9. DESIGN FOR TESTABILITY

About This Chapter

Previous chapters have dealt with the complexity of test generation and algorithms for constructing a test for a complex circuit. Test complexity can be converted into costs associated with the testing process. There are several facets to this cost, such as the cost of test pattern generation, the cost of fault simulation and generation of fault location information, the cost of test equipment, and the cost related to the testing process itself, namely the time required to detect and/or isolate a fault. Because these costs can be high and may even exceed design costs, it is important that they be kept within reasonable bounds. One way to accomplish this goal is by the process of *design for testability* (DFT), which is the subject of this chapter. The model of test being considered here is that of external testing.

The testability of a circuit is an abstract concept that deals with a variety of the costs associated with testing. By increasing the testability of a circuit, it is implied that some function of these costs is being reduced, though not necessarily each individual cost. For example, scan designs may lower the cost of test generation but increase the number of I/O pins, area, and test time.

Controllability, *observability*, and *predictability* are the three most important factors that determine the complexity of deriving a test for a circuit. The first two concepts have been discussed in Chapter 6. *Predictability* is the ability to obtain known output values in response to given input stimuli. Some factors affecting predictability are the initial state of a circuit, races, hazards, and free-running oscillators. Most DFT techniques deal with ways for improving controllability, observability, and predictability.

Section 9.1 deals briefly with some measures associated with testing, as well as methods for computing numeric values for controllability and observability. Section 9.2 discusses ad hoc DFT techniques; Section 9.3 deals with ad hoc scan-based DFT techniques. Sections 9.4 through 9.7 present structured scan-based designs, and Section 9.8 deals with board-level and system-level DFT approaches. Section 9.9 deals with advanced scan concepts, and Section 9.10 deals with the JTAG/IEEE 1149.1 proposed standards relating to boundary scan.

9.1 Testability

Testability is a design characteristic that influences various costs associated with testing. Usually it allows for (1) the status (normal, inoperable, degraded) of a device to be determined and the isolation of faults within the device to be performed quickly, to reduce both the test time and cost, and (2) the cost-effective development of the tests to determine this status. *Design for testability techniques* are design efforts specifically employed to ensure that a device is testable.

Two important attributes related to testability are controllability and observability. *Controllability* is the ability to establish a specific signal value at each node in a circuit by setting values on the circuit's inputs. *Observability* is the ability to determine the

signal value at any node in a circuit by controlling the circuit's inputs and observing its outputs.

The degree of a circuit's controllability and observability is often measured with respect to whether tests are generated randomly or deterministically using some ATG algorithm. For example, it may be difficult (improbable) for a random test pattern generation to drive the output of a 10-input AND gate to a 1, but the *D*-algorithm could solve this problem with one table lookup. Thus, the term "random controllability" refers to the concept of controllability when random tests are being used. In general, a circuit node usually has poor random controllability if it requires a unique input pattern to establish the state of that node. A node usually has poor controllability if a lengthy sequence of inputs is required to establish its state.

Circuits typically difficult to control are decoders, circuits with feedback, oscillators, and clock generators. Another example is a 16-bit counter whose parallel inputs are grounded. Starting from the reset state, 2^{15} clock pulses are required to force the most significant bit to the value of 1.

A circuit often has poor random observability if it requires a unique input pattern or a lengthy complex sequence of input patterns to propagate the state of one or more nodes to the outputs of the circuit. Less observable circuits include sequential circuits; circuits with global feedback; embedded RAMs, ROMs, or PLAs; concurrent error-checking circuits; and circuits with redundant nodes.

The impact of accessibility on testing leads to the following general observations:

- Sequential logic is much more difficult to test than combinational logic.
- Control logic is more difficult to test than data-path logic.
- Random logic is more difficult to test than structured, bus-oriented designs.
- Asynchronous designs or those with unconstrained timing signals are much more difficult to test than synchronous designs that have easily accessible clock generation and distribution circuits.

9.1.1 Trade-Offs

Most DFT techniques deal with either the resynthesis of an existing design or the addition of extra hardware to the design. Most approaches require circuit modifications and affect such factors as area, I/O pins, and circuit delay. The values of these attributes usually increase when DFT techniques are employed. Hence, a critical balance exists between the amount of DFT to use and the gain achieved. Test engineers and design engineers usually disagree about the amount of DFT hardware to include in a design.

Increasing area and/or logic complexity in a VLSI chip results in increased power consumption and decreased yield. Since testing deals with identifying faulty chips, and decreasing yield leads to an increase in the number of faulty chips produced, a careful balance must be reached between adding logic for DFT and yield. The relationship among fault coverage, yield, and defect level is illustrated in Figure 5.1. Normally yield decreases linearly as chip area increases. If the additional hardware required to support DFT does not lead to an appreciable increase in fault coverage, then the defect level will increase. In general, DFT is used to reduce test generation costs, enhance

the quality (fault coverage) of tests, and hence reduce defect levels. It can also affect test length, tester memory, and test application time.

Normally all these attributes are directly related to one another. Unfortunately, when either deterministic or random test patterns are being used, there is no reliable model for accurately predicting, for a specific circuit, the number of test vectors required to achieve a certain level of fault coverage. For many applications, test development time is a critical factor. By employing structured DFT techniques described later in this chapter, such time can often be substantially reduced, sometimes from many months to a few weeks. This reduction can significantly affect the time to market of a product and thus its financial success. Without DFT, tests may have to be generated manually; with DFT they can be generated automatically. Moreover, the effectiveness of manually generated tests is often poor.

The cost of test development is a one-time expense, and it can be prorated over the number of units tested. Hence, on the one hand, for a product made in great volume, the per unit cost of test development may not be excessively high. On the other hand, test equipment is expensive to purchase, costs a fixed amount to operate, and becomes obsolete after several years. Like test development, ATE-associated costs must be added to the cost of the product being tested. Hence, test time can be a significant factor in establishing the cost of a unit. Again, a conflict exists in that to reduce test application costs, shorter tests should be used, which leads to reduced fault coverage and higher defect levels. This is particularly true when testing microprocessors and large RAMs.

If a faulty chip is put into a printed circuit board, then the board is faulty. Identifying a faulty chip using board-level tests is usually 10 to 20 times more costly than testing a chip. This does not imply that chip-level testing should be done on all chips. Clearly this decision is a function of the defect level associated with each type of chip.

When a faulty board is put into a system, system-level tests are required to detect and isolate the faulty board. This form of testing again is about 10 to 20 times more expensive than board-level testing in identifying a faulty board. Normally, board yield is low enough to warrant that all boards be tested at the board level.

9.1.2 Controllability and Observability

In the early 70s attempts were made to quantify the concepts of controllability and observability. A precise definition for these abstract concepts and a formal method for computing their value would be desirable. The idea was to modify the design of a circuit to enhance its controllability and observability values. This modification would lead to a reduction in deterministic test generation costs. This analysis process must be relatively inexpensive, since if the computation were too involved, then the cost savings in ATG would be offset by the cost of analysis and DFT. Unfortunately, no such easily computable formal definition for these concepts has as yet been proposed. What is commonly done is that a procedure for computing controllability and observability is proposed, and then these concepts are defined by means of the procedure.

The pioneering work in this area was done by Rutman [1972] and independently by Stephenson and Grason [1976] and Grason [1979]. This work relates primarily to deterministic ATG. Rutman's work was refined and extended by Breuer [1978].

These results were then popularized in the papers describing the Sandia Controllability/Observability Analysis Program (SCOAP) [Goldstein 1979, Goldstein and Thigen 1980]. This work, in turn, formed the basis of several other systems that compute deterministic controllability and observability values, such as TESTSCREEN [Kovijanic 1979, 1981], CAMELOT (Computer-Aided Measure for Logic Testability) [Bennetts *et al.* 1980], and VICTOR (VLSI Identifier of Controllability, Testability, Observability, and Redundancy) [Ratiu *et al.* 1982].

The basic concepts used in these systems have previously been presented in Section 6.2.1.3. These programs compute a set of values for each line in a circuit. These values are intended to represent the relative degree of difficulty for computing an input vector or sequence for each of the following problems:

1. setting line x to a 1 (1-controllability);
2. setting line x to a 0 (0-controllability);
3. driving an error from line x to a primary output (observability).

Normally large values imply worse testability than smaller ones. Given these values, it is up to the designer to decide if they are acceptable or not. To reduce large values, the circuit must be modified; the simplest modifications deal with adding test points and control circuitry. These techniques will be illustrated later in this chapter. Normally the controllability and observability values of the various nodes in a circuit are combined to produce one or more testability values for the circuit.

The problem here is two-fold. First, the correlation between testability values and test generation costs has not been well established [Agrawal and Mercer 1982]. Second, it is not clear how to modify a circuit to reduce the value of these testability measures. Naive rule-of-thumb procedures, such as add test points to lines having the highest observability values and control circuitry to lines having the highest controllability values, are usually not effective. A method for automatically modifying a design to reduce several functions of these testability values, such as the maximum value and the sum of the values, has been developed by Chen and Breuer [1985], but its computational complexity is too high to be used in practice. Their paper introduces the concept of *sensitivity*, which is a measure of how the controllability and observability values of the entire circuit change as one modifies the controllability and/or observability of a given line.

Testability measures can also be derived for testing with random vectors. Here these measures deal with the probability of a random vector setting a specific node to the value of 0 or 1, or propagating an error from this node to a primary output. For this case too there is not a strong correlation between testability values and test generation costs [Savir 1983]. In summary, the testability measures that exist to date have not been useful in guiding the design process.

Recall that controllability and observability figures are used to guide decision making in deterministic ATG algorithms (see Chapter 6). These concepts have proven very useful in this context [Lioy and Messalama 1987, Chandra and Patel 1989].

9.2 Ad Hoc Design for Testability Techniques

In this section we will present several well known ad hoc designs for testability techniques. Many of these techniques were developed for printed circuit boards; some are applicable to IC design. They are considered to be ad hoc (rather than algorithmic) because they do not deal with a total design methodology that ensures ease of test generation, and they can be used at the designer's option where applicable. Their goal is to increase controllability, observability, and/or predictability.

The DFT techniques to be discussed in this section deal with the following concepts:

- test points,
- initialization,
- monostable multivibrators (one-shots),
- oscillators and clocks,
- counters/shift registers,
- partitioning large circuits,
- logical redundancy,
- breaking global feedback paths.

9.2.1 Test Points

Rule: Employ test points to enhance controllability and observability.

There are two types of test points, referred to as control points (CP) and observation points (OP). *Control points* are primary inputs used to enhance controllability; *observation points* are primary outputs used to enhance observability.

Figure 9.1(a) shows a NOR gate G buried within a large circuit. If this circuit is implemented on a printed circuit board, then the signal G can be routed to two pins, A and A', and a removable wire, called a *jumper*, can be used to connect A to A' externally. By removing the jumper, the external test equipment can monitor the signal G while applying arbitrary signals to A' (see Figure 9.1(b)). Thus A acts as an observation point and A' as a control point. Several other ways of adding test points, applicable to both boards and chips, are shown in Figure 9.1.

In Figure 9.1(c), a new control input CP has been added to gate G, forming the gate G^*. If $CP = 0$, then $G^* = G$ and the circuit operates in its normal way. For $CP = 1$, $G^* = 0$; i.e., we have forced the output to 0. This modified gate is called a *0-injection circuit* (0-*I*). The new observation test point OP makes the signal G^* directly observable.

Figure 9.1(d) illustrates the design of a 0/1 injection circuit, denoted by 0/1-*I*. Here G has been modified by adding the input $CP1$, and a new gate G' has been added to the circuit. If $CP1 = CP2 = 0$, then $G' = G$, which indicates normal operation. $CP1 = 1$ inhibits the normal signals entering G^*, sets $G^* = 0$, hence $G' = CP2$. Thus G' can be easily controlled to either a 0 or a 1. Figure 9.1(e) shows the use of a multiplexer (MUX) as a 0/1 injection circuit, where $CP2$ is the select line. In general, if G is an arbitrary signal line, then inserting either an AND gate or a NOR gate in this line

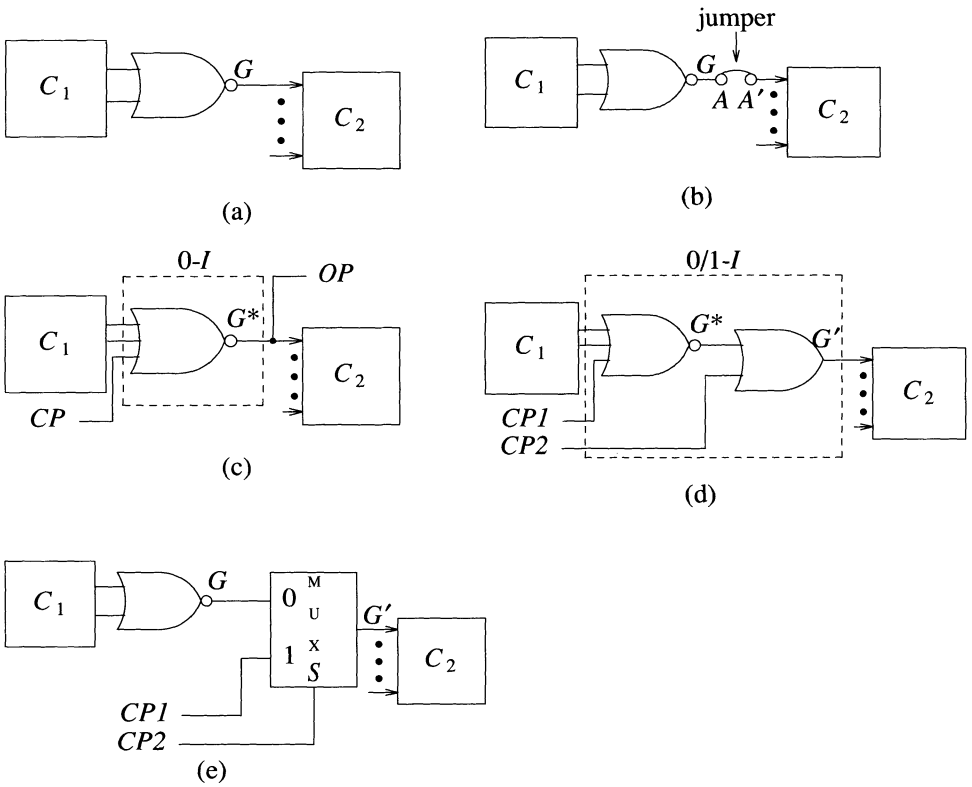


Figure 9.1 Employing test points (a) Original circuit (b) Using edge pins to achieve controllability and observability (c) Using a CP for 0-injection and an OP for observability (d) Using a 0/1 injection circuit (e) Using a MUX for 0/1 injection

creates a 0-injection circuit; inserting an OR or NAND gate creates a 1-injection circuit. Inserting a series of two gates, such as NOR-NOR, produces a *0/1-injection circuit.*

The major constraint associated with using test points is the large demand on I/O pins. This problem can be alleviated in several ways. To reduce output pins, a multiplexer can be used, as shown in Figure 9.2. Here the $N = 2^n$ observation points are replaced by a single output Z and n inputs required to address a selected observation point. The main disadvantage of this technique is that only one observation point can be observed at a time; hence test time increases. If many outputs must be monitored for each input vector applied, then the UUT clock must be stopped while the outputs are sampled. For dynamic logic this can lead to problems. To reduce the pin count even further, a counter can be used to drive the address lines of the multiplexer. Now, for each input test vector, each observation point will be sampled in turn. The counter must be clocked separately from the rest of the circuit. This DFT technique clearly suggests one trade-off between test time and I/O pins.

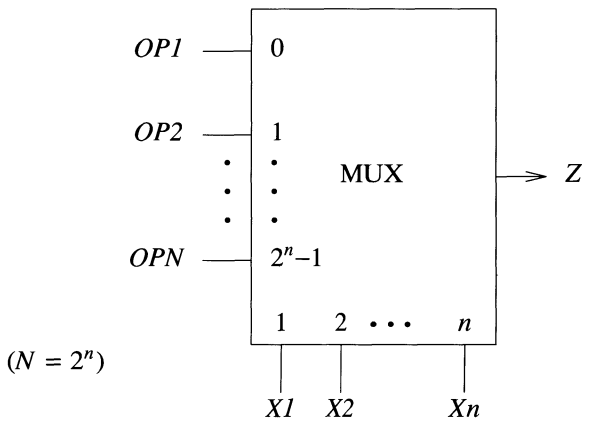


Figure 9.2 Multiplexing monitor points

A similar concept can be used to reduce input pin requirements for control inputs (see Figure 9.3). The values of the $N = 2^n$ control points are serially applied to the input Z, while their address is applied to $X1$, $X2$, ..., Xn. Using a demultiplexer, these N values are stored in the N latches that make up register R. Again, a counter can be used to drive the address lines of the demultiplexer. Also, N clock times are required between test vectors to set up the proper control values.

Another method that can be used, together with a multiplexer and demultiplexer, to reduce I/O overhead is to employ a shift register. This approach will be discussed in Section 9.3 dealing with scan techniques.

It is also possible to time-share the normal I/O pins to minimize the number of additional I/O pins required to be test points. This is shown in Figure 9.4. In Figure 9.4(a) a multiplexer has been inserted before the primary output pins of the circuit. For one logic value of the select line, the normal functional output signals are connected to the output pins; for the other value, the observation test points are connected to the output pins.

In Figure 9.4(b), a demultiplexer (DEMUX) is connected to the n primary inputs of a circuit. Again, by switching the value on the select lines, the data on the inputs can be sent to either the normal functional inputs or the control test points. This register R is used to hold the data at control test points while data are applied to the normal functional inputs.

The selection of signals to be made easily controllable or observable is based on empirical experience. Examples of good candidates for control points are as follows:

1. control, address, and data bus lines on bus-structured designs;
2. enable/hold inputs to microprocessors;

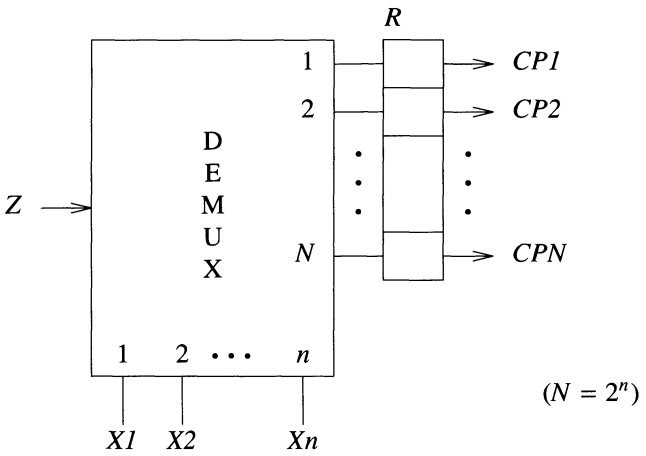


Figure 9.3 Using a demultiplexer and latch register to implement control points

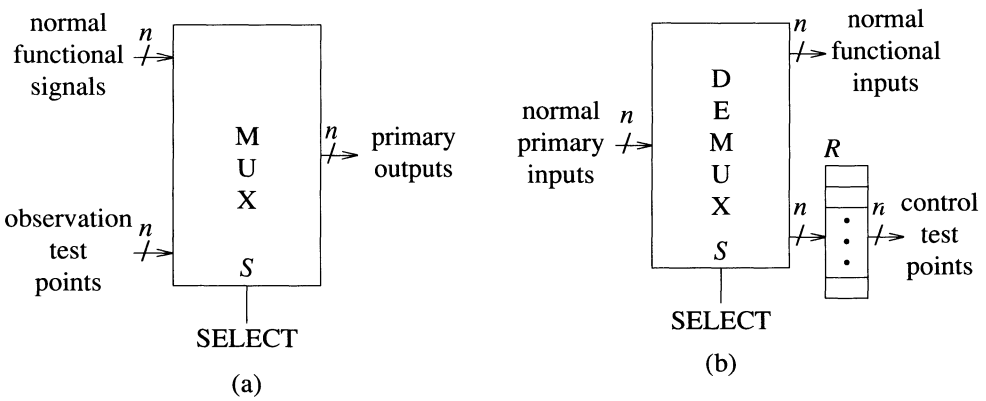


Figure 9.4 Time-sharing I/O ports

3. enable and read/write inputs to memory devices;
4. clock and preset/clear inputs to memory devices such as flip-flops, counters, and shift registers;
5. data select inputs to multiplexers and demultiplexers;
6. control lines on tristate devices.

Examples of good candidates for observation points are as follows:

1. stem lines associated with signals having high fanout;
2. global feedback paths;
3. redundant signal lines;
4. outputs of logic devices having many inputs, such as multiplexers and parity generators;
5. outputs from state devices, such as flip-flops, counters, and shift registers;
6. address, control, and data busses.

A heuristic procedure for the placement of test points is presented in [Hayes and Friedman 1974].

9.2.2 Initialization

Rule: Design circuits to be easily initializable.

Initialization is the process of bringing a sequential circuit into a known state at some known time, such as when it is powered on or after an initialization sequence is applied. The need for ease of initialization is dictated in part by how tests are generated. In this section we assume that tests are derived by means of an automatic test pattern generator. Hence, circuits requiring some clever input-initialization sequence devised by a designer should be avoided, since such sequences are seldom derived by ATG software. Later the concept of scan designs will be introduced. Such designs are easy to initialize.

For many designs, especially board designs using SSI and MSI components, initialization can most easily be accomplished by using the asynchronous preset (*PR*) or clear (*CLR*) inputs to a flip-flop. Figure 9.5 illustrates several common ways to connect flip-flop preset and clear lines for circuit initialization. In Figure 9.5(a), the preset and clear lines are driven from input pins; Figure 9.5(b) shows the same concept, but here the lines are tied via a pull-up resistor to V_{cc}, which provides for some immunity to noise. In Figure 9.5(c), the preset is controlled from an input pin while the clear line is deactivated. The configuration shown in Figure 9.5(d), which is used by many designers, is potentially dangerous because when the circuit is powered on, a race occurs and the flip-flop may go to either the 0 or 1 state.

When the preset or clear line is driven by logic, a gate can be added to achieve initialization, as shown in Figure 9.6.

To avoid the use of edge pins, preset and/or clear circuitry can be built in the circuit itself, as shown in Figure 9.7. When power is applied to the circuit, the capacitor charges from 0 to V_{cc}, and the initial low voltage on Z is used either to preset or clear flip-flops.

If the circuitry associated with the initialization of a circuit fails, test results are usually unpredictable. Such failures are often difficult to diagnose.

9.2.3 Monostable Multivibrators

Rule: Disable internal one-shots during test.

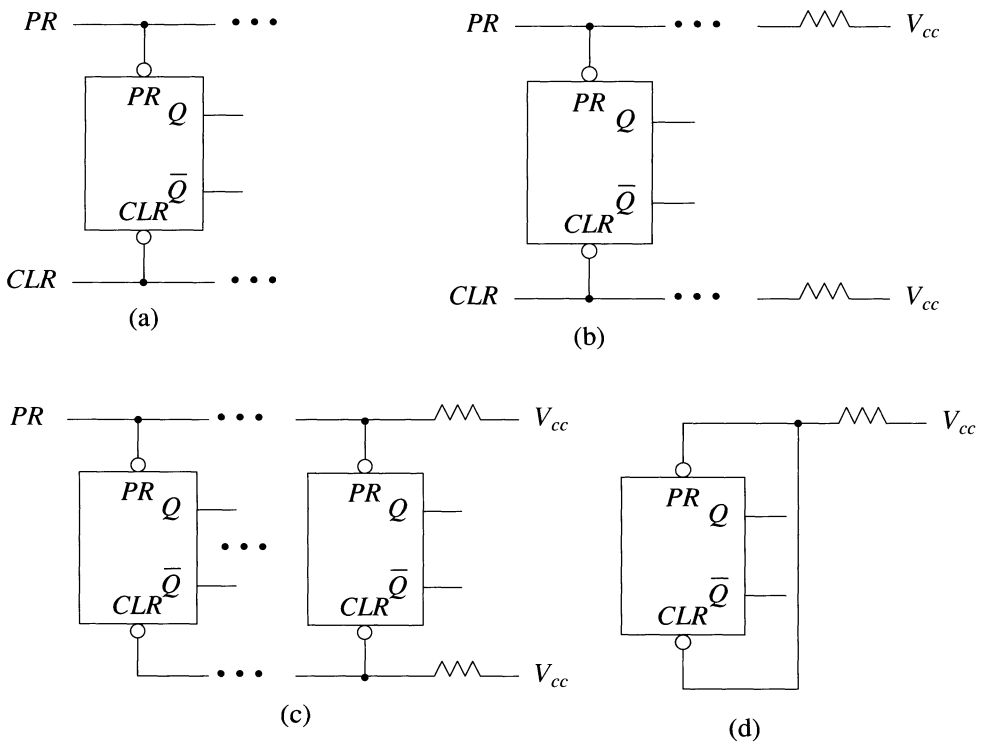


Figure 9.5 Initialization of flip-flops (a) Independent master preset/clear (b) Independent master preset/clear with pull-up resistors (c) Master preset only (d) Unacceptable configuration

Monostable multivibrators (one-shots) produce pulses internal to a circuit and make it difficult for external test equipment to remain in synchronization with the circuit being tested. There are several ways to solve this problem. On a printed circuit board, a jumper can be used to gain access to the I/O of a one-shot to disable it, control it, and observe its outputs, as well as to control the circuit normally driven by the one-shot. (See Figure 9.8(a)). Removing jumper 1 on the edge connector of the PCB allows the one-shot to be activated by the ATE via pin *A2*. The output of C_1 is also observable via pin *A1*. Pin *B2* can be driven by the ATE, and the output of the one-shot can be observed via pin *B1*.

One can also gain controllability and observability of a one-shot and its surrounding logic by adding injection circuitry, as shown in Figure 9.8(b). In this circuit, *E* is used to observe the output of the one-shot; *A* is used to deactivate the normal input to the one-shot; *B* is used to trigger the one-shot during testing; *C* is used to apply externally generated pulses to C_2 to replace those generated by the one-shot; and *D* is used to select the input appropriate to the multiplexer.

