired junction between 3-state outputs from a number of components. To test this junction, a series of tests shall be performed, each of which checks that one of the outputs can drive either a 0 or a 1 to the receiving devices. During each test, the other outputs have to be set to the opposite data value (1 or 0 respectively) with a high-impedance drive. Table 10-5 shows the pair of tests needed to check the operation of one of the outputs connected to the junction.

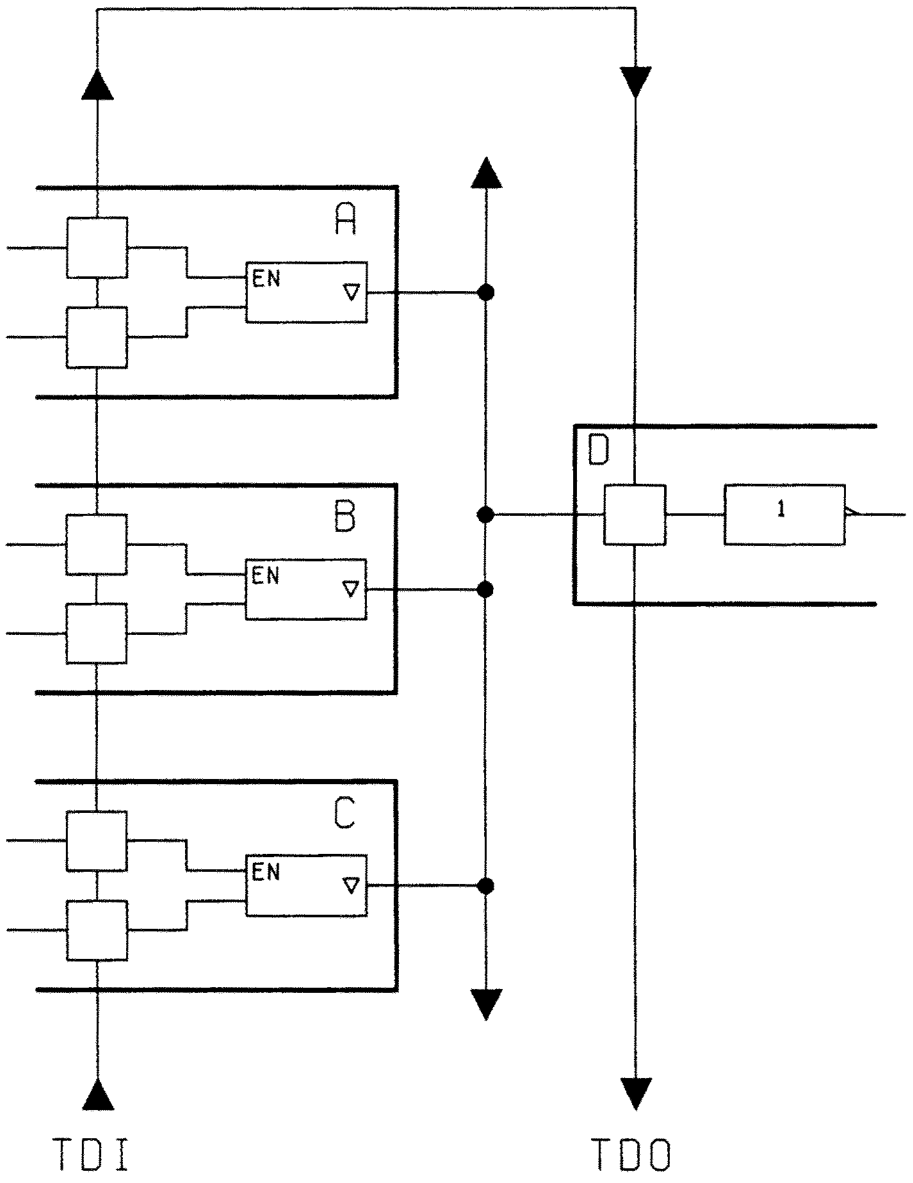


Figure 10-25—Testing Board-Level Bus Lines

Table 10-5—Test for Driver B

Stimulus applied to the bus from			Result seen at
Component A	**Component B**	**Component C**	**Component D**
1/off	0/on	1/off	0
0/off	1/on	0/off	1

To apply the test, it is necessary to be able to control both the data value at each output and whether the output is enabled. This can be done via the boundary-scan register independently of the on-chip system logic.

For similar reasons, there shall be additional boundary-scan cells associated with each 3-state bidirectional system pin that control whether it operates as input or output. As in the case of 3-state system pins, these cells may be shared across a bus or between any group of 3-state bidirectional system pins that obtain their direction control signal from a single source.

Figure 10-26 highlights the following problems that might be encountered when applying tests to logic blocks external to a component by using the boundary-scan register of the component, but that are avoided by implementing boundary-scan cells as defined in this clause.

— The logic block being tested may contain asynchronous sequential logic that will be set into undesirable states if shifting patterns appear at its inputs.

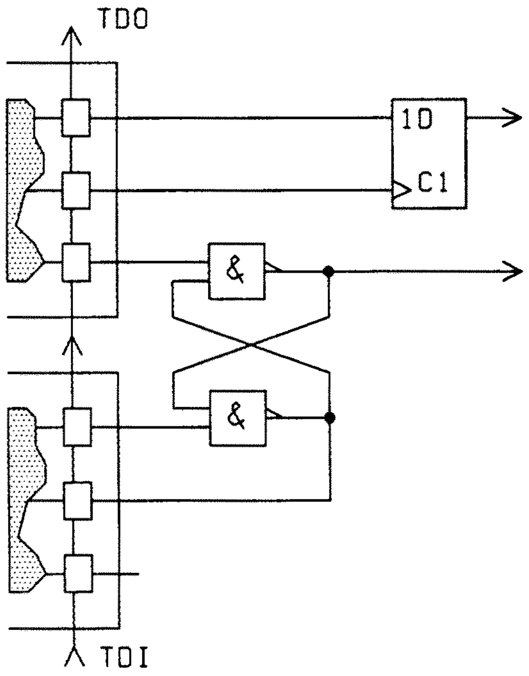


Figure 10-26—Testing External Logic via the Boundary-Scan Path

— The signals applied from the boundary-scan register may feed into clock inputs on the logic block being tested, which again will produce undesirable effects if the logic is not shielded from shifting patterns.

Since, in a generally applicable architecture, it cannot be guaranteed that such features do not exist in the circuitry under test, the boundary-scan design shall be such that these problems are guaranteed to be avoided. A design compatible with this standard ensures this by requiring a parallel output register or latch in each boundary-scan cell that can effect the state of an output driver at a system pin. The inclusion of this register or latch ensures that, while the *EXTEST* instruction is selected, the data driven from a component to neighboring circuitry changes only on completion of the shifting process.

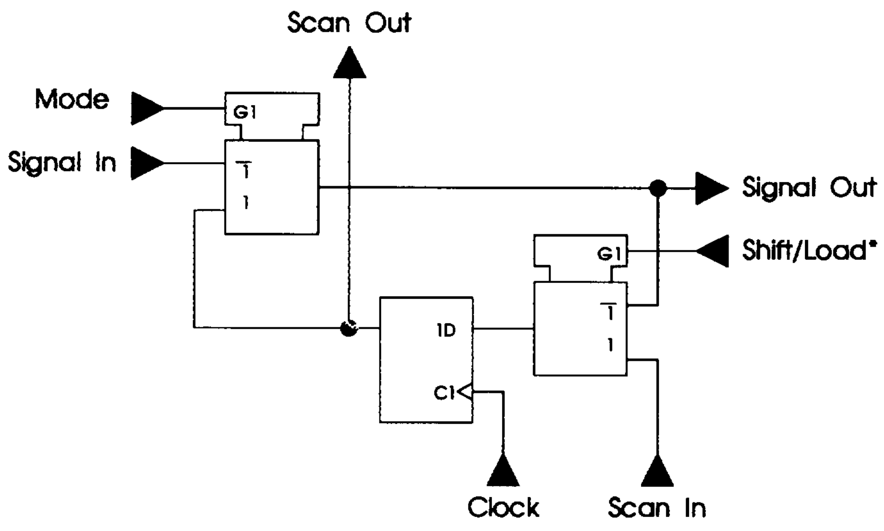


Figure 10-27—A Primitive Noncompliant Output Cell Design With Potential Problems

A further potential problem is highlighted by the primitive (noncompliant) boundary-scan cell design shown in figure 10-27. During testing of the on-chip system logic (for example, through the *INTEST* or *RUNBIST* instruction), the example cell would allow responses from the system logic to pass through the data-path multiplexer to the shift-register input of the cell. This allows the output response from the on-chip system logic to be loaded into the output boundary cells and shifted out for inspection. However, a problem arises from the fact that the cell also allows the test response from the on-chip system logic to be output from the host component and, hence, to be applied to neighboring components on a board assembly.

The application of raw test-response data from one component could have a damaging effect on other components in the circuit if it is received at clock or asynchronous data or control inputs. For example, if built-in self-testing were being performed on the memory controller of figure 10-28, there is a distinct possibility that one or more test-response patterns from the core logic of the memory controller will cause simultaneous activation of the outputs feeding the chip select (CS) inputs of the memory devices. This situation would not occur during normal operation of the complete design, either due to constraints between the logic values applied to the inputs of the memory controller or due to the design of the on-chip system logic. The design would in some way ensure that only one output from the controller was active at any time.

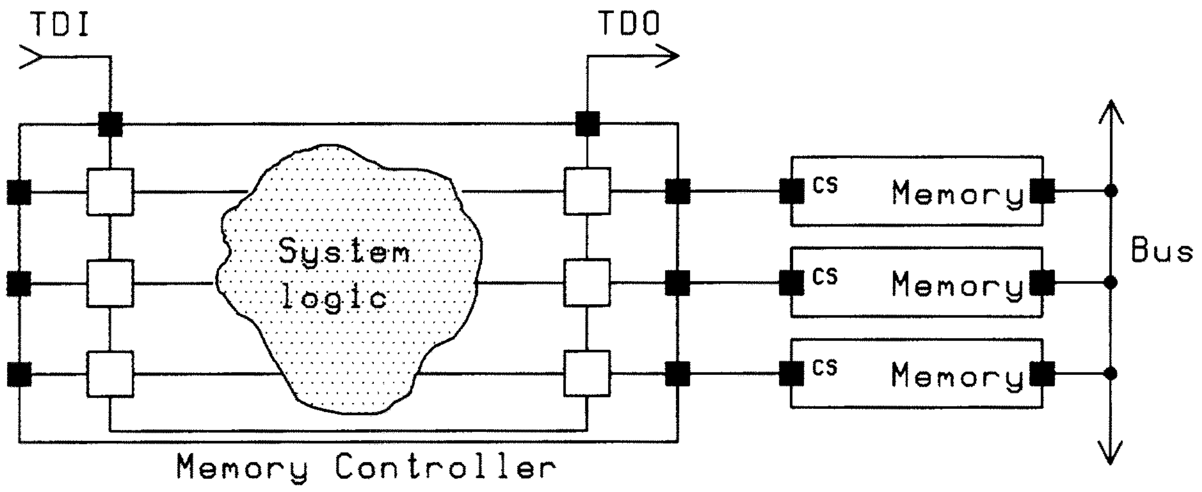


Figure 10-28—A Circuit Illustrating Potential Boundary-Scan Test Problem

The duration of an on-chip test is dependent on the type of system logic test performed. For static tests applied using the *INTEST* instruction, these potentially damaging output patterns can remain in effect over the interval between successive occurrences of the *Update-DR* controller state. For instance, in a circuit having a scan path length of 500 bits and a TCK rate of 5 MHz, the approximate interval between closest consecutive *Update-DR* controller states is 100 ms. For large board designs, the period could be sufficiently long to cause damage to drivers in contention on a bus.

One solution is to cause the output buffers of the memory controller that feed the memory CS inputs to be placed in a high impedance state during internal testing of the controller. However, floating inputs can fluctuate between high and low logic levels and are susceptible to induced voltages from adjacent board wiring interconnects. Applying a pull-up resistor on the 3-state buffers will solve the bus contention problem in external components with active-low 3-state enables, but those with active-high 3-state enables, such as the 74xx241, are still at risk.

The solution adopted in this standard is to ensure that boundary-scan register cells placed at 2-state output pins are designed such that user-defined logic values can be placed at the associated pins while system logic within the component is tested. Figure 10-29 shows a design that provides this facility and meets the rules defined in this clause. Table 10-6 shows how the Mode signal for figure 10-29 is derived for each of the boundary-scan register instructions defined in Chapter 7.

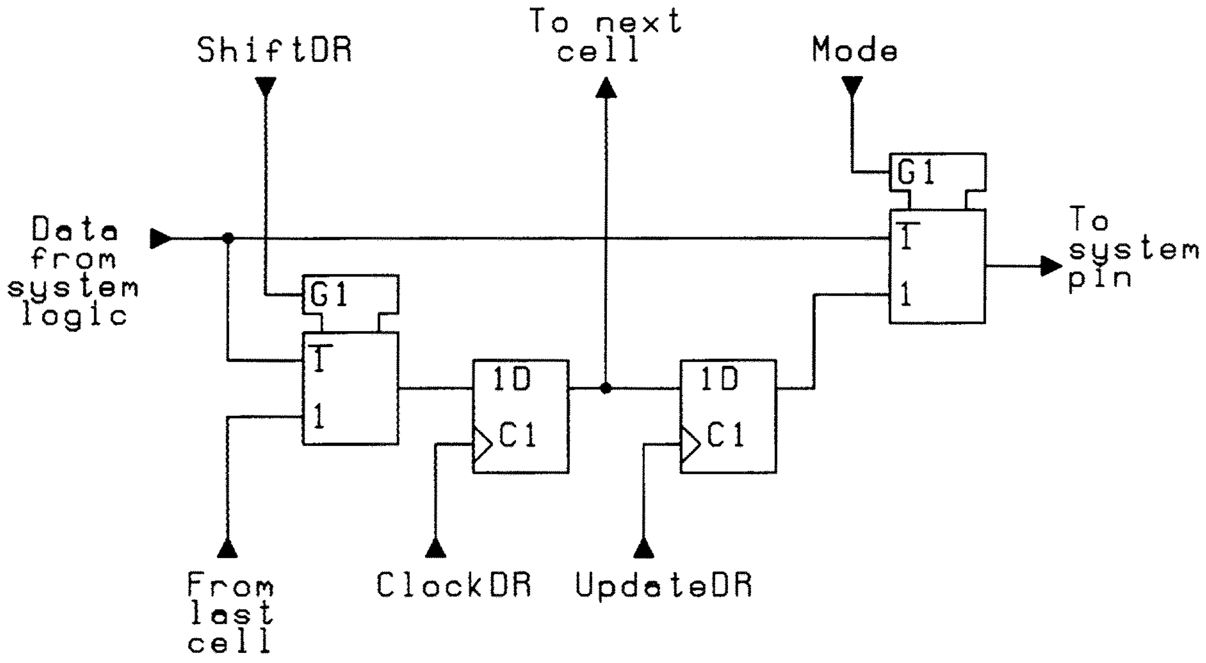


Figure 10-29—An Output Cell That Supports All Instructions

Table 10-6—Mode Signal Generation for the Example Cells in Figures 10-29, 10-31, 10-33, and 10-41

Instruction	Mode
EXTEST	1
SAMPLE/PRELOAD	0
INTEST	1
RUNBIST	1
CLAMP	1

Note that the path in figure 10-29 between the data input from the system logic and the multiplexer that feeds data to the system pin will not be used during execution of either the *EXTEST* or the *INTEST* instruction. In some cases, it may therefore be necessary to use additional test operations at the board level to test the circuitry within a component fully.

