

# An Introduction to the Boundary Scan Standard: ANSI/IEEE Std 1149.1

C.M. MAUNDER

*British Telecom Research Laboratories, Martlesham Heath, Ipswich IP5 7RE UK*

R.E. TULLOSS

*AT&T Bell Laboratories, Engineering Research Center, Princeton, New Jersey 08540*

*Received October 29, 1990*

*Editor: R.G. Bennetts*

**Abstract.** ANSI/IEEE Std 1149.1 defines a standard implementation of boundary scan that, it is hoped, will be built into many catalog and application-specific integrated circuits. The standard was developed as a solution to two continuing trends that are having a significant, adverse, impact on the task of testing loaded printed wiring boards: increasing chip complexity and greater miniaturization. The former increases the difficulty of test generation, while the latter impedes access for the bed-of-nails and hand-held probes on which many established test techniques depend.

This tutorial provides a guide to the principal features defined by the standard and to their operation. It is intended as a prelude to the standard itself, not as a substitute for it. In particular, it is recommended that readers who intend to implement integrated circuits, design tools, or test systems that support the standard read the standard document before doing so.

**Key words:** ANSI/IEEE Std 1149.1, boundary scan, JTAG, loaded-board test, self-test.

## 1. Introduction

Two continuing trends are having a significant, adverse, impact on the cost of testing loaded printed-wiring boards (PWBs):

1. *Increasing complexity:* As integrated circuits (ICs) become more complex, the difficulty of generating a test for loaded boards increases. For in-circuit testing, the generation of a test module for a new IC design could take several man-months. For functional testing, test generation times are significantly longer, due to the need to propagate test data through the complex ICs while applying tests to other chips on the board. Test lengths also increase as complexity rises, pushing up the cost of applying the finished test. In in-circuit testing, the maximum test length is constrained by a time limit imposed to ensure that components are not damaged by backdriving, so the result of increasing complexity is often that less thorough testing is possible.
2. *Greater miniaturization:* Test techniques that became widely used during the 1980s—in-circuit, functional, cluster, and emulation testing—depend

significantly on the ability to make contact with connections internal to the loaded board. Bed-of-nails or hand-held probes must be connected to such connections during test application and/or fault diagnosis. The use of surface-mount assembly techniques, particularly when coupled with double-sided component mounting and the use of buried vias to connect layers of interconnect on the PWB, reduces board geometries—making the finished product smaller and, unfortunately, more difficult to probe.

The aim of ANSI/IEEE Std 1149.1 [1]—the *Standard Test Access Port and Boundary Scan Architecture*—is to provide the basis of solutions to these problems. The key is the elimination of the need for physical probing of the loaded board, which is achieved by building an electronic test-access mechanism (the boundary scan path) into the integrated circuits themselves.

This tutorial provides a guide to the principal features defined by the standard and to their operation. It is intended as a prelude to the standard itself, not as a substitute for it. In particular, it is recommended that readers who intend to implement integrated circuits,

design tools, or test systems that support the standard should read the standard document before doing so. Also, while this tutorial shows how basic test operations can be performed, it is not a complete guide to the potential applications of the standard. For a view of the wide range of applications of the standard, the reader is directed to [2].

## 2. A Chip-Level View

Figure 1 gives an overall view of an IC design that conforms to ANSI/IEEE Std 1149.1. The circuitry can be broken down into two parts:

1. *The system logic:* This is the circuitry that performs the “normal” function for which the chip was designed. For example, it could be the logic necessary to build a microprocessor, a communications interface, or a counter.
2. *The test logic:* This is the circuitry defined by the standard and includes all the blocks in figure 1 other than the system logic. The test logic is used during

testing of the IC and of the board onto which the IC is assembled. It does not contribute to the normal operation of the system logic.

Some circuitry may be shared between the system and test logic. For example, a register in the system logic may have test modes of operation—perhaps it might be a BILBO or similar device [3]. Under these circumstances, the circuitry is regarded as system logic while it contributes to the normal chip function and as test logic while it participates in test operations.

## 3. The Test Logic Architecture

The top-level schematic of the test logic defined by IEEE Std 1149.1 includes three key circuit blocks (figure 2):

*The TAP controller:* This responds to control sequences supplied through the test access port (TAP—see below) and generates the clocks and control signals required for correct operation of the other circuit blocks.

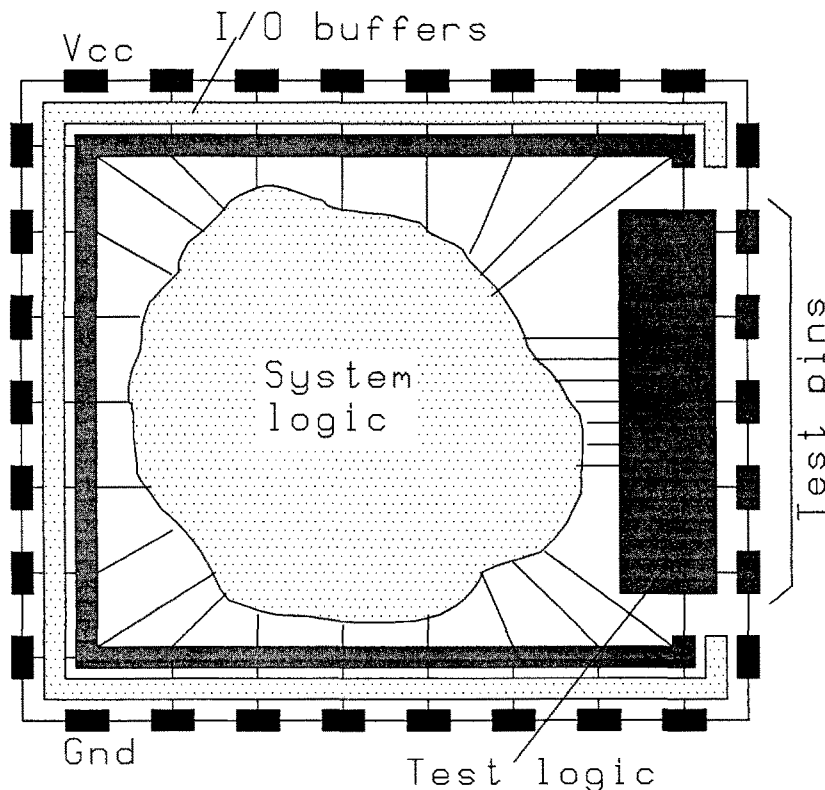


Fig. 1. Chip-level view.