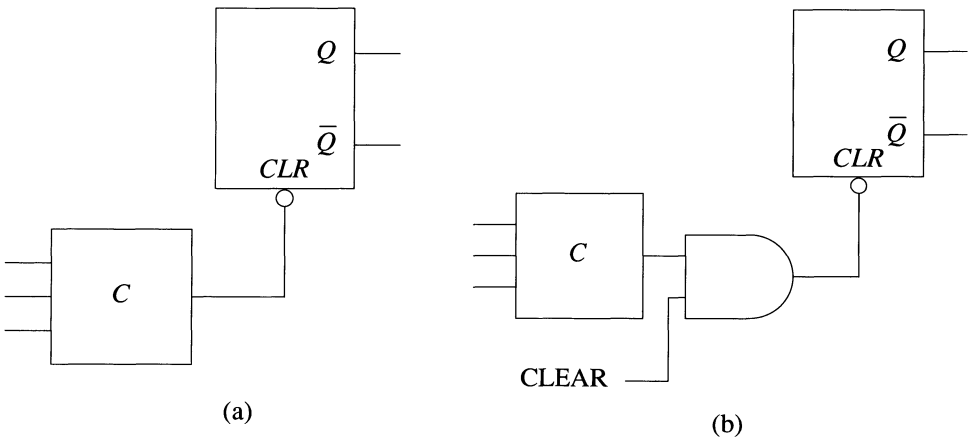


Figure 9.6 (a) Flip-flop without explicit clear (b) Flip-flop with explicit clear

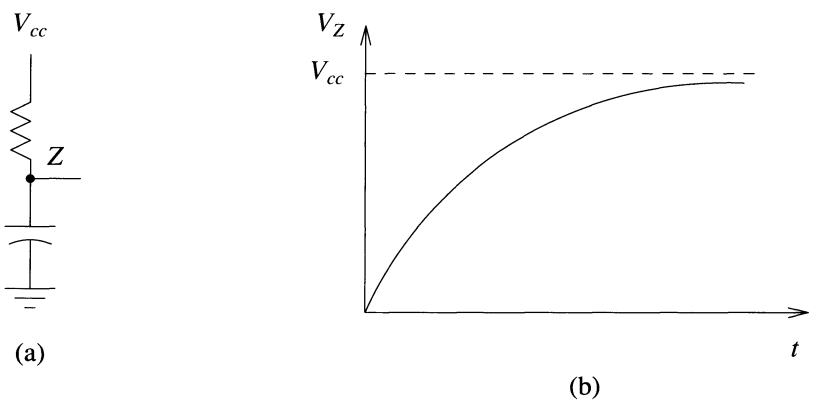
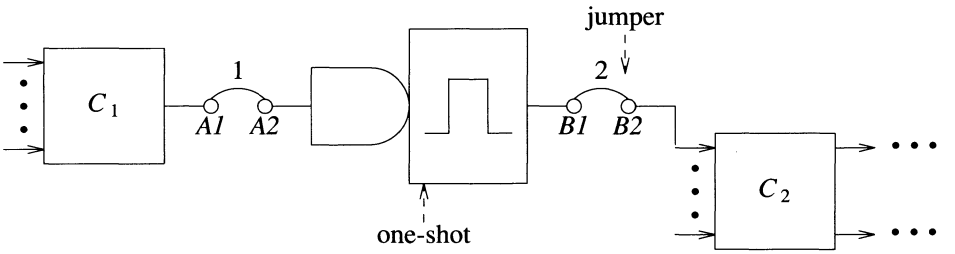


Figure 9.7 Built-in initialization signal generator

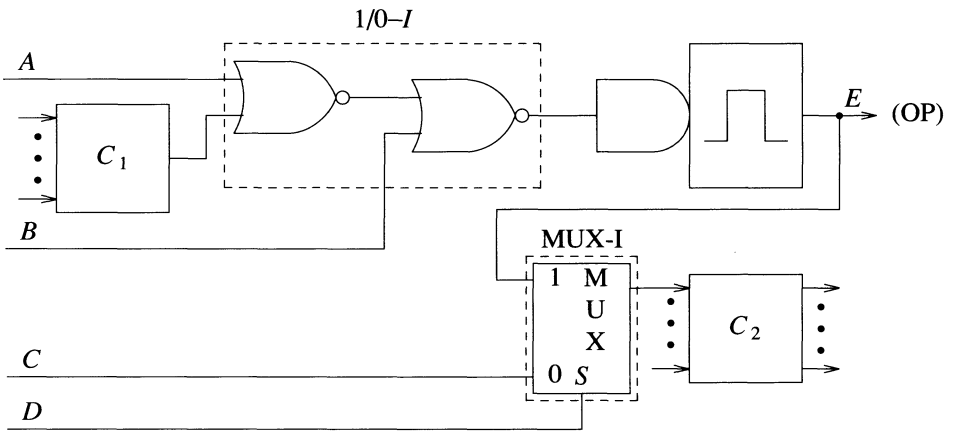
9.2.4 Oscillators and Clocks

Rule: Disable internal oscillators and clocks during test.

The use of free-running oscillators or clocks leads to synchronization problems similar to those caused by one-shots. The techniques to gain controllability and observability are also similar. Figure 9.9 shows one such DFT circuit configuration.



(a)



(b)

Figure 9.8 (a) Disabling a one-shot using jumpers (b) Logical control and disabling of a one-shot

9.2.5 Partitioning Counters and Shift Registers

Rule: Partition large counters and shift registers into smaller units.

Counters, and to a lesser extent shift registers, are difficult to test because their test sequences usually require many clock cycles. To increase their testability, such devices should be partitioned so that their serial input and clock are easily controllable, and output data are observable. Figure 9.10(a) shows a design that does not include testability features and where the register R has been decomposed into two parts, and Figure 9.10(b) shows a more testable version of this design. For example, $R1$ and $R2$ may be 16-bit registers. Here the gated clock from C can be inhibited and replaced by an external clock. The serial inputs to $R1$ and $R2$ are easily controlled and the serial output of $R2$ is easily observable. As a result, $R1$ and $R2$ can be independently tested.

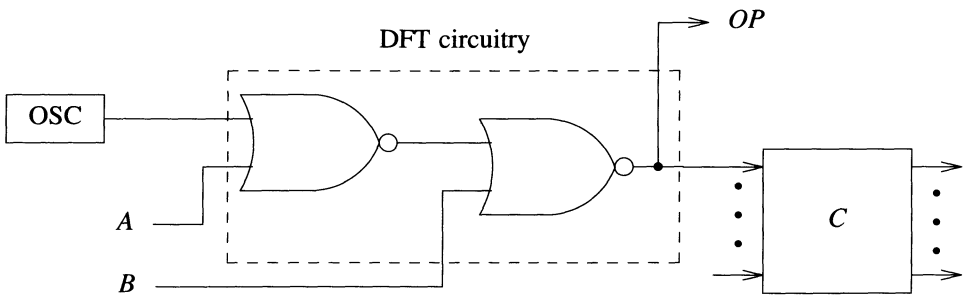


Figure 9.9 Testability logic for an oscillator

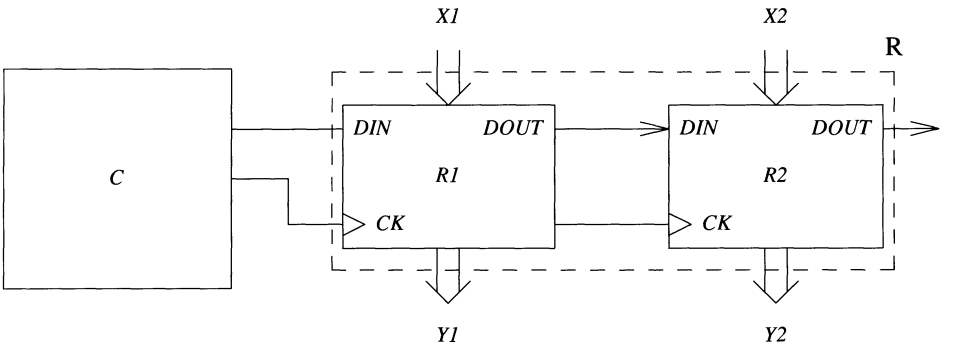
A 16-bit counter could take up to $2^{16} = 65536$ clock cycles to test. Partitioned into two 8-bit counters with enhanced controllability and observability, the new design can be tested with just $2 \times 2^8 = 512$ clock cycles. If the two partitions are tested simultaneously, even less time is required.

9.2.6 Partitioning of Large Combinational Circuits

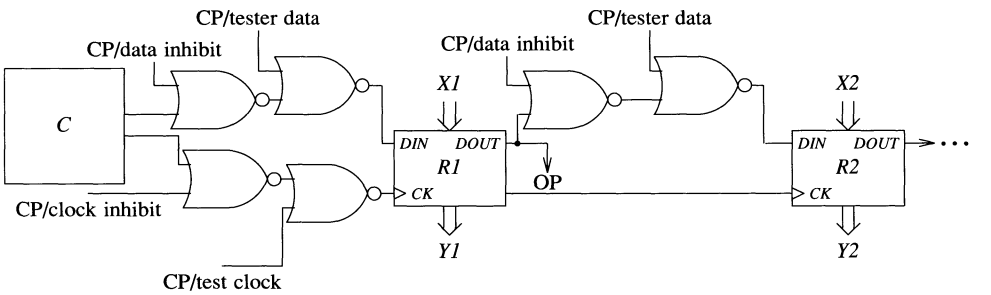
Rule: Partition large circuits into small subcircuits to reduce test generation cost.

Since the time complexity of test generation and fault simulation grows faster than a linear function of circuit size, it is cost-effective to partition large circuits to reduce these costs. One general partitioning scheme is shown in Figure 9.11. In Figure 9.11(a) we show a large block of combinational logic that has been partitioned into two blocks, C_1 and C_2; many such partitions exist for multiple-output circuits. The bit widths of some of the interconnects are shown, and we assume, without loss of generality, that $p < m$ and $q \leq n$. Figure 9.11(b) shows a modified version of this circuit, where multiplexers are inserted between the two blocks, and $A'(C')$ represents a subset of the signals in $A(C)$. For $T_1T_2 = 00$ (normal operation), the circuit operates the same way as the original circuit, except for the delay in the multiplexers. For $T_1T_2 = 01$, C_1 is driven by the primary inputs A and C'; the outputs F and D are observable at F' and G', respectively. Hence C_1 can be tested independently of C_2. Similarly C_2 can be tested independently of C_1. Part of the multiplexers are also tested when C_1 and C_2 are tested. However, not all paths through the multiplexers are tested, and some global tests are needed to ensure 100 percent fault coverage. For example, the path from D to C_2 is not tested when C_1 and C_2 are tested separately.

This partitioning scheme enhances the controllability and observability of the inputs and outputs, respectively, associated with C_1 and C_2, and hence reduces the complexity of test generation. Assume test generation requires n^2 steps for a circuit having n gates, and that C has 10000 gates and can be partitioned into two circuits C_1 and C_2 of size 5000 gates each. Then the test generation for the unpartitioned version of C requires $(10^4)^2 = 10^8$ steps, while the test generation for C_1 and C_2 requires only $2 \times 25 \times 10^6 = 5 \times 10^7$ or half the time.



(a)



(b)

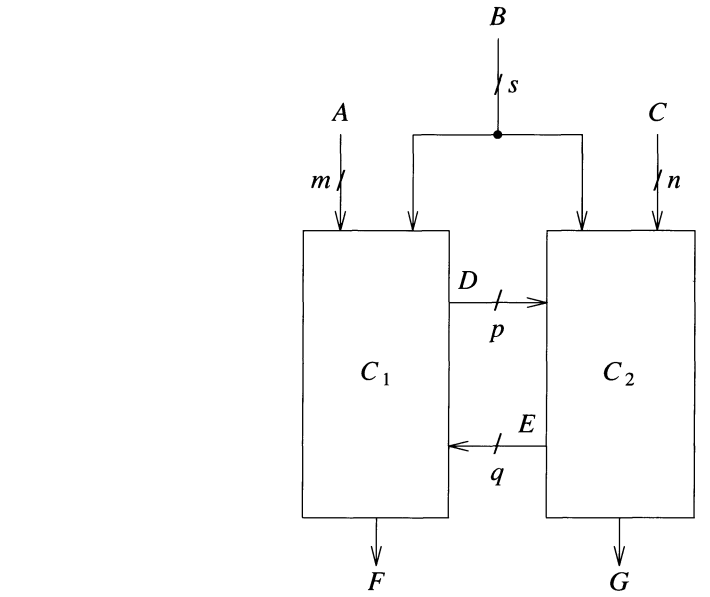
Figure 9.10 Partitioning a register

Assume it is desired to test this circuit exhaustively, i.e., by applying all possible inputs to the circuit. Let $m = n = s = 8$, and $p = q = 4$. Then to test C_1 and C_2 as one unit requires $2^{8+8+8} = 2^{24}$ test vectors. To test C_1 and C_2 individually requires $2^{8+8+4} = 2^{20}$ test vectors. The basic concept can be iterated so as to partition a circuit into more than two blocks and thus reduce test complexity by a greater factor.

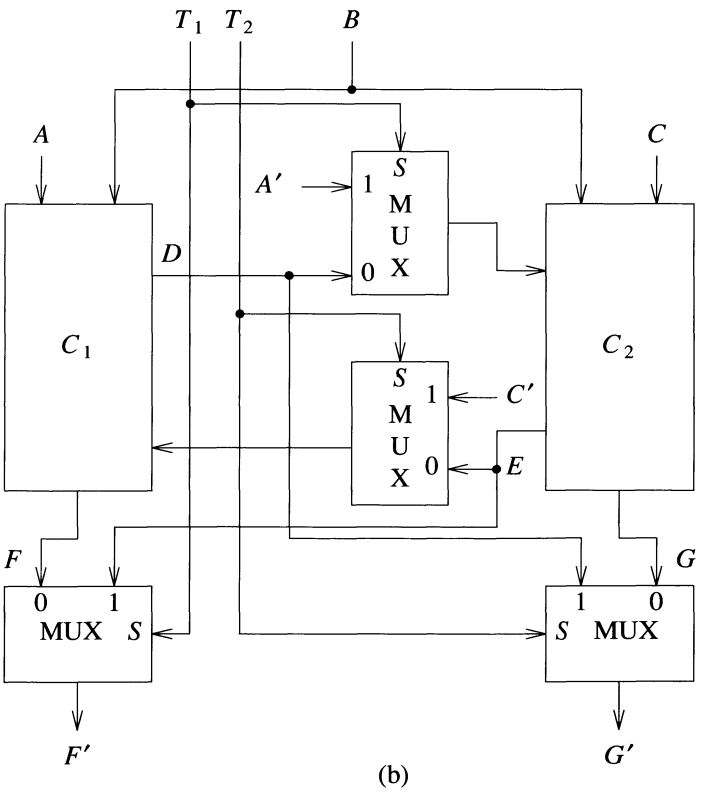
9.2.7 Logical Redundancy

Rule: Avoid the use of redundant logic.

Recall that redundant logic introduces faults which in combinational logic, and in most instances in sequential logic, are not detectable using static tests. There are several reasons such logic should be avoided whenever possible. First, if a redundant fault occurs, it may invalidate some test for nonredundant faults. Second, such faults cause difficulty in calculating fault coverage. Third, much test generation time can be spent



(a)



T_1	T_2	Mode
0	0	normal
0	1	test C_1
1	0	test C_2

(b)

Figure 9.11 Partitioning to reduce test generation cost

in trying to generate a test for a redundant fault, assuming it is not known *a priori* that it is redundant. Unfortunately, here a paradox occurs, since the process of identifying redundancy in a circuit is NP-hard. Fourth, most designers are not aware of redundancy introduced inadvertently and hence cannot easily remove it. Finally, note that redundancy is sometimes intentionally added to a circuit. For example, redundant logic is used to eliminate some types of hazards in combinational circuits. It is also used to achieve high reliability, such as in Triple Modular Redundant (TMR) systems. For some of these cases, however, test points can be added to remove the redundancy during testing without inhibiting the function for which the redundancy is provided.

9.2.8 Global Feedback Paths

Rule: Provide logic to break global feedback paths.

Consider a global feedback path in a circuit. This path may be part of either a clocked or an unclocked loop. Observability can be easily obtained by adding an observation test point to some signal along this path. Controllability can be achieved by using injection circuits such as those shown in Figure 9.1.

The simplest form of asynchronous logic is a latch, which is an essential part of most flip-flop designs. In this section, we are concerned with larger blocks of logic that have global feedback lines. Problems associated with hazards are more critical in asynchronous circuits than in synchronous circuits. Also, asynchronous circuits exhibit unique problems in terms of races. The test generation problem for such circuits is more difficult than for synchronous circuits, mainly because signals can propagate through the circuitry one or more times before the circuit stabilizes. For these reasons, asynchronous circuits other than latches should be avoided when possible. If it is not feasible to eliminate such circuits, then the global feedback lines should be made controllable and observable as described previously.

Further information on ad hoc design-for-test techniques can be found in [Davidson 1979], [Grason and Nagel 1981], [Lippman and Donn 1979], and [Writer 1975].

9.3 Controllability and Observability by Means of Scan Registers

By using test points one can easily enhance the observability and controllability of a circuit. But enhancement can be costly in terms of I/O pins. Another way to enhance observability and/or controllability is by using a *scan register* (SR). A scan register is a register with both shift and parallel-load capability. The storage cells in the register are used as observation points and/or control points. The use of scan registers to replace I/O pins deals with a trade-off between test time, area overhead, and I/O pins. Figure 9.12 shows a generic form of a scan storage cell (SSC) and corresponding scan register. Here, when $\overline{N}/T = 0$ (normal mode), data are loaded into the scan storage cell from the data input line (D); when $\overline{N}/T = 1$ (test mode), data are loaded from S_i. A scan register R shifts when $\overline{N}/T = 1$, and loads data in parallel when $\overline{N}/T = 0$. Loading data into R from line S_{in} when $\overline{N}/T = 1$ is referred to as a *scan-in* operation; reading data out of R from line S_{out} is referred to as a *scan-out* operation.

We will next look a little closer at the number of variations used to inject and/or observe test data using scan registers.

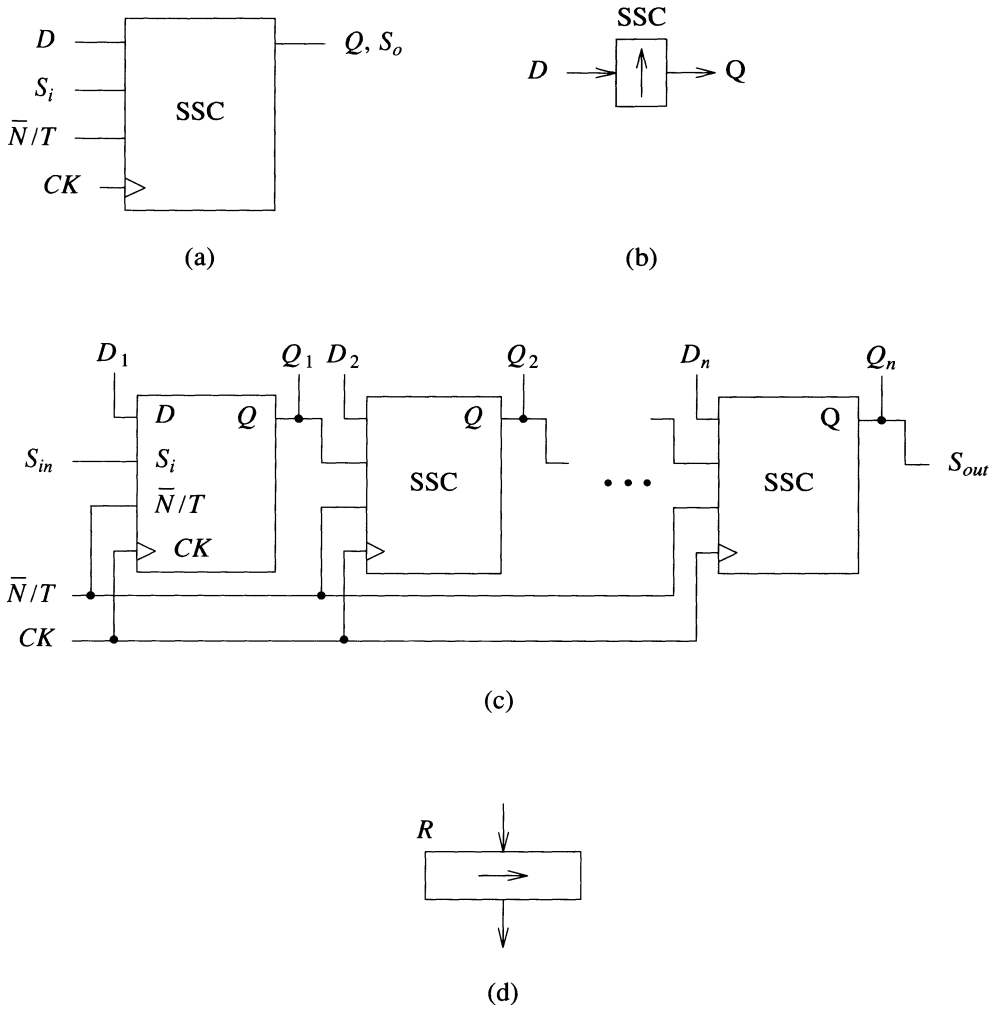


Figure 9.12 (a) A scan storage cell (SSC) (b) Symbol for a SSC (c) A scan register (SR) or shift register chain (d) Symbol for a scan register

Simultaneous Controllability and Observability

Figure 9.13(a) shows two complex circuits C_1 and C_2. They can be either combinational or sequential. Only one signal (Z) between C_1 and C_2 is shown.

Figure 9.13(b) depicts how line Z can be made both observable and controllable using a scan storage cell. Data at Z can be loaded into the SSC and observed by means of a scan-out operation. Data can be loaded into the SSC via a scan-in operation and then injected onto line Z'. Simultaneous controllability and observability can be achieved. That is, the scan register can be preloaded with data to be injected into the circuit.

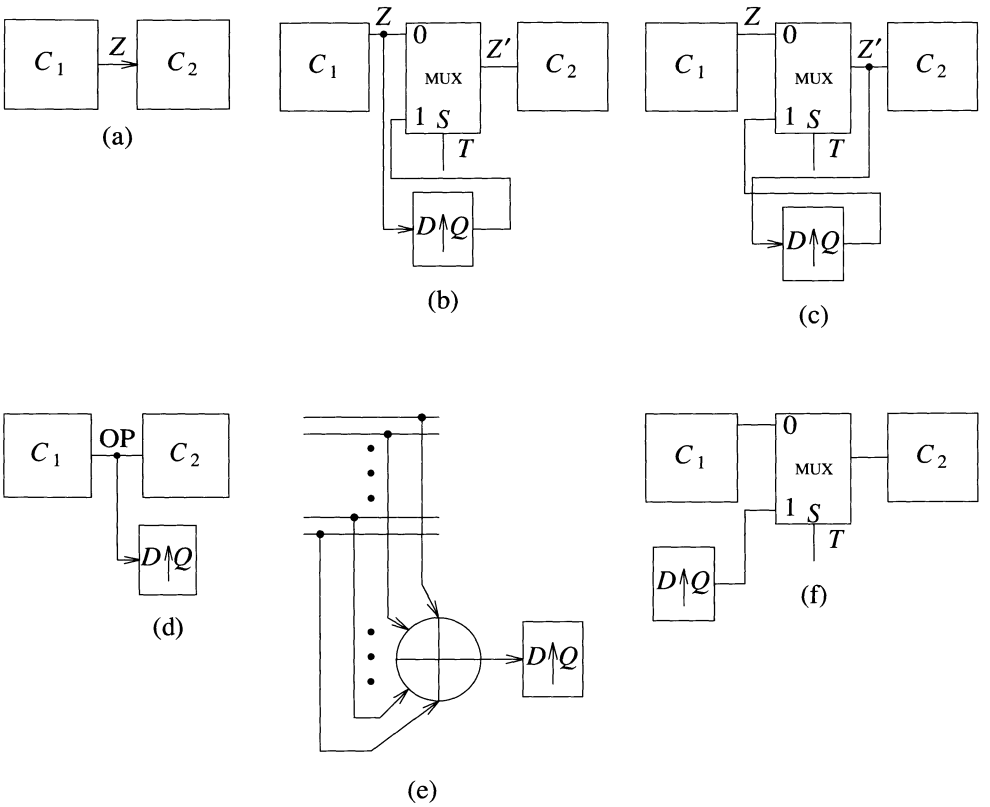


Figure 9.13 (a) Normal interconnect (b) Simultaneous C/O (c) Separate C/O (d) Observability circuit (e) Compacting data (f) Controllability circuit

The circuit can be run up to some time t. At time $t + 1$, if $T = 1$, then the data in the SSC will be injected onto Z'; if $\overline{N}/T = 0$, the data at Z will be loaded into the SSC.

Nonsimultaneous Controllability and Observability

In Figure 9.13(c) we see a variant on the design just presented. Here we can either observe the value Z' by means of the scan register or control the value of line Z'. Both cannot be done simultaneously.

Figure 9.14 shows a more complex scan storage cell for use in sequential circuits that are time sensitive, such as asynchronous circuits. The Q_2 flip-flop is used as part of the scan chain; i.e., by setting $T2 = 1$ and clocking $CK2$ the scan storage cells form a shift register. By loading the Q_1 latch and setting $T1 = 1$, signals can be injected into the circuit. Similarly, by setting $T2 = 0$ and clocking $CK2$, data can be loaded into the scan register. One scenario for using this cell is outlined below.