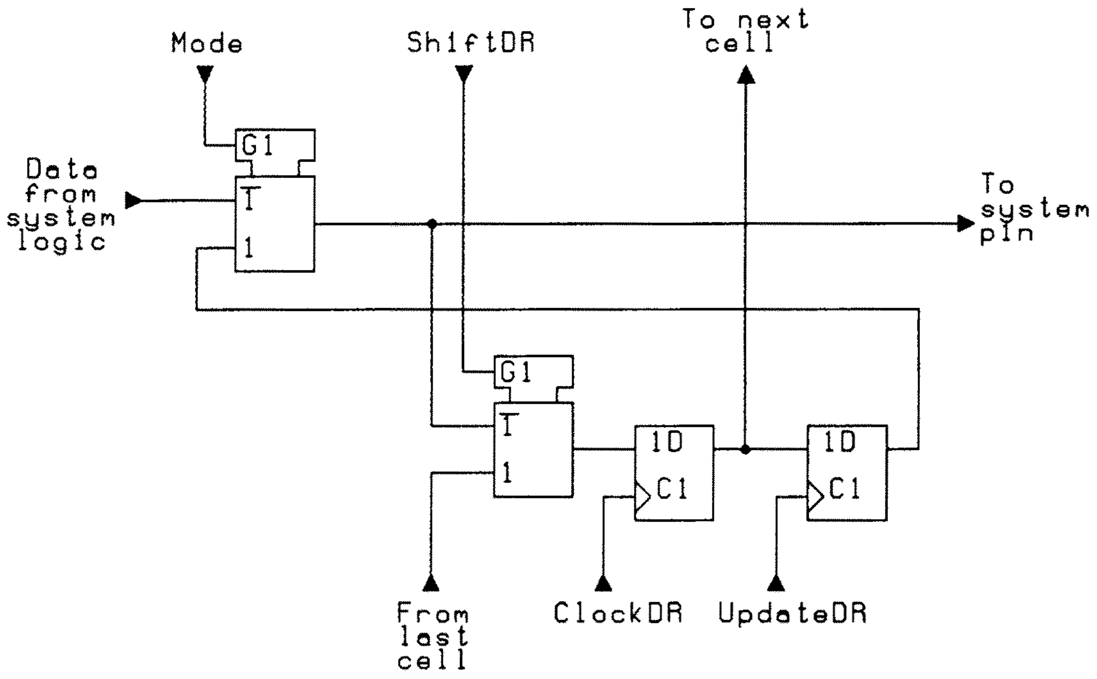


Figure 10-30—An Output Cell That Supports SAMPLE/PRELOAD, EXTEST, and RUNBIST

The example cell design of figure 10-30 could be used where the *INTEST* instruction is not supported by a component, since this design does not permit rule 10.6.1g to be met. Note that for rule 10.6.1g to be met in respect to the *SAMPLE/PRELOAD* instruction, the cell shall drive off-chip via an output buffer. This is necessary to ensure that the signal value captured using the *SAMPLE/PRELOAD* instruction is that intended to be driven off-chip, not that actually on the off-chip connection at the time. The latter may be affected by faults on the off-chip connection or, for bus connections, by the combination of drivers active at the time. By ensuring that the signal that should have been driven from the chip is sampled at the driving end, while the signal actually driven is sampled at the receiving end, additional diagnostic information is obtained.

Table 10-7 shows how the Mode signal for figure 10-30 is derived for each of the boundary-scan register instructions supported by the cell design.

Table 10-7—Mode Signal Generation for the Example Cells in Figure 10-30

Instruction	Mode
EXTEST	1
SAMPLE/PRELOAD	0
RUNBIST	1
CLAMP	1

Where a component has 3-state system output pins, these may feed onto a wired junction at the board level. In order to test the interconnections forming the wired junction using the *EXTEST* instruction, it shall be possible to drive independently onto the junction from each of the possible driving pins. As was discussed earlier in this clause, to achieve this it is necessary to be able to control the output enable or direction control signals fed to the output drivers at 3-state or bidirectional system pins.

In addition, it is necessary to ensure that contention does not occur on board-level interconnections when the on-chip system logic is tested using the *INTEST* or *RUNBIST* instruction. This requirement can be met in either of two ways:

a) The state of a system pin can be fully defined by the user by shifting data into the boundary-scan register.
b) A system pin can be forced into the inactive drive state. This additional option is possible since the board-level circuit design shall necessarily be designed such that components driven from the 3-state bus do not erroneously respond to high-impedance conditions during normal system operation. Therefore, the inactive drive state can be safely driven during testing of the system logic within a component.

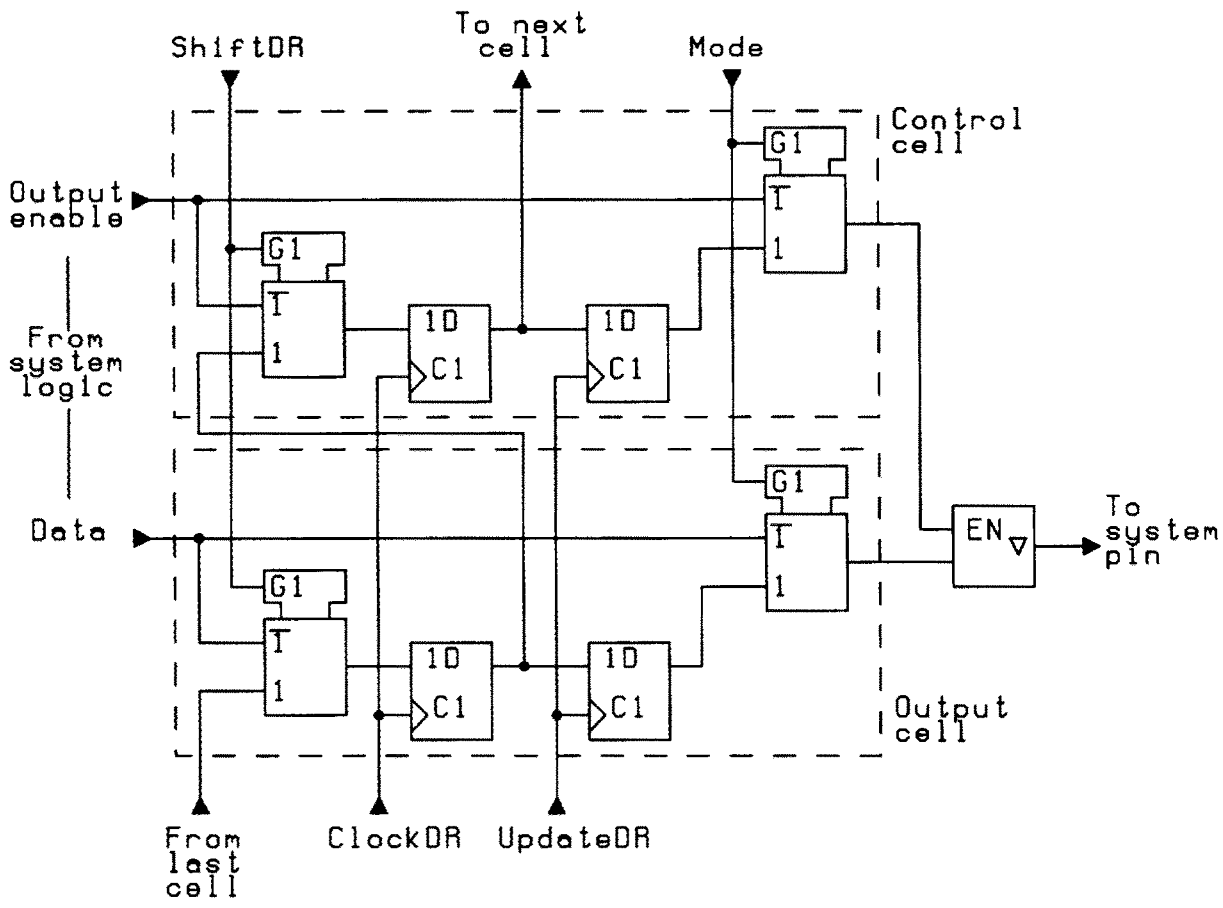


Figure 10-31—Boundary-Scan Cells at a 3-State Output—Example 1

The options listed for the *INTEST* and *RUNBIST* instructions in rale 10.6.1g cover these two possibilities. Figures 10-31 and 10-32 give example designs for a boundary-scan register cell that could be used at a 3-state system output pin. Figure 10-31 implements option (a), while figure 10-32 implements option (b). In figure 10-31, the Mode signal should be controlled as shown in table 10-6.

In figure 10-32, the design of the circuitry around the shift-register stages is such that all paths can be tested if both the *EXTEST* and *INTEST* instructions are executed with appropriate data. The Mode_1 and Mode_2 signals should be controlled as shown in table 10-8.

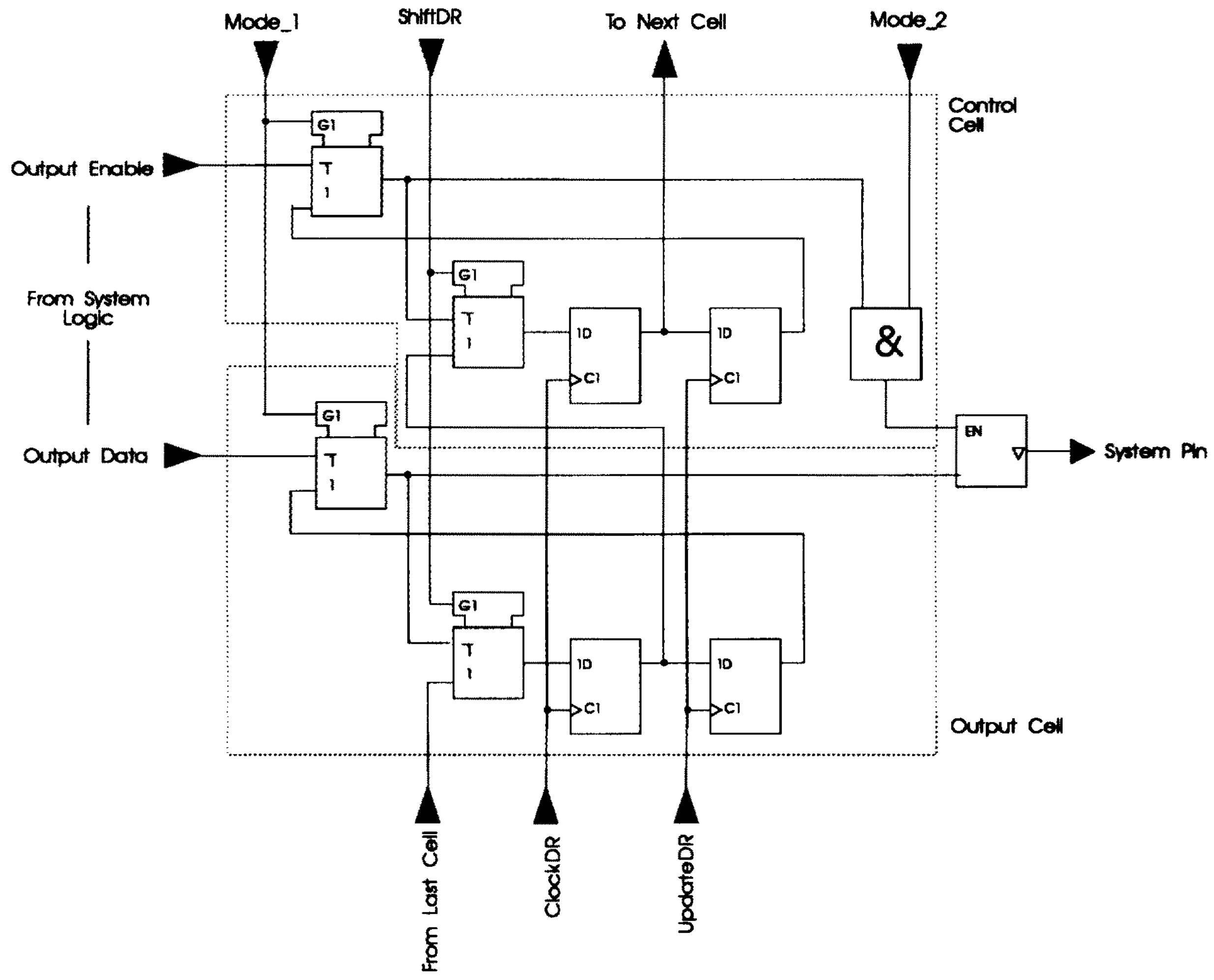


Figure 10-32—Boundary-Scan Cells at a 3-State Output—Example 2

Table 10-8—Mode Signal Generation for the Example Cell in Figure 10-32

Instruction	Mode 1	Mode 2
EXTEST	1	1
SAMPLE/PRELOAD	0	1
INTEST	0	0
RUNBIST	0	0
CLAMP	1	1
HIGHZ	0	0

10.7 Bidirectional Signals

10.7.1 Specifications

Rules

a) Boundary-scan register cells shall be provided at bidirectional system pins such that
   i) Whenever the pin is an input pin, all rules are met for cells provided at system input pins and inputs to the on-chip system logic (see 10.5).
   ii) Whenever the pin is an output pin, all rules are met for cells provided at outputs of the on-chip system logic that drive data inputs of system output buffers (see 10.6).
   iii) All rules are met for cells provided at outputs of the on-chip system logic that drive control inputs of buffers at system output pins (see 10.6).

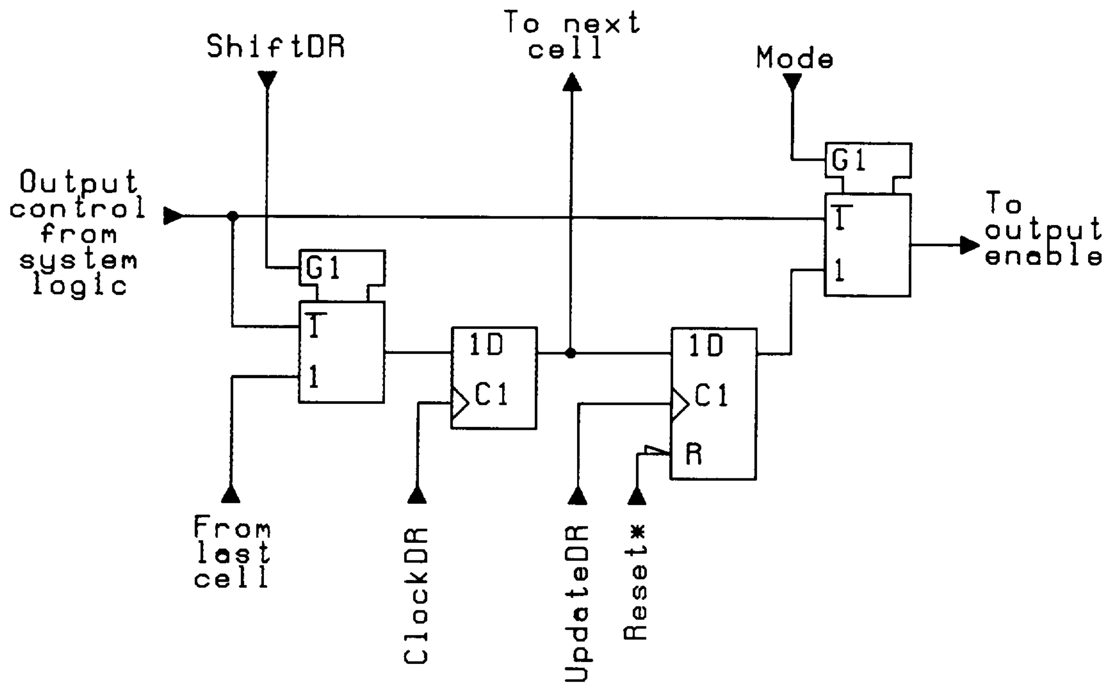
NOTE — In cases where the direction of signal flow is determined by an output *O* of the on-chip system logic, a boundary-scan register cell will exist in the signal path between *O* and the system pin. When the *EXTEST*, *CLAMP*, *INTEST*, or *RUNBIST* instruction is selected, the direction of signal flow will be determined by the data held in the latched parallel output of the shift-register stage of the boundary-scan register cell.