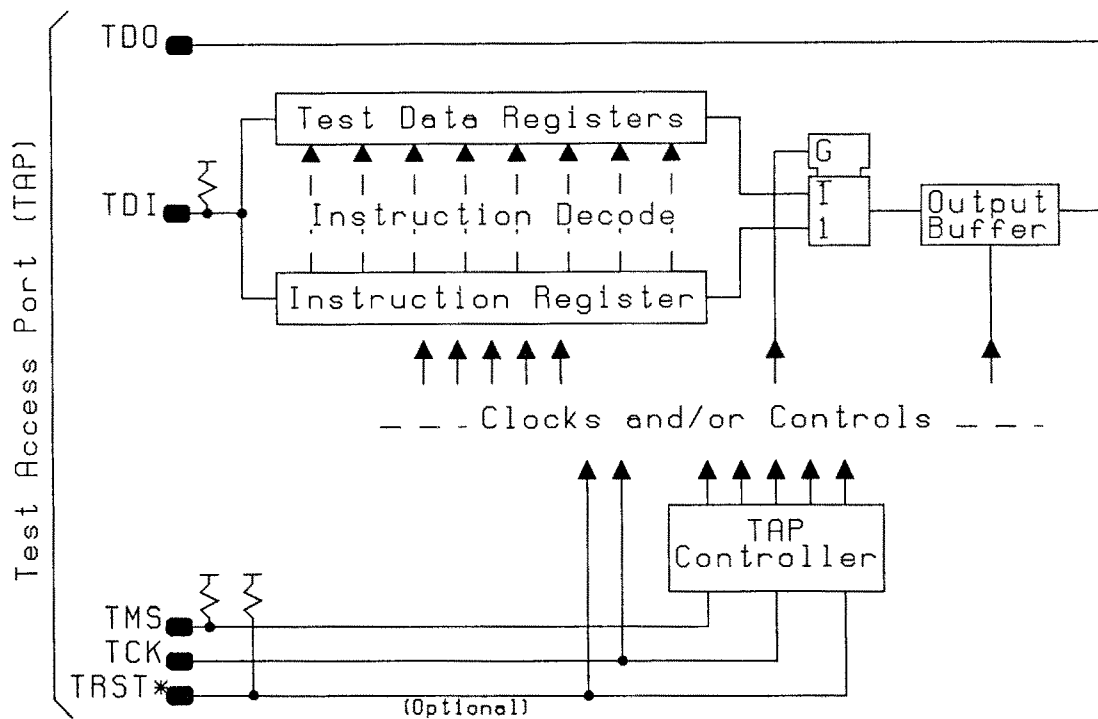


Fig. 2. IEEE Std 1149.1 test logic.

**The instruction register:** This shift-register-based circuit is serially loaded with the instruction that selects a test to be performed.

**The test data registers:** This is a bank of shift-register based circuits (see section 7). The stimuli or conditioning values required by a test are serially loaded into the test data register selected by the current instruction. Following execution of the test, results in test data registers can be shifted out for examination.

These circuit blocks are connected to a TAP which includes the four or, optionally, five signals used to control the operation of tests and to allow serial loading and unloading of instructions and test data. The TAP on an IC is directly analogous to the “diagnostic” socket provided on many automobiles—it allows an external test processor to control and to communicate with the various test features built into the product. When a number of ICs that implement the standard are combined on a PWB, they can be arranged in a single daisy chain with:

- the TDI input of the first IC in the chain connected to the board edge or to the output of an on-board master device;
- subsequent ICs each having their TDI connected to their predecessor's TDO; and

- the last IC's TDO being fed to the board edge or to the input of the on-board master device.

This arrangement supports the serial transfer of test data and instructions.

#### 4. The TAP

The TAP contains four or, optionally, five pins. These are:

**The test clock input (TCK):** This is independent of the system clock(s) for the chip so that test operations can be synchronized between the various chips on a PWB.

**The test mode select input (TMS):** The operation of the test logic is controlled by the sequence of 1s and 0s applied at this input. The sequence on TMS directs the TAP controller in its generation of the clock and control signals required by the other test logic blocks.

**The test data input (TDI):** Data applied at this serial input are fed either into the instruction register or into a test data register, depending on the sequence previously applied at TMS.

*The test data output (TDO):* This serial output from the instruction register or from a test data register depending on the sequence previously applied at TMS. During shifting, data applied at TDI will appear at TDO after a number of cycles of TCK determined by the length of the register included in the serial path. When data are not being shifted through the chip, TDO is set to an inactive drive state (e.g., high-impedance).

*The optional test reset input (TRST\*):* The test logic is designed so that it can be reset synchronously under control of TCK and TMS. TRST\* provides a supplementary reset mechanism. The test logic is reset when a 0 is applied at TRST\*.

The TDI, TMS, and TRST\* inputs are either equipped with a pull-up resistor or otherwise designed such that, when they are not driven from an external source, the test logic perceives a logic 1.

## 5. The TAP Controller

A key goal during the development of IEEE Std 1149.1 was to keep the number of pins in the TAP to a minimum, based on the knowledge that many ICs are pin- (rather than silicon-) limited. As test engineers are only too aware, designers are always reluctant to allocate pins for test purposes.

The TAP controller allows this goal to be met. It is a 16-state finite state machine that operates according to the state diagram shown in figure 3. Note that in the states whose names end *-DR*, the test data registers operate, while in those whose names end *-IR*, the instruction register operates. A move along a state transition arc occurs on every rising edge of TCK. The 0s and 1s shown adjacent to the state transition arcs show the value that must be present on TMS at the time of the next rising edge of TCK for the particular transition to occur.

Eight of the 16 controller states determine operation of the test logic, allowing the following test functions to be performed:

*Test-Logic-Reset:* All test logic is reset, allowing normal operation of the chip to occur without interference. Regardless of the starting state of the TAP controller, the *Test-Logic-Reset* controller state is reached by holding the TMS input at 1 and applying five rising edges at TCK. Alternatively, where TRST\* is provided, it can be used to force the

controller asynchronously into the *Test-Logic-Reset* controller state at any desired point during circuit operation.

*Run-Test/Idle:* The operation of the test logic in this controller state depends on the instruction held in the instruction register. When the instruction is, for example, one that activates a self-test, then the self-test will be run in this state. In another case, if the instruction in the instruction register is one that selects a data register for scanning, then the test logic will be idle.

*Capture-DR:* Each instruction must identify one or more test data registers that are enabled to operate in test mode when the instruction is selected. In this controller state, data are loaded from the parallel inputs of the selected test data registers into their shift-register paths.

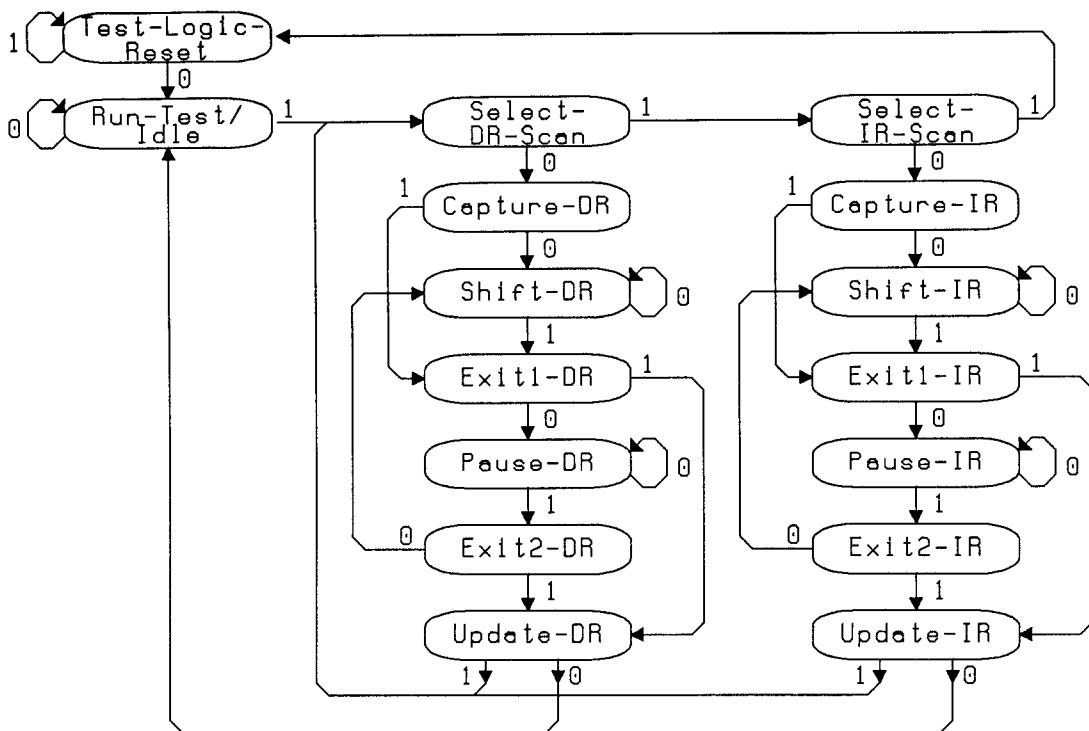
*Shift-DR:* Each instruction must identify a single test data register that is to be used to shift data between TDI and TDO in the *Shift-DR* controller state. Shifting allows the previously captured data to be examined and new test input data to be entered.