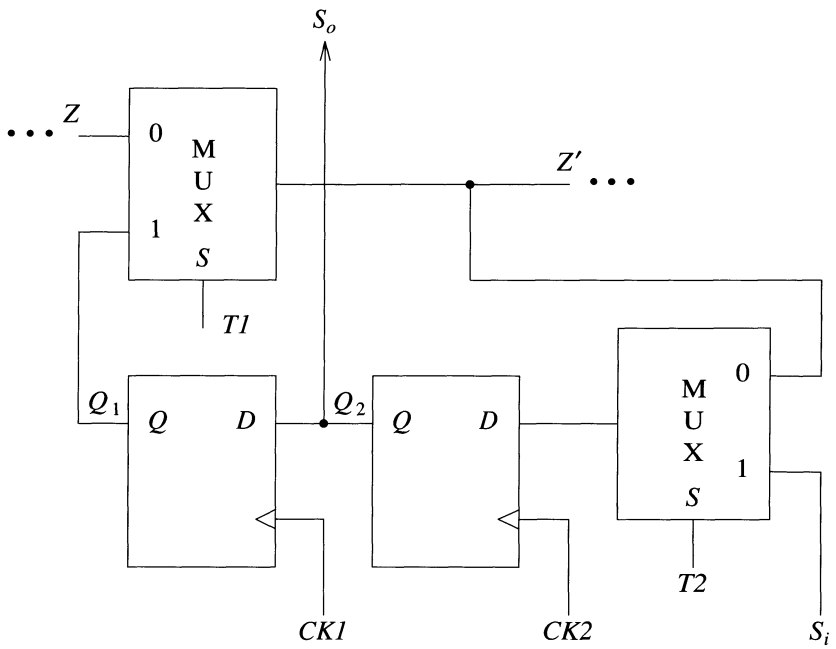


Figure 9.14 Complex scan storage cell

1. Load the scan register with test data by setting $T2 = 1$ and clocking $CK2$.
2. Drive the UUT to a predefined state with $T1 = 0$.
3. Load Q_2 into latch Q_1.
4. Optional
   a. Load $Z' = Z$ into Q_2 by setting $T2 = 0$ and clocking $CK2$ once.
   b. Scan-out the data in the Q_2 flip-flops.
5. Inject signals into the circuit by setting $T1 = 1$.
6. (Optional) Clock the UUT one or more times.
7. Observe the observation points by setting $T2 = 0$ and clock $CK2$ once.
8. Scan out these data by setting $T2 = 1$ and clocking $CK2$.

This last operation corresponds to step 1; i.e., data can be scanned-in and scanned-out of a scan register at the same time.

Observability Only

Figure 9.13(d) shows how an observation point can be tied to a storage cell in a scan register to gain observability. In Figure 9.13(e) the number of storage cells has been reduced by combining many observation points through a parity tree into a single scan

cell. This technique detects single errors but may fail to detect faults affecting two signals.

Controllability Only

If only controllability is required, then the circuit of Figure 9.13(f) can be used.

Note that in all cases shown it was assumed that controllability required the ability to inject either a 0 or a 1 into a circuit. If this is not the case, then the MUX can be replaced by an AND, NAND, OR, or NOR gate.

Applications

Figure 9.15 shows a sequential circuit S having inputs X and outputs Z. To enhance controllability, control points have been added, denoted by X'. These can be driven by a scan register R_1. To enhance observability, observation points have been added, denoted by Z'. These can be tied to the data input ports of a scan register R_2. Thus X' act as pseudo-primary inputs and Z' as pseudo-primary outputs. Using X' and Z' can significantly simplify ATG. Assume one has run a sequential ATG algorithm on S and several faults remain undetected. Let f be such a fault. Then by simulating the test sequence generated for the faults that were detected, it is usually possible to find a state s_0 such that, if signals were injected at specific lines and a specific line were made observable, the fault could be detected. These points define where CPs and OPs should be assigned. Now to detect this fault the register R_1 is loaded with the appropriate scan data. An input sequence is then applied that drives the circuit to state s_0. Then the input X' is applied to S. The response Z and Z' can then be observed and the fault detected. By repeating this process for other undetected faults, the resulting fault coverage can be significantly increased. Of course, this increase is achieved at the expense of adding more scan storage cells. Note that the original storage cells (latches and flip-flops) in the circuit need not be modified. Finally, using the appropriate design for the scan storage cells, it is possible to combine registers R_1 and R_2 into a single register R where each scan storage cell in R is used both as a CP and an OP. The scan storage cell shown in Figure 9.14 is appropriate for this type of operation.

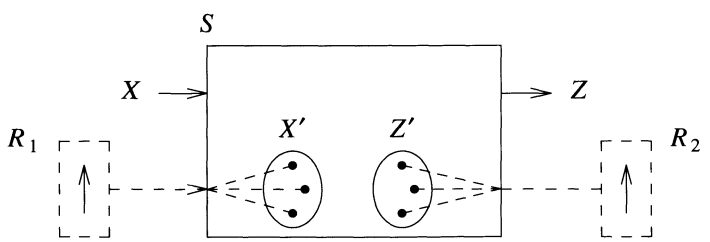


Figure 9.15 General architecture using test points tied to scan registers

Figure 9.16 illustrates a circuit that has been modified to have observation and control points to enhance its testability. The original circuit consists of those modules denoted by heavy lines. The circuits denoted by C_1, C_2, ..., are assumed to be complex logic blocks, either combinational or sequential. Most of their inputs and outputs are not shown. To inject a 0 into the circuit, an AND gate is used, e.g., G_1; to inject a 1, an OR gate is used, e.g., G_2. If such gates do not already exist in the circuit, they can be added. To inject either a 0 or a 1, a MUX is used, e.g., G_3 and G_4. Assume that the scan register R is able to hold its contents by disabling its clock. One major function of the test hardware shown is to observe data internal to the circuit via the OP lines. This process is initiated by carrying out a parallel load on R followed by a scan-out operation. The other major function of this circuit is to control the CP lines. These data are loaded into R via a scan-in operation. The data in Q_4 and Q_5 do not affect the circuit until $T = 1$. Then if $Q_1 = 1$, the data in Q_4 and Q_5 propagate through the MUXs; if Q_2 is 1, a 0 is injected at the output of G_1; if $Q_3 = 1$, a 1 is injected at the output of G_2. Since signals $\overline{N}/T$, S_{in}, S_{out}, T, and CK each require a separate pin, five additional I/O pins are required to implement this scheme. The number of CPs and OPs dictates the length of the register R. Note that the same storage cell can be used both to control a CP and accept data from an OP. The problem with this scheme is that the test data as supplied by R cannot change each clock time. Normally the circuit is logically partitioned using the CPs. N test vectors are then applied to the circuit. Data are then collected in R via the OPs and shifted out for observations. If the register is of length n, then n clock periods occur before the next test vector is applied. This scheme is particularly good for PCBs, where it is not easy to obtain CPs.

Several variations of this type of design philosophy can be incorporated into a special chip designed specifically to aid in making PCBs more testable. One such structure is shown in Figure 9.17. The normal path from A to B is broken when $B1 = 1$ and the top row of MUXs is used to inject data into a circuit from the scan register. The lower row of MUXs is used for monitoring data within the circuit. This circuit can be further enhanced by using MUXs and DEMUXs as shown earlier to concentrate several observation points into one or to control many control points from one scan storage cell.

9.3.1 Generic Boundary Scan

In designing modules such as complex chips or PCBs, it is often useful for purposes of testing and fault isolation to be able to isolate one module from the others. This can be done using the concept of *boundary scan*, which is illustrated in Figure 9.18. Here, the original circuit S has n primary inputs X and m primary outputs Y (see Figure 9.18(a)). In the modified circuit, shown in Figure 9.18(b), R_1 can be now used to observe all input data to S, and R_2 can be used to drive the output lines of S. Assume all chips on a board are designed using this boundary scan architecture. Then all board interconnects can be tested by scanning in test data into the R_2 register of each chip and latching the results into the R_1 registers. These results can then be verified by means of a scan-out operation. A chip can be tested by loading a test vector into the R_2 registers that drive the chip. Thus normal ATE static chip tests can be reapplied for in-circuit testing of chips on a board. The clocking of S must be inhibited as each test is loaded into the scan path. Naturally, many test vectors need to be processed fully to test a board. There are many other ways to implement the concept of boundary scan. For example, the storage cell shown in Figure 9.14 can be

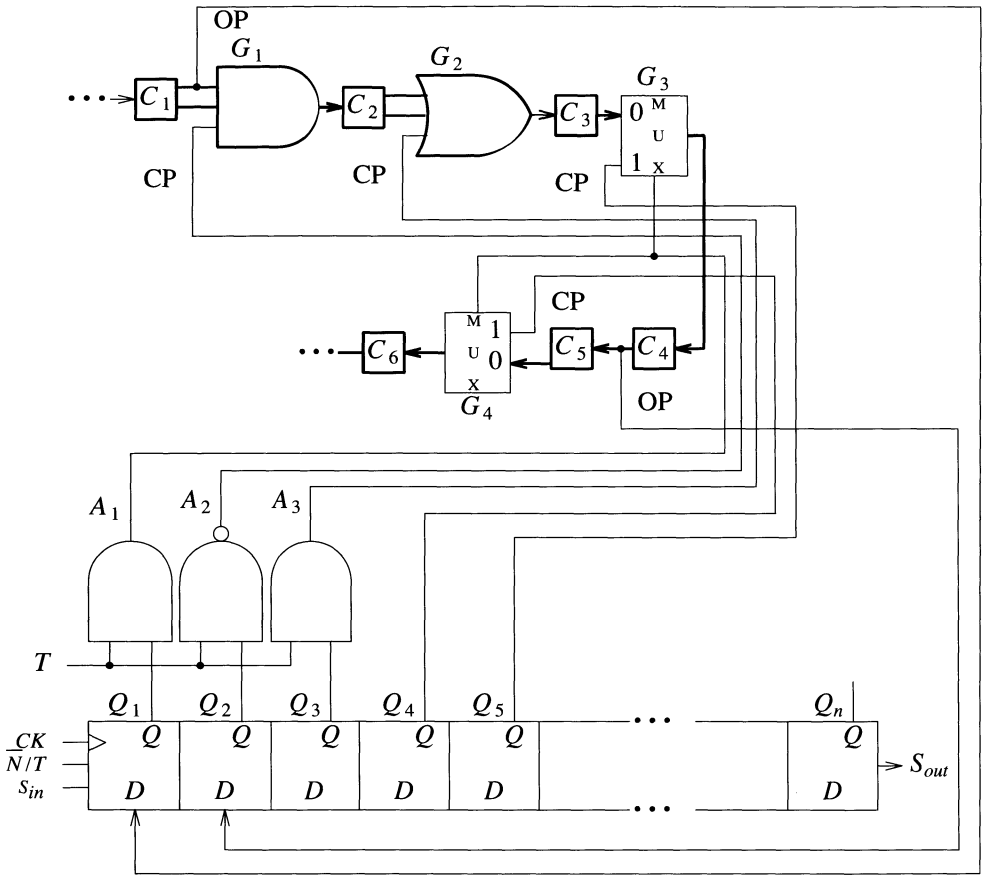


Figure 9.16 Adding controllability and observability to a circuit

inserted in every I/O line. The scan registers discussed so far fall into the category of being isolated scan or shadow registers. This is because they are not actually part of the functional circuitry itself. Scan registers that are part of the functional registers are referred to as integrated scan registers. They will be discussed next. Aspects of interconnect testing and related design-for-test techniques are discussed in [Goel and McMahon 1982].

9.4 Generic Scan-Based Designs

The most popular structured DFT technique used for external testing is referred to as *scan design* since it employs a scan register. We assume that the circuit to be made testable is synchronous. There are several forms of scan designs; they differ primarily in how the scan cells are designed. We will illustrate three generic forms of scan design and later go into the details for how the registers can be designed.

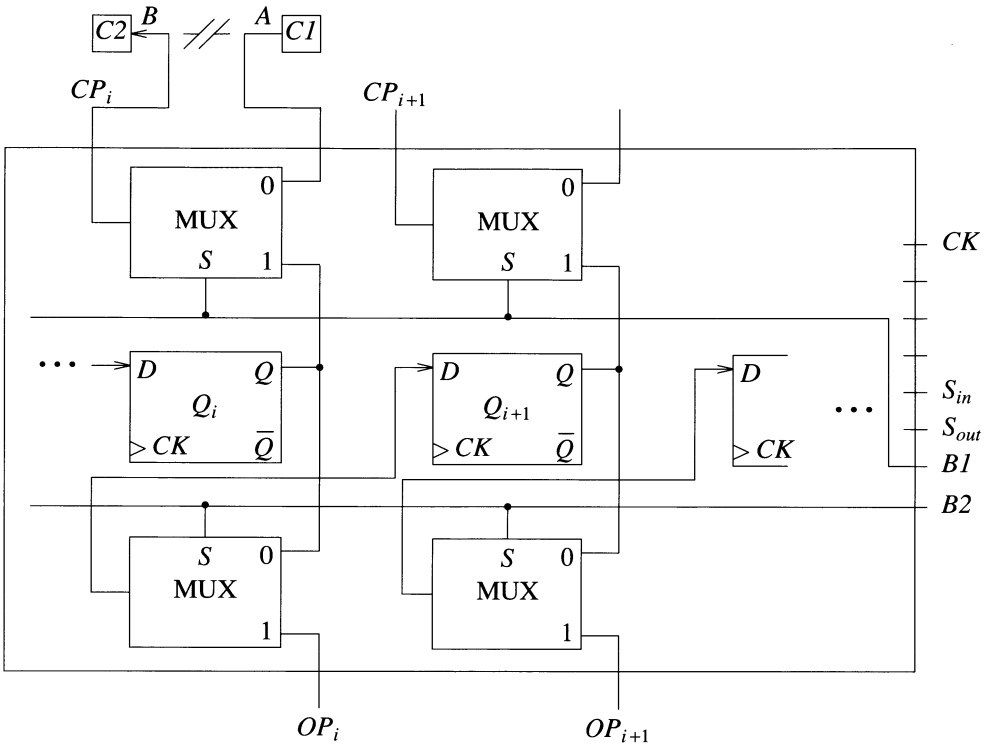


Figure 9.17 Controllability/observability circuitry with a scan chain

Usually these designs are considered to be *integrated* scan-based designs because all functional storage cells are made part of the scan registers. Thus the selection of what lines to be made observable or controllable becomes a moot point. So these techniques are referred to as *structured* rather than ad hoc.

9.4.1 Full Serial Integrated Scan

In Figure 9.19(a) we illustrate the classical Huffman model of a sequential circuit, and in Figure 9.19(b) the full scan version of the circuit. Structurally the change is simple. The normal parallel-load register R has been replaced by a scan register R_s. When $\overline{N}/T = 0$ (normal mode), R_s operates in the parallel-latch mode; hence both circuits operate the same way, except that the scan version may have more delay. Now Y becomes easily controllable and E easily observable. Hence test generation cost can be drastically reduced. Rather than considering the circuit S of Figure 9.19(a) as a sequential circuit, test generation can proceed directly on the combinational circuit C using any of a variety of algorithms, such as PODEM or FAN. The result is a series of test vectors (x_1,y_1), (x_2,y_2), ... and responses (z_1,e_1), (z_2,e_2), ... To test S^*, set $\overline{N}/T = 1$ and scan y_1 into R_s. During the k-th clock time, apply x_1 to X. Now the first test pattern, $t_1 = (x_1,y_1)$, is applied to C. During the $(k+1)st$ clock time, set

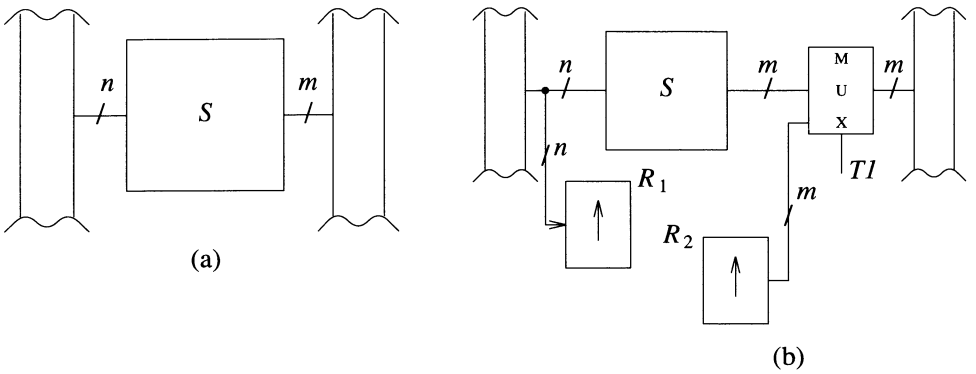


Figure 9.18 Boundary scan architecture (a) Original circuit (b) Modified circuit

$\overline{N}/T = 0$ and load the state of E, which should be e_1, into R_s while observing the response on Z, which should be z_1. This process is then repeated; i.e., while y_2 is scanned into R_s, e_1 is scanned out and hence becomes observable. Thus the response $r_1 = (z_1,e_1)$ can be easily observed. The shift register is tested by both the normal test data for C and by a shift register test sequence, such as 01100xx ..., that tests the setting, resetting, and holding of the state of a storage cell. It is thus seen that the complex problem of testing a sequential circuit has been converted to a much simpler one of testing a combinational circuit.

This concept is referred to as *full serial integrated scan* since all the original storage cells in the circuit are made part of the scan register, and the scan register is used as a serial shift register to achieve its scan function. Normally the storage cells in scan designs do not have reset lines for global initialization; instead they are initialized by means of shifting data into the SCAN register.

9.4.2 Isolated Serial Scan

Isolated serial scan designs differ from full serial integrated scan designs in that the scan register is not in the normal data path. A common way of representing this architecture is shown in Figure 9.20; it corresponds closely to that shown in Figure 9.15.

This scan architecture is somewhat ad hoc since the selection of the CPs and OPs associated with the scan register R_s is left up to the designer. Hence S may remain sequential, in which case test generation may still be difficult. If R_s is used both to observe and control all the storage cells in S, then the test generation problem again is reduced to one of generating tests for combinational logic only. This design is shown in Figure 9.21 and is referred to as *full isolated scan*. Here, S' consists of the circuit C and register R', and R has been modified to have two data input ports. The testing of this circuit is now similar to that for full serial scan designs. A test vector y_1 is scanned (shifted) into R_s, loaded into R', and then applied to the circuit C. The

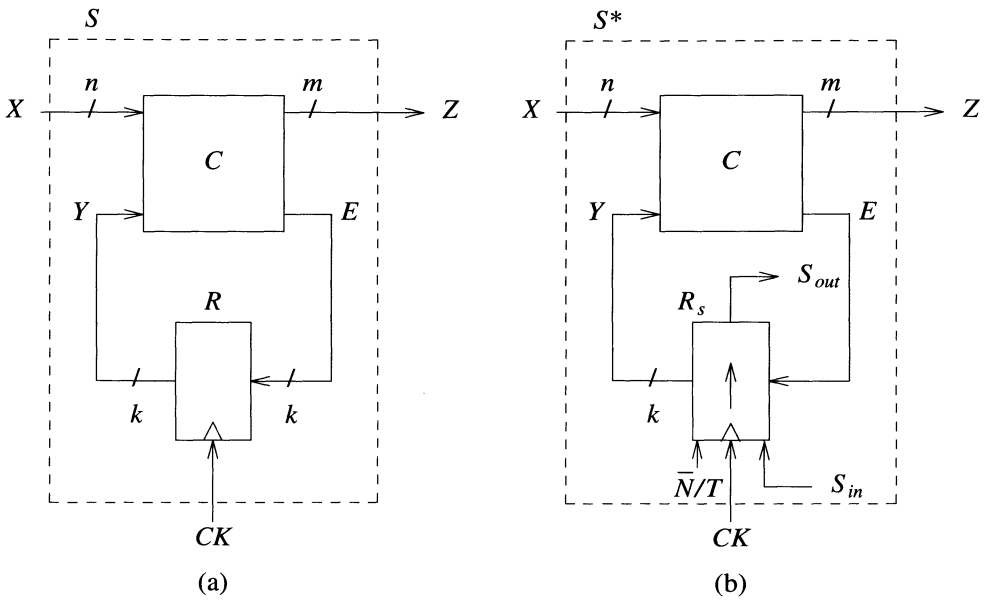


Figure 9.19 (a) Normal sequential circuit S (b) Full serial integrated scan version for circuit

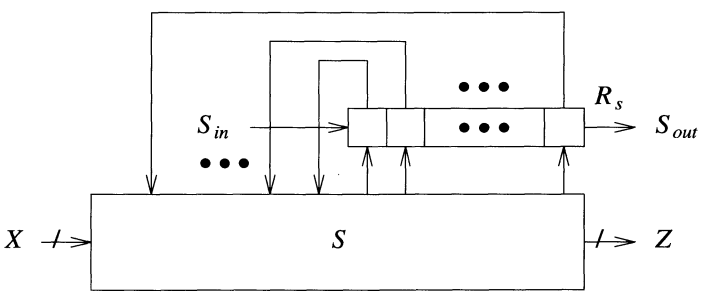


Figure 9.20 Isolated serial scan (scan/set)

response e_1 can be loaded into R', transferred to R_s, and then scanned out. The register R_s is said to act as a *shadow register* to R'. The overhead for this architecture is high compared to that for full serial integrated scan designs. Isolated scan designs have several useful features. One is that they support some forms of real-time and on-line testing. Real-time testing means that a single test can be applied at the operational clock rate of the system. In normal full serial scan, a test vector can only be applied at intervals

of k clock periods. On-line infers that the circuit can be tested while in normal operation; i.e., a snapshot of the state of the circuit can be taken and loaded into R_S. This data can be scanned out while S continues normal operation. Finally this architecture supports latch-based designs; i.e., register R and hence R' can consist of just latches rather than flip-flops. It is not feasible to string these latches together to form a shift register; hence adding extra storage cells to form a scan register is required.

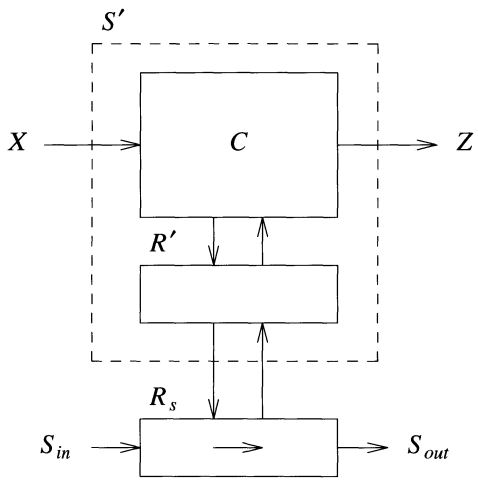


Figure 9.21 Full isolated scan

9.4.3 Nonserial Scan

Nonserial scan designs are similar to full serial integrated scan designs in that they aim to give full controllability and observability to all storage cells in a circuit. The technique differs from the previous techniques in that a shift register is not used. Instead the storage cells are arranged as in a random-access bit-addressable memory. (See Figure 9.22.) During normal operation the storage cells operate in their parallel-load mode. To scan in a bit, the appropriate cell is addressed, the data are applied to S_{in}, and a pulse on the scan clock *SCK* is issued. The outputs of the cells are wired-ORed together. To scan out the contents of a cell, the cell is addressed, a control signal is broadcast to all cells, and the state of the addressed cell appears at S_{out}. The major advantage of this design is that to scan in a new test vector, only bits in R that need be changed must be addressed and modified; also selected bits can be observed. This saves scanning data through the entire register. Unfortunately the overhead is high for this form of scan design. There is also considerable overhead associated with storing the addresses of the cells to be set and/or read.

9.5 Storage Cells for Scan Designs

Many storage cell designs have been proposed for use in scan cell designs. These designs have several common characteristics. Because a scan has both a normal data input and a scan data input, the appropriate input can be selected using a multiplexer controlled by a

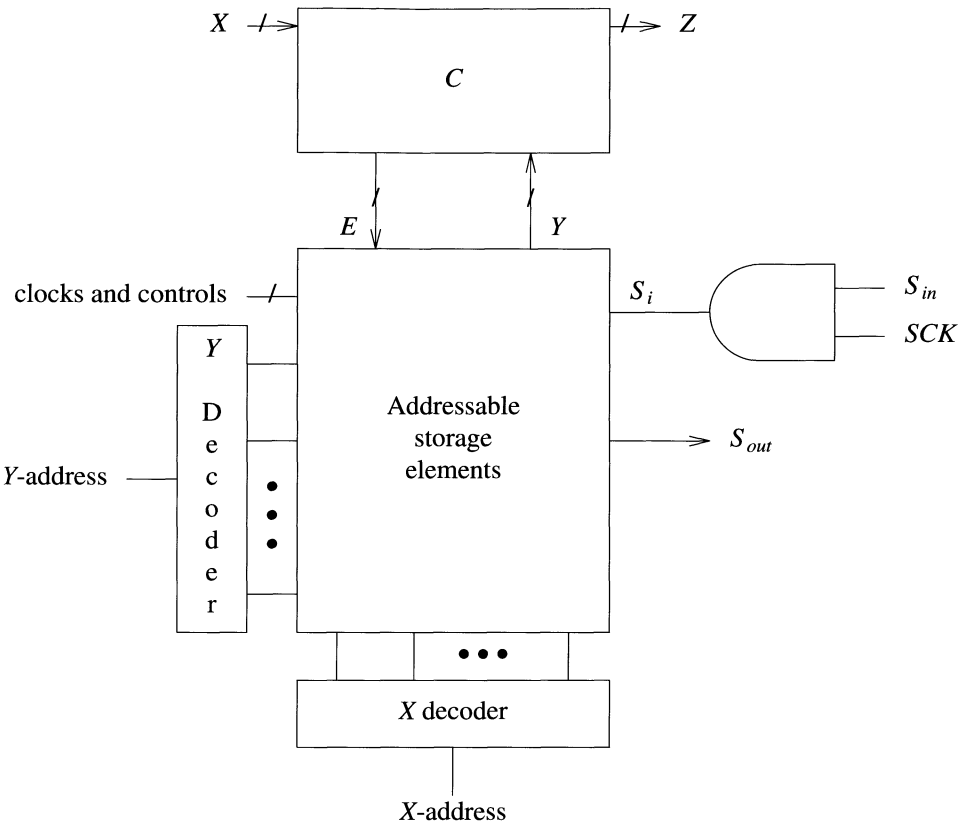


Figure 9.22 Random-access scan

normal/test ($\overline{N}/T$) input or by a two-clock system. Also, the cell can be implemented using a clocked edge-triggered flip-flop, a master-slave flip-flop, or level-sensitive latches controlled by clocks having two or more phases. Designs having up to four clocks have been proposed. In our examples master-slave rather than edge-triggered flip-flops will be used. Also only D flip-flops and latches will be discussed.

Several sets of notation will be used. Multiple data inputs will be denoted by *D1*, *D2*, ..., *Dn*, and multiple clocks by *CK1*, *CK2*, ..., *CKm*. In addition, if a latch or flip-flop has a single-system clock, the clock will be denoted by *CK*, a single-scan clock by *SK*, a scan data input by S_i, and a scan data output by S_o. Also, sometimes the notation used in some of the published literature describing a storage cell will be indicated.

Figure 9.23(a) shows a NAND gate realization of a clocked D-latch, denoted by *L*. This is the basic logic unit in many scan storage cell designs.

Figure 9.23(b) shows a two-port clocked master-slave flip-flop having a multiplexer on its input and denoted by (*MD-F/F*). Normal data (*D*) enter at port *1D* when $\overline{N}/T = 0$. The device is in the scan mode when $\overline{N}/T = 1$, at which time scan data (S_i) enter at port *2D*.