10.7.2 Description

These requirements represent a merging of those for 2-state or 3-state output pins with those for system input pins.

Figures 10-33, 10-34, and 10-35 give examples of the provision of boundary-scan register cells at 3-state bidirectional pins.

a) Figure 10-33 allows the state of the pin to be fully controlled while the *INTEST* or *RUNBIST* instruction is selected. The Mode signal shown in figure 10-33 should be controlled as indicated in table 10-6. The Reset* signal is fed from the example TAP controller of figure 5-5 to the parallel output register of the direction control cell in accordance with permission 10.3.1g.



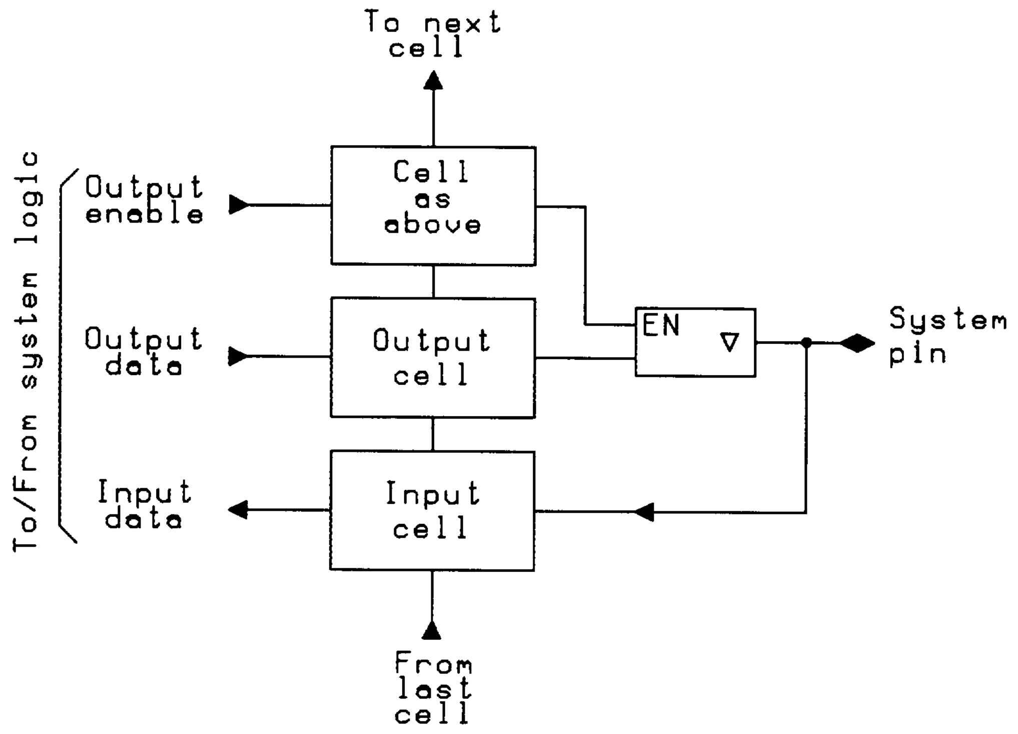


Figure 10-33—Boundary-Scan Cells at a Bidirectional Pin—Example 1

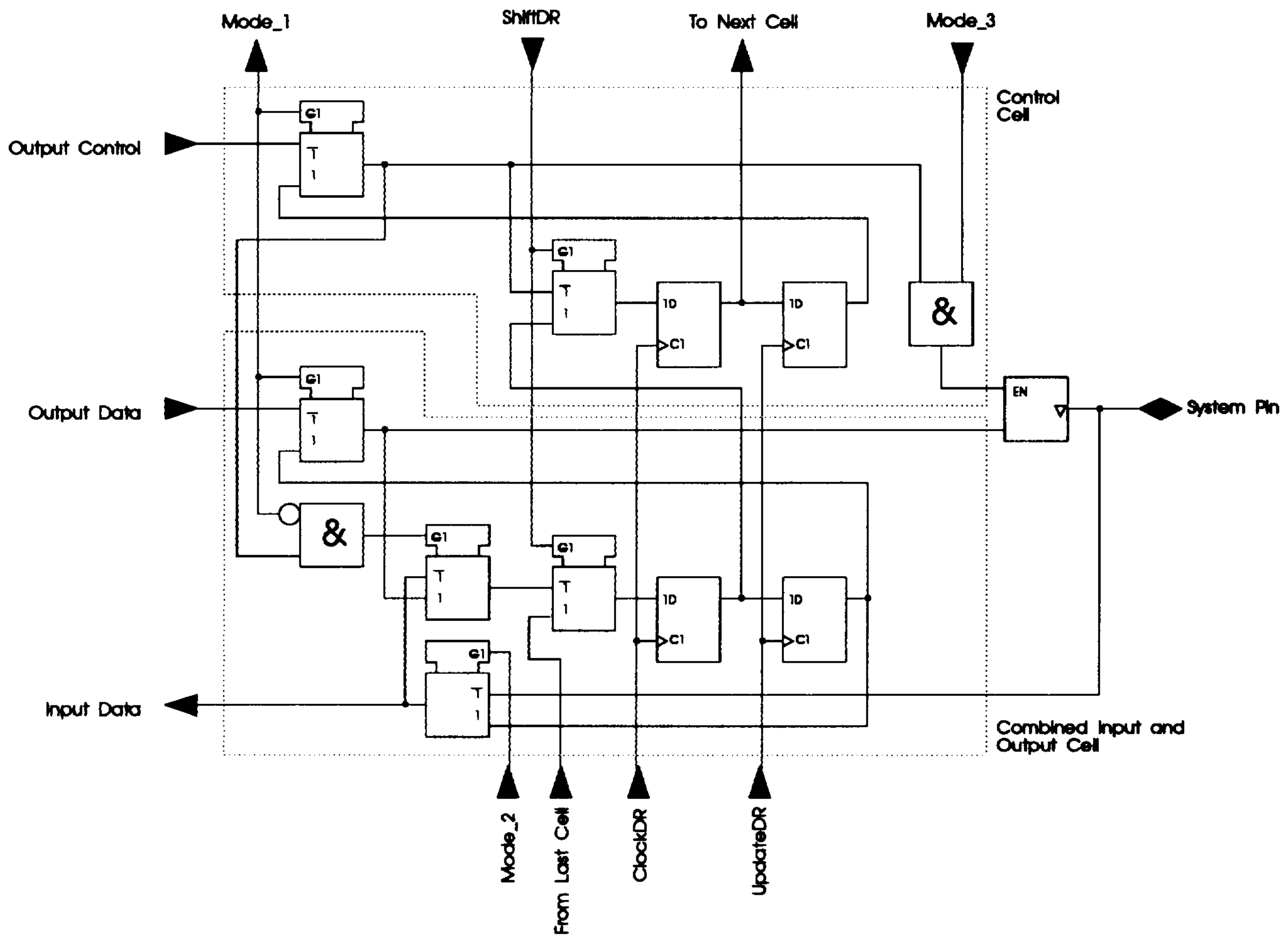


Figure 10-34—Boundary-Scan Cells at a Bidirectional Pin—Example 2

Table 10-9—Mode Signal Generation for the Example Cells in Figure 10-34

Instruction	Mode 1	Mode 2	Mode 3
EXTEST	1	0	1
SAMPLE/PRELOAD	0	0	1
INTEST	0	1	0
RUNBIST	X	X	0
CLAMP	1	X	1
HIGHZ	X	X	0

b) In figure 10-34, a single boundary-scan register cell is used to control and observe both output and input data. This cell meets the requirements of 10.6.1 when the pin is defined to be an output and the requirements of 10.5.1 when it is defined to be an input. The various control signals used by the cell should be controlled as shown in table 10-9. As discussed in connection with figure 10-32, the design of the circuitry around the shift-register stages in figure 10-34 permits all circuitry in the cell to be tested if the *EXTEST* and *INTEST* instructions are executed with appropriate data.

c) Figure 10-35 is similar to figure 10-34. Note that, while this design conforms fully to the rules set out in this standard, it is not recommended for use in new component designs. This is because the combined input and output cell does not capture as much data as possible about the external interconnection when the *EXTEST* instruction is selected. The example design of figure 10-34 is superior in this respect.

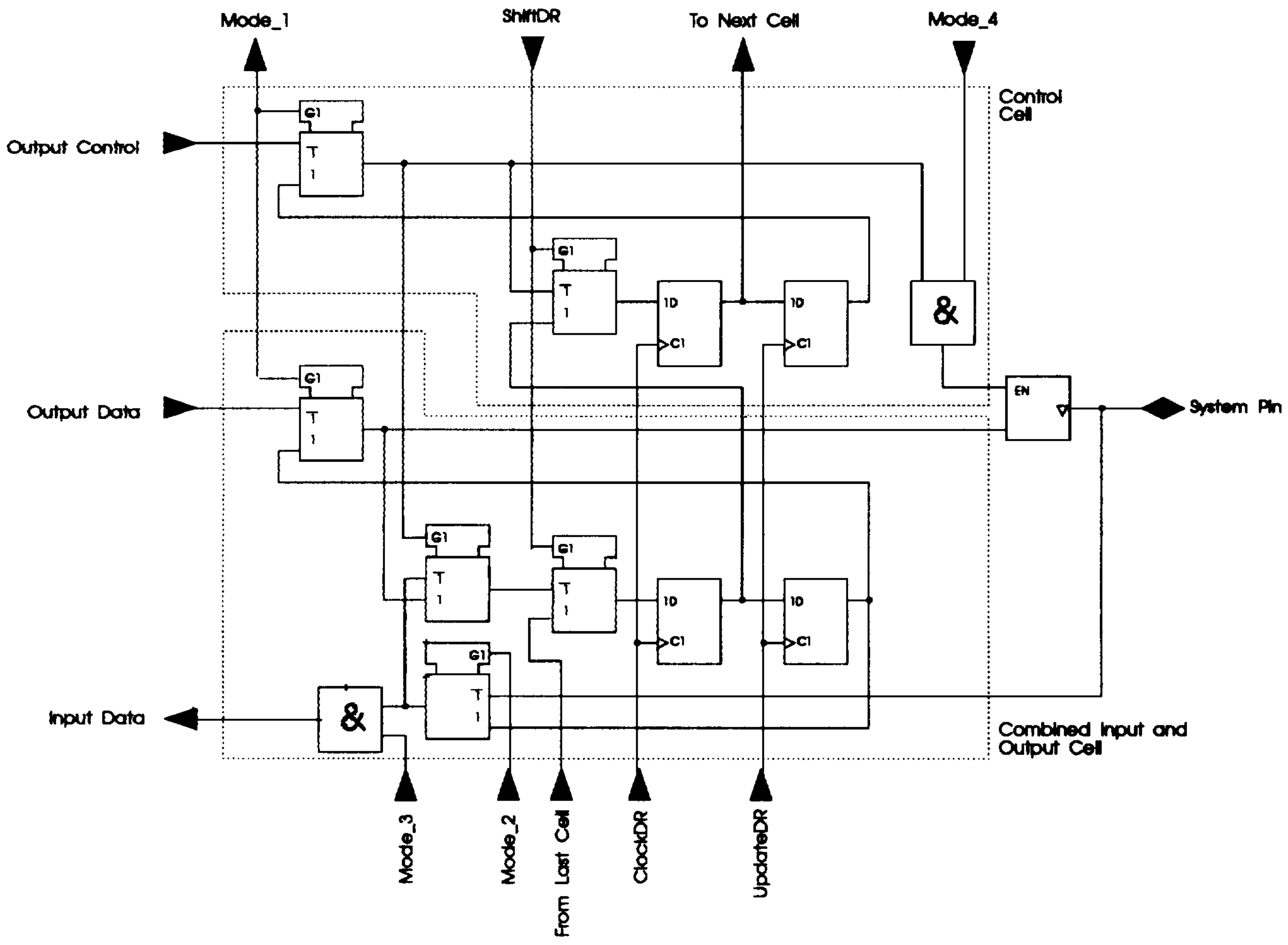


Figure 10-35—Boundary-Scan Cells at a Bidirectional Pin—Example 3

Table 10-10—Mode Signal Generation for the Example Cells in Figure 10-35

Instruction	Mode 1	Mode 2	Mode 3	Mode 4
EXTEST	1	0	0	1
SAMPLE/PRELOAD	0	0	1	1
INTEST	0	1	1	0
RUNBIST	X	X	X	0
CLAMP	1	X	0	1
HIGHZ	X	X	0	0

Figure 10-36 shows how boundary-scan register cells may be provided at an open-collector bidirectional pin.