*Update-DR:* This controller state marks the completion of the shifting process. Some test data registers may be provided with a latched parallel output to prevent signals applied to the system logic, or through the chip's system pins, from rippling while new data are shifted into the register. Where such test data registers are selected by the current instruction, the new data is transferred to their parallel outputs in this controller state.

*Capture-IR, Shift-IR, and Update-IR:* These controller states are analogous to *Capture-DR*, *Shift-DR*, and *Update-DR*, respectively, but cause operation of the instruction register. By entering these states, a new instruction can be entered and applied to the test data registers and/or other specialized circuitry. The new instruction becomes "current" in the *Update-IR* controller state.

Note that in the *Update-DR* and *Update-IR* controller states, the described action takes place on the falling edge of TCK. In all other states, the described action takes place on the rising edge of TCK just before the controller leaves the state (see figure 4). Also note that TDO is active only in the *Shift-DR* and *Shift-IR* controller states.

In the remaining eight controller states, no operation of the test logic occurs—that is, the test logic is "idle." The "pause" states (*Pause-DR* and *Pause-IR*)



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.

Fig. 3. State diagram for the TAP controller.

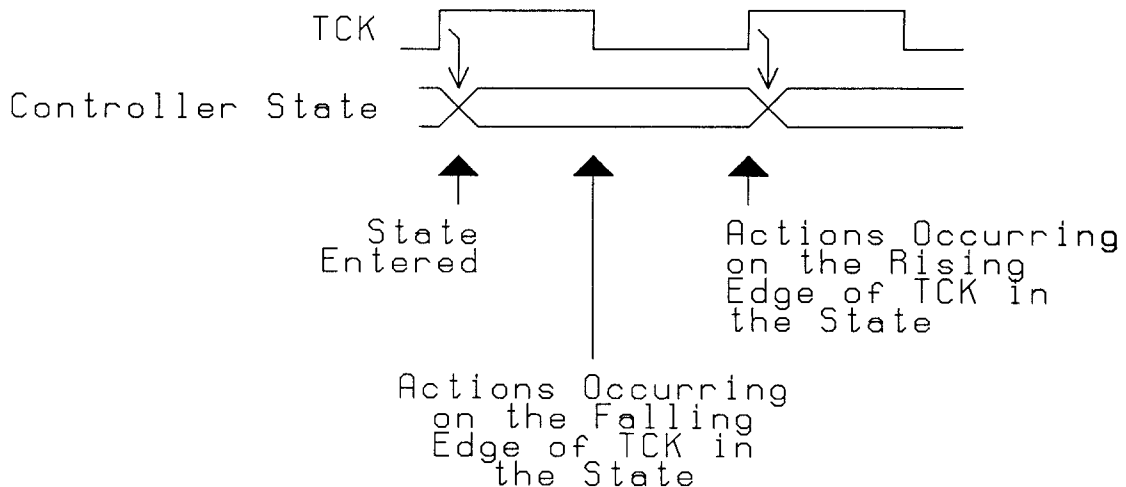


Fig. 4. The timing of events within a controller state.

are provided to allow the shifting process to be temporarily halted. This might occur while an ATE or other equipment controlling the test logic fetches more test data from backup memory (e.g., disc).

The final six controller states (*Select-DR-Scan*, *Select-IR-Scan*, *Exit1-DR*, *Exit1-IR*, *Exit2-DR*, and *Exit2-IR*) are decision points that allow choices to be made as to the route to be followed around the

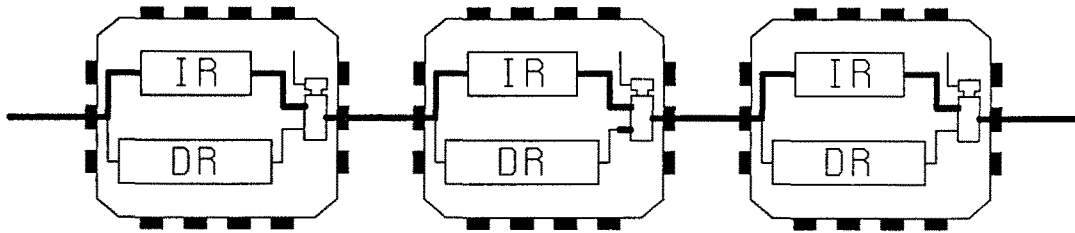


Fig. 5. Daisy-chain connection of instruction registers.

controller's state diagram. For example, in the *Exit-DR* controller state a choice is made between entry into the *Pause-DR* state or entry into the *Update-DR* state.

## 6. The Instruction Register

The instruction register provides one of the alternate serial paths between TDI and TDO. It operates when the instruction scanning portion of the controller state diagram is entered (i.e., the portion where state names end -IR).

The instruction register allows test instructions to be entered into each IC along the board-level path. At the board level, the instruction registers are daisy-chained together in the *Shift-IR* controller state (figure 5), so a different instruction can be loaded into each chip on the path if required.

The instruction register is a parallel-in, parallel-out shift-register. The parallel output is latched so that a new instruction can be shifted in without altering the instruction applied to the remainder of the test logic. The latched output is updated from the shift-register path in the *Update-IR* controller state; at this time, the new instruction becomes "current." In the *Test-Logic-Reset* controller state, the latched output is reset—to either the IDCODE or the BYPASS instruction depending on the set of test data registers built into the particular IC (see sections 8 and 9).

The instruction register must contain at least two stages (shown cross-hatched in figure 6). No maximum length is defined, since this will be determined by the number of test instructions provided by the particular IC. Stages  $I_1$  and  $I_0$  (i.e., the stages located nearest to the serial output) must be set to 0 and 1 respectively in the *Capture-IR* controller state. These fixed values assist in detecting and locating faults in the serial path through chips on a board [2]. Instruction register stages numbered  $I_2$  or greater are optional and can have a parallel input from which data (typically, status information) are loaded.

## 7. The Test Data Registers

The test logic design provides for a bank of test data registers as shown in figure 7.

IEEE Std 1149.1 specifies the design of three test data registers, two of which must be included in the design. The mandatory test data registers are the bypass and boundary scan registers. The provision of a device identification register is optional and further design-specific test data registers can be added as appropriate to a given design.