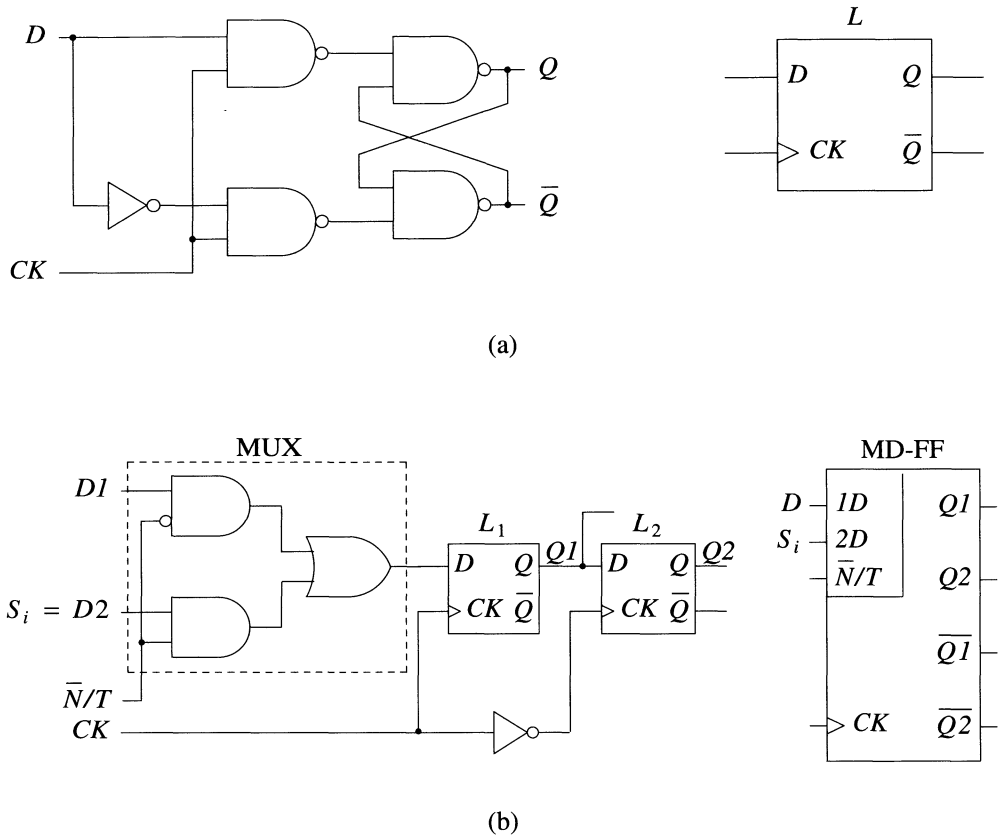


Figure 9.23 Some storage cell designs (a) Clocked D-latch and its symbol (b) Multiplexed data flip-flop and its symbol (MD-FF)

Sometimes it is useful to separate the normal clock from the clock used for scan purposes. Figure 9.24(a) shows a two-port clocked flip-flop (2P-FF), which employs two clocks and a semimultiplexed input unit.

It is often desirable to insure race-free operation by employing a two-phase nonoverlapping clock. Figure 9.24(b) shows a two-port shift register latch consisting of latches L_1 and L_2 along with a multiplexed input (MD-SRL). This cell is not considered a flip-flop, since each latch has its own clock. Also, since this type of design is used primarily in scan paths, the term "shift" is used in its name.

To avoid the performance degradation (delay) introduced by the MUX in an MD-SRL, a two-port shift register latch (2P-SRL) can be used, as shown in Figure 9.25. This circuit is the NAND gate equivalent to the shift register latch used in a level-sensitive scan design (LSSD) methodology employed by IBM [Eichelberger and Williams 1977]. Note that three clocks are used. The inputs have the following functions (the notation in parenthesis is used by IBM):

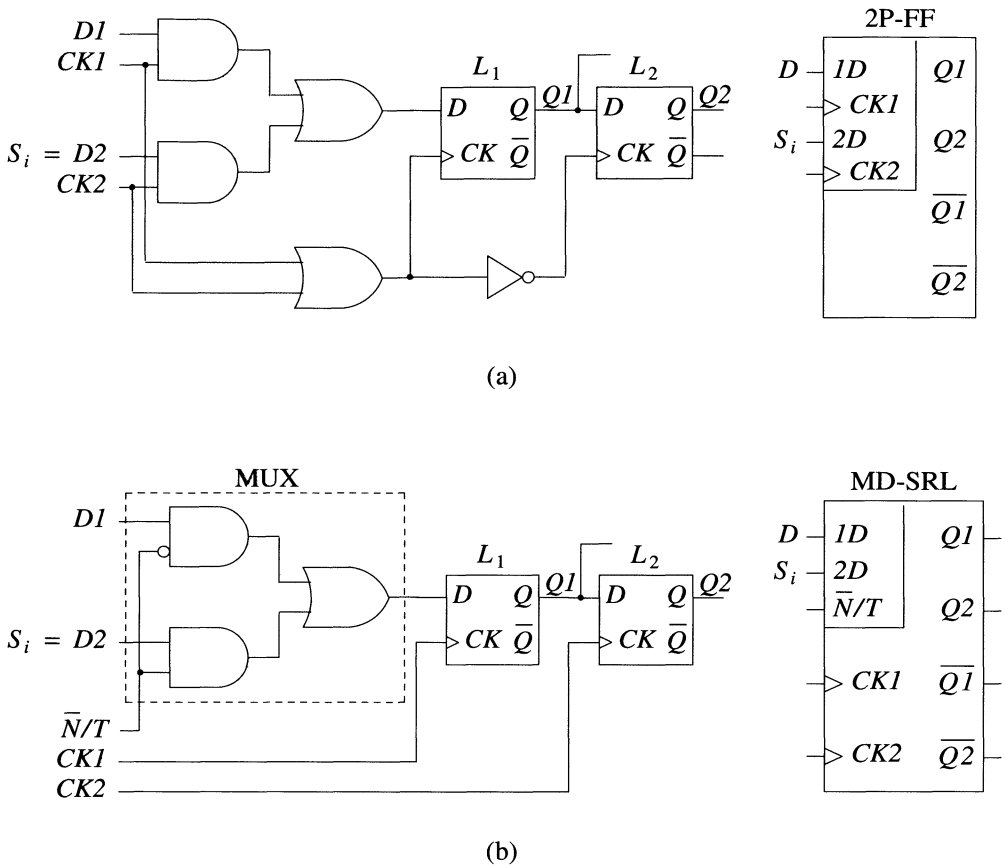


Figure 9.24 (a) Two-port dual-clock flip-flop and its symbol (2P-FF)
(b) Multiplex data shift register latch and its symbol (MD-SRL)

D1(*D*) is the normal data input.
D2 or S_i(*I*) is the scan data input.
CK1(*C*) is the normal system clock.
CK2(*A*) is the scan data input clock.
CK3(*B*) is the L_2 latch clock.

If *CK1* and *CK2* are NORed together and used to clock L_2, then *CK3* can be deleted. The result would be a two-port flip-flop.

Figure 9.26 shows a raceless master-slave *D* flip-flop. (The concept of races in latches and ways to eliminate them will be covered when Figure 9.28 is discussed.) Two clocks are used, one (*CK*) to select and control normal operation, the other (*SK*) to select scan data and control the scan process. In the normal mode, $SK = 1$ blocks scan data on S_i from entering the master latch; i.e., $G_1 = 1$. Also $G_7 = 1$. $CK = 0$ enables the value of the data input *D* to be latched into the master latch. When *CK* goes to 1, the state of the

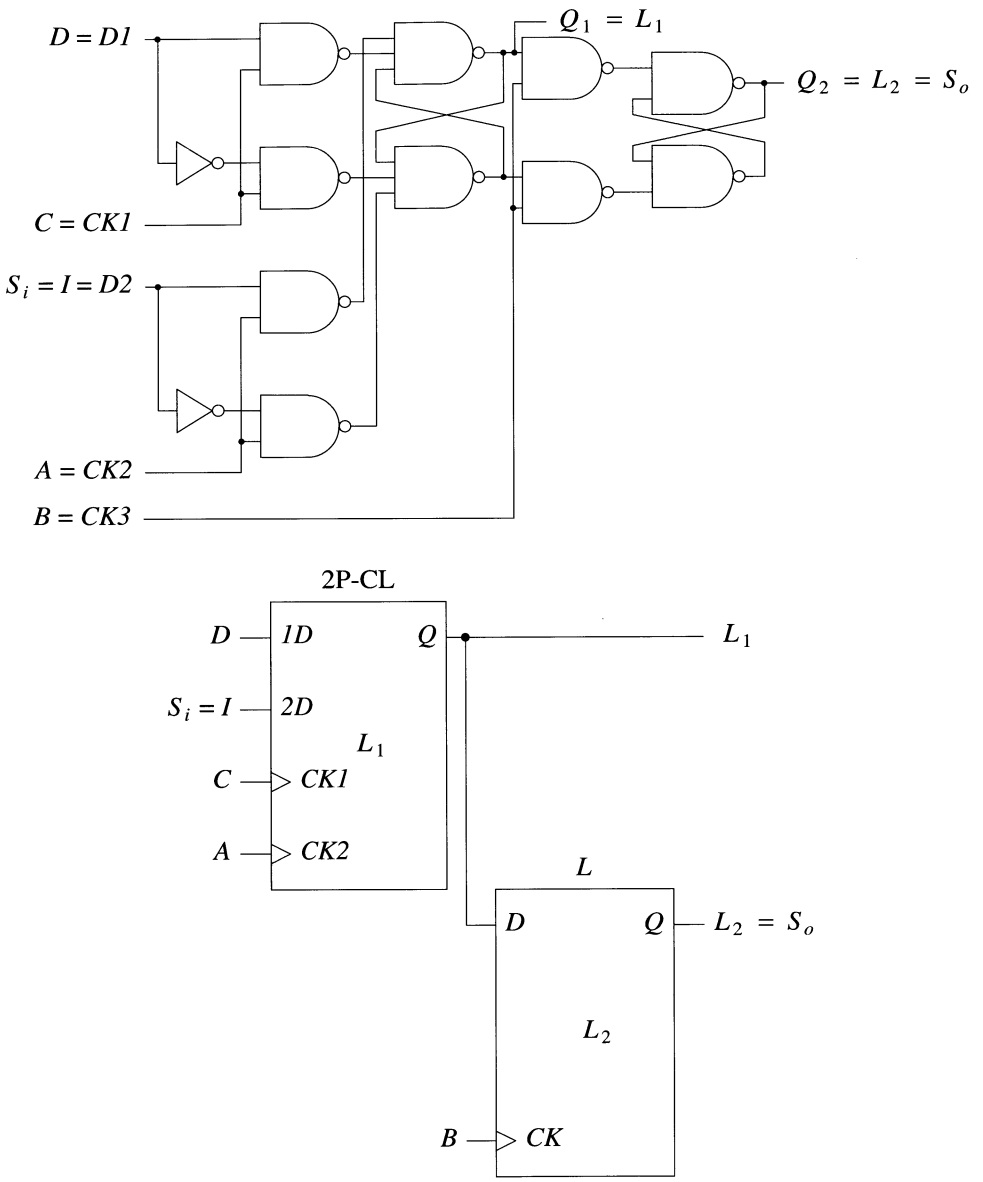


Figure 9.25 Two-port shift register latch and its symbol (2P-SRL)

master is transferred to the slave. Similarly, when $CK = 1$ and a pulse appears on SK, scan data enter the master and are transferred to the slave.

The random-access scan design employs an addressable polarity-hold latch. Several latch designs have been proposed for this application, one of which is shown in Figure 9.27.

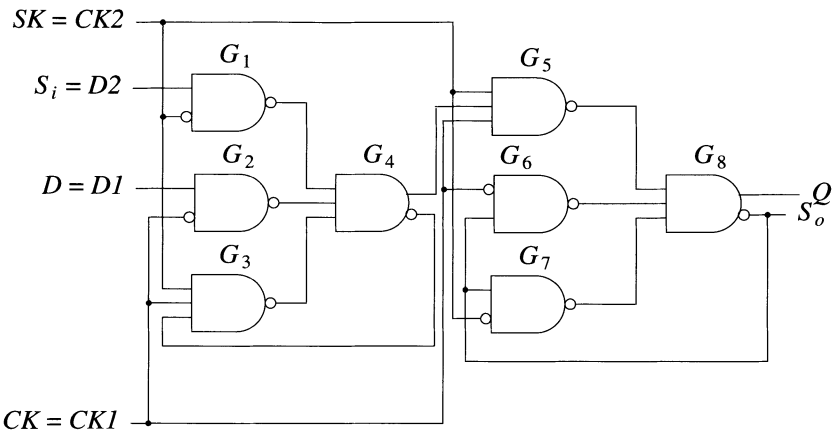


Figure 9.26 Raceless two-port D flip-flop

Since no shift operation occurs, a single latch per cell is sufficient. Note the use of a wired-AND gate. Latches that are not addressed produce a scan-out value of $S_o = 1$. Thus the S_o output of all latches can be wired-ANDed together to form the scan-out signal S_{out}.

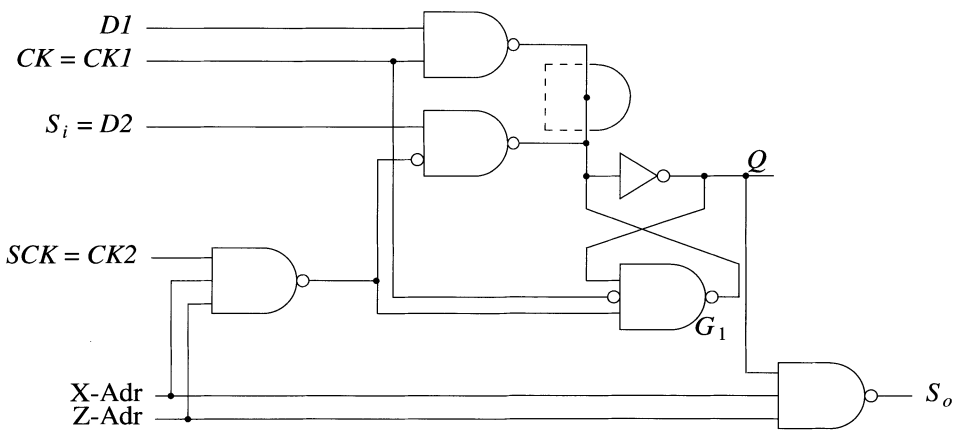


Figure 9.27 Polarity-hold addressable latch

The design of the storage cell is a critical aspect of a scan-based design. It is clearly important to achieve reliable operation. Hence a scan cell must be designed so that races and hazards either do not affect it or do not occur. In practice, storage cells are designed

at the transistor level, and the gate-level circuits shown represent approximations to the actual operation of a cell. We will next consider one cell in somewhat more detail. Figure 9.28(a) shows a simple latch design. Figure 9.28(b) shows a logically equivalent design where G_1 has been replaced by a wired-AND and an inverter. The resulting latch requires only two NAND gates and two inverters, rather than the latch shown in Figure 9.23(a), which has four NAND gates and one inverter. Q is set to the value of D when $CK = 0$. Consider the case when $CK = 0$ and $D = 1$ (see Figure 9.28(b)). Let CK now go from 0 to 1. Because of the reconvergent fanout between CK and G_1', a race exists. If the effect of the clock transition at G_5 occurs before that at G_3, then $G_5 = 0$ and G_6 remains stable at 1. If, however, the effect at G_3 occurs first, then $G_3 = 1$, G_6 goes to 0, and this 0 keeps G_5 at a 1. One can attempt to rectify this situation by changing the threshold level of G_2 so that G_5 changes before G_2. This solution works as long as the fabrication process is reliable enough accurately to control the transition times of the gates. A somewhat simpler solution is to add a gate G_4 to eliminate the inherent hazard in this logic (see Figure 9.28(c)). Now, $G_4 = 0$ keeps G_6 at 1 as the clock drops from 1 to 0. The problem with this design is that G_4 is redundant; i.e., there is no static test for G_4 *s-a*-1.

Many other storage cells have been devised to be used in scan chains. Some will be discussed later in this chapter. In the next section we will discuss some specific scan approaches proposed by various researchers.

9.6 Classical Scan Designs

Scan Path

One of the first full serial integrated scan designs was called Scan Path [Kobayashi *et al.* 1968, Funatsu *et al.* 1975]. This design employs the generic scan architecture shown in Figure 9.19(b) and uses a raceless master-slave D flip-flop, such as the one shown in Figure 9.26.

Shift Register Modification

The scan architecture shown in Figure 9.19(b) using a MD-FF (see Figure 9.23(b)) was proposed by Williams and Angell [1973] and is referred to as Shift Register Modification.

Scan/Set

The concept of Scan/Set was proposed by Stewart [1977, 1978] and uses the generic isolated scan architectures shown in Figures 9.20 and 9.21. Stewart was not specific in terms of the types of latches and/or flip-flops to use.

Random-Access Scan

The Random-Access Scan concept was introduced by Ando [1980] and uses the generic nonserial scan architecture shown in Figure 9.22 and an addressable storage cell such as the one shown in Figure 9.27.

Level-Sensitive Scan Design (LSSD)

IBM has developed several full serial integrated scan architectures, referred to as Level-Sensitive Scan Design (LSSD), which have been used in many IBM systems [Eichelberger and Williams 1977, 1978, DasGupta *et al.* 1981]. Figure 9.25 shows one

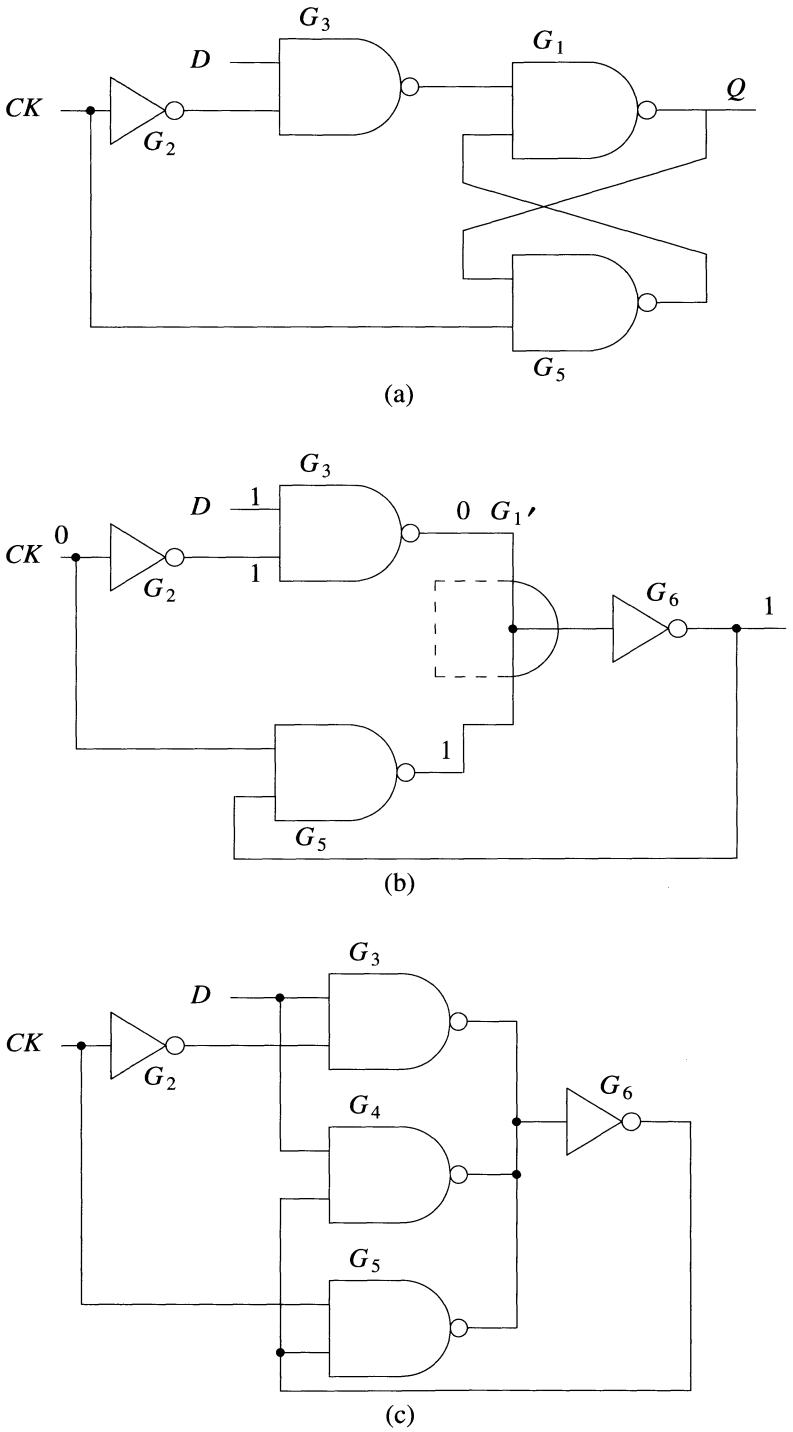


Figure 9.28 Analysis of a latch

design that uses a polarity-hold, hazard-free, and level-sensitive latch. When a clock is enabled, the state of a latch is sensitive to the level of the corresponding data input. To obtain race-free operation, clocks C and B as well as A and B are nonoverlapping.

Figure 9.29 shows the general structure for an LSSD *double-latch design*. The scan path is shown by the heavy dashed line. Note that the feedback Y comes from the output of the L_2 latches. In normal mode the C and B clocks are used; in test mode the A and B clocks are used.

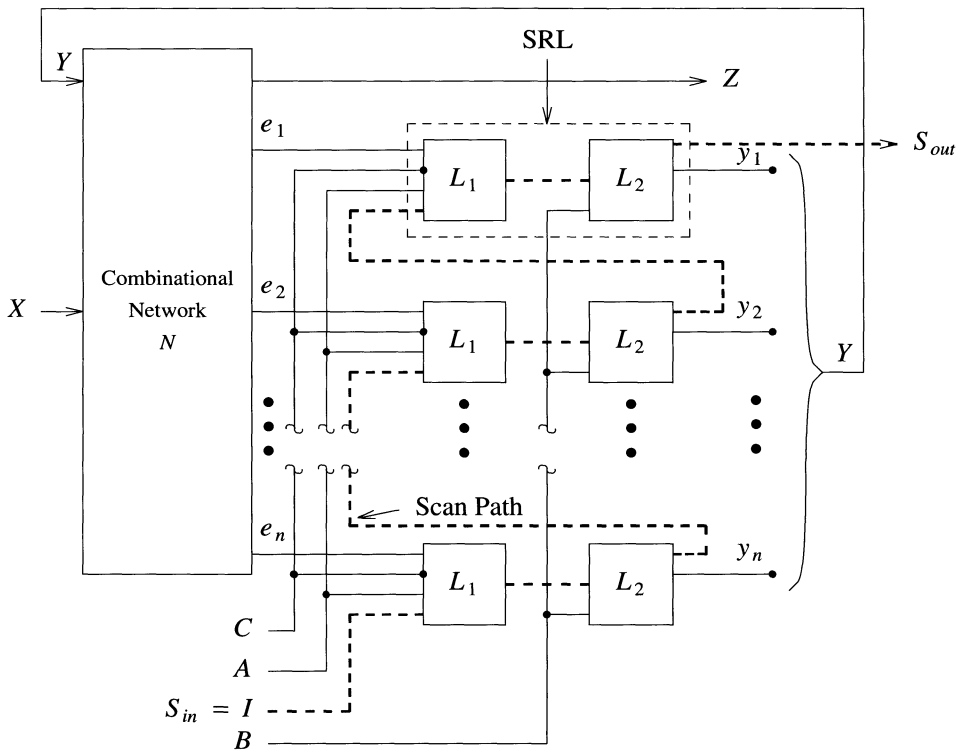


Figure 9.29 LSSD double-latch design

Sometimes it is desired to have combinational logic blocks separated by only a single latch rather than two in sequence. For such cases a *single-latch design* can be used. Figure 9.30 shows such a design. Here, during normal operation two-system clocks C_1 and C_2 are used in a nonoverlapping mode. The A and B clocks are not used; hence the L_2 latches are not used. The output Y_1 is an input to logic block N_2, whose outputs Y_2 are latched by clock C_2 producing the output at Y_2. The L_2 latches are used only when in the scan test mode; then the SRLs are clocked by A and B.