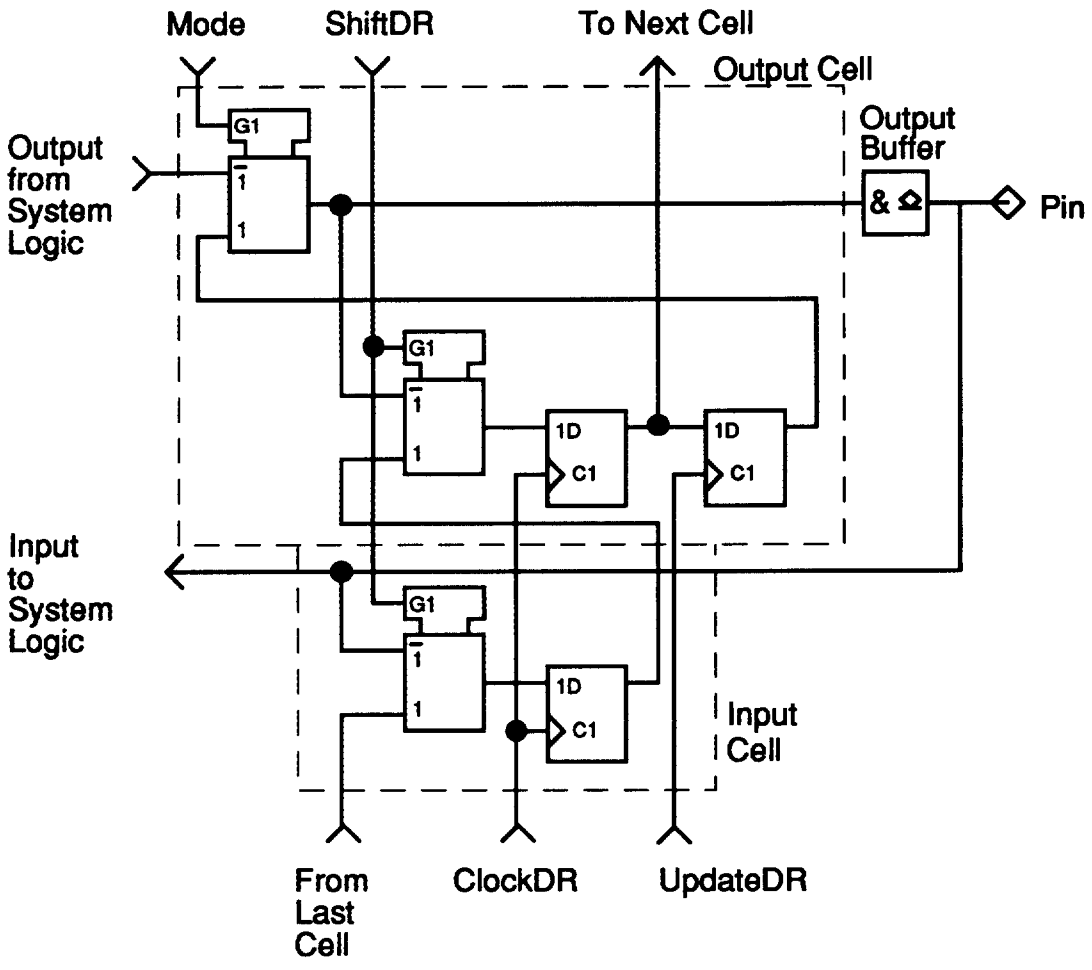


Figure 10-36—Boundary-Scan Cells at an Open-Collector Bidirectional Pin

10.8 Redundant Cells

Redundant cells may exist in a component design for a number of reasons. For example:

a) They may be observe-only cells that observe a signal (input or output) that is observed by another boundary-scan register cell.
b) (b) They may be parts of boundary-scan register cells designed for bidirectional system pins, in cases where the pin has been programmed or otherwise customized to be permanently an input pin or an output pin. For example, a programmable component may be provided with three boundary-scan register cells at each system pin, sufficient to permit each system pin to be programmed as an input pin, 2-state or 3-state output pin, or bidirectional pin. Following programming, certain of these cells may not be logically connected either to a given system pin or to a system logic input or output or both. Alternatively, a vendor of application-specific components may build a boundary-scan register into the basic component design (i.e., the design before the component is "committed") that provides for a fully bidirectional signal at each possible system pin. When the basic component is "committed," these cells will be constrained such that only the required functionality is connected.

10.8.1 Specifications

Rules

a) The results of tests performed using the boundary-scan register shall not be dependent on the value held in a redundant boundary-scan register cell at the start of the test.

Recommendations

b) The number of redundant boundary-scan register cells included in a component should be minimized.
c) Redundant cells should be designed such that the data shifted out through TDO following loading of the shift-register stage in the *Capture-DR* controller state is either a constant or the data just previously shifted into the cell.

10.8.2 Description

Some programmable components (e.g., programmable gate arrays or application-specific ICs) offer input/output circuits that can be programmed as input, output, 3-state, or bidirectional pins. To permit programming as a 3-state or bidirectional pin, two or more boundary-scan register cells would need to be included in each configurable cell to allow access to the data and control signals. However, when the cell is programmed as an input or 2-state output pin, only one cell will be required. In some implementations, the cells not associated with the programmed system function of a pin may be logically disconnected from the pin and from the system logic. Under such circumstances, the disconnected cells could no longer be used during testing and would became redundant. Rule 10.2.1h requires that the unused cells remain in the boundary-scan register so that the register has a fixed length regardless of how the component is programmed.

NOTE — In many programmable devices, programmed lack of logical connection(s) may occur only with regard to a boundary-scan register cell and the on-chip system logic. The cells provided for a particular programmable pin may remain logically connectable to that pin during testing, and the bidirectional control cell would then remain functional. The rules of this chapter do not prohibit this "excess" functionality at a pin. Indeed, interconnect test generation may actually be easier when all pins on a board-level net *appear* from *outside* the components to be provided with full bidirectional boundary-scan capability.

To minimize the number of redundant cells contained in the boundary-scan register of a component, the register should only contain cells that, in some programmed configuration of the component, can provide access to signals at the boundary of the on-chip system logic. For example, a cell that receives its parallel data input from the on-chip system logic and sends its parallel data output into the on-chip system logic should not be included in the boundary-scan register (e.g., as shown in figure 10-37).

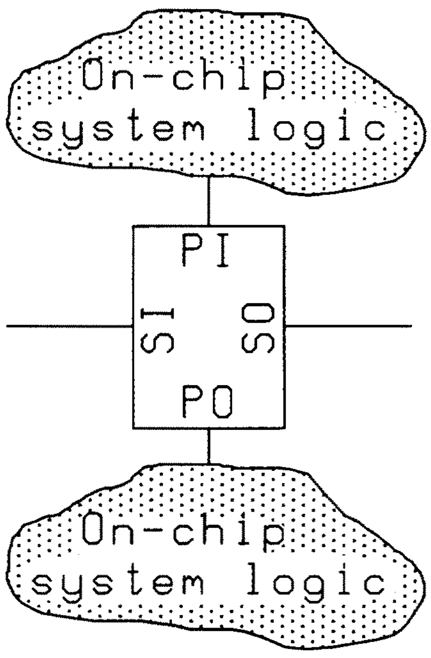


Figure 10-37—A Cell That Should Not Be Included in the Boundary-Scan Register

10.9 Special Cases

10.9.1 Specifications

Permissions

a) In a case in which a system input pin is used solely as a source of control or solely as a source of data for a system output pin, a single cell may be provided that meets the rules of 10.5.1 (for the input pin) and 10.6.1 (for the output pin).

10.9.2 Description

Where the signal received at a system input pin is used solely to provide data or control for a system output pin, it is possible to use a single boundary-scan cell to meet both sets of requirements. A common example of a situation where this might arise is one in which a system input pin is used solely to provide an output or direction-control signal for 3-state or bidirectional system pins. In such a case, either

a) Two separate boundary-scan cells may be included, as shown in figure 10-38; or
b) The functions of both cells may be combined into a single cell as shown in figure 10-39.

In the latter case, care should be taken in the design of the cell to ensure that it conforms to all the rules for the set of boundary-scan test instructions supported by the component.

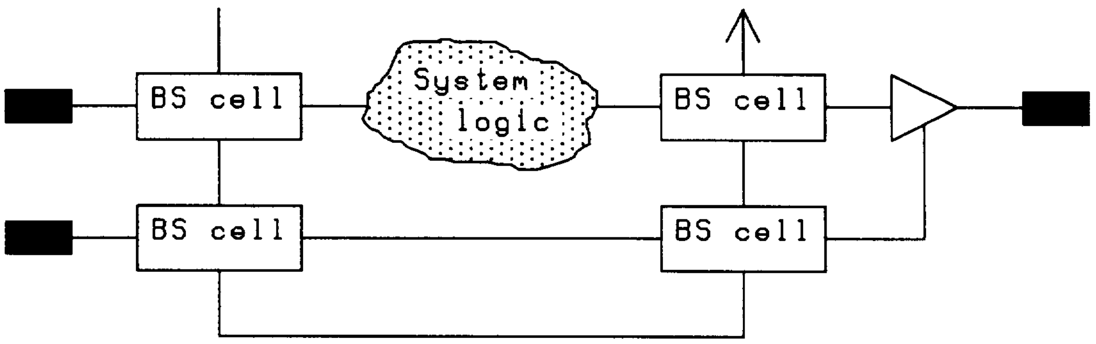


Figure 10-38—Input Pins Used Only to Control Output Pins—Case A

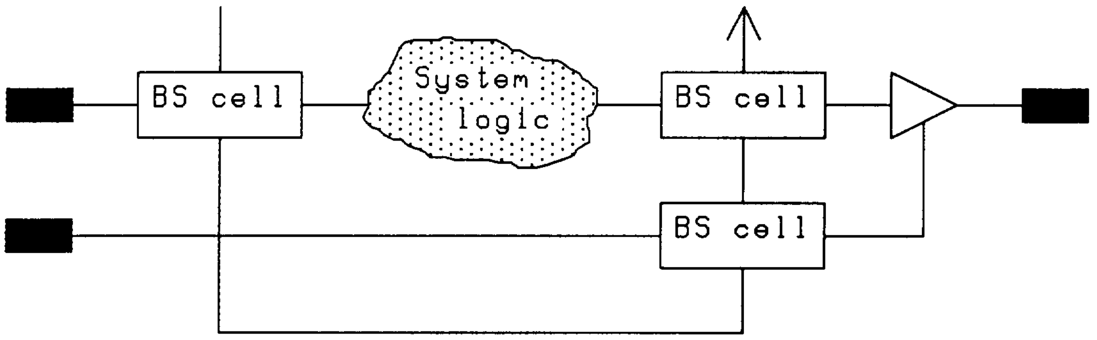


Figure 10-39—Input Pins Used Only to Control Output Pins—Case B

Note that the situation illustrated in figure 10-40 violates the rules of this standard. In this case, the signal received from the system input pin is used both as an output or direction control and as an input to the on-chip system logic.

In a case in which the signal from a system input pin is used only as an output enable for the 3-state output buffer and the option has been taken to provide a single boundary-scan cell as shown in figure 10-18, the top cell in figure 10-31 has to be modified if recommendation 7.8.1f is to be met. Specifically, the cell has to reload its own state in the *Capture-DR* controller state when the *INTEST* instruction is selected to avoid taking on a value dependent on off-chip circuitry. Figure 10-41 shows how this could be achieved. For this design, the Mode signal should be controlled as shown in table 10-6.

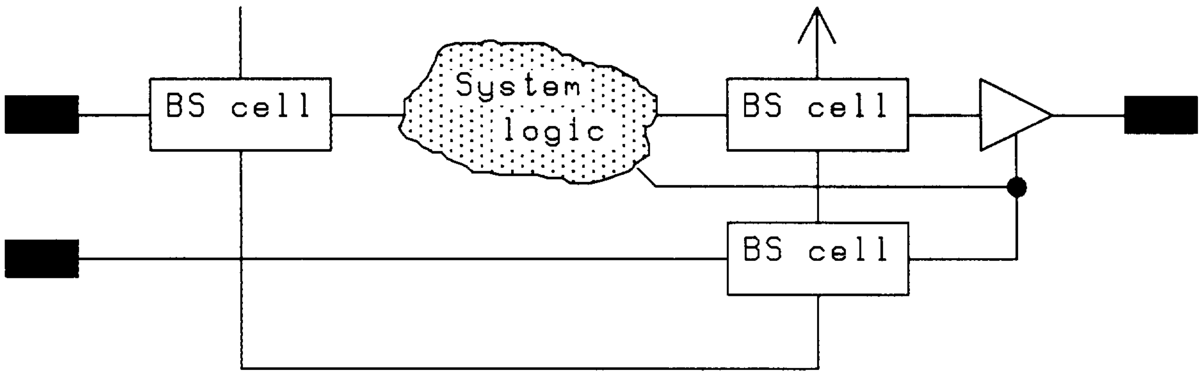


Figure 10-40—Illegal Use of a Single Cell for Output Control and Data

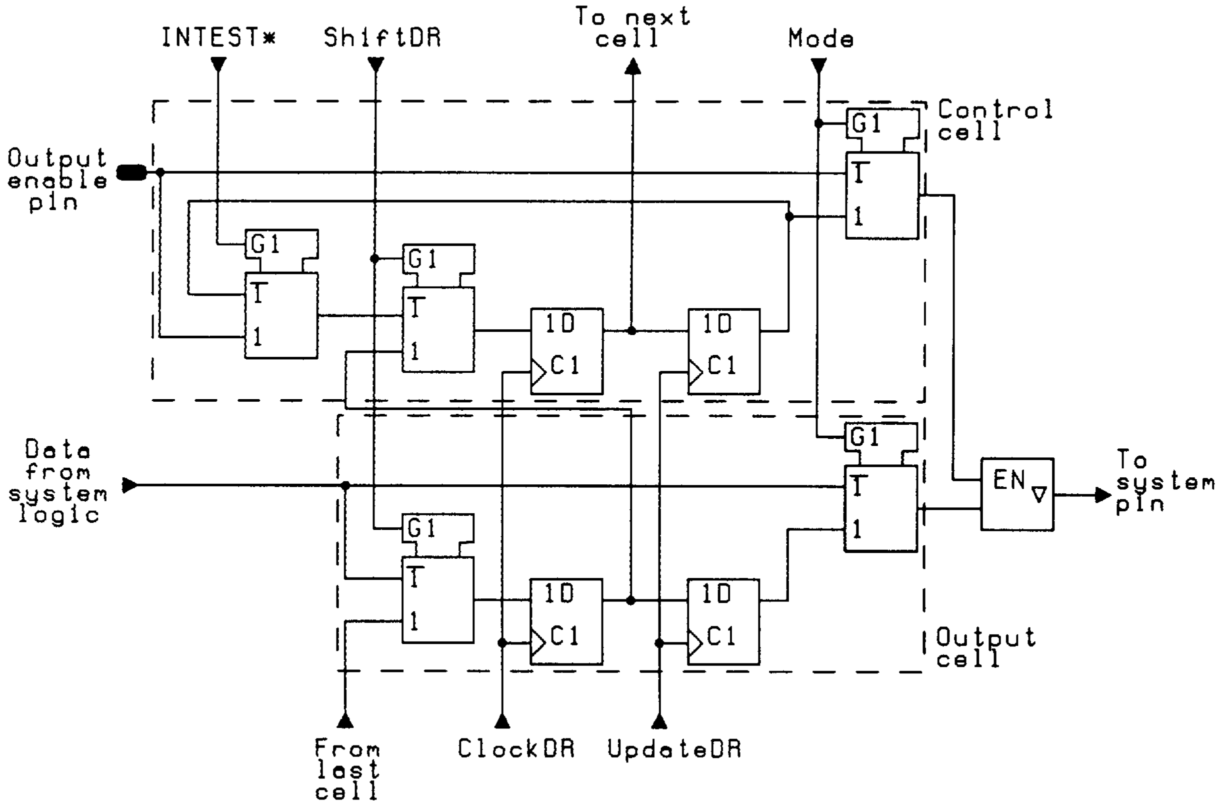


Figure 10-41—Boundary-Scan Cells at a 3-State Pin Where Output Control Is From a System Pin

11. The Device Identification Register

This chapter defines the design and operation of the optional device identification register. If provided, this register allows the manufacturer, part number, and version of a component to be determined through the TAP. One application of the device identification register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. As more components emerge that conform to this standard, it may become desirable to allow for a system diagnostic controller unit to blindly interrogate a board design in order to determine the type of each component in each location. The need to do this becomes more apparent if one considers systems that are configurable by the addition of option boards, or by programming certain components, etc. This information is also available for factory process monitoring and for failure mode analysis of assembled boards.

NOTE — The design requirements contained in this chapter apply only when the optional device identification register is included in a component.