All test data registers are shift register based and operate according to the same principles:

- Operation of the various test data registers is controlled according to the current instruction. An instruction can place several test data registers into their test mode of operation, but it can select only one register for connection as the serial path between TDI and TDO in the *Shift-DR* controller state. On the other hand, it is important to note that one or more physical registers can be configured as one (virtual) test data register by a given instruction.
- Registers that are not enabled for test operation by the current instruction are configured so that they do not interfere with operation of the on-chip system logic (in the case of a register that can operate in either system or test mode, the system mode will be selected).
- Registers enabled for test operation by the current instruction will load data from their parallel inputs (if any) in the *Capture-DR* controller state and will make any new data available at their latched parallel outputs (if any) in the *Update-DR* controller state. In other words, the results of a test are sampled in the *Capture-DR* controller state and the new test stimulus is available, at the latest, in the *Update-DR* controller state.
- Where test execution is required between the *Update-DR* and *Capture-DR* controller states (e.g., execution of a self-test), this occurs in the *Run-Test/Idle* state.

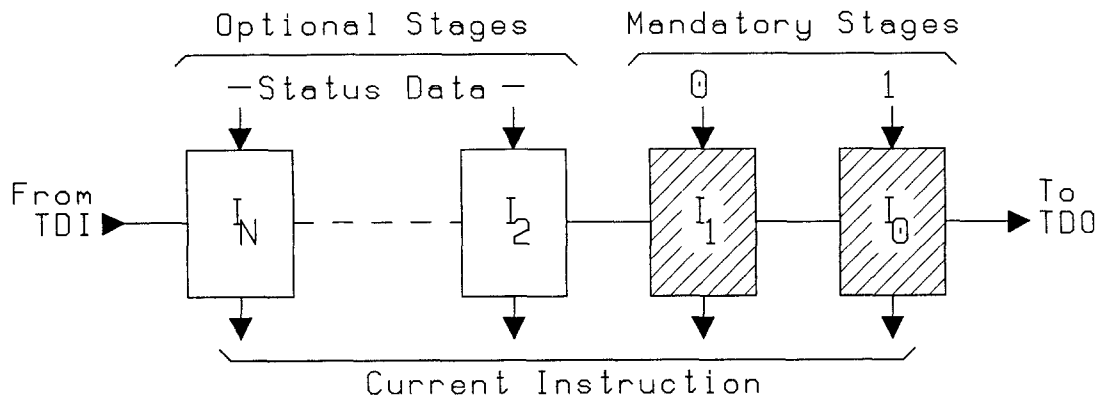


Fig. 6. The instruction register.

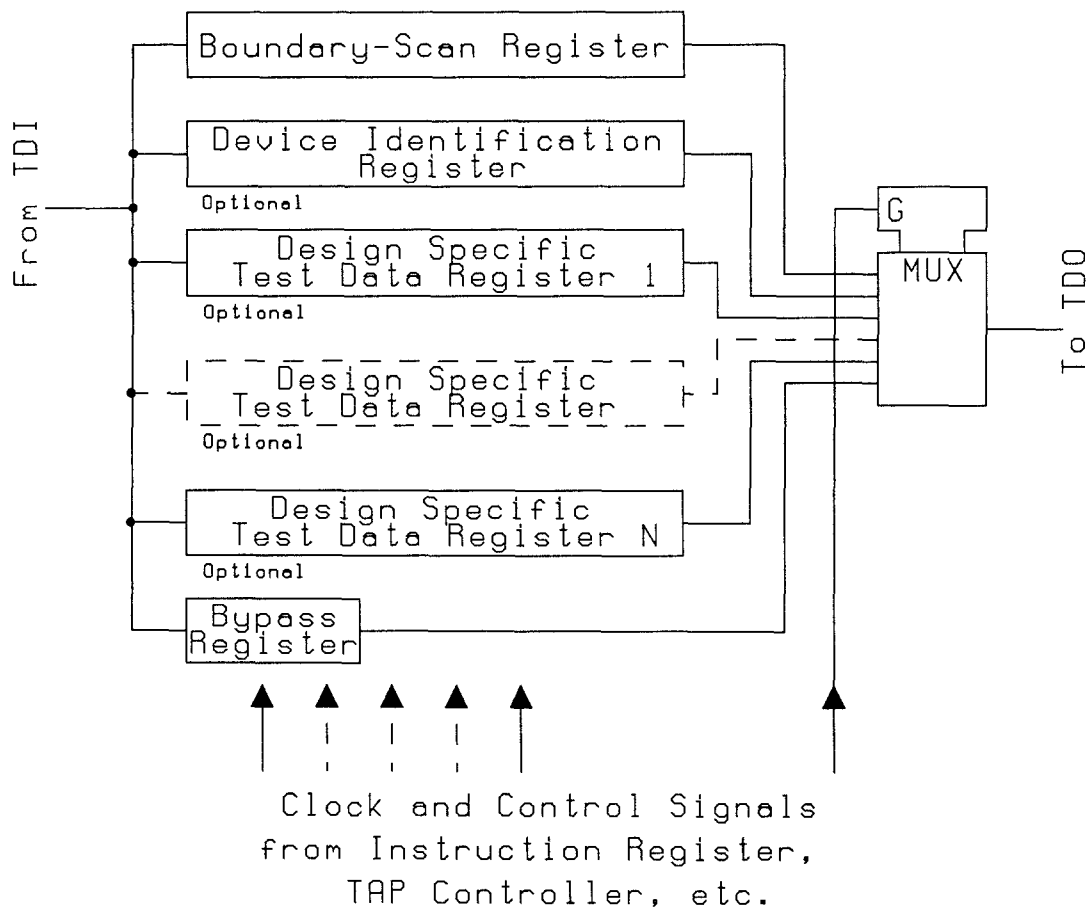


Fig. 7. Test data registers.

- The register selected by the instruction to be the serial path between TDI and TDO will shift data from TDI towards TDO in the *Shift-DR* controller state. All other test data registers enabled for test operation will hold their state while shifting occurs.

## 8. The BYPASS Instruction

Every IC that conforms to the standard must support the BYPASS instruction. A value (but not necessarily the only value) for the BYPASS instruction must be

“all-1s” (i.e., a logic 1 entered into each stage of the instruction register). In ICs that do not include the optional device identification register, the BYPASS instruction is forced onto the instruction register’s output in the *Test-Logic-Reset* state and thus becomes the current instruction whenever the test logic is reset.

The BYPASS instruction selects the bypass register as the serial path between TDI and TDO during the *Shift-DR* controller state. This register consists of a single parallel-in, serial-out shift-register stage that loads a constant logic 0 in the *Capture-DR* controller state when the BYPASS instruction is selected. The bypass register does not have a parallel data output so there is no significance to the data present in the register when shifting is completed. The operation of the register cannot interfere with that of the on-chip system logic.

As an example of an occasion when the bypass register might be used, consider a board containing 100 ICs, all with boundary scan and connected into a single serial chain, a small part of which is shown in figure 8. Assume that a need arises to access a test data register located in IC57, but that it is desired not to interfere with the operation of the remaining 99 ICs. (An example of such a situation might be when the target chip includes a “shadow” test data register that permits the state of its key internal registers to be read.) In this case, the required instruction would be loaded into IC57, with the BYPASS instruction being loaded into the other ICs. The serial bit stream shifted into TDI during the instruction scanning cycle would be:

111.....1111CCC...CCC111.....111

where CCC...CCC is the instruction to be loaded into IC57. As a result of use of the “all-1s” value for the BYPASS instruction, the complexity of the bit stream input to the serial path is considerably reduced. This is an important consideration, since it reduces the data

storage requirement for the automatic test equipment (ATE) or bus master chip that controls the operation of the board during test.

Once the instructions are loaded, a minimum-length serial path to and from the target chip is set up that allows access to the chip of interest in the minimum possible time, increasing test throughput.