The LSSD latch design shown in Figure 9.25 has two problems. One is logic complexity. Also, when it is used in a single-latch design (see Figure 9.30), only L_1 is used during

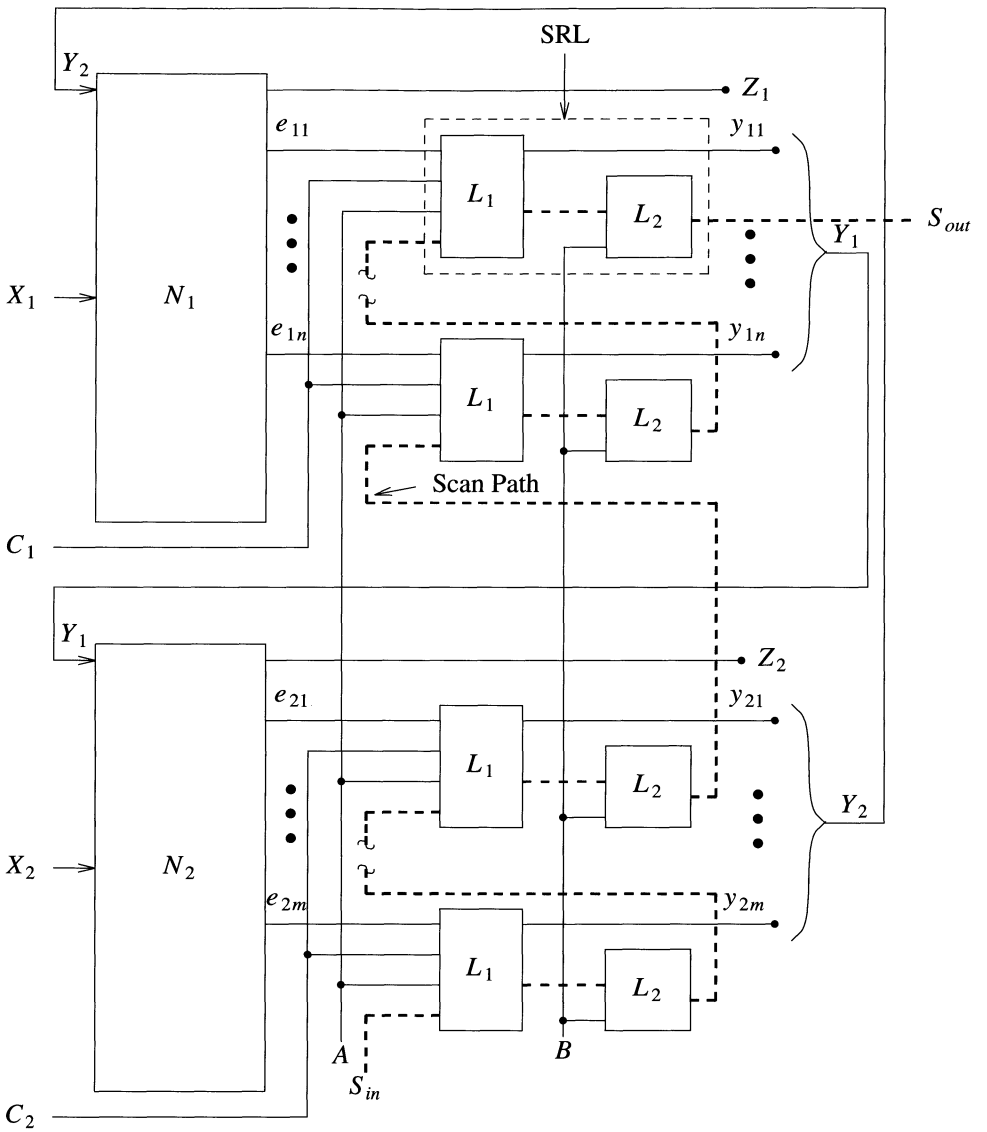


Figure 9.30 LSSD single-latch design using conventional SRLs

normal operation; L_2 is used only for shifting test data. A variation on this latch, known as the L_2^* latch, which reduces gate overhead, was reported in [DasGupta *et al.* 1982] and is shown in Figure 9.31. The primary difference between the *2P-SRL* of Figure 9.25 and L_2^* of Figure 9.31 is that L_2^* employs an additional clocked data port D^* and an additional clock C^*.

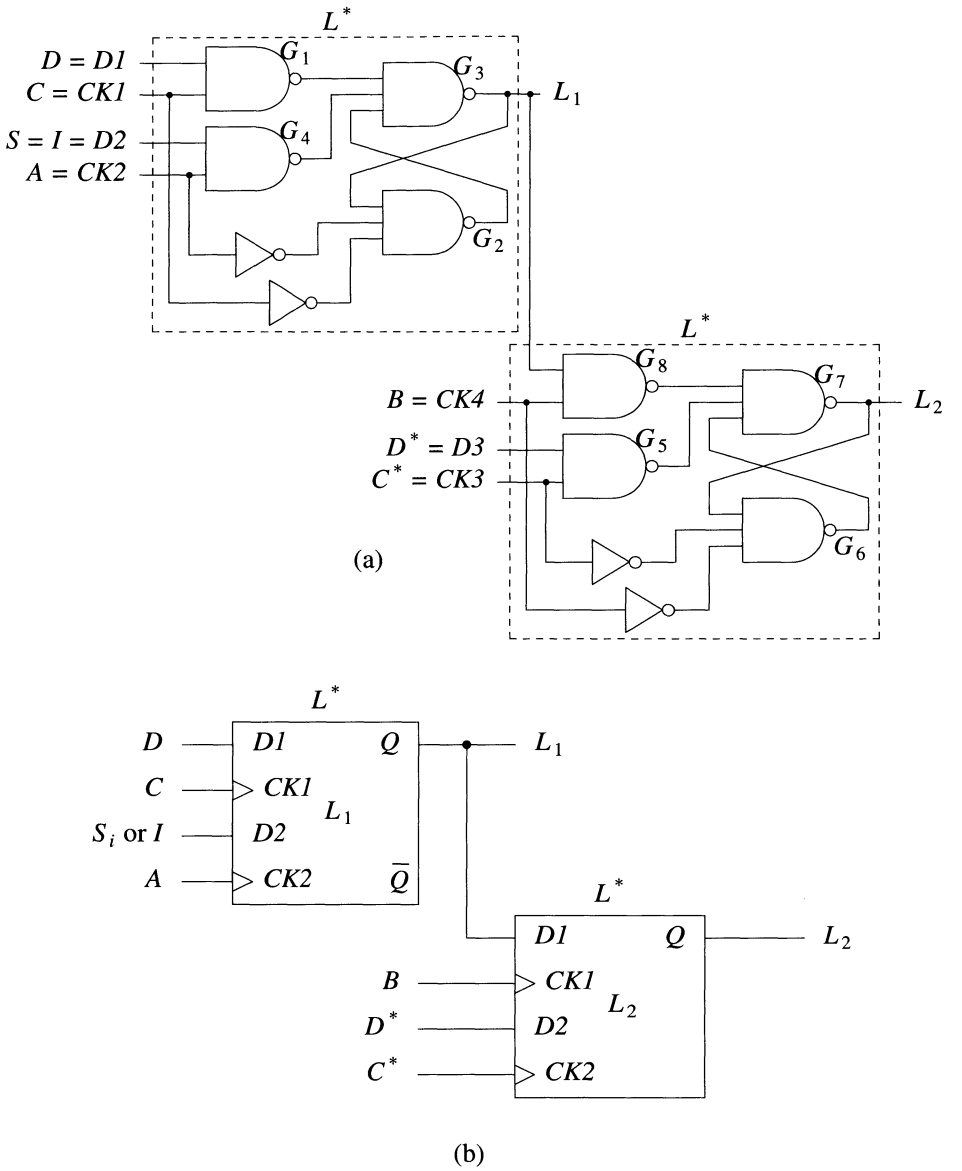


Figure 9.31 SRL using L_2^* latch with two data ports (a) Gate model (b) Symbol

Using this latch in a single-latch SRL design, the outputs of N_1 are the D inputs to an SRL latch, while the outputs of N_2 are the D^* inputs (see Figure 9.32).

It is important to note that for the design shown in Figure 9.32, it is not possible to test N_1 and N_2 at the same time. That is, since an L_1 and L_2 pair of latches make up one

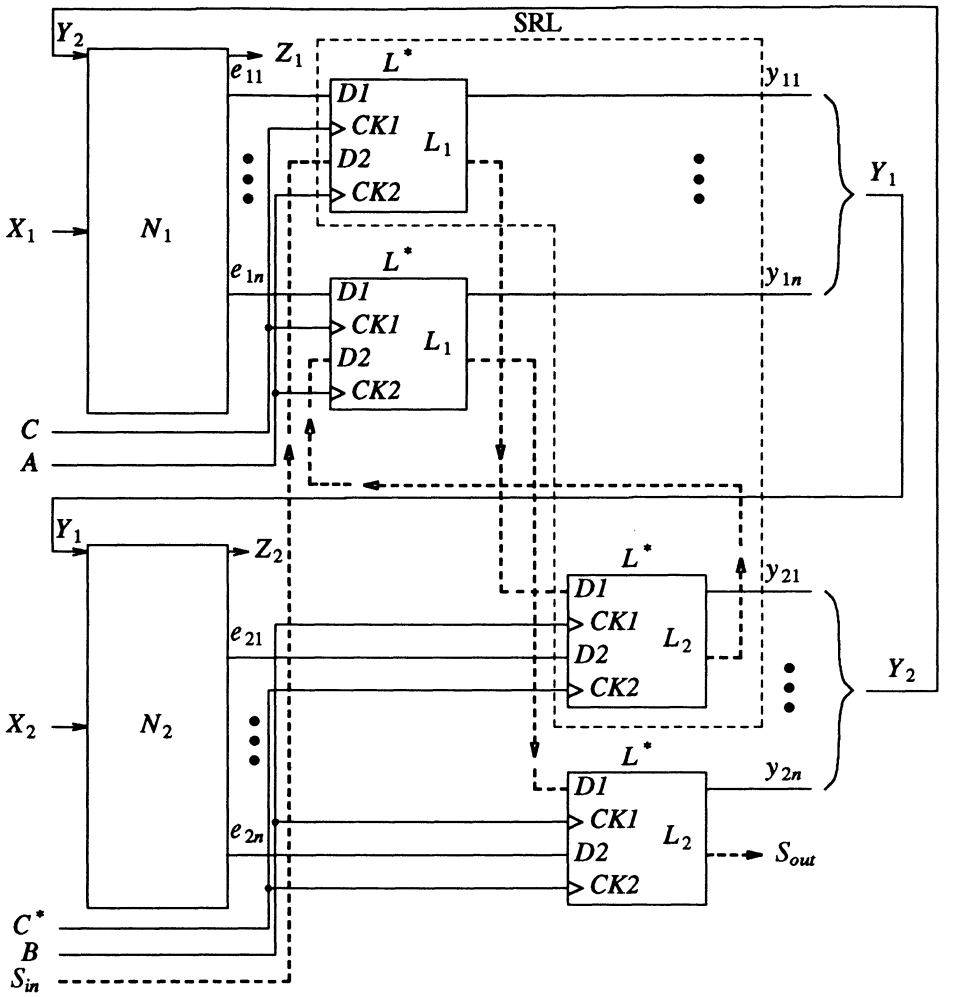


Figure 9.32 Single-latch scan design using SRLs with the L_2^* latch

SRL, only a test vector corresponding to Y_1 or Y_2 can be shifted through the scan chain, but not both Y_1 and Y_2.

LSSD Gate Overhead

Gate overhead is one important issue related to scan design. We will next compute this overhead for the three LSSD schemes presented. Let K be the ratio of combinational logic gates to latches in a nonscan design, such as that of Figure 9.30 without the L_2 latches. Assume a race-free single-latch architecture; i.e., phase 1 clocked latches feed a logic network N_2, which feeds phase 2 clocked latches, which feed a logic network N_1, which feeds the phase 1 clocked latches, again as in Figure 9.30.

Referring to Figure 9.31, a latch requires gates G_1, G_2, and G_3. Gates G_4, G_6, G_7, and G_8 are required to form an SRL. Thus for every latch in the original design, four extra gates are required to produce an SRL, resulting in the gate overhead of $(4/(K+3)) \times 100$ percent. For the L_2^* design, the L_1 and L_2 latches are both used during system operation; hence, the only extra gates are G_4 and G_5, leading to an overhead of $(1/(K+3)) \times 100$ percent; i.e., there is only one extra gate per latch. If the original design uses a double-latch storage cell with three gates per latch, then the overhead required to produce a double-latch scan design is $\left\lceil \frac{1/2}{K+3} \right\rceil \times 100$ percent. Figure 9.33 shows a plot of overhead as a function of K for these three scan designs.

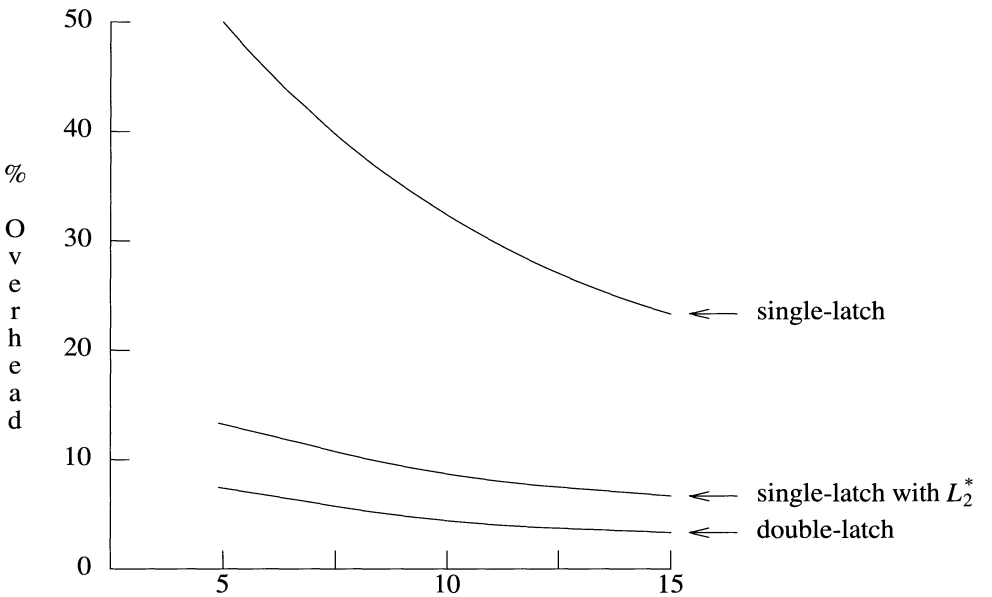


Figure 9.33 Gate overhead for single-latch design with and without L_2^* latches and double-latch design

LSSD Design Rules

In addition to producing a scan design, the design rules that define an LSSD network are intended to insure race-free and hazard-free operation. This is accomplished in part by the use of a level-sensitive rather than an edge-sensitive storage cell, as well as by the use of multiple clocks.

A network is said to be *level-sensitive* if and only if the steady state response to any of the allowed input changes is independent of the transistor and wire delays in that network.

Hence races and hazards should not affect the steady-state response of such a network. A level-sensitive design can be achieved by controlling when clocks change with respect to when input data lines change.

The rules for level-sensitive scan design are summarized as follows:

1. All internal storage elements must consist of polarity-hold latches.
2. Latches can be controlled by two or more nonoverlapping clocks that satisfy the following conditions.
   a. A latch X may feed the data port of another latch Y if and only if the clock that sets the data into latch Y does not clock latch X.
   b. A latch X may gate a clock C_i to produce a gated clock C_{ig}, which drives another latch Y if and only if clock C_{ig}, or any clock C_{ig} produced from C_{ig}, does not clock latch X.
3. There must exist a set of clock primary inputs from which the clock inputs to all SRLs are controlled either through (1) single-clock distribution trees or (2) logic that is gated by SRLs and/or nonclock primary inputs. In addition, the following conditions must hold:
   a. All clock inputs to SRLs must be at their "off" states when all clock primary inputs are held to their "off" states.
   b. A clock signal at any clock input of an SRL must be controlled from one or more clock primary inputs. That is, the clock signal must be enabled by one or more clock primary inputs as well as setting the required gating conditions from SRLs and/or nonclocked primary inputs.
   c. No clock can be ANDed with either the true or the complement of another clock.
4. Clock primary inputs cannot feed the data inputs to latches, either directly or through combinational logic. They may only feed clock inputs to latches or primary outputs.

   A network that satisfies these four rules is *level-sensitive*. The primary rule that provides for race-free operation is rule 2a, which does not allow one latch clocked by a given clock to feed another latch driven by the same clock. Rule 3 allows a test generation system to turn off system clocks and use the shift clocks to force data into and out of the scan latches. Rule 4 is also used to avoid races.

   The next two rules are used to support scan.

5. Every system latch must be part of an SRL. Also, each SRL must be part of some scan chain that has an input, output, and shift clocks available as primary inputs and/or outputs.
6. A scan state exists under the following conditions:
   a. Each SRL or scan-out primary output is a function of only the preceding SRL or scan-in primary input in its scan chain during the scan operation
   b. All clocks except the shift clocks are disabled at the SRL inputs
   c. Any shift clock to an SRL can be turned on or off by changing the corresponding clock primary input.

9.7 Scan Design Costs

Several attributes associated with the use of scan designs are listed below.

1. Flip-flops and latches are more complex. Hence scan designs are expensive in terms of board or silicon area.
2. One or more additional I/O are required. Note that some pins can be multiplexed with functional pins. In LSSD, four additional pins are used (S_{in}, S_{out}, A, and B).
3. With a given set of test patterns, test time per pattern is increased because of the need to shift the pattern serially into the scan path. The total test time for a circuit also usually increases, since the test vector set for a scan-path design is often not significantly smaller than for a nonscan design.
4. A slower clock rate may be required because of the extra delay in the scan-path flip-flops or latches, resulting in a degradation in performance. This performance penalty can be minimized by employing storage cells that have no additional delay introduced in series with the data inputs, such as the one shown in Figure 9.26.
5. Test generation costs can be significantly reduced. This can also lead to higher fault coverage.
6. Some designs are not easily realizable as scan designs.

9.8 Board-Level and System-Level DFT Approaches

By a *system* we mean a collection of modules, such as PCBs, which consist of collections of ICs. Many of the ad hoc DFT techniques referred to earlier apply to the board level. Structural techniques, such as scan, can also be applied at the board and system levels, assuming chips are designed to support these techniques. The primary system-level DFT approaches use existing functional busses, scan paths, and boundary scan.

9.8.1 System-Level Busses

This DFT approach makes use of a module's or system's functional bus to control and observe signals during functional level testing. A test and/or maintenance processor, such as the ATE, appears as another element attached to the system's busses. Figure 9.34 shows a simple bus-oriented, microprocessor-based system. During testing, the ATE can take control of the system busses and test the system. This is often done by emulating the system's processing engine. The ATE can also emulate the various units attached to the bus, monitor bus activity, and test the processing engine. In general, complex, manually generated functional tests are used.

9.8.2 System-Level Scan Paths

Figure 9.35 shows how the concept of scan can be extended to the board and system levels. Here the scan path of each chip on a board is interconnected in a daisy chain fashion to create one long scan path on each board. The boards all share a common S_{in}, $\overline{N/T}$, and CK input. Their S_{out} lines are wired-ORed together. The testing of such a system is under the control of a system-maintenance processor, which selects that board to be attached to the board-level scan line S_{out}. The interconnect that is external to the boards can be considered to be a test bus. It is seen that starting with a structured DFT approach at the lowest level of the design, i.e., at the chip level, leads to a hierarchical

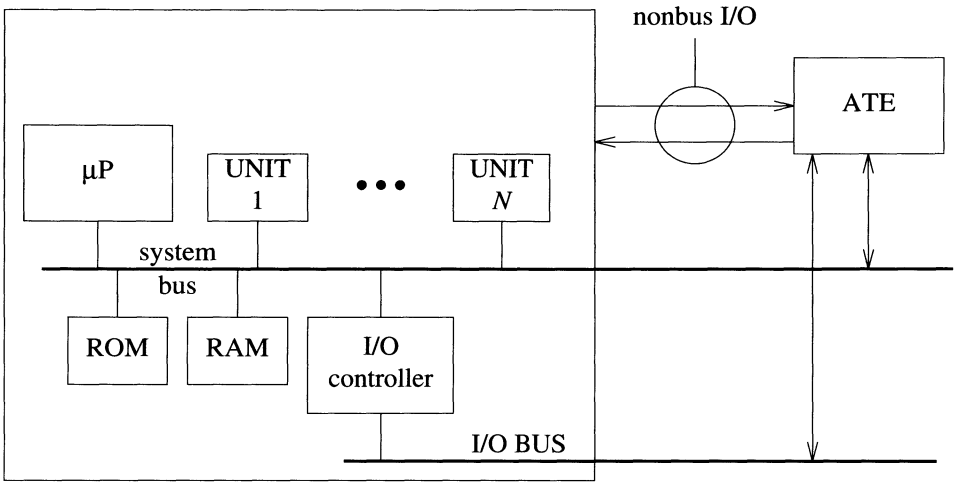


Figure 9.34 System-level test using system bus

DFT methodology. Assuming chips have boundary scan, tests developed for chips at the lowest level can be used for testing these same components at the higher levels. These tests must be extended to include tests for the interconnect that occurs at these higher levels.

9.9 Some Advanced Scan Concepts

In this section several advanced concepts related to scan-based designs are presented. The topics include the use of multiple test sessions and partial scan. *Partial scan* refers to a scan design in which a subset of the storage cells is included in the scan path. These techniques address some of the problems associated with full-scan designs discussed in Section 9.7.

9.9.1 Multiple Test Session

A *test session* consists of configuring the scan paths and other logic for testing blocks of logic, and then testing the logic using the scan-test methodology. Associated with a test session are several parameters, including the number of test patterns to be processed and the number of shifts associated with each test pattern. Often test application time can be reduced by using more than one test session. Consider the circuit shown in Figure 9.36, where C_1 has 8 inputs and 4 outputs and can be tested by 100 test patterns, and C_2 has 4 inputs and 8 outputs and can be tested by 20 test patterns. Thus the entire circuit can be tested by 100 test patterns, each of length 12, and the total test time is approximately 100×12 = 1200 clock cycles, where we have ignored the time required to load results into R_2 and R_4 as well as scan out the final result. Note that when a new test pattern is scanned into R_1 and R_3, a test result is scanned out of R_2 and R_4. We have just described one way of testing C_1 and C_2, referred to as the *together mode*. There are two other ways for testing C_1 and C_2, referred to as *separate mode* and *overlapped mode*.

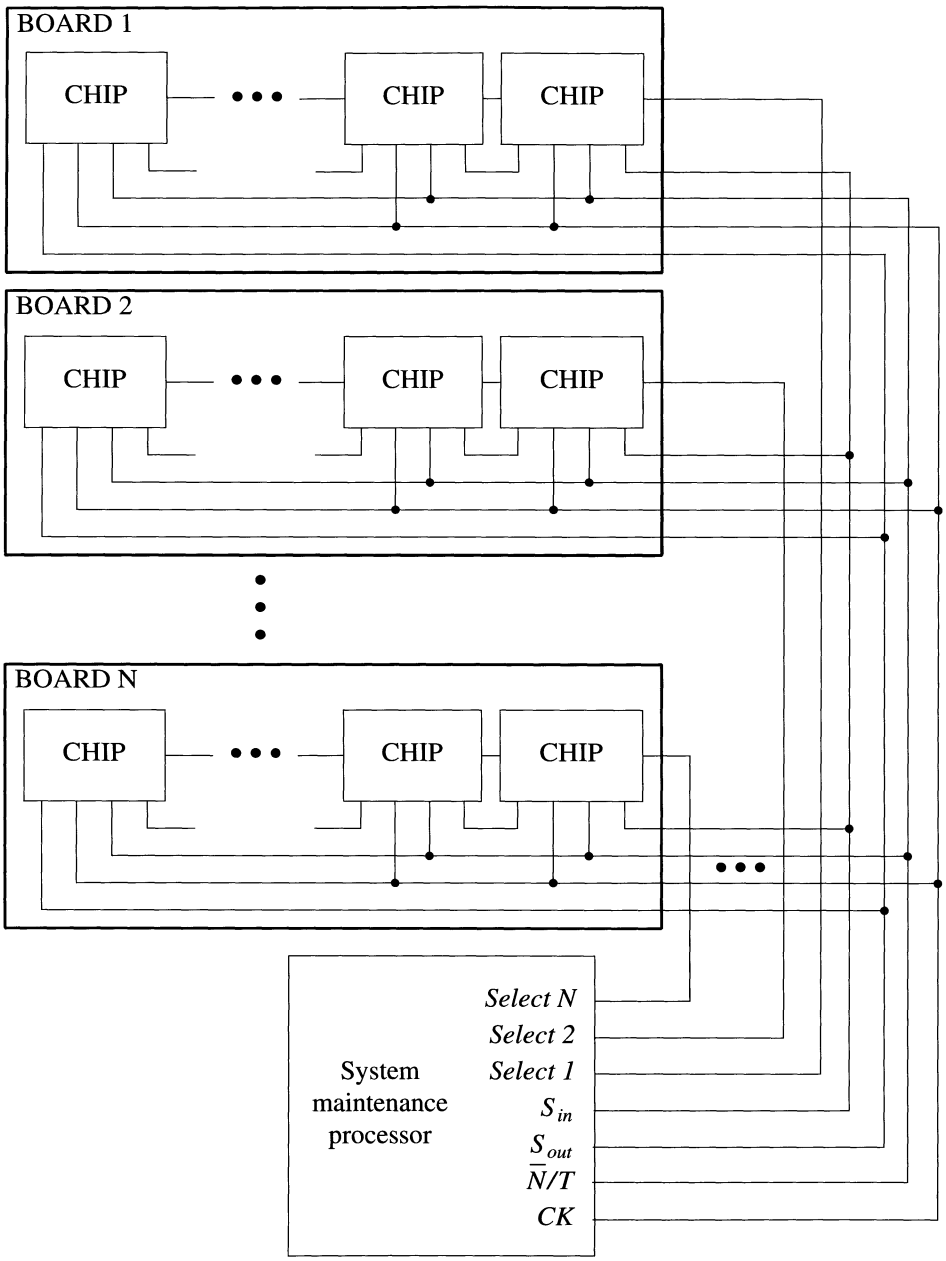


Figure 9.35 Scan applied to the system level

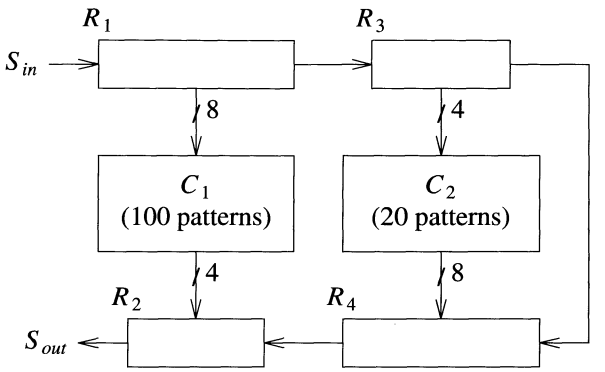


Figure 9.36 Testing using multiple test sessions

Separate Mode

The blocks of logic C_1 and C_2 can be tested *separately*. In this mode, while C_1 is being tested, C_2, R_3, and R_4 are ignored. To test C_1 it is only necessary to load R_1 with a test pattern, and capture and scan out the result in R_2. Let $|R_i|$ be the length of register R_i. Since max $\{|R_1|, |R_2|\} = 8$, only $8 \times 100 = 800$ clock cycles are required to test C_1. To test C_2 in a second test session the scan path needs to be reconfigured by having the scan output of R_4 drive a primary output. This can also be accomplished by disabling the parallel load inputs from C_1 to R_2. We will assume that the latter choice is made. Now C_2 can be tested by loading test patterns into the scan path formed by R_1 and R_3. Each test pattern can be separated by four don't-care bits. Thus each test pattern appears to be eight bits long. When a new test is loaded into R_3, a test result in R_4 must pass through R_2 before it is observed.