11.1 Design and Operation of the Device Identification Register

11.1.1 Specifications

Rules

a) The device identification register shall be a shift-register based path that has a parallel input, but no parallel output.
b) The circuitry used to implement shift-register stages in the device identification register shall not be used to perform any system function (i.e., it shall be a dedicated part of the test logic).
c) On the rising edge of TCK in the *Capture-DR* controller state, the device identification register shall be set such that subsequent shifting causes an identification code to be presented in serial form at TDO.
d) The component shall contain a vendor-defined identification code, containing four fields (see figure 11-1), which is accessed when the *IDCODE* instruction is entered.
e) For user-programmable components where programming cannot be completely determined by use of the test logic defined by this standard, the capability shall be provided to permit the user to program a supplementary identification code that will be loaded into the device identification register in response to the *USERCODE* instruction.
f) The operation of the device identification register shall have no effect on the operation of the on-chip system logic.

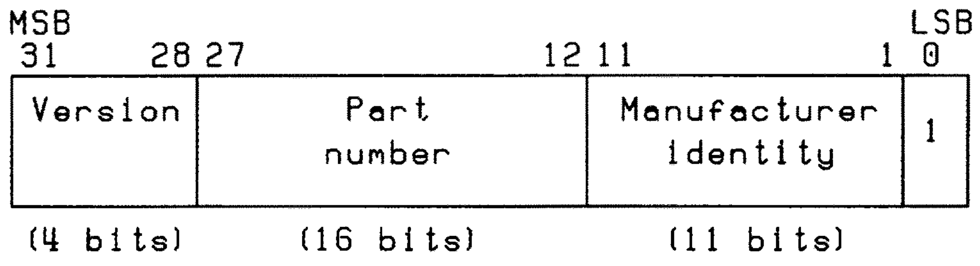


Figure 11-1—Structure of the Device Identification Register

11.1.2 Description

Figure 11-2 shows a design for a device identification register cell that satisfies these requirements.

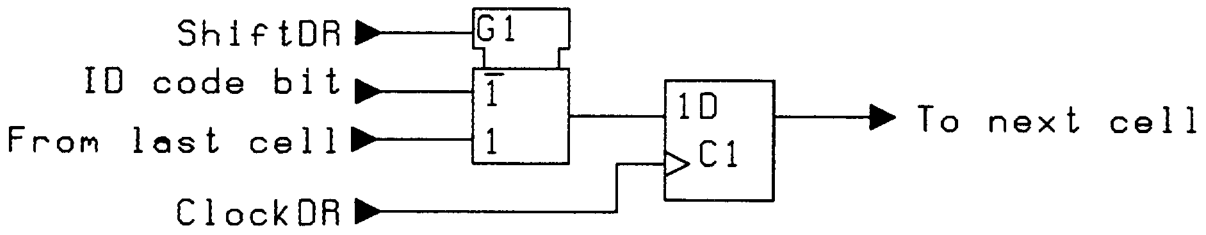


Figure 11-2—Device Identification Register Cell Design

The identification code loaded into the device identification register in response to the *IDCODE* instruction allows the manufacturer, part number, and variant for the component to be read in a serial binary form. In situations where blind interrogation of a product is necessary, this information allows the structure of the board to be determined and also, by reference to stored data, the instruction set and other details for each component. (It is assumed that the components in a product will be selected from a limited set.)

Examination of the identification code also allows the structure of the boundary-scan register to be deduced, including the positioning of cells at input and output pins and the location of cells that control 3-state or bidirectional pins. This information is valuable in ensuring that contention between drivers at the board level is avoided (for example, as discussed in 10.6).

For programmable components, however, the configuration of pins as inputs, outputs, etc. may be determined by programming, rather than by the basic design of the component. In such cases, therefore, a supplementary identification code is required to indicate how the component has been programmed. This supplementary code shall be user programmable and accessed through the device identification register in response to the *USERCODE* instruction.

NOTE — The supplementary identification code is required only in cases when the component cannot be reprogrammed through the test logic defined by this standard. In cases where such reprogramming is possible, the ATE or master device controlling the operation of the component can ensure that it is programmed to the correct state at the start of the test sequence.

Since the bypass register (which is selected in the absence of a device identification register by the instruction loaded in the *Test-Logic-Reset* controller state) loads a logic 0 at the start of a scan cycle, whereas a device identification register will load a constant logic 1 into its LSB, examination of the first bit of data shifted out of a component during a test data scan sequence immediately following exit from the *Test-Logic-Reset* controller state will show whether a device identification register is included in the design.

A requirement of the *IDCODE* and *USERCODE* instructions is that, when they are used, the on-chip system logic shall continue its normal operation undisturbed. Rule 11.1.1b is included so that this requirement can be met. Note, however, that provided rule 11.1.1b is met, the shift-register stages may be shared resources used by several of the registers defined by this standard and also by any design-specific test data register.

11.2 Manufacturer Identity Code

11.2.1 Specifications

Rules

a) The manufacturer identity code shall be a compressed form of the JEDEC Publication 106-A [1][2] generated from the JEDEC (the Joint Electron Device Engineering Council) code as follows:
   1) Identification code bits 7-1. The seven LSBs are derived from the last byte of the JEDEC code by discarding the parity bit.
   2) Identification code bits 11-8. The four MSBs provide a binary count of the number of bytes in the JEDEC code that contain the continuation character (hex 7F). Where the number of continuation characters exceeds 15, these four bits contain the modulo-16 count of the number of continuation characters.

b) The manufacturer code 00001111111 shall not be used in components that are otherwise compatible with this standard.

Recommendations

c) Where the component is an application-specific integrated circuit (ASIC), the manufacturer ID code should be that of the manufacturer of the component, rather than that of the designer.

11.2.2 Description

This scheme utilizes the manufacturer coding scheme administered by JEDEC [1].

The JEDEC code is formed from a variable number of eight bit bytes. Each byte contains seven data bits and an odd parity bit (the MSB). Bytes other than the last contain continuation characters (hex 7F), while the last contains 127 different codes that, together with a knowledge of the number of preceding continuation code bytes, allow the manufacturer's identity to be determined.

The compressed form of the JEDEC code used within the device identification register limits the number of bits needed in the device identification register to contain the manufacturer identity code and allows the length of the code to be standardized. The length of the compressed JEDEC code is fixed at 11 bits (see 11.1), which allows for 2032 different manufacturer codes. (Note that 16 codes are unused since these correspond to the hex 7F code in the seven LSBs—the JEDEC continuation character).

One of the unused codes (00001111111) should be treated as illegal for components compatible with this standard. By shifting a dummy device identification code containing this manufacturer identity code from the bus master (ATE, board-level controller, etc.) into the board-level serial path set up by moving directly from the *Test-Logic-Reset* controller state into scanning of the test data registers, it is possible to detect the end of the identity code sequence.

When test data register scanning is entered in this way, the serial path at the board level comprises:

a) The device identification registers of components that provide them; and
b) The bypass registers of components that do not include a device identification register.

As discussed in 11.1, the fact that identification codes begin with a logic 1, whereas the bypass registers load a logic 0, allows the identification codes in the serial stream read out of the board to be detected. By feeding in the dummy identification code at the board's serial input and checking the serial output for the invalid manufacturer identity code 00001111111, it is possible to locate the end of the identification code sequence for a board containing an unknown number of components.

[2]The numbers in brackets refer to those of the references listed in 2.4.

11.3 Part-Number Code

11.3.1 Specifications

Rules

a) The part-number code shall consist of 16 bits.
b) The manufacturer shall ensure that no two component types that are offered in the same package with the TAP pins in the same location have the same part-number code.

11.3.2 Description

The part-number code may be used to verify the type of the component inserted in a particular location on an assembled product. The use of a 16-bit value for this code gives an acceptably low chance that an incorrect component inserted in the location will return a correct part-number code.

Part-number codes could, for example, be generated from the textual part-number code using a data compaction scheme.

11.4 Version Code

11.4.1 Specifications

Rules

a) The version code shall consist of 4 bits.

Recommendations

b) The value of the version code for a component should be assigned to identify the variant of a component type.

12. Conformance and Documentation Requirements

12.1 Claiming Conformance to This Standard

The level of conformance to this standard can vary according to the range of test operations supported.

12.1.1 Specifications

Rules

a) Components that claim conformance to this standard shall comply with all relevant rules in the Specifications subclauses of this standard.

   NOTE — Components that were designed before publication of this standard and conform fully with the requirements of this standard except in the control of the TDO output driver with regard to the controller states in which it is active (see the note following table 5-2) may also claim conformance to this standard.

b) When claiming that a component conforms to this standard, the claim shall clearly identify the subset of the public instructions defined in this standard that is supported, as listed in table 12-1 and defined in 7.4.1.

Table 12-1—Public Instructions

Instruction	Status
BYPASS	Mandatory
CLAMP	Optional
EXTEST	Mandatory
HIGHZ	Optional
IDCODE	Optional
INTEST	Optional
RUNBIST	Optional
SAMPLE/PRELOAD	Mandatory
USERCODE	Optional

Recommendations

c) It is recommended that components support either the *INTEST* or the *RUNBIST* instruction or both.

Permissions

d) ASIC vendors may claim conformance to this standard by illustrating an interconnection of cells that, if built, would produce a component that meets the requirements of this standard.

12.1.2 Description

The minimum requirement for conformance to this standard is set to ensure that the user of an integrated circuit can perform two basic functions using the test logic: examine the operation of a prototype system and test assembled products for assembly-induced defects during manufacturing.

To enable efficient and comprehensive verification of internal component operation at the board and system level, it is strongly recommended that either the *INTEST* or *RUNBIST* instruction or both is supported.

12.2 Prime and Second Source Components

12.2.1 Specifications

Rules

a) With the sole exception of the device identification code, the publicly accessible test logic for second source components shall operate in the same manner as that of the prime source component in response to all public instructions.

12.2.2 Description

It is essential that both the system and the test logic of prime and second source components operate in the same manner in the component purchaser's environment. This ensures that test programs created for a printed circuit board containing multiply source components produce consistent results regardless of the source of individual components.

The only exceptions to this requirement are the optional device identification register and any test logic that is accessed only in response to private instructions. In the former case, the identification code shall vary to identify the source of the particular component, its part number, and its revision (see Chapter 11). In the latter case, test logic that is not publicly accessible is not intended for use other than by the component vendor; therefore, this test logic should not be operated by a board-level test program.

12.3 Documentation Requirements

12.3.1 Specifications

Rules

a) For any component that claims conformance to this standard, the operation of all test logic accessed in response to public instructions shall be fully documented.

b) The following information, required by the component purchaser for use in test development and other activities, shall be supplied by the component manufacturer:

   i) Instruction register. The following information pertaining to the instruction register is required:
      - Its length.
      - The pattern of fixed values loaded into the register during the *Capture-IR* controller state.
      - The significance of each design-specific data bit presented at a parallel input, where provided.

   ii) Instructions. For each public instruction offered by a component, the following information is required:
      - The binary code(s) for the instruction.
      - A list of test data registers placed in a test mode of operation by the instruction.
      - The name of the serial test data register path enabled to shift data by the instruction.
      - A definition of any data values that shall be written into test data registers prior to selection of the instruction, and the order in which these values shall be loaded.
      - The effect of the instruction. Any system pins whose drivers become inactive as a result of loading the instruction should be clearly identified.
      - A definition of the test data registers that will hold the result of applying a test and of how they are to be examined.
      - A description of the method of performing the test and of how data inputs and their corresponding data outputs are to be computed.

      If private instructions are utilized in a component, the vendor shall clearly identify any instruction codes that, if selected, would cause hazardous operation of the component.

   iii) Self-test operation. For each instruction that causes operation of a self-test function, the following information is required in addition to that listed under rule 12.3.1b(ii):
      - The minimum duration (e.g., a number of cycles of TCK) required to ensure completion of the test.
      - A definition of the test data registers whose states are altered during execution of the test.
      - A definition of the results of executing the self-test on a fault-free component.
      - An estimate of the percentage (e.g., to the nearest 5%) of the single stuck-at faults in the component's circuitry that will be detected by the self-test function or a description of the operation of the self-test Function and the circuitry exercised.

   iv) Test data registers. For each test data register available for public use and access in a component, the following information is required:
      - The name of the register, used for reference in other parts of the data sheet.
      - The purpose of the register.
      - The length.
      - A full description of the operating modes of the register.
      - The result of setting each bit at the parallel output of the register.
      - The significance of each bit loaded from the parallel input of the register.

   v) Boundary-scan register. The following information is required in addition to that listed under rule 12.3.1b(iv):
      - The correspondence between boundary-scan register bits and system pins, system direction controls, or system output enables.
      - Whether each pin is an input, a 2-state output, a 3-state output, or a bidirectional pin.
      - For each boundary-scan register cell at an input pin, whether the cell can apply tests to the on-chip system logic.
      - For each boundary-scan register cell associated with an output or direction control signal, a list of the pins controlled by the cell and the value that shall be loaded into the cell to place the driver at

each pin in an inactive state or will be observed using the *SAMPLE/PRELOAD* or *INTEST* instructions when the on-chip system logic causes the driver to be inactive.
- The method by which single-step operation is to be achieved while the *INTEST* instruction is selected, if this instruction is supported.
- The method of providing clocks to the on-chip system logic while the *RUNBIST* instruction is selected, if this instruction is supported.
- For each redundant cell, whether the cell returns either the value shifted in or a constant following loading of the cell in the *Capture-DR* controller state.

vi) Device-identification register. Where a device identification register is included in a component, the following information is required in addition to that listed under rule 12.3.1b(iv):
- The value of the manufacturer's identification code.
- The value of the part number code.
- The value of the version code.
- The method of programming the value of the supplementary identification code, where required.

vii) Performance. The performance of the test logic should be fully defined, including the following information:
- The maximum acceptable TCK clock frequency.
- A full set of timing parameters for the test logic.
- The logic switching thresholds for TAP input and output pins.
- The load presented by the TCK, TMS, TDI, and TRST* pins.
- The drive capability of the TDO output pin.
- The extent to which the TDO driver may be overdriven when active (e.g., using an in-circuit test system).
- Whether TCK may be stopped in the logic 1 state.

viii) If a component has compliance-enable inputs as defined in 4.2.2, then the following documentation shall be provided:
- A complete list of these inputs labeled as compliance-enable inputs.
- A complete list of those logic patterns that, when applied at the compliance-enable inputs, will enable compliance to this standard.
- A clear indication of any patterns that, if applied to the compliance-enable inputs, would cause hazardous operation of the component.

12.3.2 Description

Figures 12-1 and 12-2 show how set-up and hold timing parameters and propagation delays should be measured relative to the test clock TCK and a reference voltage V_{ref}. Note that such timing parameters are required for TMS, TDI, and TDO and also for system pins that can be driven from the test logic (e.g., the system data input set-up time for the boundary-scan register before the rising edge of TCK in the *Capture-DR* controller state).