## 9. The IDCODE and USERCODE Instructions

The IDCODE and USERCODE instructions select use of the optional device-identification register. In every IC that includes the device-identification register, the IDCODE instruction is forced onto the instruction register’s output in the *Test-Logic-Reset* state and thus becomes the current instruction whenever the test logic is reset.

The device-identification register allows a binary data pattern to be read from the chip that identifies the manufacturer, the part number, the variant, and (where appropriate) the programmed state. This information might be used to:

- adjust test program execution, depending on the source and/or variant of each chip present on the board;
- verify that the correct IC (or correctly programmed IC) has been mounted in each board location;
- establish which member of a plug-compatible family of boards is being tested.

The register contains 32 parallel-in, serial-out shift-register stages. Like the bypass register, the device-identification register does not have a parallel output and, in consequence, there is no significance to the data contained in the register when shifting terminates. Also like the bypass register, operation of the device-identification register can occur without interfering with normal chip operation.

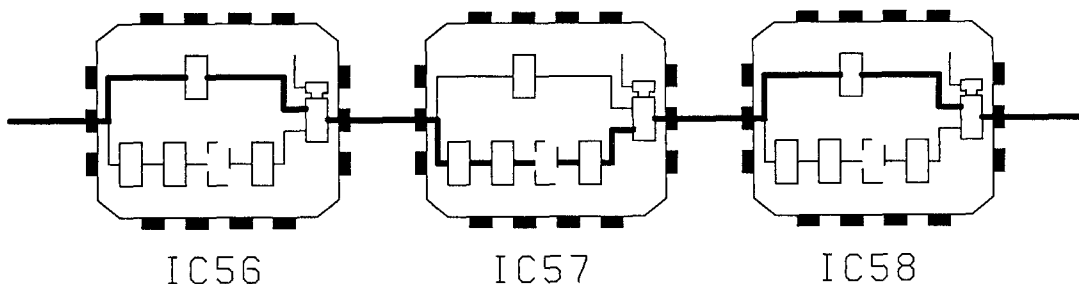


Fig. 8. Use of the bypass register.

Where an IC is programmed off-line (e.g., by blowing fuses or through some other nonreversible process), each stage must have a pair of alternative data inputs so that two different 32-bit codes can be loaded—one to identify the device and one to identify its programming. The former is loaded when the IDCODE instruction is selected, while the latter is loaded when the USERCODE instruction is selected. In all other types of IC, only one data input is required and the USERCODE instruction is not provided.

The structure of the data loaded into the device identification register in response to the IDCODE instruction is shown in figure 9. There are four separate fields:

1. *The header:* ID<sub>0</sub> loads a constant logic 1.
2. *The manufacturer code:* ID<sub>11</sub>–ID<sub>1</sub> load an 11-bit manufacturer code. This code is derived from a scheme managed by the Joint Electron Device Engineering Council (JEDEC) [1, 4]. The code can uniquely identify up to 2032 manufacturers (since 16 codes are not used). If more than 2032 manufacturer codes are issued by JEDEC, then the scheme will result in reuse of some code values within the manufacturer code field. However, the chance that a component from an incorrect manufacturer will have the same code *and* the same test functionality is acceptably low.
3. *The part-number code:* ID<sub>27</sub>–ID<sub>12</sub> provide a 16-bit part number, chosen by the manufacturer to distinguish a chip from the others that the company sells. In cases where more than 2<sup>16</sup> chip types are offered by a manufacturer, part-number codes might have to be reused. The objective is to minimize the chance that an incorrect chip in a given position on a board will have the same part number as the correct chip type—not to provide absolute identification of the IC.

4. *The version number code:* For chips that are manufactured in several different versions through their lives, bits ID<sub>31</sub>–ID<sub>28</sub> can be used to distinguish up to 16 variants.

The data loaded by the USERCODE instruction may be organized as the part's programmer sees fit. It must be programmable at the same time (and in the same way) that the function of the chip is programmed.

## 10. Boundary Scan Register Instructions

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 10. An example implementation for a cell that could be used in each of the locations shown in figure 10 is given in figure 11. The connections labeled PI, PO, SI, and SO in figure 11 are connected to adjacent cells, the on-chip system logic, and the system pins as shown in figure 10. The cell in figure 11 is designed to respond to the clock and control signals generated by the TAP controller. The mode input is controlled according to the type of pin connected to the cell (input, output, etc.) and the specific instruction selected.

Use of this sample 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 described below. Other cell designs are possible that meet

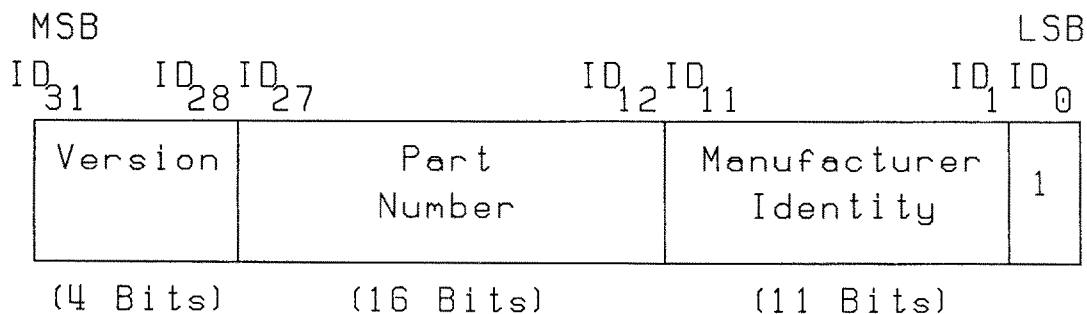


Fig. 9. Structure of the device identity code.

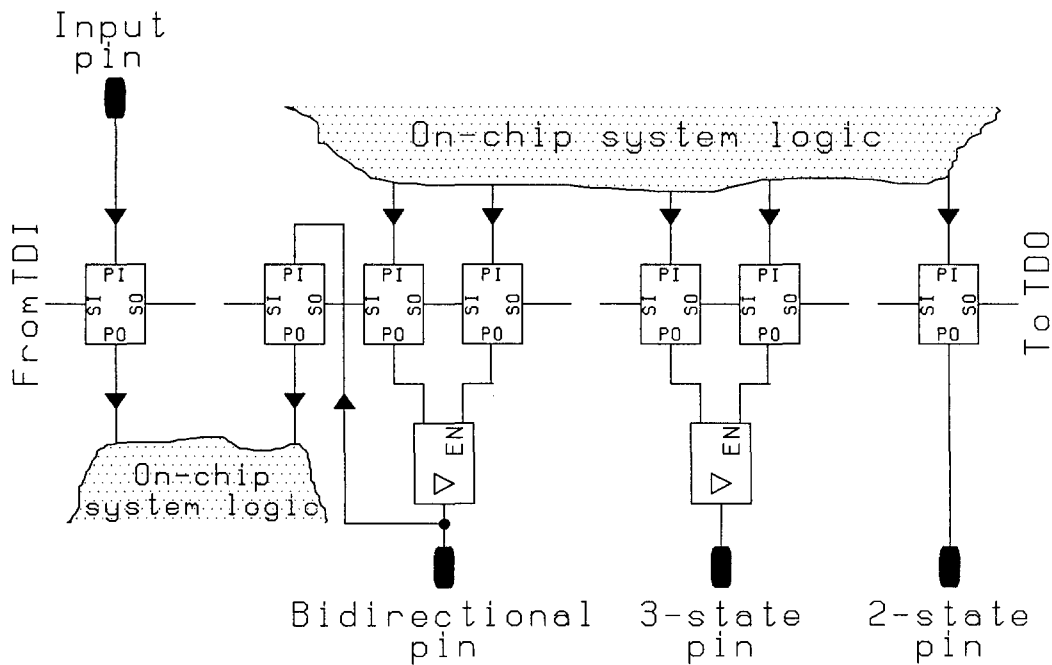


Fig. 10. A simplified view of the boundary scan register.