The result in R_4, when shifted eight times, makes four bits of the result available at S_{out} and leaves the remaining four bits in R_2. To test C_2 requires $20 \times 8 = 160$ clock cycles. The total time to test C_1 and C_2 is thus 960 clock cycles, compared to the previous case, which required 1200 clock cycles.

Overlapped Mode

C_1 and C_2 can be tested in an *overlapped mode*, that is, partly as one block of logic, and partly as separate blocks. Initially C_1 and C_2 can be combined and tested with 20 patterns, each 12 bits wide. This initial test requires $12 \times 20 = 240$ clock cycles. Now C_2 is completely tested and C_1 can be tested with just 80 of the remaining 100 test patterns. To complete the test of C_1, the scan path need not be reconfigured, but the length of each test pattern is now set to 8 rather than 12. The remaining 80 test patterns are now applied to C_1 as in the separate mode; this requires $80 \times 8 = 640$ clock cycles for a total of 880 clock cycles.

Therefore, by testing the various partitions of logic either together, separately, or in an overlapped mode, and by reorganizing the scan path, test application time can be reduced. No one technique is always better than another. The test time is a function of the scan-

path configuration and relevant parameters such as number of test patterns, inputs, and outputs for each block of logic. More details on the efficient testing of scan-path designs can be found in [Breuer *et al.* 1988a].

Another way to reduce test application time and the number of stored test patterns is embodied in the method referred to as *scan path with look-ahead shifting* (SPLASH) described in [Abadir and Breuer 1986] and [Abadir 1987].

9.9.2 Partial Scan Using I-Paths

I-Modes

Abadir and Breuer [1985 a,b] introduced the concept of I-modes and I-paths to efficiently realize one form of partial scan. A module S with input port X and output port Y is said to have an *identity mode* (I-mode) between X and Y, denoted by IM $(S{:}X \rightarrow Y)$, if S has a mode of operation in which the data on port X is transferred (possibly after clocking) to port Y. A time tag t and activation-condition tags C and D are associated with every I-mode, where t is the time (in clock cycles or gate delays) for the data to be transferred from X to Y, C denotes the values required on the input control lines of S to activate the mode, and D denotes any values required on data inputs to ensure I-mode operation.

Latches, registers, MUXs, busses, and ALUs are examples of modules with I-modes. There are two I-modes associated with the multiplexer shown in Figure 9.37(a), denoted by [IM(MUX:$A \rightarrow C$); $x = 0$; $t = 10ns$], and [IM(MUX:$B \rightarrow C$); $x = 1$; $t = 10ns$]. There are several I-modes associated with the ALU shown in Figure 9.37(b); one is denoted by [IM(ALU:$A \rightarrow C$); $x_1x_2 = 00, t = 20ns$], where $x_1x_2 = 00$ is the condition code for the ALU to pass data from A to C; another I-mode is denoted by [IM(ALU:$A \rightarrow C$); $x_1x_2 = 01$; $B = 0$; $C_{in} = 0$], where $x_1x_2 = 01$ is the condition code for the ALU to operate as an adder. The I-mode for the register shown in Figure 9.37(c) is denoted by [IM(Register:$A \rightarrow B$); t = 1 clock cycle].

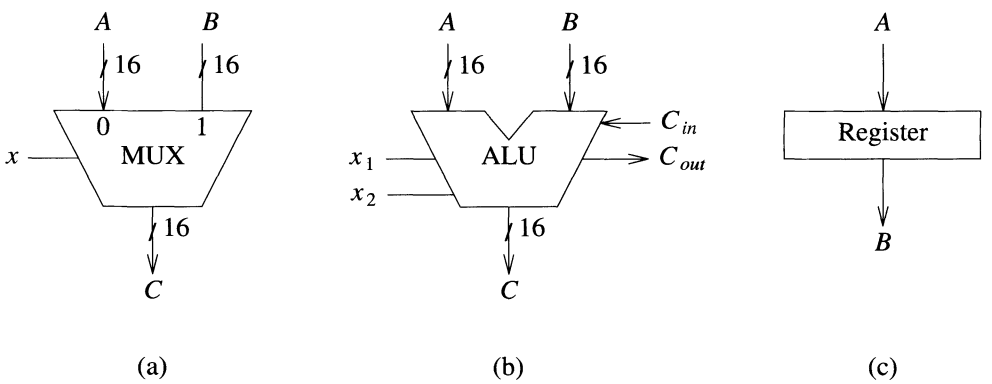


Figure 9.37 Three structures having I-modes

I-Paths

An *identity-transfer path* (I-path) exists from output port X of module $S1$ to input port Y of module $S2$, denoted by IP($S1{:}X \rightarrow S2{:}Y$), if data can be transferred unaltered, but possibly delayed, from port X to port Y. Every I-path has a time tag and activation plan. The time tag indicates the time delay for the data to be transferred from X to Y; the activation plan indicates the sequence of actions that must take place to establish the I-path. An I-path consists of a chain of modules, each of which has an I-mode.

Testing Using I-paths

Example 9.1: Consider the portion of a large circuit shown in Figure 9.38. All data paths are assumed to be 32 bits wide. Note that from the output port of the block of logic C, I-paths exist to the input ports of R_1, R_2, and R_3. Also, I-paths exist from the output ports of R_1, R_2, R_3, and R_4 to the input port of C. Register R_2 has a hold-enable line $\overline{H}/L$, where if $\overline{H}/L = 0$ R_2 holds its state, and if $\overline{H}/L = 1$ the register loads. Let $T = \{T_1, T_2, \ldots, T_n\}$ be a set of test patterns for C. Then C can be tested as shown in Figure 9.39. We assume that only one register can drive the bus at any one time, and a tristate driver is disabled (output is in the high impedance state) when its control line is high.

This process can now be repeated for test pattern T_{i+1}. While T_{i+1} is shifted into R_1, the response Z_i to T_i is shifted out over the S_{out} line.

In this partial scan design not only is the hardware overhead much less than for a full-scan design, but also the scan register has no impact on the signal delay in the path from R_2 to R_3. □

This partial-scan approach leads to the following design problems.

1. Identifying a subset of registers to be included in the scan path.
2. Scheduling the testing of logic blocks. Since hardware resources, such as registers, busses, and MUXs are used in testing a block of logic it is usually impossible to test all the logic at one time. Hence after one block of logic is tested, another can be tested. As an example, for the circuit shown in Figure 9.38, R_1 along with other resources are first used to test C. This represents one test session. Later, R_1 and other resources can be used to test some other block of logic.
3. Determining efficient ways to activate the control lines when testing a block of logic.
4. Determining ways of organizing the scan paths to minimize the time required to test the logic.

The solution to some of these problems are discussed in [Breuer *et al.* 1988a,b].

More Complex Modes and Paths

The I-mode discussed previously is a parallel-to-parallel (P/P) I-mode, since data enter and exit modules as n-bit blocks of information. Other types of I-modes exist, such as serial-to-serial (S/S), serial-to-parallel (S/P), and parallel-to-serial (P/S). An example of a P/S I-mode would be a scan register which loads data in parallel and transmits them serially using its shift mode. Concatenating structures having various types of I-modes produces four types of I-paths, denoted by P/P, P/S, S/P, and S/S.

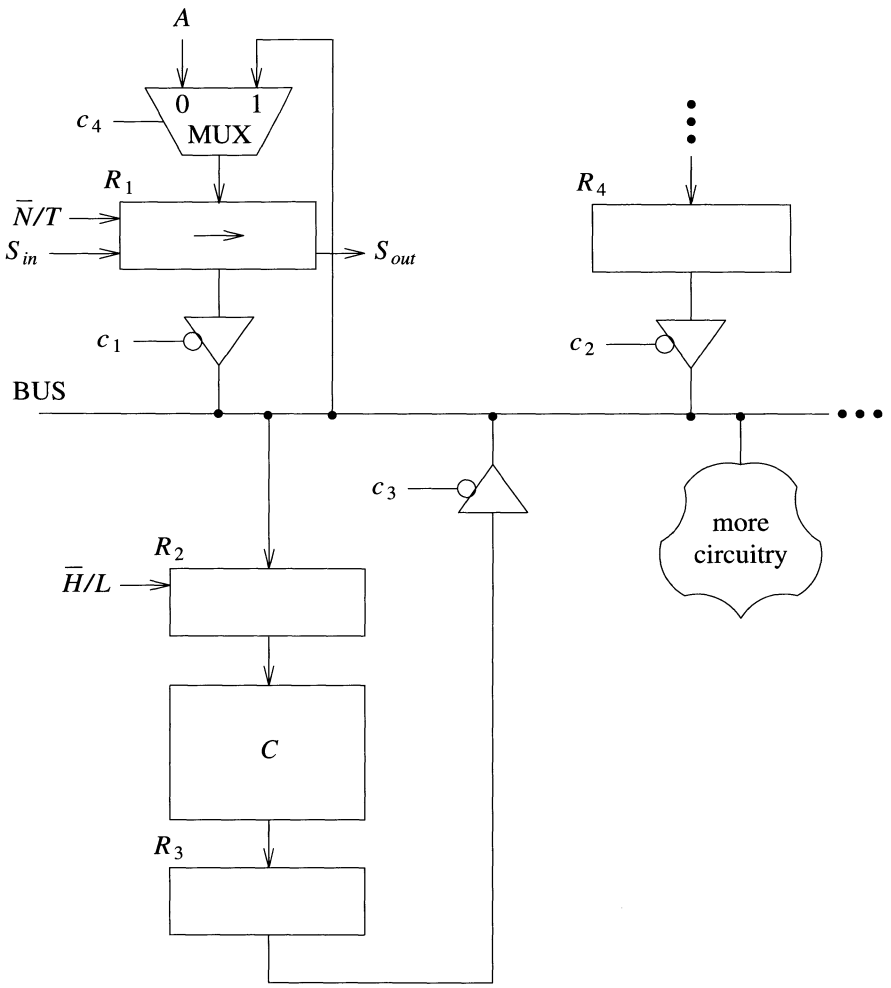


Figure 9.38 Logic block to be tested using I-path partial scan

In addition to I-modes, several other modes can be defined to aid in testing and in reducing the number of scan registers. A module S is said to have a *transfer-mode* (T-mode) if an onto mapping exists between input port X of S and output port Y of S. A trivial example of a structure having a T-mode is an array of inverters that maps the input vector X into NOT(X). A T-path consists of a chain of modules having zero or more I-modes and at least one T-mode. I-paths and T-paths are used primarily for transmitting data from a scan register to the input port of a block of logic to be tested.

A module V having an input port X and an output port Y is said to have a *sensitized mode* (S-mode) if V has a mode of operation such that an error in the data at port X produces an error in the data at port Y. An example is a parallel adder defined by the equation

Time	Controls	Operation
$t_1 \ldots t_{32}$	$\overline{N}/T = 1$	Scan T_i into R_1
t_{33}	$c_1 = 0$ $\overline{H}/L = 1$	Contents of R_1 are loaded onto the bus; data on bus are loaded into R_2
t_{34}	—	Test pattern T_i is applied to C; Response Z_i from C is loaded into R_3
t_{35}	$c_3 = 0$ $c_4 = 1, \overline{N}/T = 0$	Z_i is loaded onto bus; Z_i passes from bus through MUX and is loaded into R_1

Figure 9.39 Test process for block C

SUM=A+B. If B is held at any constant value, then an error in A produces an error in *SUM*. An S-path consists of a chain of structures having zero or more I-modes and at least one S-mode. S-modes correspond to one-to-one mappings.

When testing a block of logic, its response must be transmitted to a scan register or primary outputs. To transmit these data, I-paths and S-paths can be used.

More details on I-paths, S-paths, and T-paths can be found in [Breuer *et al.* 1988a]. Freeman [1988] has introduced the concept of F-paths, which correspond to "one-to-one" mappings, and S-paths, which correspond to "onto" mappings, and has shown how these paths can be effectively used in generating tests for data-path logic. He assumes that functional tests are used and does not require scan registers.

9.9.3 BALLAST — A Structured Partial Scan Design

Several methods have been proposed for selecting a subset of storage cells in a circuit to be replaced by scan-storage cells [Trischler 1980, Agrawal *et al.* 1987, Ma *et al.* 1988, Cheng and Agrawal 1989].

The resulting circuit is still sequential and in most cases sequential ATG is still required. But the amount of computation is now reduced. It is difficult to identify and/or specify the proper balance between adding storage cells to the scan path and reducing ATG cost. However, it appears that any heuristic for selecting storage cells to be made part of a scan path will lead to reduce ATG computation.

BALLAST (Balanced Structure Scan Test) is a structured partial scan method proposed by Gupta *et al.* [1989a,b]. In this design approach, a subset of storage cells is selected and made part of the scan path so that the resulting circuit has a special *balanced* property. Though the resulting circuit is sequential, only combinational ATG is required, and complete coverage of all detectable faults can be achieved.

The test plan associated with BALLAST is slightly different from that employed in a full-scan design, in that once a test pattern is shifted into the scan path, more than one normal system clock may be activated before the test result is loaded into the scan path and subsequently shifted out. In addition, in some cases the test data must be held in the scan path for several clock cycles while test data propagate through the circuitry.

Circuit Model

In general a synchronous sequential circuit S consists of blocks of combinational logic connected to each other, either directly or through registers, where a register is a collection of one or more storage cells. The combinational logic in S can be partitioned into maximal regions of connected combinational logic, referred to as *clouds*. The inputs to a cloud are either primary inputs or outputs of storage cells; the outputs of clouds are either primary outputs or inputs to storage cells. A group of wires forms a vacuous cloud if (1) it connects the outputs of one register directly to the inputs of another, (2) it represents circuit primary inputs feeding the inputs of a register, or (3) it represents the outputs of a register that are primary outputs. Storage cells can be clustered into registers as long as all storage cells in the register share the same control and clock lines. Storage cells can also be grouped together so that each register receives data from exactly one cloud and feeds exactly one cloud. However, a cloud can receive data from more than one register and can feed more than one register.

Figure 9.40 illustrates these concepts. C_1, C_2, and C_3 are nonvacuous clouds; A_1, A_2, and A_3 are vacuous clouds; c_1, c_2, ..., c_5 are blocks of logic, and R_1, R_2, ..., R_5 are registers. In forming clouds and registers, it is important first to create the clouds and then cluster storage cells into registers. If all storage cells are first clustered into one register, it is possible that only one cloud will be identified. A circuit can be partitioned into clouds in linear time.

A synchronous sequential circuit S is said to be *balanced*, denoted as a B-structure, if for any two clouds v_1 and v_2 in S, all signal paths (if any) between v_1 and v_2 go through the same number of registers. This condition also implies that S has an acyclic structure. The structure shown in Figure 9.40 is balanced. The structures shown in Figure 9.41 are not balanced.

Let S^* be a sequential circuit. There always exists a subset of registers which, if replaced by PIs and POs, creates a new circuit S, which is balanced. Let this subset of registers be made part of a scan path. This scan path represents pseudo-primary inputs and outputs to S. S is said to be the *kernel* of logic that is to be tested.

Given a B-structure S^B, its *combinational equivalent* C^B is the combinational circuit formed from S^B by replacing each storage cell (assumed to be a D flip-flop) in S^B by a wire. For simplicity we assume that only the Q output of the cell is used. Define the *depth* d of S^B as the largest number of registers on any path in S^B between any two clouds.

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a complete test set for all detectable stuck at faults in C^B. Then the circuit S is tested as follows. Each test pattern $t_i = (t_i^a, t_i^b)$ consists of two parts, where t_i^a is applied to the primary inputs to S, and t_i^b is applied to the pseudo-primary inputs to S, that is, t_i^b is the data in the scan path. Then S^B can be tested as follows.

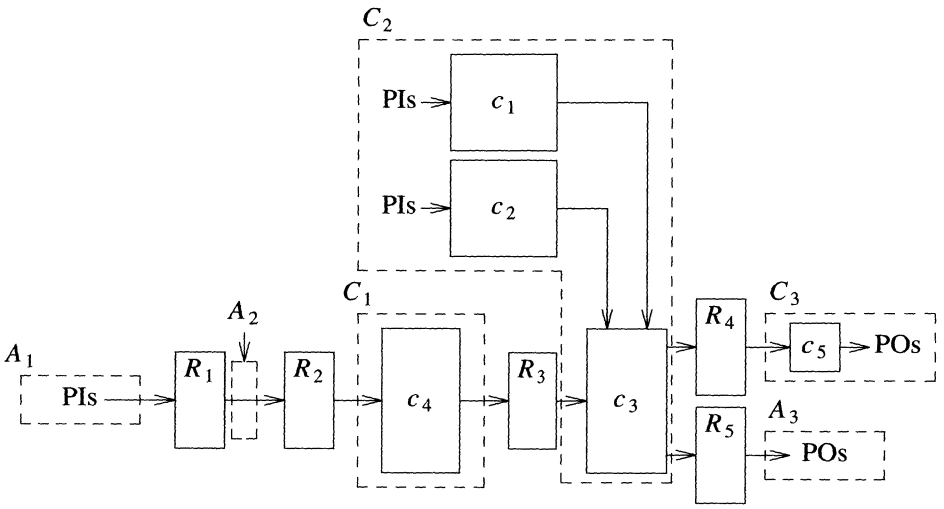


Figure 9.40 A partitioned circuit showing clouds and registers

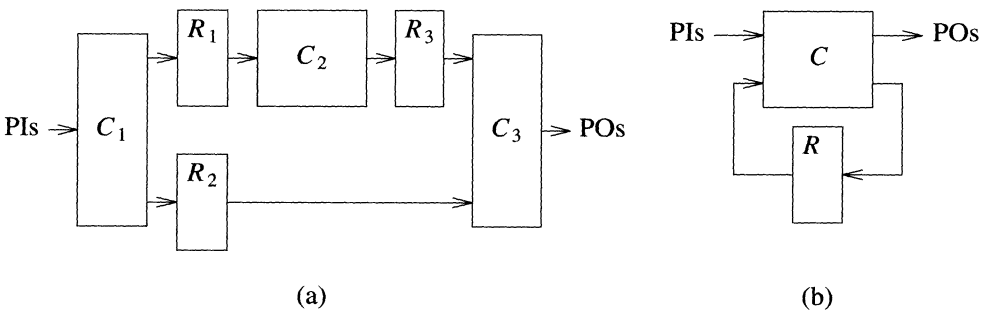


Figure 9.41 Nonbalanced structures (a) Unequal paths between C_1 and C_3 (b) A self-loop (unequal paths between C and itself)

Step 1: Scan in the test pattern t_i^b.

Step 2: Apply t_i^a to the primary inputs to S.

Step 3: While holding t_i^a at the primary inputs and t_i^b in the scan path, clock the registers in S d times.

Step 4: Place the scan path in its normal mode and clock it once.

Step 5: Observe the value on the primary outputs.

Step 6: Simultaneously, scan out the results in the scan paths and scan in t^b_{i+1}.

Note that in this test plan the scan path has three modes of operation, namely normal (parallel load), hold, and shift.

Example 9.2: Consider the circuit S shown in Figure 9.42(a). Selecting R_3 and R_6 to be scan registers produces the partial scan design shown in Figure 9.42(b). Replacing the scan path by PIs and POs results in the B-structure S^B, shown in Figure 9.42(c), which has depth $d = 2$.

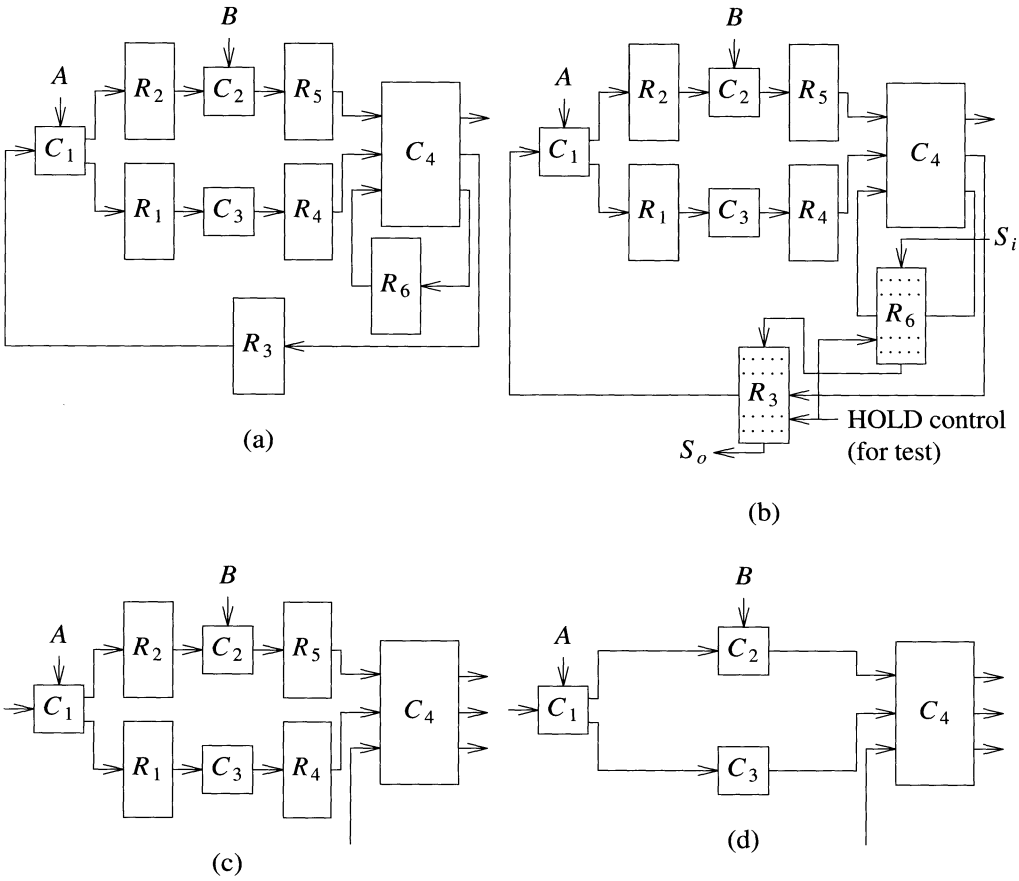


Figure 9.42 (a) Synchronous circuit S (b) A partial scan design of S (c) Kernel K of S (d) Combinational equivalent of K

The combinational equivalent C^B is shown in Figure 9.42(d). To test S^B, a pattern is shifted into the scan path R_3 and R_6 and held there two clock periods while a test pattern is also applied to the primary inputs A and B. After one clock period test results are captured in registers R_1 and R_2. After the second clock period test results are captured in R_4 and R_5. Finally test results are captured in the scan path R_3 and R_6. □

It has been shown that any single stuck fault that is detectable in the combinational logic in S is detectable by a test vector for C^B using the procedure just described [Gupta *et al.* 1989]. Thus, restricting one's attention to single stuck faults in the clouds, no fault coverage is sacrificed by employing the BALLAST test method. However, some additional test may be required to detect shorts between the I/O of storage cells.

A circuit can be transformed into a B-structure by removing an appropriate set of registers, which then become part of the scan path. Several criteria can be used to select these registers, such as information on critical paths in the circuit or the number of cells in the registers. For example, one criterion is to select a minimal number of storage cells to be made part of the scan path so that the resulting circuit is a B-structure.

Unfortunately, the problem just cited is NP-complete. A heuristic procedure for its solution is presented in [Gupta *et al.* 1989].

Figure 9.43 shows several different scan designs for the circuit shown in Figure 9.42(a), other than the one shown in Figure 9.42(b). Let n_i be the number of storage cells in register R_i. The design shown in Figure 9.43(a) can be considered superior to the one shown in Figure 9.42(b) if $n_1 + n_2 < n_3$, since fewer storage cells exist in the scan path. Assume that a scan storage cell introduces delay into a path terminating at its output port. Then the designs shown in Figure 9.43(c) and (d) may be well suited for the case where critical paths exist between R_5 and R_3 and between R_3 and R_1. Note that the depth for the design in Figure 9.43(d) is 3. Finally, Figure 9.43(e) shows a full-scan design. In all these designs, R_6 was made part of the scan path. For some partial scan designs it may not be necessary to transform each storage cell in R_6 into a scan-storage cell. The sequential circuit consisting of C_4 and R_6 can be replaced by a lower-level design description, where R_6 is replaced by its individual storage cells and C_4 is segmented into the cones of logic that feed each storage cell. The resulting circuit can then be made into a partial scan design using the BALLAST design methodology.

The importance of using a balanced structure is illustrated in the next example.

Example 9.3: Consider the circuit shown in Figure 9.44(a). This is a nonbalanced circuit having the structure shown in Figure 9.41(a). Consider the fault b *s-a*-1. The test pattern for this fault at time t within cloud C_3 consists of $(Q_3,Q_4,Q_5) = (0,0,1)$. This implies that at time $(t-1)$, $(Q_1,Q_2) = (1,0)$ and $A = 0$. This in turn implies that at time $(t-2)$, $(A,B) = (1,0)$. One test sequence for this fault is shown in Figure 9.44(a). The combinational equivalent of this circuit is shown in Figure 9.44(b), where now the fault b *s-a*-1 is redundant. □

It is seen that some nonbalanced circuits may require sequential ATG.

In some cases, it is not necessary that the scan registers have a hold mode. Methods for removing hold modes as well as dealing with designs where the registers in the original circuit have hold modes are dealt with in [Gupta *et al.* 1989b]. For example, for the situation shown in Figure 9.42(b), let $t(3)$ and $t(6)$ be the test patterns to be loaded into R_3 and R_6, respectively. Consider the test pattern $t(3)$ *xx* $t(6)$ where two don't-care bits have been placed between $t(3)$ and $t(6)$. Now the scan path can continue shifting while the test data propagate through the circuit. Assume at time t that $t(3)$ is in R_3. At time $(t + 1)$, test results are latched into R_2 and R_3, at time $(t + 2)$ test pattern $t(6)$ is in R_6 and test results are latched into R_4 and R_5. At time $(t + 3)$ test results are latched into R_3 and R_6.