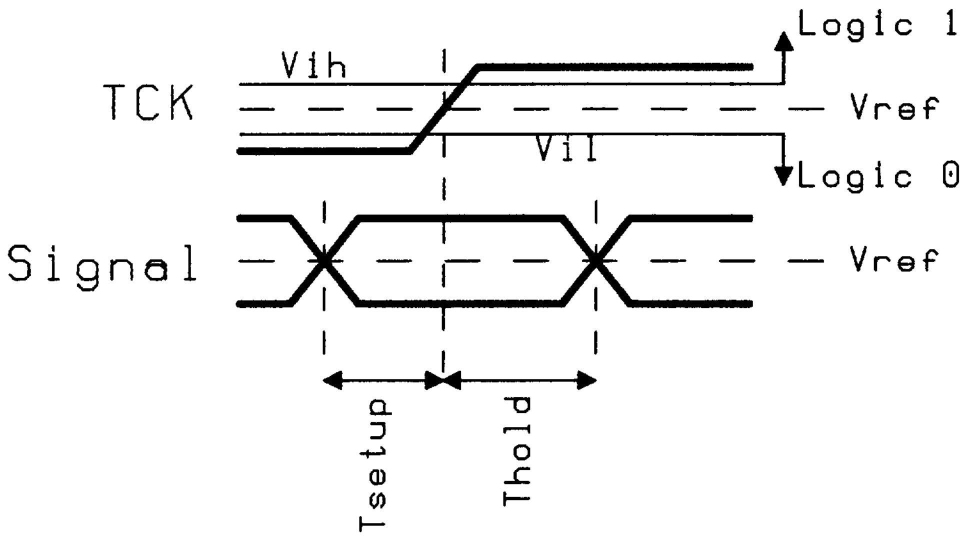


Figure 12-1—Measuring Set-Up and Hold Timing

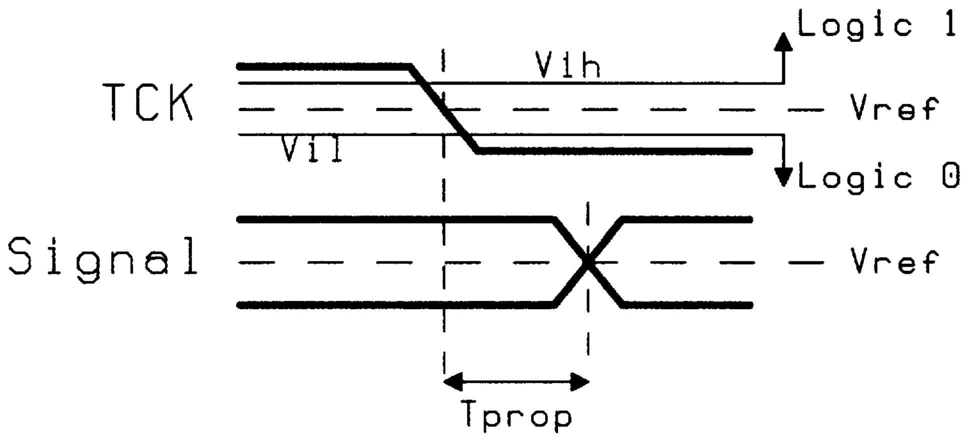


Figure 12-2—Measuring Propagation Delays

Annex A An Example Implementation Using Level-Sensitive Design Techniques (Informative)

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

To illustrate how a circuit might be constructed to meet the requirements of IEEE Std 1149.1-1990, example designs are included in the standard. These examples form a consistent set and could be used as the basis of an implementation. However, it is important to emphasize that the designs contained in the standard are neither mandatory nor recommended in preference to any other implementation. Many other implementations are possible. For example, this Appendix illustrates one of many possible implementations of the test logic that could be based on Level-Sensitive Scan Design (LSSD) techniques. This implementation has two modes of operation:

a) A "chip-on-board" mode where the component responds to the signals received at the TAP inputs in the manner required by the standard; and
b) A "stand-alone" mode that allows the entire component (including both the on-chip system logic and the test logic) to be tested using LSSD techniques, for example, as a part of a postproduction test.

The stand-alone mode extends the functionality of the component beyond that required by the standard, while maintaining compatibility with the standard for chip-on-board testing.

A.1 Top-Level Test Logic Design

The design for the test logic is shown in figure A-1. The following design features should be noted:

a) All stored-state devices are constructed from level-sensitive latches. Shift-register stages contained in the test logic require two such latches controlled from a pair of nonoverlapping clocks. The clock generator circuit shown in figure A-2 generates these clocks as shown in figures A-3 and A-4 1.
b) An internal scan path is provided that visits all shift-register latches in the design, including those in the test logic. The internal scan path is shown as a bold line in figure A-1.
c) For chip-on-board operation, the LSSD clocks LSSD_A, LSSD_B, and LSSD_P are held at 0, while LSSD_C1 and LSSD_C2 are held at 1. This allows the test logic to operate in response to signals received at the TMS, TDI, and TCK inputs, as defined in IEEE Std 1149.1-1990.

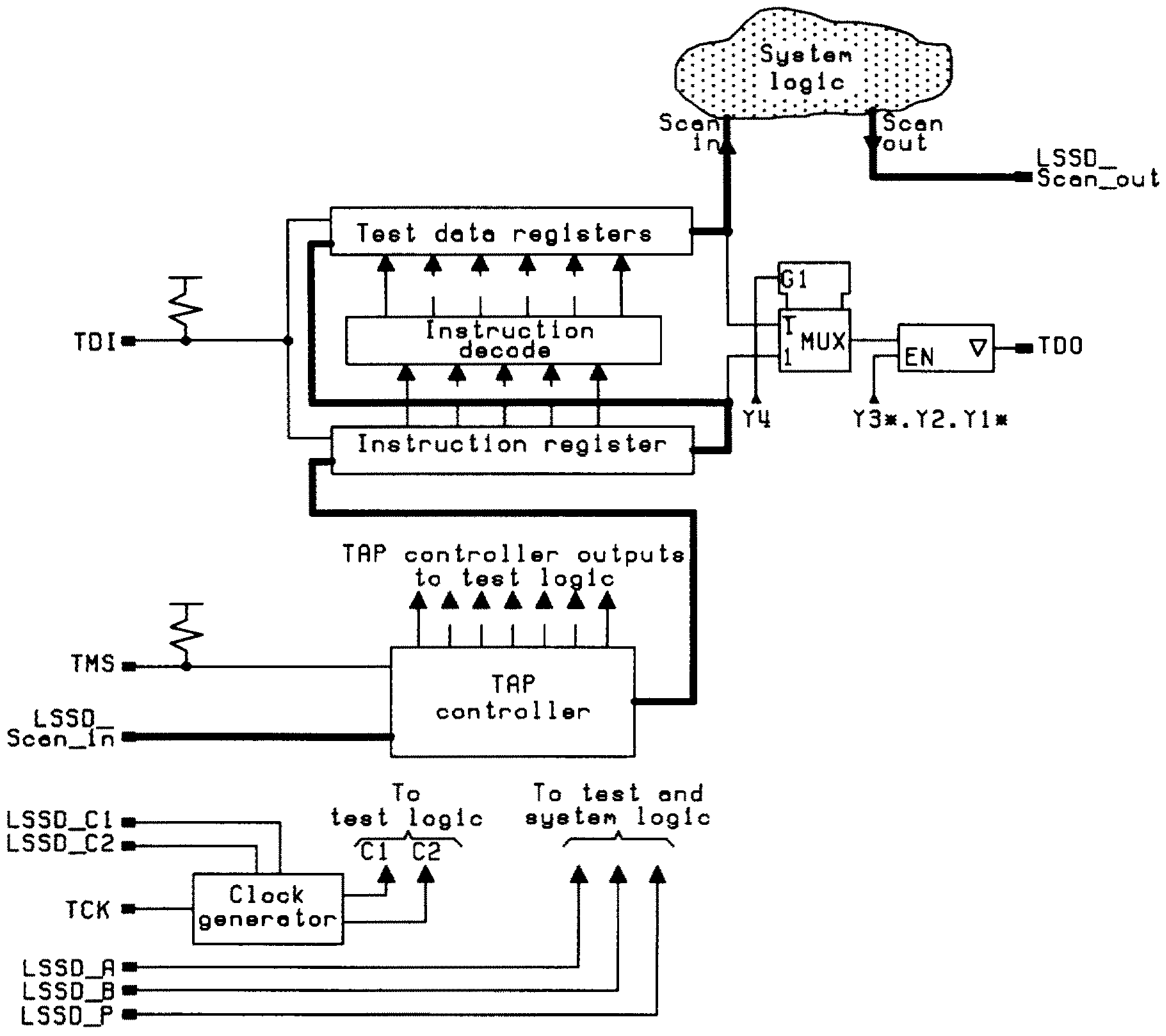


Figure A-1—Test Logic Schematic

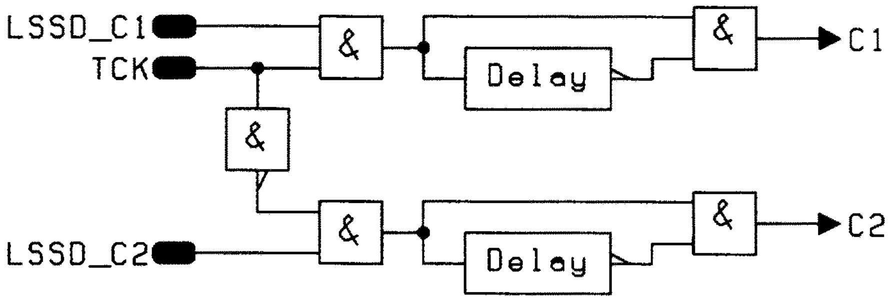


Figure A-2—Generation of Nonoverlapping Clocks From TCK

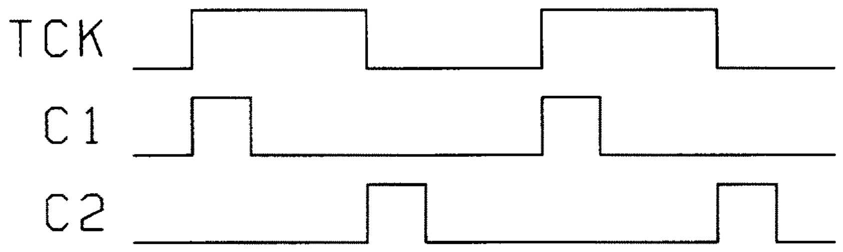


Figure A-3—Operation of the Clock Generator

d) For stand-alone component testing using the internal scan path, TCK and the clocks for the on-chip system logic are operated in concert with the LSSD clocks (LSSD_A, LSSD_B, LSSD_P, LSSD_C1, and LSSD_C2). To ensure that LSSD testing of the test logic is correctly synchronized to that of the on-chip system logic, the signals LSSD_C1 and LSSD_C2 are used. These signals are controlled in concert with the remaining LSSD clocks that enable shifting along the scan path. In this mode, the TCK input to the clock generator is used as a control signal that can enable or disable the signals supplied to LSSD_C1 and LSSD_C2. For example, to permit a positive-going pulse on LSSD_C1 to propagate through to C1, a logic 1 must first be applied at TCK. Similarly, positive-going pulses at LSSD_C2 are allowed through to C2 if TCK is first set to 0. Figure A-4 shows the expected relationships between the various clock signals during LSSD stand-alone testing.

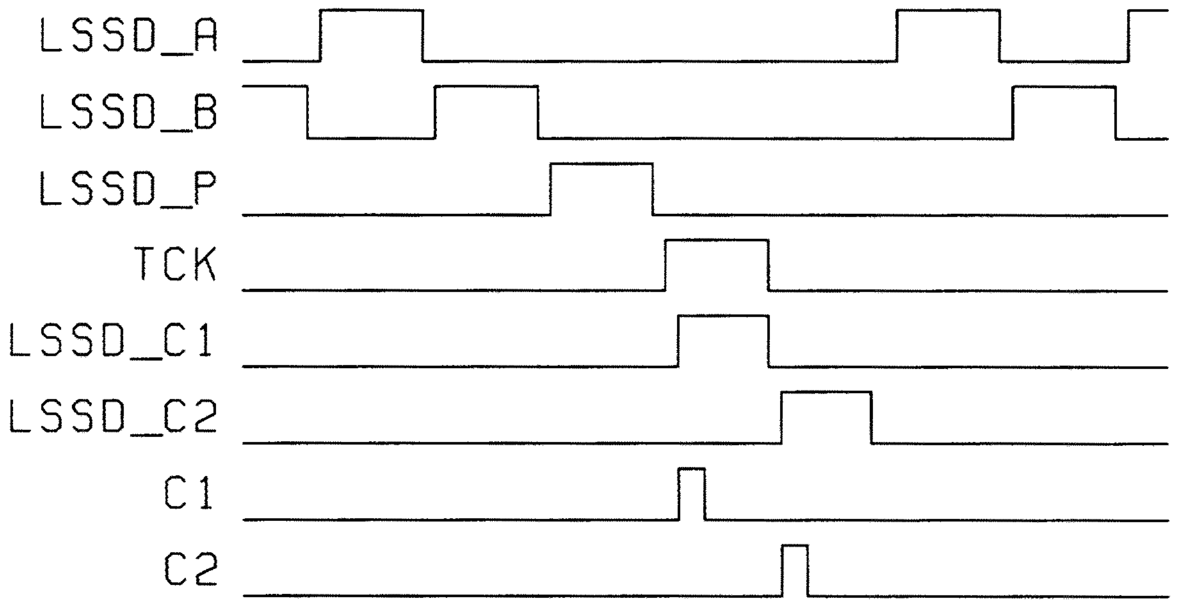


Figure A-4—Control of Clocks for “Stand-Alone” Component Testing

e) Since the serial outputs of the instruction and test data registers change state on the falling edge of TCK due to the master-slave operation of the latches that form the shift-register stages, it is not necessary to retime the output fed to TDO.

Note that the capture of input signals to the test logic (e.g., those received at TMS, TDI, and at system input pins) occurs on the falling edge of C1, which itself occurs at a fixed delay following receipt of a rising edge at TCK. Provided that the width of pulse C1 is independent of the frequency of TCK, the requirements of the standard will be met (see 3.2.2).