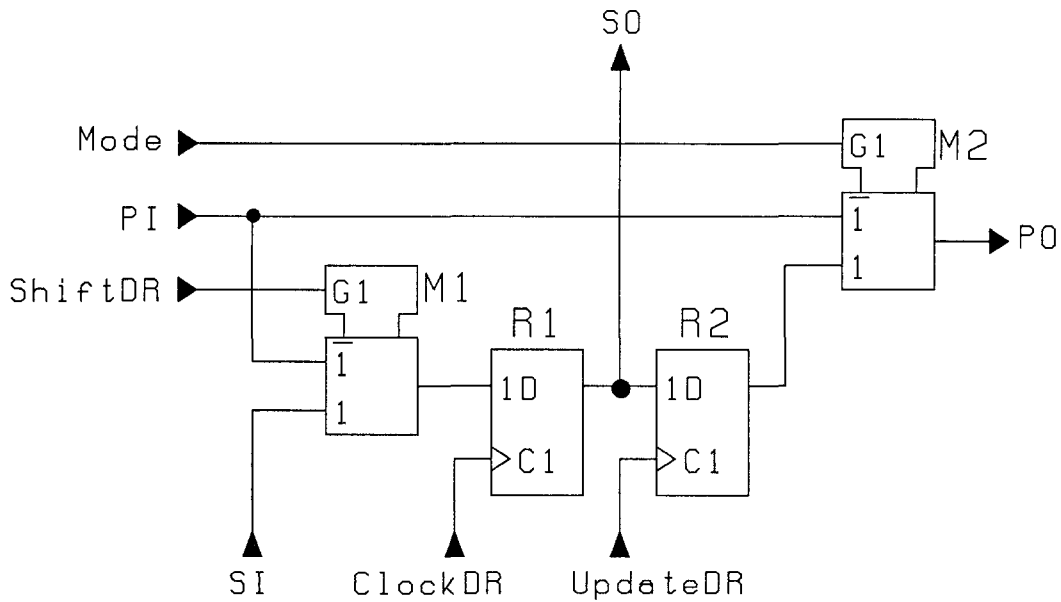


Fig. 11. An example boundary scan cell design.

the requirements of this standard for different sets of instructions. For example, if the *INTEST* instruction were not supported in a given design, R2 and M2 would be omitted from cells that feed data from a system pin to the on-chip system logic. The input, PI, would then be connected directly through the cell to the output, PO.

As the boundary scan register instructions are discussed, we note for each:

- whether the instruction is mandatory or optional;
- which test data registers can be connected in the serial path between TDI and TDO;

- c. restrictions (if any) on the choice of binary codes for the instruction; and
- d. flow of data between the component's system pins, the boundary scan register cells, and the on-chip system logic.

### 10.1. Sample/Preload

This mandatory instruction allows a snapshot of the normal operation of the component to be taken and examined. The taking of this snapshot is arranged so as to have no effect on the system operation that it monitors. The instruction also allows data values to be loaded onto the latched parallel outputs of the boundary scan shift register prior to selection of other boundary scan test instructions. This instruction must select only the boundary scan register to be connected between TDI and TDO in the *Shift-DR* controller state. There is no required binary value that must be decoded as the SAMPLE/PRELOAD instruction. When SAMPLE/PRELOAD is the current instruction, test logic operation is not permitted to have any effect on operation of system logic or on the flow of signals between the IC's system pins and on-chip system logic (i.e., the Mode signals for all cells should be set to 0).

The sampling mechanism of the SAMPLE/PRELOAD instruction has the effect of loading the states of all signals flowing through system pins into their corresponding boundary scan register cells on the rising edge of the clock feeding TCK during the *Capture-DR* controller state. This capability allows the state of the interconnect network of a machine to be captured at a desired moment (e.g., upon a failed parity check) by having the requisite edge of the signal feeding TCK triggered by a specified system event.

The preloading mechanism allows an initial data pattern to be placed at the latched parallel outputs of boundary scan register cells (e.g., in cells connected to pins driving off-chip) prior to selection of some other boundary scan test operation. For example, the EXTEST instruction (to be discussed next) is used for testing the interconnection of chips. It is very desirable that predetermined and nondamaging signals be driven out of chips on a board while the first test pattern to be applied to the interconnect is being scanned into the boundary scan register. This aim can be achieved by preloading the same boundary scan register. As soon as the EXTEST instruction has been transferred to the parallel output of the instruction register, preloaded data will be driven through the system output pins of each

IC; this situation will remain in effect while scanning of external test patterns is going on.

Shifting of data for sampling and preloading can occur concurrently when required—while sampled data is shifted out, preloading of new data can occur.

### 10.2. EXTEST

One of the principal motivations for the development of the Standard was the need for a noncontact method of testing board (and system) interconnect. [5, 6, 7] The EXTEST instruction is the key to the Standard's response to that need. The EXTEST instruction is mandatory. There may be more than one binary code that is interpreted as EXTEST; however, one of these codes must be the "all 0's" instruction code.

When EXTEST is the current instruction, the boundary scan register is the one and only register that is to be connected between TDI and TDO for data-scanning purposes. While the EXTEST instruction is selected in a chip,

- a. system logic of the chip must be controlled such that it cannot be damaged as a result of signals received at the system input or system clock input pins;
- b. the state of all signals driven from system output pins is completely defined by the data shifted into the boundary scan register and changes only on the falling edge of TCK in the *Update-DR* controller state; and
- c. the state of all signals received at system input pins is loaded into the boundary scan register on the rising edge of TCK in the *Capture-DR* controller state.

The EXTEST instruction allows circuitry external to the component package to be tested. Typically such circuitry would be the board interconnect. Clusters of components that lack boundary scan registers can be statistically tested using the same functionality although complex clusters would require more sophisticated diagnostic systems than would interconnect alone. In use of the EXTEST instruction, boundary scan register cells at output pins are used to apply test stimuli, while those at input pins capture test results. Captured results are scanned out of the serially linked boundary scan registers of a board while the next set of test input values is scanned in.

As was suggested above, the first test stimulus to be applied using the EXTEST instruction should be shifted into the boundary scan register using the SAMPLE/PRELOAD instruction. This is the most judicious

approach because, when the change to the EXTEST instruction takes place in the next occurring *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 should 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 a second time.

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.

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. As in the case of the BYPASS instruction, this may simplify demands on the ATE or bus master device which controls a test.

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. Where followed, this recommendation ensures that data shifted out of the component in response to the EXTEST instruction is not altered by the presence of faults in a chip's 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 boundary scan registers used to apply the test.

### 10.3. INTEST

INTEST is one of two optional instructions defined by the Standard that allow testing of on-chip system logic after a component is assembled on a board. The binary

value(s) that are decoded as the INTEST instruction may be selected by the component designer's specification. Using the INTEST instruction, test stimuli are shifted in one at a time and applied to the on-chip system logic. To achieve this, on-chip system logic must be capable of single-step operation while INTEST is the current instruction. Internal test results are captured in the IC's boundary scan register and are examined by subsequent shifting. Initial data set-up for such a test can be achieved using the SAMPLE/PRELOAD instruction.