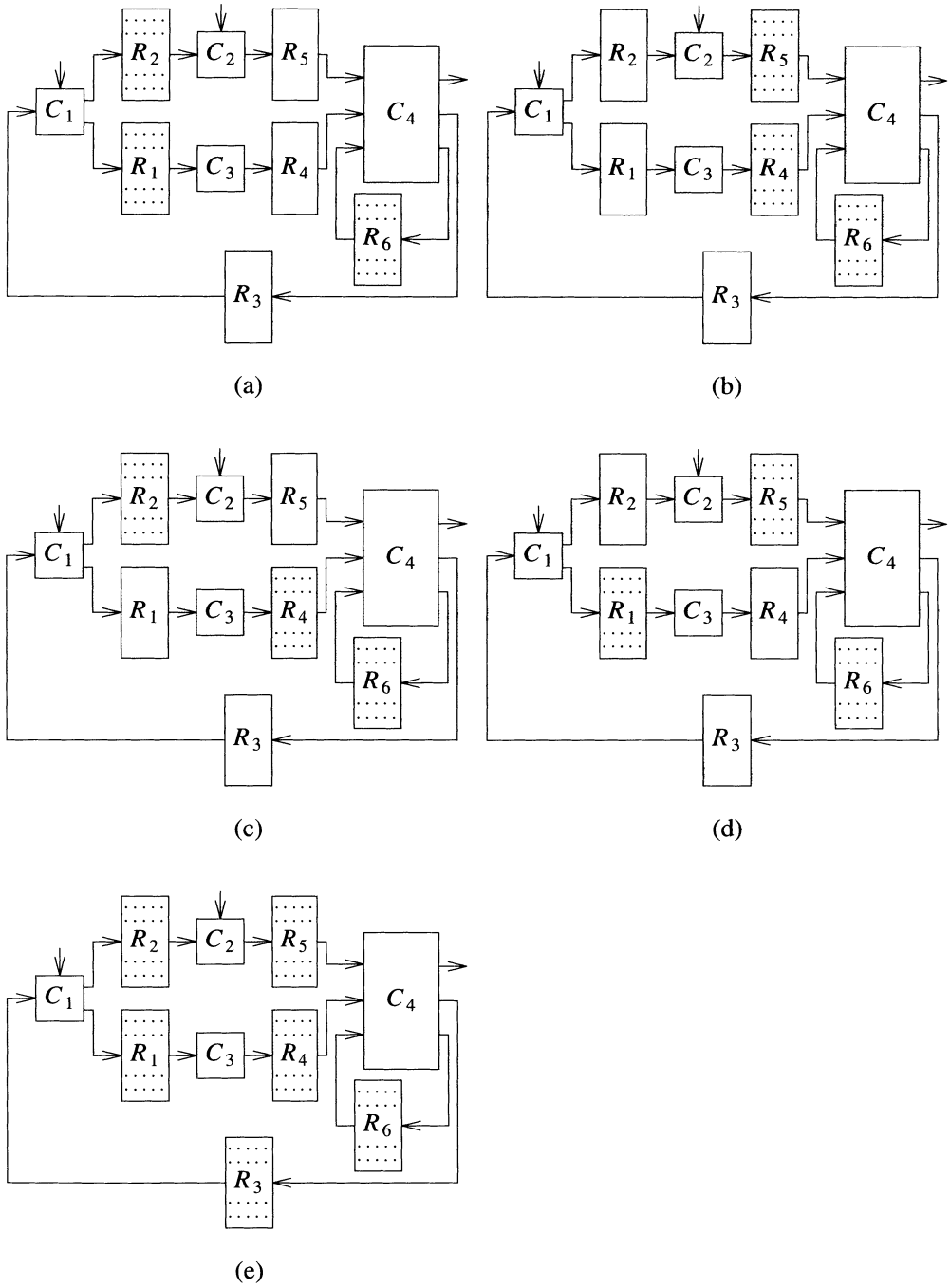


Figure 9.43 Different partial and full scan designs for S

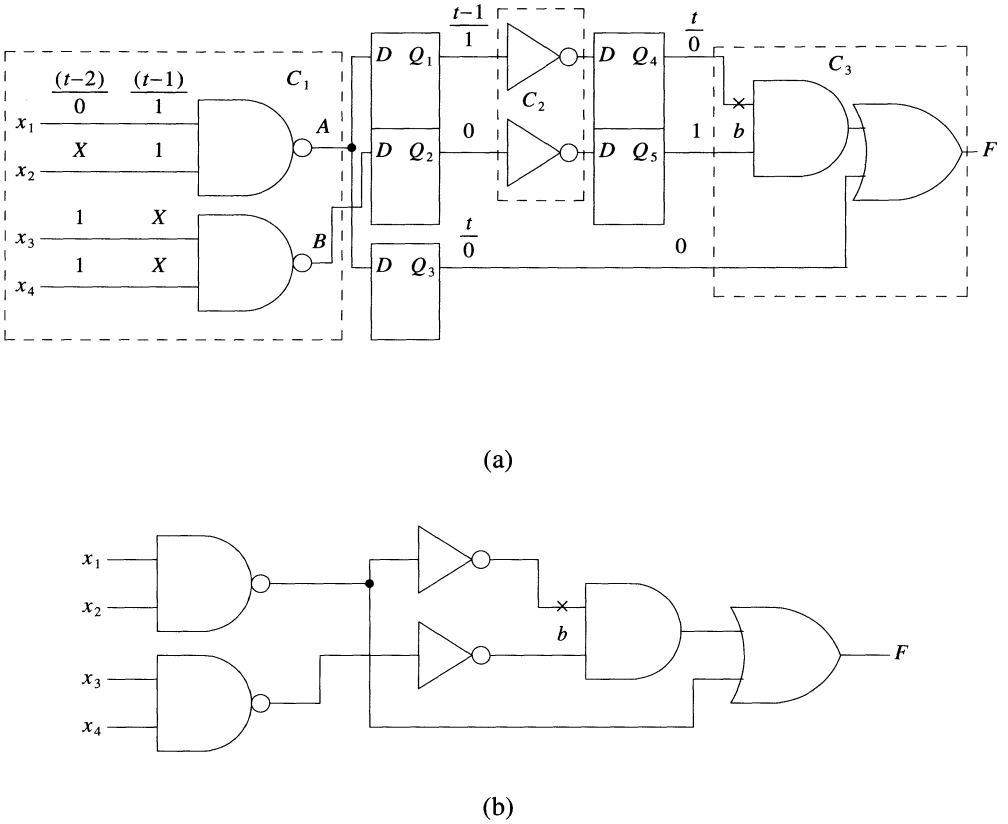


Figure 9.44 (a) A nonbalanced circuit (b) Its combinational equivalent

9.10 Boundary-Scan Standards

9.10.1 Background

To better address problems of board-level testing, several DFT standards have been and are currently being developed. The primary goal of these proposed standards is to ensure that chips of LSI and VLSI complexity contain a common denominator of DFT circuitry that will make the test development and testing of boards containing these chips significantly more effective and less costly. Some of these initiatives are known as the Joint Test Action Group (JTAG) Boundary Scan standard [JTAG 1988], the VHSIC Element-Test and Maintenance (ETM) — Bus standard [IBM *et al.* 1986a], the VHSIC Test and Maintenance (TM) — Bus standard [IBM *et al.* 1986b], and the IEEE 1149.1 Testability Bus Standard [IEEE 1989].

These standards deal primarily with the use of a test bus which will reside on a board, the protocol associated with this bus, elements of a bus master which controls the bus, I/O ports that tie a chip to the bus, and some control logic that must reside on a chip to interface the test bus ports to the DFT hardware residing on the application portion of the chip. In addition, the JTAG Boundary Scan and IEEE 1149.1 standards also require that a boundary-scan register exist on the chip.

The primary reasons for using boundary scan are to allow for efficient testing of board interconnect and to facilitate isolation and testing of chips either via the test bus or by built-in self-test hardware. With boundary scan, chip-level tests can be reused at the board level.

The description of a board-level test bus presented in this section is based on IEEE 1149.1. There are several major components associated with IEEE 1149.1, namely (1) the physical structure of the test bus and how it can be interconnected to chips, (2) the protocol associated with the bus, and (3) the on-chip test bus circuitry associated with a chip. The latter includes the boundary-scan registers and the test access port (TAP) controller which is a finite-state machine that decodes the state of the bus.

Figure 9.45 shows a general form of a chip which supports IEEE 1149.1. The application logic represents the normal chip design prior to the inclusion of logic required to support IEEE 1149.1. This circuitry may include DFT or BIST hardware. If so, the scan paths are connected via the test-bus circuitry to the chip's scan-in and scan-out ports. This is illustrated by the connection from the test data input line *TDI* to S_{in}, and S_{out} to the test data output line *TDO*. The normal I/O terminals of the application logic are connected through boundary-scan cells to the chips I/O pads.

The test-bus circuitry, also referred to as the bus slave, consists of the boundary-scan registers, a 1-bit bypass register, an instruction register, several miscellaneous registers, and the TAP.

The boundary-scan bus consists of four lines, namely a test clock (*TCK*), a test mode signal (*TMS*), the *TDI* line, and the *TDO* line.

Test instructions and test data are sent to a chip over the *TDI* line. Test results and status information are sent from a chip over the *TDO* line to whatever device is driving the bus. This information is transmitted serially. The sequence of operations is controlled by a bus master, which can be either ATE or a component that interfaces to a higher-level test bus that is part of a hierarchical test and maintenance system. Control of the test-bus circuitry is primarily carried out by the TAP, which responds to the state transitions on the *TMS* line.

Briefly, the test bus and associated logic operates as follows.

1. An instruction is sent serially over the *TDI* line into the instruction register.
2. The selected test circuitry is configured to respond to the instruction. In some cases this may involve sending more data over the *TDI* line into a register selected by the instruction.
3. The test instruction is executed. Test results can be shifted out of selected registers and transmitted over the *TDO* line to the bus master. It is possible to shift new data into registers using the *TDI* line while results are shifted out and transmitted over the *TDO* line.

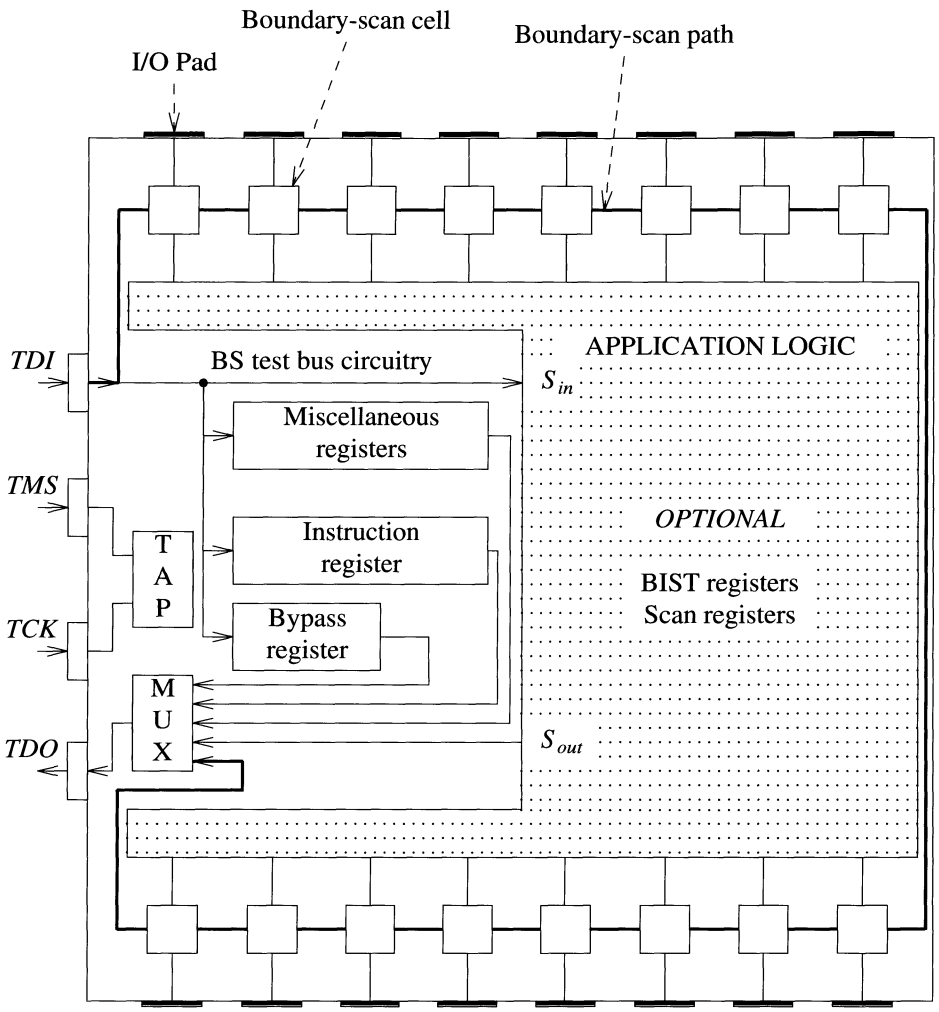


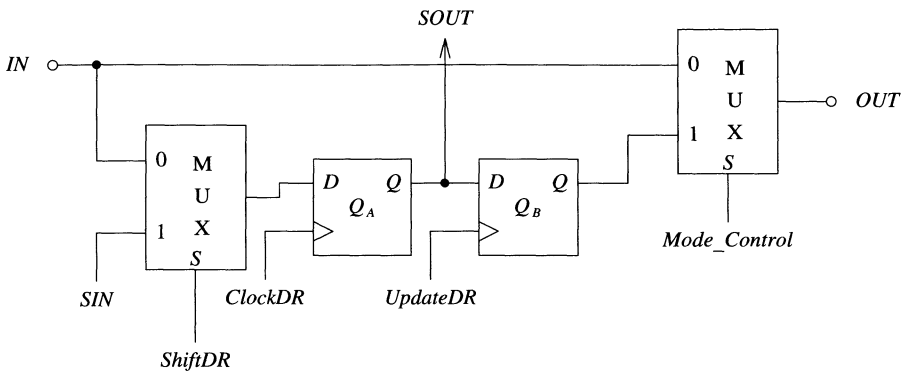
Figure 9.45 Chip architecture for IEEE 1149.1

9.10.2 Boundary-Scan Cell

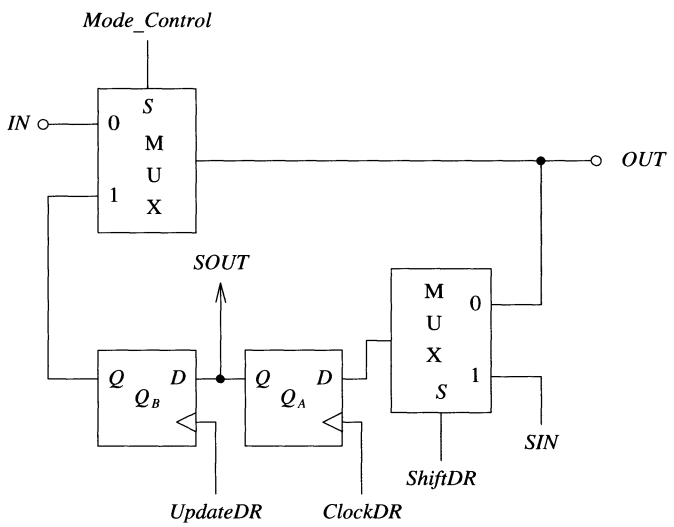
Two possible boundary-scan cell designs are shown in Figure 9.46. These cells can be used as either output or input cells. Other cell designs exist for bidirectional I/O ports and tristate outputs.

As an input boundary-scan cell, *IN* corresponds to a chip input pad, and *OUT* is tied to a normal input to the application logic. As an output cell, *IN* corresponds to the output of the application logic, and *OUT* is tied to an output pad. The cell has several modes of operations.

Normal Mode: When *Mode_Control*=0, data passes from port *IN* to port *OUT*; then the cell is transparent to the application logic.



(a)



(b)

Figure 9.46 Two example boundary-scan cells

Scan Mode: The boundary-scan cells are interconnected into a scan path, where the *SOUT* terminal of one cell is connected to the *SIN* terminal of the next cell in the path. The first cell is driven by *TDI*, and the last one drives *TDO*. In the scan mode, *ShiftDR*=1 and clock pulses are applied to *ClockDR*. *DR* denotes a data register; *IR* denotes the instruction register.

Capture Mode: The data on line *IN* can be loaded into the scan path by setting *ShiftDR*=0 and applying one clock pulse to *ClockDR*. Thus the Q_A storage cells act as a shadow or snapshot register. The *OUT* terminal can be driven by either *IN* or the output of Q_B.

Update Mode: Once the Q_A storage cell is loaded, either by a capture or scan operation, its value can be applied to the *OUT* port by setting *Mode_Control* = 1 and applying a clock pulse to *UpdateDR*.

If a boundary-scan cell drives an output pad, then the select line of the multiplexer driving the output is driven by a signal denoted by *Output_Mode_Control*; if the cell is driven by an input pad the select line is driven by a signal labeled *Input_Mode_Control*.

It is easy to extend these design concepts to handle both tristate and bidirectional I/O pads.

9.10.3 Board and Chip Test Modes

Figure 9.47 shows a board containing four chips which support IEEE 1149.1. The boundary-scan cells are interconnected into a single scan path, where the *TDO* of one chip is tied to the *TDI* of another chip, except for the initial *TDI* and *TDO* ports which are tied to distinct terminals of the board. Some of the normal interconnect between chip pads is also shown. Using this configuration various tests can be carried out, including (1) interconnect test, (2) snapshot observation of normal system data, and (3) testing of each chip. To implement these tests, three test modes exist, namely external test, sample test, and internal test.

External Test Mode

To test the interconnect and/or logic external to chips supported by IEEE 1149.1 the circuit configuration shown in Figure 9.48 is used. Here a test pattern can be loaded into the Q_A cells of chips 1 and 2, for example, Q_A(chip 1) = 1 and Q_A(chip 2) = 0. Chip 1 can carry out an Update operation, where the data in Q_A(chip 1) are to drive the output pad. Chip 2 can carry out a Capture operation, where the data on its input pad are loaded into Q_A(chip 2). The data in the scan path can be shifted out to see if the correct response was received. By selecting test data appropriately, tests for shorts, opens, and stuck-at faults can be carried out. By using appropriate boundary-scan cells and test data, interconnect tests can be carried out for tristate logic and bidirectional pads.

Sample Test Mode

Figure 9.49 shows how the I/O data associated with a chip can be sampled during normal system operation. This sampled data can be scanned out while the board remains in normal operation. The boundary-scan circuitry must be designed so that the boundary-scan cells in the signal path between a chip I/O pins and the application logic do not interfere with the operation of the board logic.

Internal Test Mode

Figure 9.50 shows the configuration of the boundary-scan cells when in the internal test mode. In this configuration the inputs to the application logic are driven by the input boundary-scan cells, and the response can be captured in the output boundary-scan cells. Since these cells are part of a scan path, this mode of operation gives complete controllability and observability of the I/O pads of a chip. The chip may also have internal scan paths and in fact be designed to have built-in self-test (BIST) capability. While in the internal test mode, the internal scan paths and BIST operations can be activated to test the chip. This configuration replaces some forms of testing normally carried out by bed-of nails test equipment.

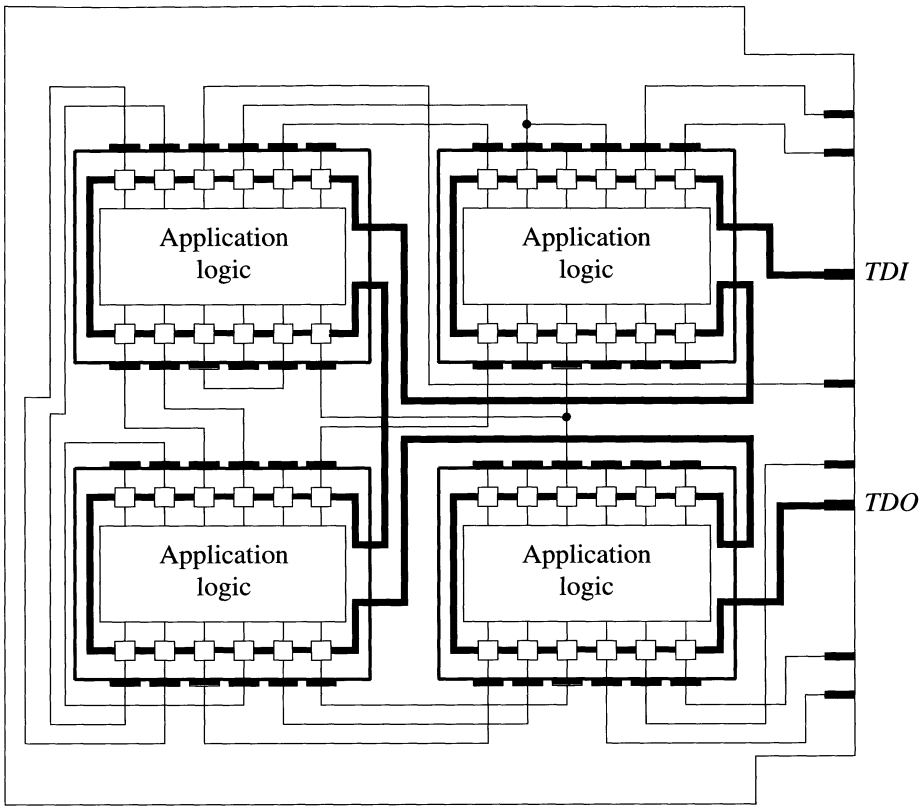


Figure 9.47 A printed circuit board with a IEEE 1149.1 test bus

9.10.4 The Test Bus

A board supporting IEEE 1149.1 contains a test bus consisting of at least four signals. (A fifth signal can be used to reset the on-chip test-bus circuitry.) These signals are connected to a chip via its test-bus ports. Each chip is considered to be a bus slave, and the bus is assumed to be driven by a bus master.

The minimum bus configuration consists of two broadcast signals (*TMS* and *TCK*) driven by the master, and a serial path formed by a "daisy-chain" connection of serial scan data pins (*TDI* and *TDO*) on the master and slave devices. Figure 9.51(a) shows a ring configuration; Figure 9.51(b) shows a star configuration, where each chip is associated with its own *TMS* signal. Star and ring configurations can be combined into hybrid configurations. The four bus signals associated with a slave TAP are defined as follows.

TCK — Test Clock. This is the master clock used during the boundary-scan process.

TDI — Test Data Input. Data or instructions are received via this line and are directed to an appropriate register within the application chip or test bus circuitry.

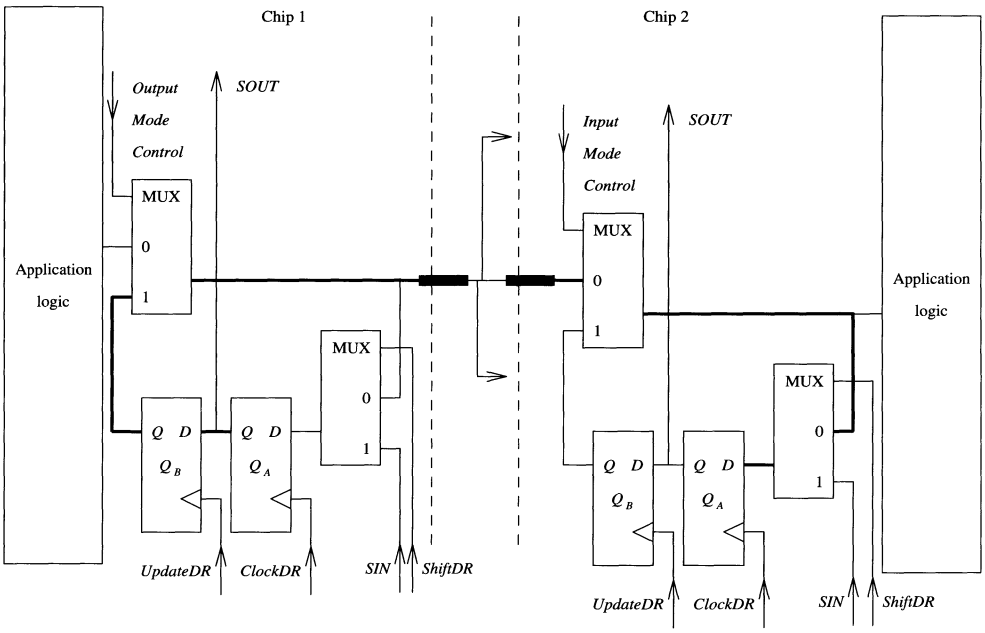


Figure 9.48 External test configuration

TDO — Test Data Output. The contents of a selected register (instruction or data) are shifted out of the chip over TDO.

TMS — Test Mode Selector. The value on TMS is used to control a finite state machine in a slave device so that this device knows when to accept test data or instructions.

Though the IEEE 1149.1 bus employs only a single clock, *TCK*, this clock can be decoded on-chip to generate other clocks, such as the two-phase nonoverlap clocks used in implementing LSSD.

9.10.5 Test-Bus Circuitry

The on-chip test-bus circuitry allows access to and control of the test features of a chip. A simplified version of this circuitry is shown in Figure 9.45, and more details are shown in Figure 9.52. This circuitry consists of four main elements, namely (1) a test access port (TAP) consisting of the ports associated with *TMS, TCK, TDI*, and *TDI*, (2) a TAP controller, (3) a scannable instruction register and associated logic, and (4) a group of scannable test data registers. We will next elaborate on some of these features.

9.10.5.1 The TAP Controller

The TAP controller is a synchronous finite-state machine whose state diagram is shown in Figure 9.53. It has a single input, labeled *TMS*, and its outputs are signals corresponding to a subset of the labels associated with the various states, such as Capture-IR. The state diagram shows that there are two parallel and almost identical

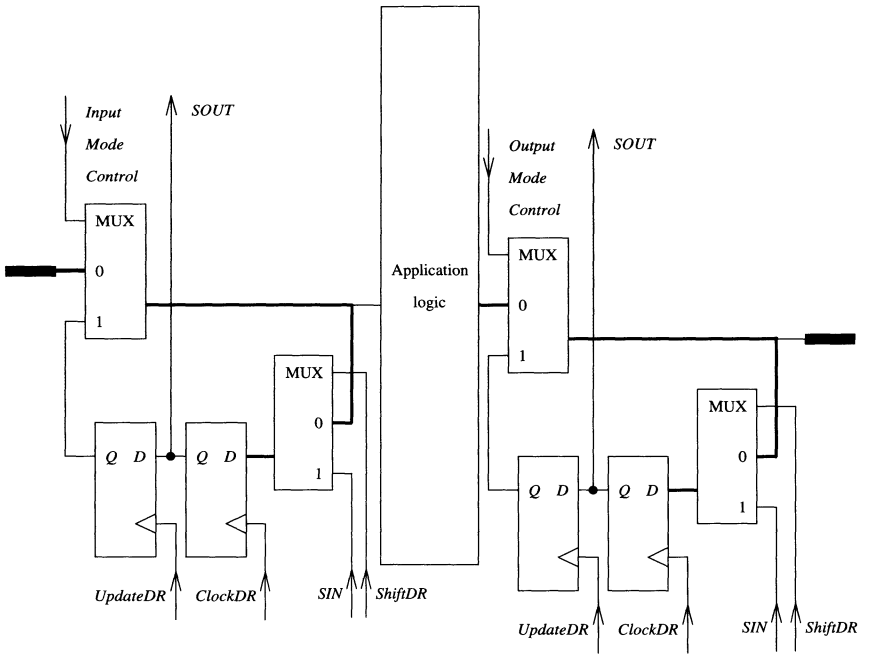


Figure 9.49 Sample test configuration

subdiagrams, one corresponding to controlling the operations of the instruction register, the other to controlling the operation of a data register. The controller can change state only when a clock pulse on *TCK* occurs; the next state is determined by the logic level of line *TMS*. The function of some of these control states is described below.