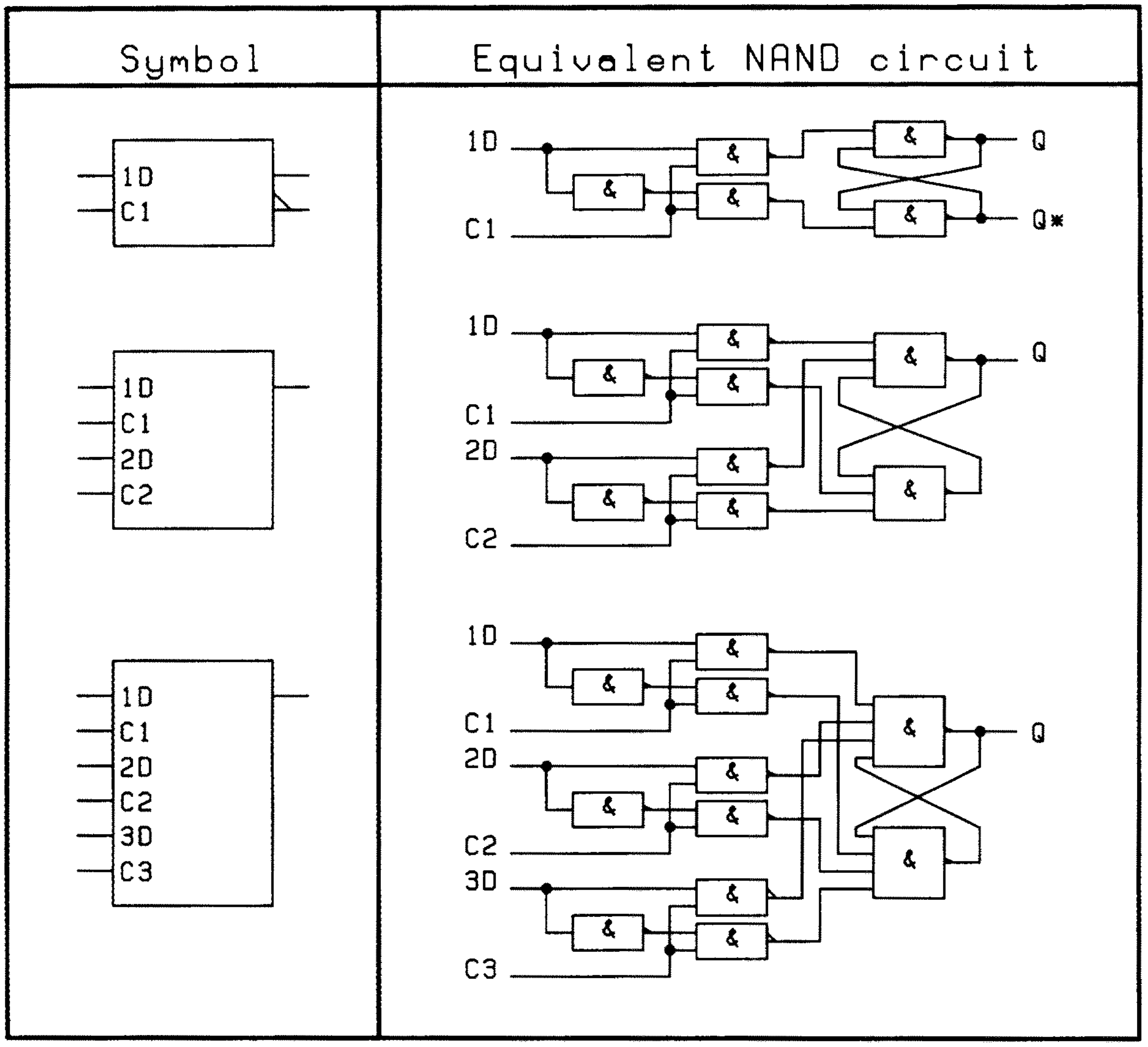


Figure A-5—Schematics for Level-Sensitive Latches

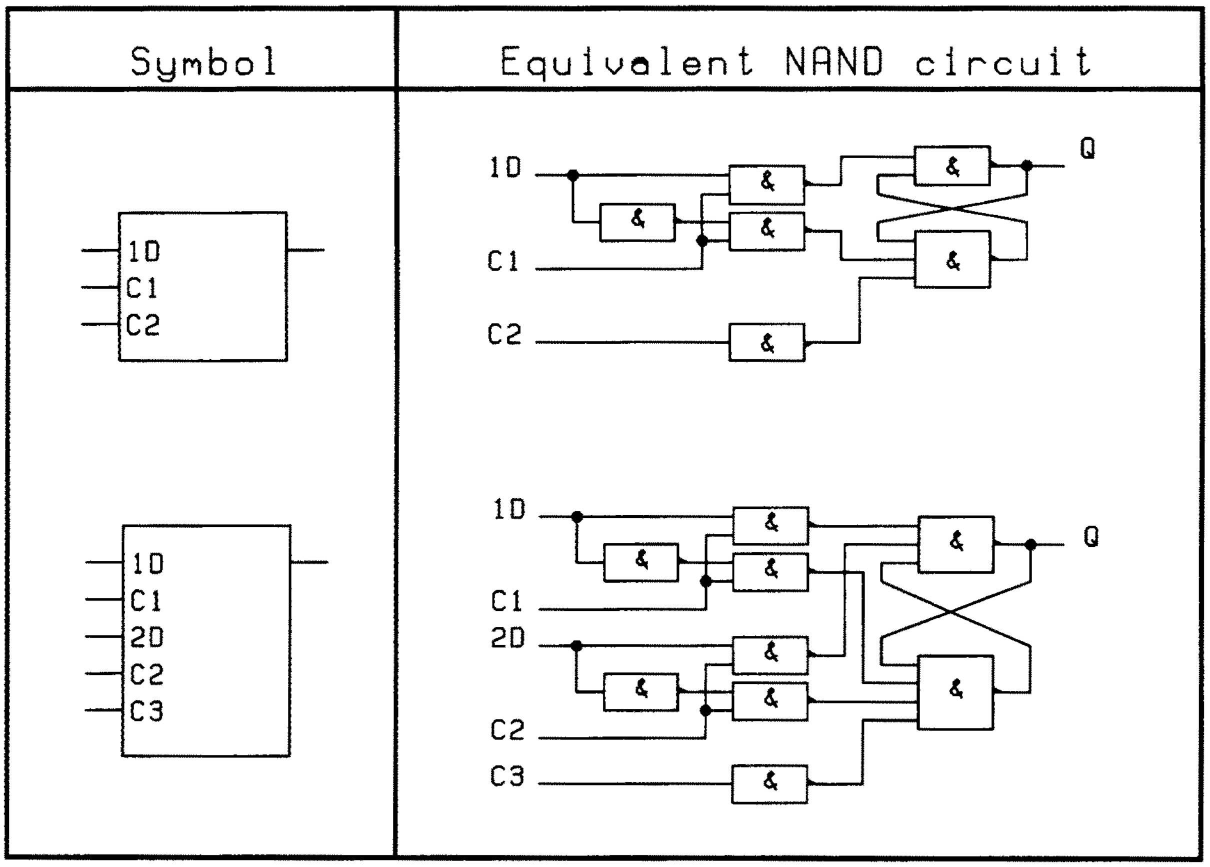


Figure A-5 —Schematics for Level-Sensitive Latches, Continued

A.2 Latch Designs

Figure A-5 gives NAND gate equivalent circuits for the latches used in the remainder of the schematics in this Appendix.

A.3 TAP Controller Implementation

Figures A-6 and A-7 show the implementation of the TAP controller. Note that, for this example, the output decoding logic is defined for each register in the following clauses of this Appendix. The assignments of logic states to controller states are as for the previous implementation—see table 5-3 in Chapter 5 of the standard.

Note the inclusion of the scan test path through the controller latches, which allows the controller to be fully tested as a part of a scan test on the complete integrated circuit.

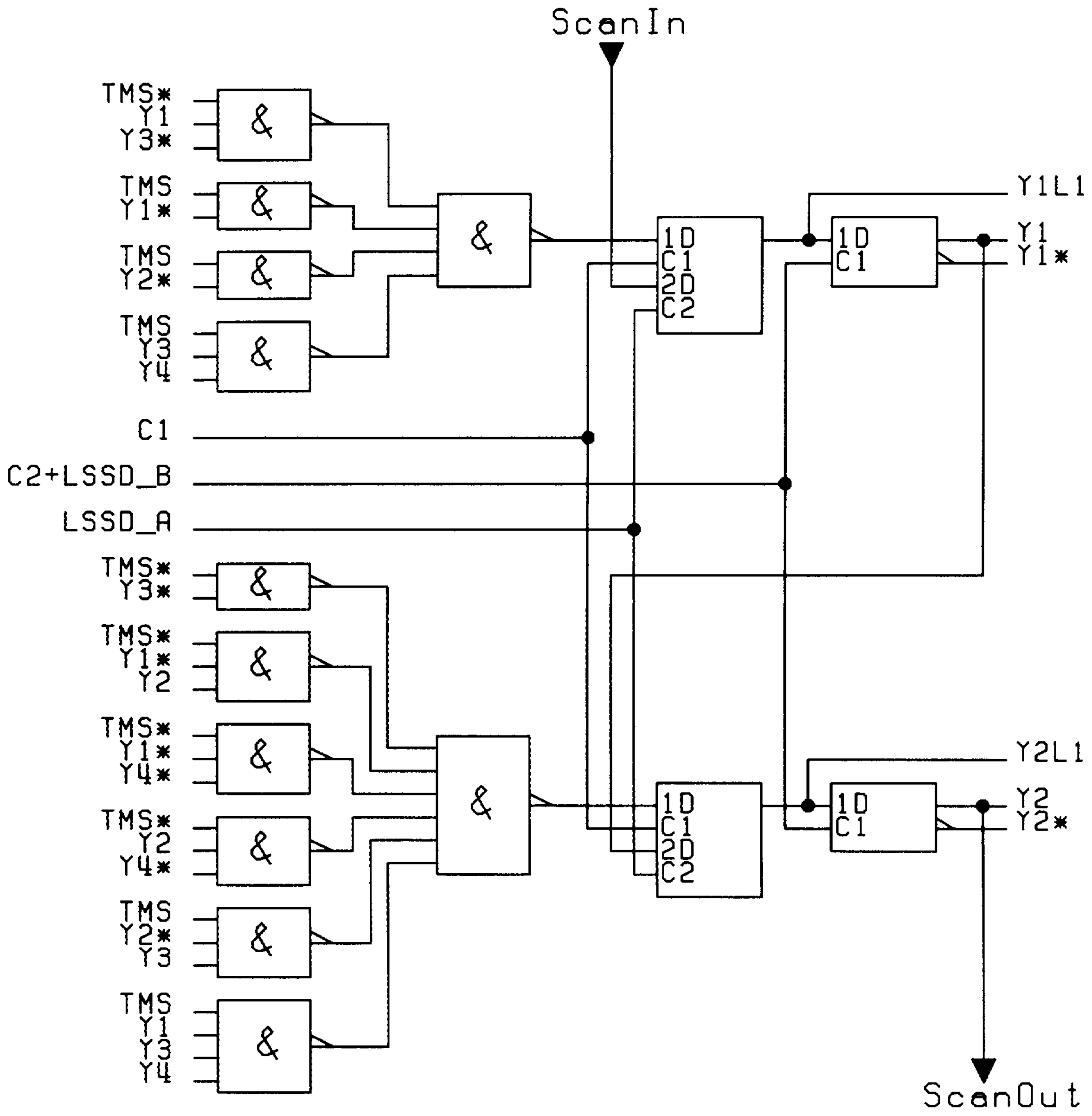


Figure A-6—TAP Controller—A

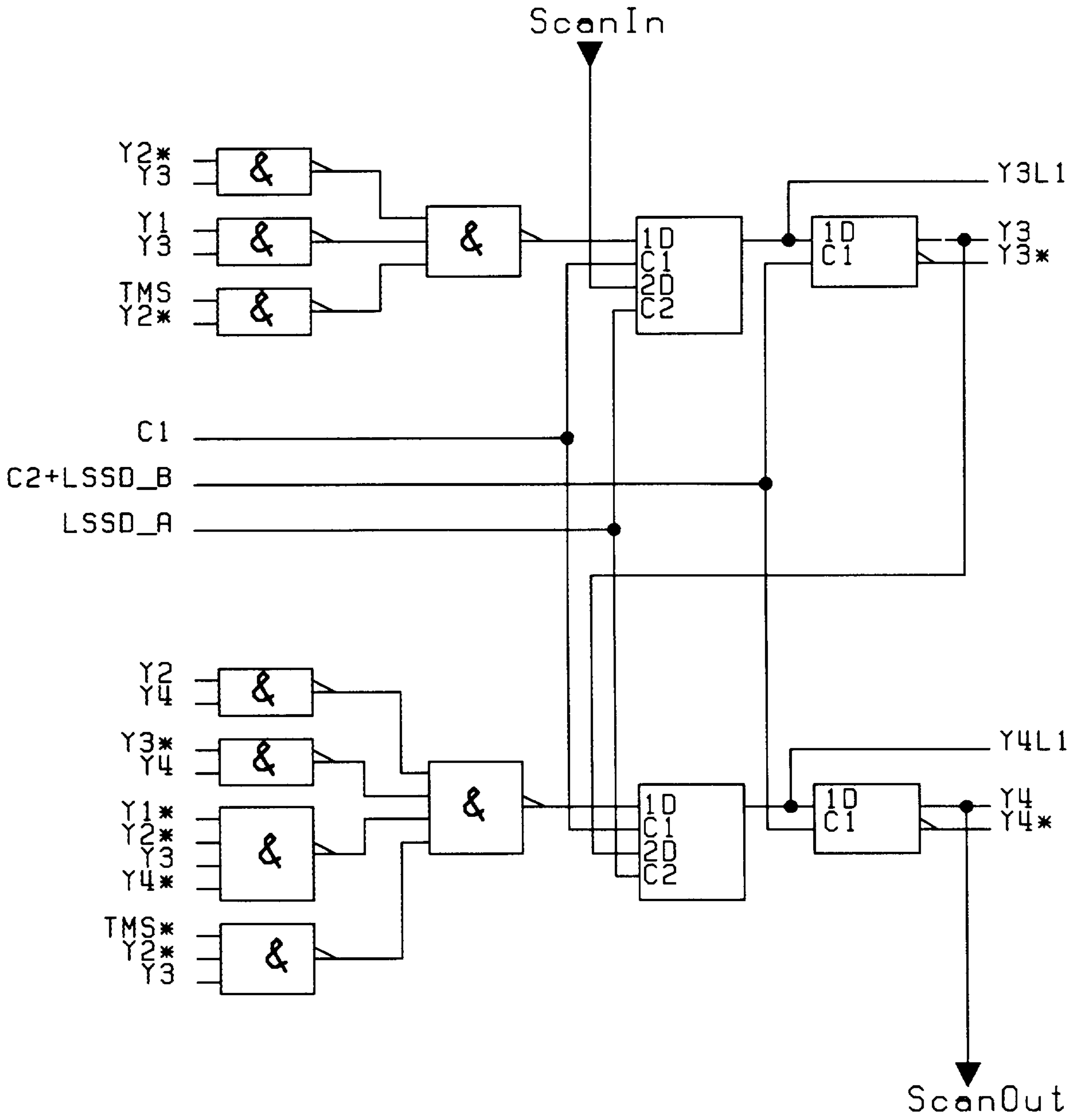


Figure A-7—TAP Controller—B

A.4 Instruction Register Implementation

To allow for the scan path input to the instruction register, the design of the cell nearest to TDI will differ from that of other cells in the register. Figure A-8 shows a design for the cell nearest to TDI, while figure A-9 shows a design for other cells.