The INTEST instruction must select the boundary scan register to be the one and only register connected between TDI and TDO for shifting access. When INTEST is the current instruction:

- the state of all signals driven from system output pins must be completely defined by data previously shifted into the boundary scan register and is allowed to change only on the falling edge of TCK in the *Update-DR* controller state;
- the state of all nonclock signals driven into the system logic from the boundary scan register must be completely defined by data previously shifted into the register;
- the state of all signals output from the system logic to the boundary scan register must be loaded into the register on the rising edge of TCK in the *Capture-DR* controller state.

The INTEST instruction allows slow-speed (static) testing of on-chip system logic. Each test pattern and response must be shifted through the boundary scan register. While an INTEST-based test is proceeding, the logic values at the component output pins are defined from the boundary scan register. This requirement ensures that surrounding components on an assembled board are supplied known signal levels while on-chip system logic testing is in progress. A consistent, "safe" 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 "safe" data pattern is then reloaded into boundary scan control cells and the cells associated with chip outputs each time a new INTEST test pattern is shifted into the boundary scan register.

As noted above, the approach taken with the INTEST instruction requires that 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. Note

that, for each such test step, the latched parallel output of the boundary scan cell at the system output pin is updated from data shifted in before the contents of the shift-register is overwritten with a new test response.

While the *INTEST* instruction is selected, an IC's boundary scan register assumes the role of the fixture and pin electronics of an ATE used for stand-alone component testing. Cells at nonclock system input pins are used to apply test stimuli, while those at system output pins capture responses. Stimuli and responses are moved into and out of the circuit by shifting the boundary scan register.

To achieve single-step operation, on-chip system logic can be expected to receive a sequence of clock events between application of the stimulus and capture of the response. A designer's specification of boundary scan cells for system clock input pins would allow clocks for on-chip system logic to be provided in several ways while *INTEST* is the current instruction. Here are some examples:

- a. The signals received at system clock pins can be fed directly to the on-chip system logic as they would be during nontest operation. When this approach is taken, off-chip clock sources should be handled in such a way that, during internal testing via *INTEST*, clock signals received by the component change state only in the *Run-Test/Idle* controller state. In this way, on-chip system logic operation can be inhibited while test data is shifted through the boundary scan register. Figure 12 illustrates how a system clock applied to a component should be controlled during *INTEST*-based testing of on-chip system logic.
- b. On-chip system logic can be supplied with clock signals derived from the input signal to TCK in the *Run-Test/Idle* controller state. In all other controller states, such clocks should not change state (figure 13).
- c. Circuitry may be built into the component, which allows on-chip system logic to complete one step of operation upon entry into the *Run-Test/Idle* controller state. For example, if the component were a microprocessor, it would be permitted to complete a single processing cycle; this might be achieved by internal generation of a pulse on the processor's hold signal. In such a case, the clock(s) applied at system clock pin(s) during the test could be free-running.
- d. Clock signals can be loaded serially via the boundary scan path in the same manner in which nonclock signals for the on-chip system logic are supplied. This would require the boundary scan register to be loaded for each distinct clock signal state—twice for a single-phase clock. This may be a hazard-prone operation for certain circuit designs.

The Standard recommends that 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* or *RUNBIST* (see below) instruction is selected should be independent of the operation of off-chip circuitry or board-level interconnections. Where followed, this approach ensures that data shifted out of a component in response to the *INTEST* instruction are not altered by the presence of faults in off-chip

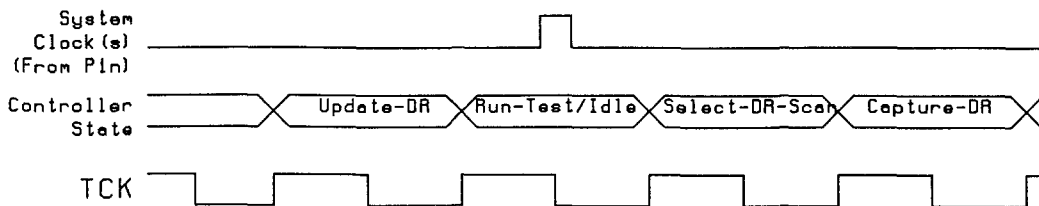


Fig. 12. Control of applied system clock during *INTEST*.

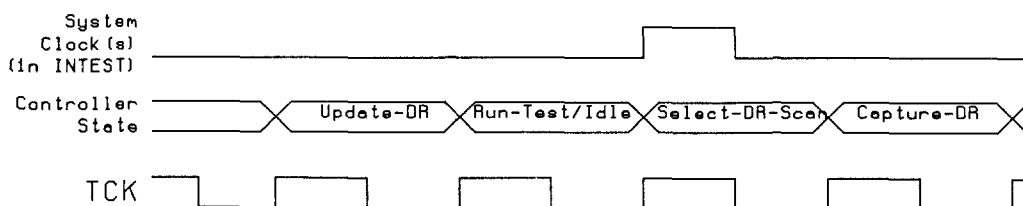


Fig. 13. Use of TCK as clock for on-chip system logic during *INTEST*.

system logic, board-level interconnections, etc. This will simplify diagnosis; any errors in the output bit-stream can be caused only by faults in on-chip system logic or the boundary scan register.

#### 10.4. RUNBIST

In many cases, it may be desirable to put more of the burden of test into the product itself, rather than on an external tester. Contemporary Built-In Self-Test (BIST) technology is beginning to make this possible with relatively little circuit overhead or performance penalty [8]. The optional RUNBIST instruction causes execution of a self-contained self-test of the component. It's binary encoding is left to the discretion of the designer. Use of the RUNBIST instruction allows a component user to determine the health of a component without the need to load complex data patterns and without the need for single-step operation (as required for the INTEST instruction). With this capability in place, it becomes possible for all components on a board that offer the RUNBIST instruction to execute their self-tests concurrently or in groups limited by power consumption or heat dissipation requirements; thus, a rapid health check for assembled boards can be provided.

RUNBIST is as an important instance of the flexibility that the Standard offers in providing a gateway (the TAP) through which powerful instructions can be passed to an IC when it has become a component of an assembled system.

The Standard allows the development of public and private instructions which could cause execution of BIST in various subsections of an IC. (Such capabilities could contribute to support of failure mode analysis). The RUNBIST instruction is intended to serve as a model for other BIST instructions and, when it is feasible to do so, to be a vehicle by means of which designers could link all self-testing circuitry within an IC into a single BIST process for the entire entity. The Standard recommends that RUNBIST be implemented wherever possible.

Self-test operation accessed via the RUNBIST instruction must execute only in the *Run-Test/Idle* controller state; and when RUNBIST is the current instruction, the test data register into which the results of the self-test(s) will be loaded must be connected for serial access between TDI and TDO in the *Shift-DR* controller state. (This register can be the boundary scan register.) The result of the self-test(s) executed in response to the RUNBIST instruction must be loaded

into the designated test data register no later than the rising edge of signal input to TCK in the *Capture-DR* controller state.

The developers of the Standard intended the RUNBIST instruction to be an encouragement for designers to create as simple an interface as possible for self-testing ICs embedded in systems. Therefore, they required that where a test data register (other than the boundary scan register) must be initialized prior to execution of the self-test, this must occur at the start of the self-test without any requirement to shift data into the component—there cannot be any requirement to enter seed values into any test data register other than the boundary scan register. Moreover, where a component includes multiple self-test functions, these may be executed in response to the RUNBIST instruction either concurrently or in a sequence determined by the component manufacturer. In the latter case, all sequencing must be taken care of within the component without requiring alteration of instruction register contents or other external direction.

A number of the requirements in the Standard are there for purposes of isolation—to assure a repeatable self-test result for fault-free circuits and to protect off-chip circuitry during self-test. The design of the component must ensure that results of self-tests executed in response to the RUNBIST instruction are not affected by signals received at nonclock system input pins; and data shifted out of a component following completion of execution of a self-test accessed using the RUNBIST instruction must be independent of the operation of off-chip circuitry or board-level interconnections. (The latter clause ensures that BIST result data shifted out of the chip are not altered by the presence of faults in off-chip system logic, board-level interconnections, etc. As discussed under INTEST, above, this simplifies diagnosis.)

When RUNBIST is the current instruction, the state of all signals driven from system output pins must be completely defined by data previously shifted into the boundary scan register (for example, by use of the *SAMPLE/PRELOAD* instruction as described under INTEST, above); and states of parallel output registers or latches in boundary scan register cells located at system output pins (2-state, 3-state, or bidirectional) are not permitted to change while the RUNBIST instruction is selected. In contrast to the INTEST instruction, the data values driven through the system output pins are held while the RUNBIST instruction is selected. For a boundary scan register cell located at a system output pin (see figure 11), the *Update-DR*

signal should be held at 0 while the RUNBIST instruction is selected and the Mode input should be held at 1.

While the RUNBIST instruction is selected, boundary scan register cells associated with nonclock system inputs of a chip may simply be loaded with constant, safe values. Alternatively, the boundary scan register may act as a pattern generator and/or signature compactor in the *Run-Test/Idle* controller state—provided the states of parallel output registers or latches are unchanging as required above.

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 they would be during nontest operation of the component. Where this is done, the design of the component must ensure that self-test executes only in the *Run-Test/Idle* controller state. However, the clock may be active in other controller states.
- b. The on-chip system logic can be supplied with clock signals derived from the signal input at TCK in the *Run-Test/Idle* controller state. In this instance, in all other controller states, the clocks should not change state.

There are certain, additional, commonsense requirements placed on an IC that implements RUNBIST and on such an IC's documentation. All stages of the test data register selected by the RUNBIST instruction must be set to determinate logic states (0 or 1) in the *Capture-DR* controller state (i.e., while the test result is loaded). A duration for the test executed in response to the RUNBIST instruction must be specified (e.g., by citing a number of rising edges of the signal at TCK or of a system clock). Because it is likely that self-test in one component may complete before the self-test in another, the test results produced by the execution of the RUNBIST instruction and deposited in the specified test data register ready for shifting out of the IC must be stable through the period of delay before the *Capture-DR* controller state is entered, which period is likely to be unpredictable at the time of the component's design. Use of the RUNBIST instruction must give the same result in all versions of a component. These requirements were included in the standard to ensure that the test for an assembled board is independent of the versions of components mounted on it. Such

independence is an important consideration when working in a maintenance or repair environment, where the versions of components used on a board may very well be unknown. The Standard's requirement 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 other test data register connected between TDI and TDO during the time the RUNBIST instruction is current.

## 11. Conclusion

We have outlined the principal features of ANSI/IEEE Std 1149.1 to give the reader a basic understanding of the circuitry defined by the standard and its operation. Those readers intending to implement ICs, tools, or ATE systems to support the standard should consult the standard document [1] before doing so, since the standard contains much more information than it has been possible to include here.

## References

1. ANSI/IEEE Std 1149.1—*Standard Test Access Port and Boundary-Scan Architecture*, IEEE, New York, 1990.
2. C.M. Maunder and R.E. Tulloss, *An Introduction to the Boundary-Scan Standard*, IEEE Computer Society Press: Los Alamitos, CA, 1990.
3. B. Konemann et al., "Built-in logic block observation techniques," *IEEE Test Conf. Proc.*, IEEE Computer Society Press: Los Alamitos, CA, 1979, pp. 37–41.
4. Joint Electron Device Engineering Council, "Standard Manufacturer's Identification Code," *JEDEC Publication 106-A*, July 1986. (Obtainable from JEDEC, 2001 Eye Street, N.W., Washington, D.C. 20006, USA.)
5. F.P.M. Beenker, "Systematic and structured methods for digital board testing," *IEEE Test Conf. Proc.*, IEEE Computer Society Press: Los Alamitos, CA, 1985, pp. 380–385.
6. C.W. Yau and N. Jarwala, "A new framework for analyzing test generation and diagnosis algorithms," *IEEE Test Conf. Proc.*, IEEE Computer Society Press: Los Alamitos, CA, 1989, pp. 63–70.
7. N. Jarwala and C.W. Yau, "A unified theory for designing optimal test generation and diagnosis algorithms for board interconnect," *IEEE Test Conf. Proc.*, IEEE Computer Society Press: Los Alamitos, CA, 1989, pp. 71–77.
8. H.N. Scholtz et al., "ASIC implementations of boundary-scan and BIST," *8th International Custom Microelectronics Conference Proceedings*, Prodex Ltd.: Saffron Walden, U.K., 1988, pp. 43.1–43.9.

**Colin Maunder** is an Engineering Adviser at the British Telecommunications Research Laboratories, Martlesham Heath, Ipswich, U.K. His work includes design-for-test consultancy on chip and system development projects and new techniques for the maintenance and repair of electronic equipment.

Mr. Maunder has been involved in research into test generation and design-for-testability since 1976; and has lectured on these topics in Europe and North America. He worked (1979–1981) on the development of the CAMELOT testability assessment program; contributed (1983–1985) to the development of HITEST—the first commercial knowledge-based test generation system; and provided the design-for-test features of several British Telecom integrated circuits.

In 1986 he helped draft the first technical proposal of the Joint Test Action Group (JTAG) and later became chairman of JTAG's Technical Subcommittee and of the IEEE P1149.1 Working Group. In January 1991 he became vice-chair of the working group. He is a charter member of the IEEE Computer Society Test Technology Technical Committee's Testability Bus Standards Steering Committee, and is on the organizing committees for the European Design-for-Test Workshop and the European Test Conference.

Mr. Maunder received a B.Sc. in Physics from Imperial College, London, in 1973. He is a Chartered Engineer, a member of the IEE (U.K.), and a Senior Member of the IEEE.

**Rodham Tulloss** is a Distinguished Member of Technical Staff at AT&T Bell Laboratories, Engineering Research Center, Princeton, NJ. During 11 years as Supervisor, he initiated and led the development of research into fault simulation, test data translation, automated test generation, built-in self-test, and boundary scan. He is presently involved with standards development and the introduction of built-in self-test and boundary scan into AT&T designs.

Dr. Tulloss founded the North American JTAG interest group and has served as cochair and vice-chair of the IEEE P1149.1 Working Group. He became the chair of the working group in January 1991. He is a Senior Member of the IEEE, a charter member of the IEEE Computer Society Testability Bus Standards Steering Committee; and, from 1981 to 1983, was coeditor of the Society TTTC Newsletter.

As technical consultant to the 1989 Satellite Seminar and to the program to develop educational transatlantic television programs in electronic engineering, he has played a significant role in educating the engineering community about boundary scan.

Dr. Tulloss received his Ph.D. from the University of California, Berkeley, in 1971. He also holds an M.S. in Mathematics from the University of California and a B.S. in Mathematics from Union College, Schenectady, NY. Outside of engineering, he is a prize-winning poet and a recognized expert on fungi of the genus *Amanita*.