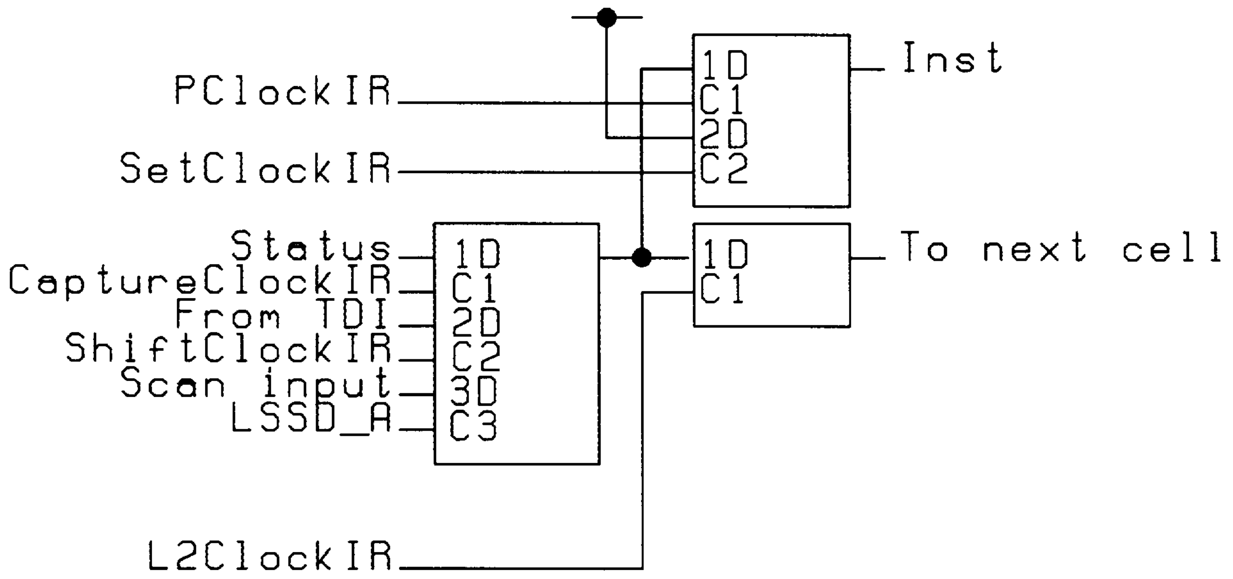


Figure A-8—Instruction Register Cell Nearest to TDI

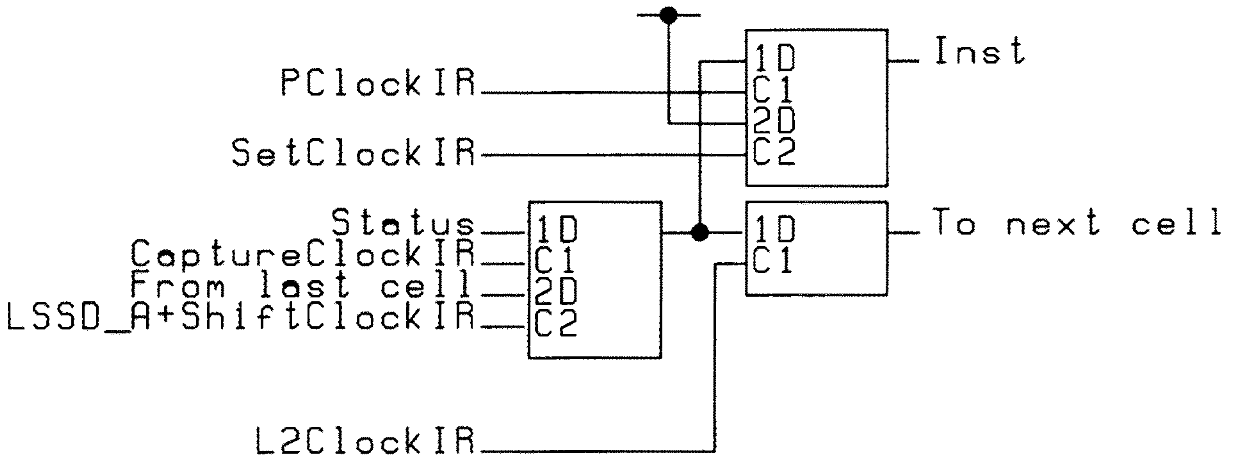


Figure A-9—Other Instruction Register Cells

The control signals for these cell designs are generated from the TAP controller outputs and the various clock signals as follows:

CaptureClockIR = C1.Y4.Y3.Y2.Y1*

ShiftClockIR = C1.Y4.Y3*.Y2.Y1*

SetClockIR = C2.Y4L1.Y3L1.Y2L1.Y1L1

L2ClockIR = C2 + LSSD_B

PClockIR = C2.Y4L1.Y3L1.Y2L1*.Y1L1 + LSSD_P

A.5 Bypass Register Implementation

The bypass register can be implemented as shown in figure A-10.

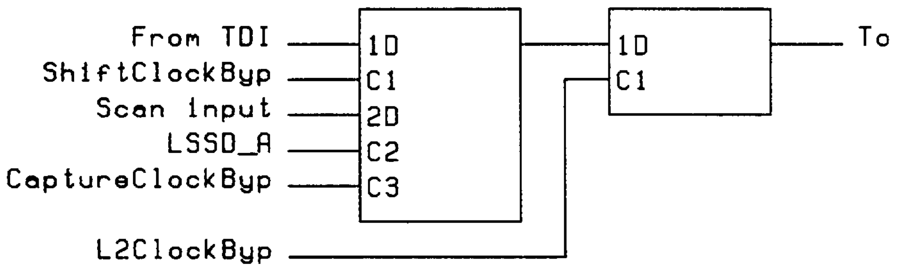


Figure A-10—Bypass Register

Control signals for this implementation are generated as follows:

CaptureClockByp = C1.Y4*.Y3.Y2.Y1*.BYPASS

ShiftClockByp = C1.Y4*.Y3*.Y2.Y1*.BYPASS

L2ClockByp = C2 + LSSD_B

Note that the variable BYPASS is true when the BYPASS instruction is present at the instruction register outputs.

A.6 Boundary-Scan Register Implementation

A set of boundary-scan cell designs is included in the following figures. The control signals required by these cells are:

CaptureClockBS = C1.Y4*.Y3.Y2.Y1*.BST

ShiftAClockBS = C1.Y4*.Y3*.Y2.Y1*.BST + LSSD_A

L2CIockBS = C2 + LSSD_B

PClockBS = C2.Y4L1*.Y3L1.Y2L1*.Y1L1.BST+ LSSD_P

ResetClockBS = C2.Y4L1.Y3L1.Y2L1.Y1L1

The variable BST is true whenever a boundary-scan test instruction is present at the instruction register's outputs. The signals labeled DriveOut and LogicTest are controlled as follows:

a) DriveOut is true when the *EXTEST, INTEST,* or *RUNBIST* instruction is selected.
b) LogicTest is true when the *INTEST* or *RUNBIST* instruction is selected.

Note that the boundary-scan register nearest to TDI should have its first latch modified to accept either the signal received from TDI (when C1.Y4*.Y3*.Y2.Y1*.BST is true) or the signal at the scan input to the register (when LSSD_A is true).

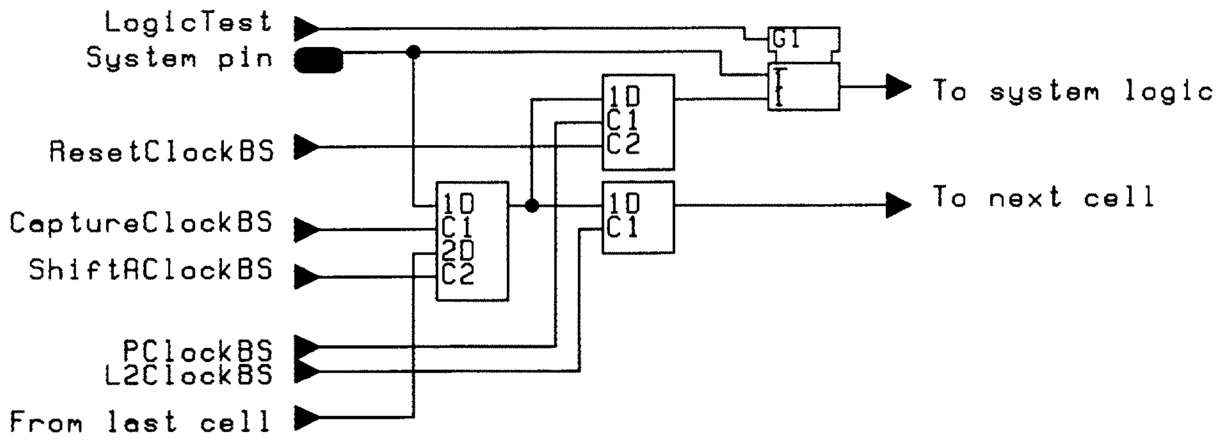


Figure A-11—A Level-Sensitive Input Cell Design

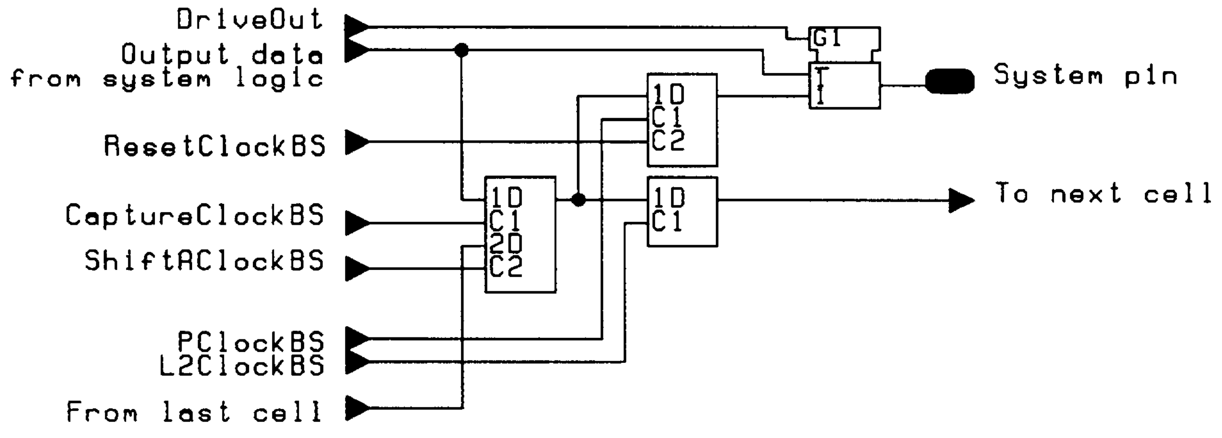


Figure A-12—A Level-Sensitive Output Cell Design

The parallel output latches in the control cells in figures A-13 and A-14 are reset in the *Test-Logic-Reset* controller state as allowed by permission 10.3.1g and rule 10.6.1o. A similar reset capability is provided in figures A-11 and A-12.

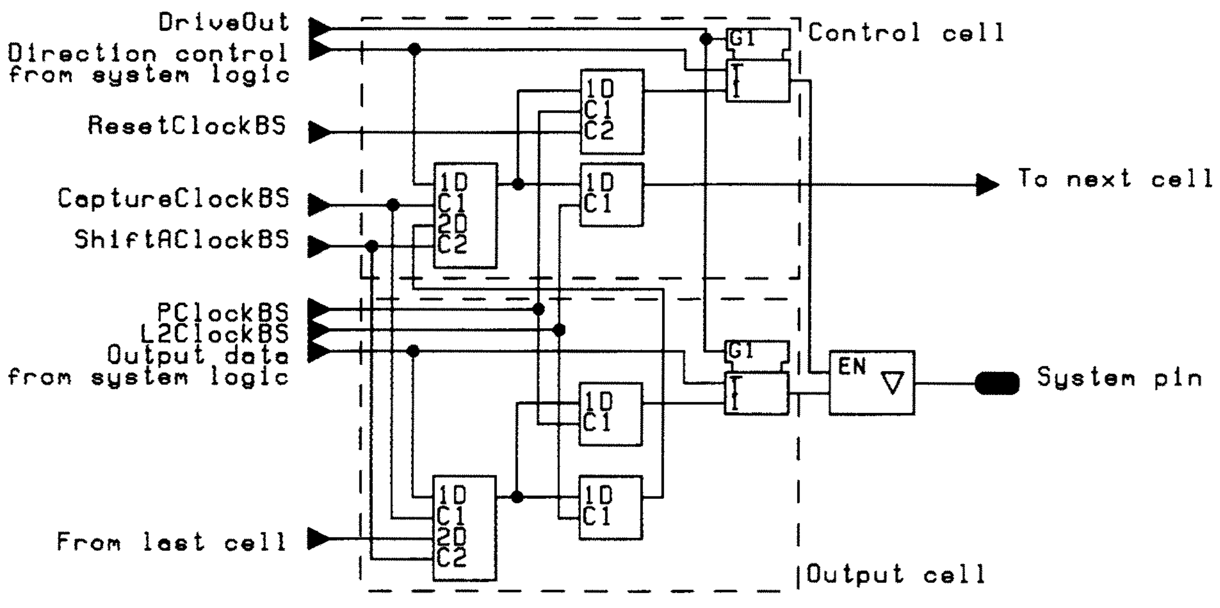


Figure A-13—Level-Sensitive Cells at a 3-State Output

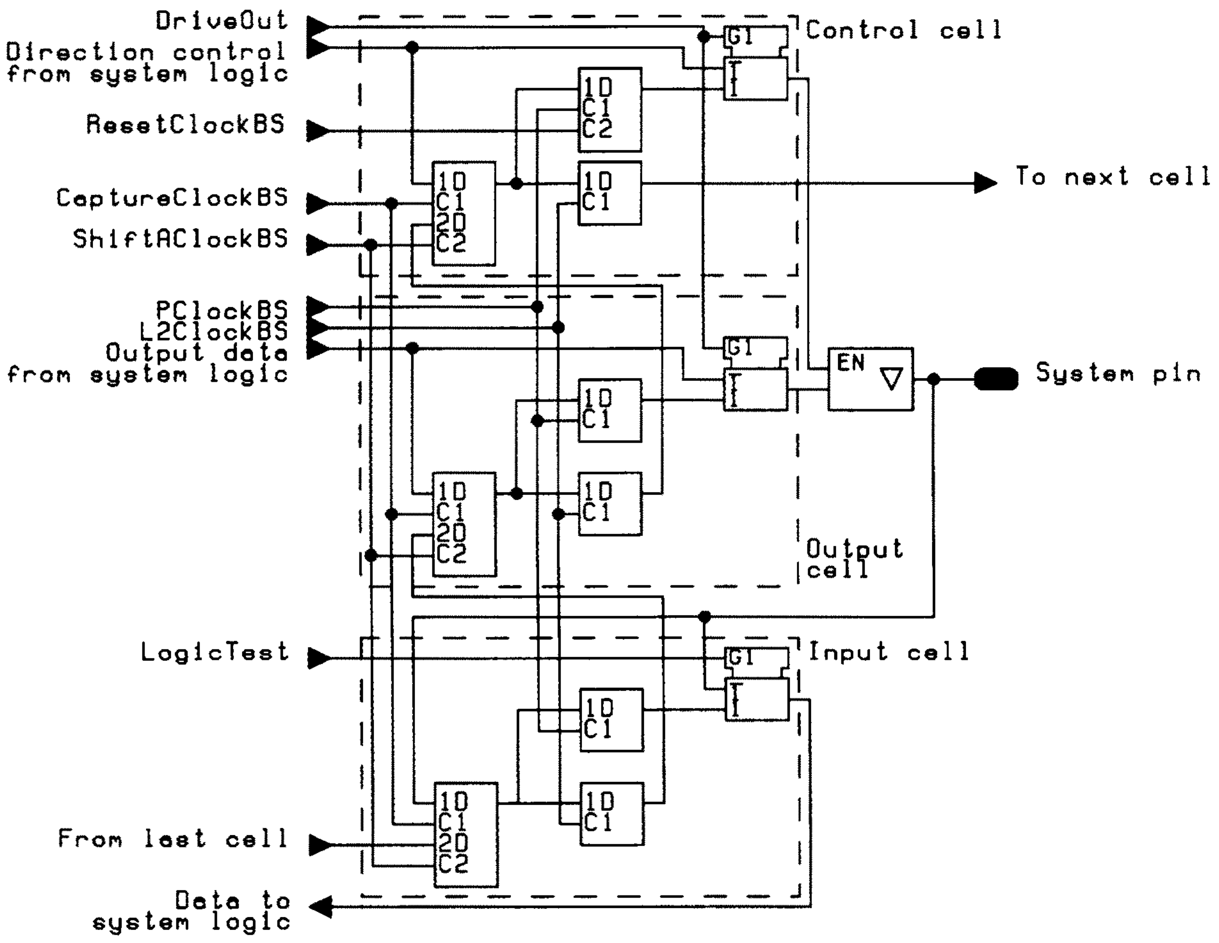


Figure A-14—Leve1-Sensitive Cells at a Bidirectional Pin