Test-Logic-Reset: In this state the test logic (boundary-scan) is disabled so that the application logic can operate in its normal mode.

Run-Test/Idle: This is a control state that exists between scan operations, and where an internal test, such as a built-in self-test, can be executed (see instruction RUNBIST).

A test is selected by first setting the instruction register with the appropriate information. The TAP controller remains in this state as long as *TMS*=0.

Select-DR-Scan: This is a temporary controller state. If *TMS* is held low then a scan-data sequence for the selected test-data register is initiated, starting with a transition to the state Capture-DR.

Capture-DR: In this state data can be loaded in parallel into the test-data registers selected by the current instruction. For example, the boundary-scan registers can be loaded while in this state. Referring to Figure 9.49, to capture the data on the input pad shown and the output of the application logic, it is necessary to set *ShiftDR* low and to activate the clock line *ClockDR*.

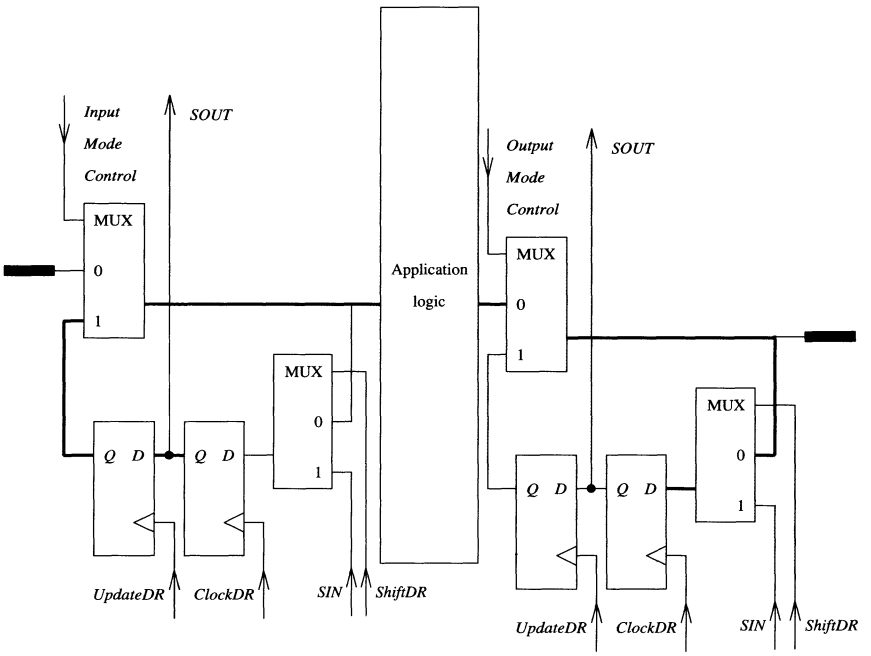


Figure 9.50 Internal test configuration

Shift-DR: In this state test-data registers specified by the data in the instruction register, and which lie between *TDI* and *TDO*, are shifted one position. A new data value enters the scan path via the *TDI* line, and a new data value now is observed at the *TDO* line. Other registers hold their state.

Exit1-DR: In this state all test-data registers selected by the current instruction hold their state. Termination of the scan process can be achieved by setting *TMS* high.

Pause-DR: In this control state the test-data registers in the scan path between *TDI* and *TDO* hold their state. The Pause-DR and Pause-IR states can be used to temporarily halt the scan operation and allow the bus master to reload data. This is often necessary during the transmission of long test sequences.

Update-DR: Some registers have a latched parallel output so that the output does not change during a scan process. In this control state test-data registers specified by the current instructions and having a latched parallel-output feature are loaded from their associated shift registers.

Referring to Figure 9.50, once the boundary-scan register has been loaded with scan data while the TAP controller is in state Shift-DR, these data can be applied to the inputs of the application logic when the controller is in state Update-DR by simply activating the clock *UpdateDR*.

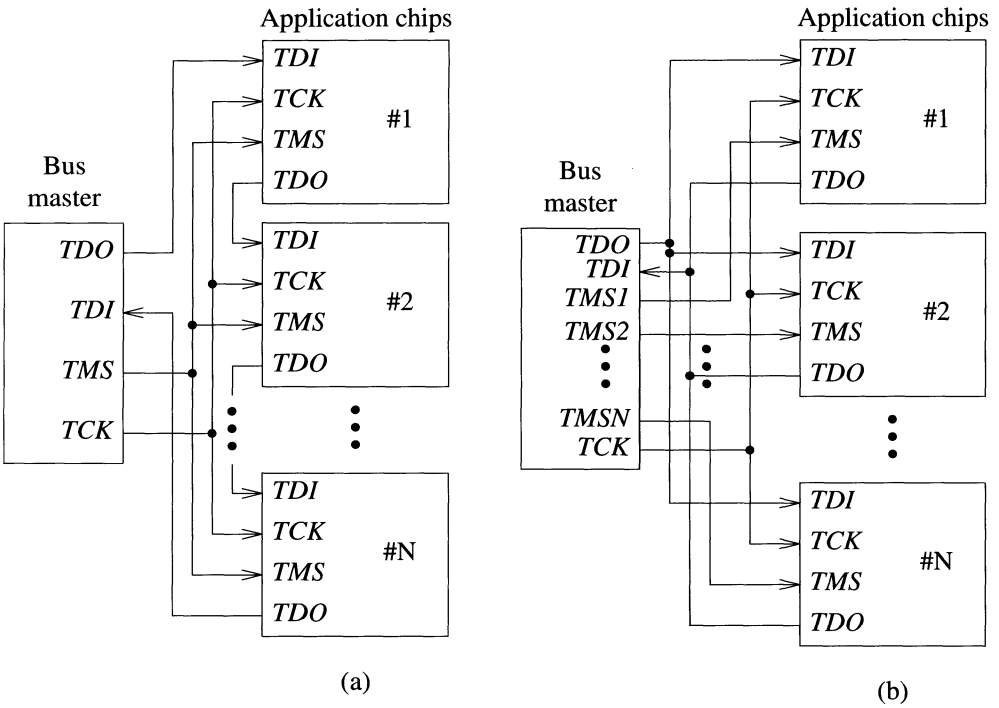


Figure 9.51 (a) Ring configuration (b) Star configuration

The states that control the instruction register operate similarly to those controlling the test-data registers. The instruction register is implemented using a latched parallel-output feature. This way a new instruction can be scanned into the scan path of the instruction register without affecting the output of the register. The Select-IR-Scan state is again a temporary transition state. In the Capture-IR state, the shift register associated with the instruction register is loaded in parallel. These data can be status information and/or fixed logic values. In the Shift-IR state this shift register is connected between *TDI* and *TDO* and shifts data one position. In the Update-IR control state the instruction shifted into the instruction register is latched onto the parallel output of the instruction register. This instruction now becomes the current instruction.

The TAP controller can be implemented using four flip-flops and about two dozen logic gates.

9.10.5.2 Registers

The Instruction Register and Commands

The instruction register has the ability to shift in a new instruction while holding the current instruction fixed at its output ports. The register can be used to specify operations to be executed and select test-data registers. Each instruction enables a single serial test-data register path between *TDI* and *TDO*. The instructions BYPASS, EXTEST, and

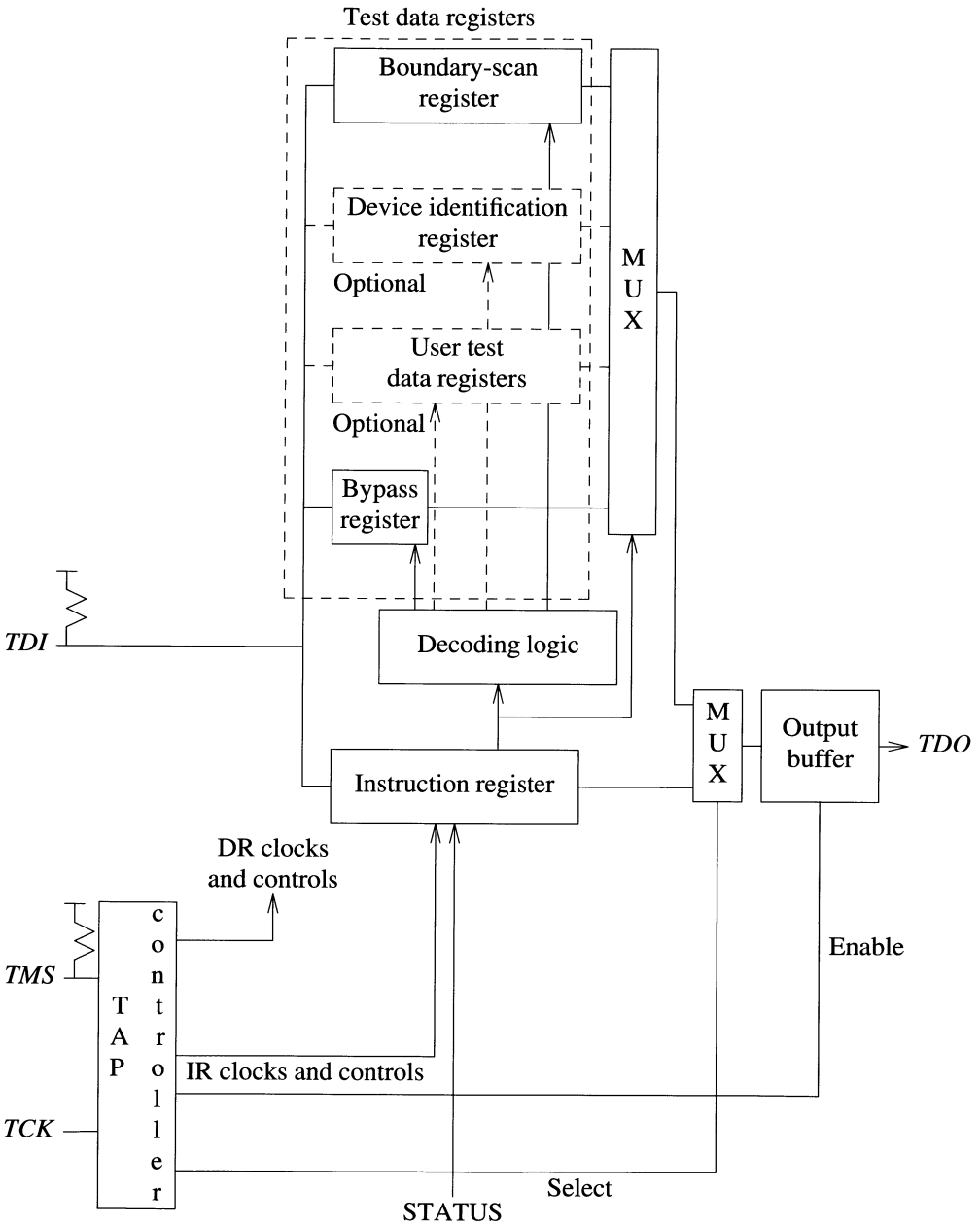


Figure 9.52 IEEE 1149.1 test bus circuitry

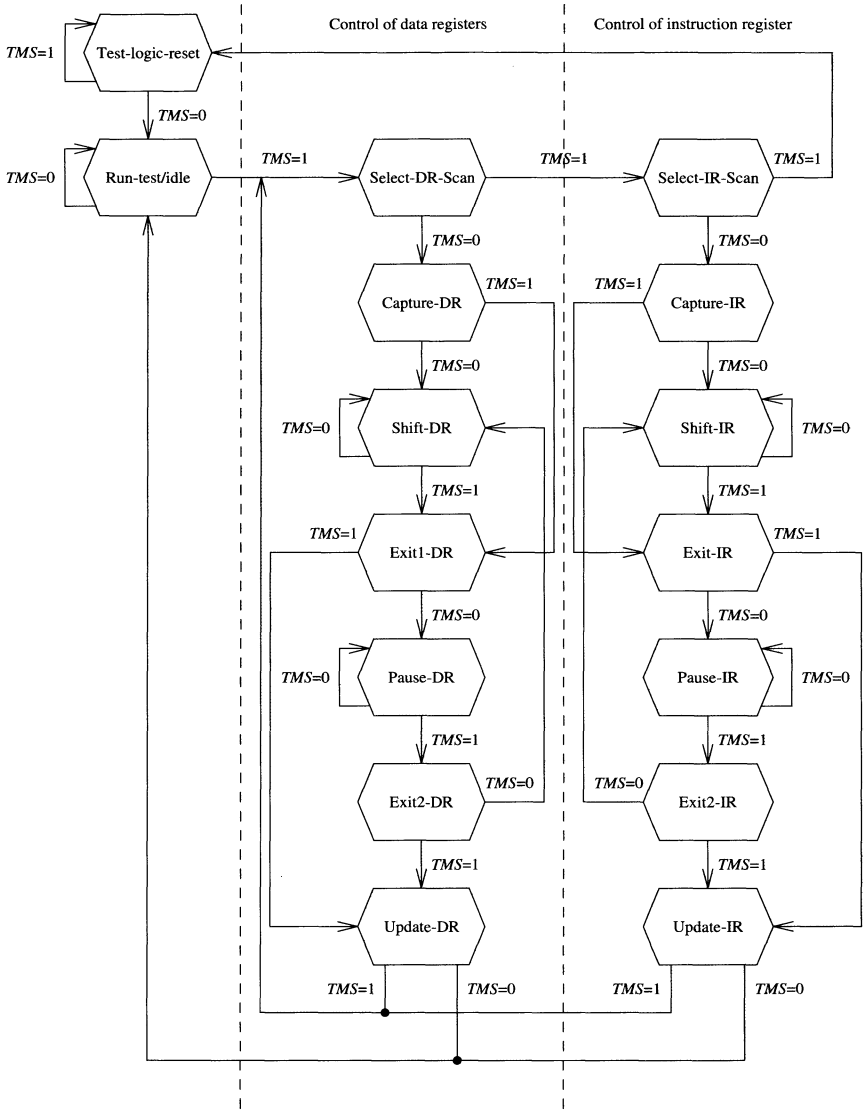


Figure 9.53 State diagram of TAP controller

SAMPLE are required; INTEST or RUNBIST are recommended. A brief description of these instructions follows.

BYPASS Instruction

Every chip must have a BYPASS register, which is a test-data register of length 1. Consider a board having 100 chips that are connected as shown in Figure 9.51(a). Test

data must pass through all the chips when testing any one chip, say the *i*th chip. To reduce the length of the scan path, all chips except for the *i*th chip can be put into the bypass mode, where now only a single storage cell lies between the *TDI* and *TDO* ports of these chips. The BYPASS instruction is used to configure a chip in this bypass mode.

EXTEST Instruction

EXTEST is primarily used to test circuitry external to a chip, such as the board interconnect. When this instruction is executed, boundary-scan cells at output pads are used to drive the pads. Those cells at input pads capture test results when the TAP controller enters the Capture-DR state (see Figure 9.48).

SAMPLE Instruction

The SAMPLE instruction allows the data on the I/O pads of a chip to be sampled and placed in the boundary-scan register during normal board operation (see Figure 9.49).

INTEST Instruction

The INTEST instruction is used to apply a test vector to the application logic via the boundary-scan path, and to capture the response from this logic. First the INTEST instruction is loaded into the instruction register. Then the boundary-scan register is loaded with test data. Because of the INTEST instruction, the inputs to the application logic are driven by the input boundary-scan cells, and the output pads of the chip are driven by the output boundary-scan cells. The scan path is loaded while in the control state Shift-DR. In the Update-DR state these data get applied to the inputs of the application logic. Repeating a load-test-data register cycle, the test results are captured when in state Capture-DR. After this, another test pattern can be loaded into the boundary-scan path while the results are sent back to the bus master. This cycle is repeated for each test pattern.

RUNBIST Instruction

The RUNBIST instruction allows for the execution of a self-test process of the chip. This test is executed while the TAP controller is in the Run-Test/Idle state. The RUNBIST instruction must select the boundary-scan register to be connected between *TDI* and *TDO*. All inputs to the application logic are driven by the boundary-scan register during the execution of this instruction.

Test-Data Registers

The test-bus circuitry contains at least two test-data registers, namely the bypass and the boundary-scan registers. In addition, other scan registers can be accessed and connected between *TDI* and *TDO*, such as device identification registers and registers that are part of the application logic itself. Thus if the application logic is designed according to the full-scan DFT methodology, then this logic can be completely tested by the IEEE 1149.1 architecture.

More information on the use of boundary-scan and related interface logic can be found in [Avra 1987], [Beenker 1985], [Breuer *et al.* 1988], [Breuer and Lien 1988a, b], [Lagemaat and Bleeker 1987], [Lien and Breuer 1989], [Maunder and Beenker 1987], [Wagner 1987], and [Whetsel 1988a, b].

REFERENCES

[Abadir and Breuer 1985a] M. S. Abadir and M. A. Breuer, "Constructing Optimal Test Schedules for VLSI Circuits Having Built-in Test Hardware," *Proc. 15th Intn'l. Fault-Tolerant Computing Conf.*, pp. 165-170, June, 1985.

[Abadir and Breuer 1985b] M. S. Abadir and M. A. Breuer, "A Knowledge Based System for Designing Testable VLSI Chips," *IEEE Design & Test of Computers*, Vol. 2, No. 4, pp. 56-68, August, 1985.

[Abadir and Breuer 1986a] M. S. Abadir and M. A. Breuer, "Scan Path With Look Ahead Shifting," *Proc. Intn'l. Test Conf.*, pp. 699-704, September, 1986.

[Abadir and Breuer 1986b] M. S. Abadir and M. A. Breuer, "Test Schedules for VLSI Circuits," *IEEE Trans. on Computers*, Vol. C-35, No. 5, pp. 361-367, April, 1986.

[Abadir and Breuer 1987] M. S. Abadir and M. A. Breuer, "Test Schedules for VLSI Circuits Having Built-in Test Hardware," *Intn'l. Journal of Computers and Mathematics with Applications*, Vol. 13, No. 5/6, pp. 519-536, 1987.

[Abadir 1987] M. S. Abadir, "Efficient Scan Path Testing Using Sliding Parity Response Compaction," *Proc. Intn'l. Conf. on Computer Aided Design,* pp. 332-335, November, 1987.

[Agrawal and Mercer 1982] V. D. Agrawal and M. R. Mercer, "Testability Measures — What Do They Tell Us?," *Digest of Papers 1982 Intn'l. Test Conf.*, pp. 391-396, November, 1982.

[Agrawal *et al.* 1987] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin, "A Complete Solution to the Partial Scan Problem," *Proc. Intn'l. Test Conf.*, pp. 44-51, September, 1987.

[Ando 1980] H. Ando, "Testing VLSI with Random Access Scan," *Proc. COMPCON*, pp. 50-52, 1980.

[Avra 1987] L. Avra, "A VHSIC ETM-BUS Compatible Test and Maintenance Interface," *Proc. Intn'l. Test Conf.*, pp. 964-971, September, 1987.

[Beenker 1985] F. Beenker, "Systematic and Structured Methods for Digital Board Testing," *Proc. Intn'l. Test Conf.*, pp. 380-385, November, 1985.

[Bennetts 1984] R. G. Bennetts, *Design of Testable Logic Circuits*, Addison-Wesley, Reading, Massachusetts, 1984.

[Bennetts *et al.* 1981] R. G. Bennetts, C. M. Maunder, and G. D. Robinson, "CAMELOT: A Computer-Aided Measure for Logic Testability," *IEE Proc.*, Vol. 128, Part E, No. 5, pp. 177-189, 1981.

[Breuer 1978] M. A. Breuer, "New Concepts in Automated Testing of Digital Circuits," *Proc. EEC Symp. on CAD of Digital Electronic Circuits and Systems*, Brussels, pp. 69-92, 1978.

[Breuer *et al.* 1988a] M. A. Breuer, R. Gupta, and R. Gupta, "AI Aspects of TEST: A System for Designing Testable VLSI Chips," *IFIP Workshop on*

Knowledge-Based Systems for Test and Diagnosis, pp. 29-75, September 27-29, 1988.

[Breuer *et al.* 1988b] M. A. Breuer, R. Gupta, and J. C. Lien, "Concurrent Control of Multiple BIT Structures," *Proc. Intn'l. Test Conf.*, pp. 431-442, September, 1988.

[Breuer and Lien 1988a] M. A. Breuer and J. C. Lien, "A Test and Maintenance Controller for a Module Containing Testable Chips," *Proc. Intn'l. Test Conf.*, pp. 502-513, September, 1988.

[Breuer and Lien 1988b] M. A. Breuer and J. C. Lien, "A Methodology for the Design of Hierarchically Testable and Maintainable Digital Systems," *Proc. 8th Digital Avionics Systems Conf. (DASC)*, pp. 40-47, October 17-20, 1988.

[Chandra and Patel 1989] S. J. Chandra and J. H. Patel, "Experimental Evaluation of Testability Measures for Test Generation," *IEEE Trans. on Computer-Aided Design*, Vol. CAD-8, No. 1, pp. 93-97, January, 1989.

[Chen and Breuer 1985] T-H. Chen and M. A. Breuer, "Automatic Design for Testability Via Testability Measures," *IEEE Trans. on Computer-Aided Design*, Vol. CAD-4, pp. 3-11, January, 1985.

[Cheng and Agrawal 1989] K.-T. Cheng and V. D. Agrawal, "An Economical Scan Design for Sequential Logic Test Generation," *Proc. 19th Intn'l. Symp. on Fault-Tolerant Computing*, pp. 28-35, June, 1989.

[DasGupta *et al.* 1981] S. DasGupta, R. G. Walther, and T. W. Williams, "An Enhancement to LSSD and Some Applications of LSSD in Reliability, Availability, and Serviceability," *Digest of Papers 11th Annual Intn'l. Symp. on Fault-Tolerant Computing*, pp. 32-34, June, 1981.

[DasGupta *et al.* 1982] S. DasGupta, P. Goel, R. G. Walther, and T. W. Williams, "A Variation of LSSD and Its Implications on Design and Test Pattern Generation in VLSI," *Proc. Test Conf.*, pp. 63-66, November, 1982.

[Davidson 1979] R. P. Davidson, "Some Straightforward Guidelines Help Improve Board Testability," *Electronic Design News*, pp. 127-129, May 5, 1979.

[Eichelberger 1983] E. B. Eichelberger, "Latch Design Using 'Level Sensitive Scan Design,'" *Proc. COMPCON*, pp. 380-383, February, 1983.

[Eichelberger and Williams 1977] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testing," *Proc. 14th Design Automation Conf.*, pp. 462-468, June, 1977.

[Eichelberger and Williams 1978] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *Journal Design Automation & Fault-Tolerant Computing*, Vol. 2, No. 2, pp. 165-178, May, 1978.

[Eichelberger *et al.* 1978] E. B. Eichelberger, T. W. Williams, E. I. Muehldorf, and R. G. Walther, "A Logic Design Structure for Testing Internal Array," *3rd USA-JAPAN Computer Conf.*, pp. 266-272, October, 1978.

[Freeman 1988] S. Freeman, "Test Generation for Data-path Logic: The F-Path Method," *IEEE Journal of Solid-State Circuits*, Vol. 23, No. 2, pp. 421-427, April, 1988.

[Funatsu *et al.* 1978] S. Funatsu, N. Wakatsuki,, and A. Yamada, "Designing Digital Circuits with Easily Testable Consideration," *Proc. Test Conf.*, pp. 98-102, September, 1978.

[Goel and McMahon 1982] P. Goel and M. T. McMahon, "Electronic Chip-in-Place Test," *Digest of Papers 1982 Intn'l. Test Conf.*, pp. 83-90, November, 1982.

[Goldstein 1979] L. H. Goldstein, "Controllability/Observability Analysis of Digital Circuits," *IEEE Trans. on Circuits and Systems*, Vol. CAS-26, No. 9, pp. 685-693, September, 1979.

[Goldstein and Thigen 1980] L. M. Goldstein and E. L. Thigen, "SCOAP: Sandia Controllability/Observability Analysis Program," *Proc. 17th Design Automation Conf.*, pp. 190-196, June, 1980.

[Grason 1979] J. Grason, "TMEAS -- A Testability Measurement Program," *Proc. 16th Design Automation Conf.*, pp. 156-161, June, 1979.

[Grason and Nagel 1981] J. Grason and A. W. Nagel, "Digital Test Generation and Design for Testability," *Journal Digital Systems*, Vol. 5, No. 4, pp. 319-359, 1981.

[Gupta *et al.* 1989a] R. Gupta, R. Gupta, and M. A. Breuer, "BALLAST: A Methodology for Partial Scan Design," *Proc. 19th Intn'l. Symp. on Fault-Tolerant Computing*, pp. 118-125, June, 1989.

[Gupta *et al.* 1989b] R. Gupta, R. Gupta, and M. A. Breuer, "An Efficient Implementation of the BALLAST Partial Scan Architecture," *Proc. IFIP Intn'l. Conf. on Very Large Scale Integration (VLSI 89)*, pp. 133-142, August 16-18, 1989.

[Hayes and Friedman 1974] J. P. Hayes and A. D. Friedman, "Test Point Placement to Simplify Fault Detection," *IEEE Trans. on Computers*, Vol. C-33, pp. 727-735, July, 1974.

[IBM *et al.* 1986a] IBM, Honeywell, and TRW, "VHSIC Phase 2 Interoperability Standards," ETM-BUS Specification, December, 1986.

[IBM *et al.* 1986b] IBM, Honeywell and TRW, "VHSIC Phase 2 Interoperability Standards," TM-BUS Specification, December, 1986.

[IEEE 1989] "Standard Test Access Port and Boundary-Scan Architecture," Sponsored by Test Technology Technical Committee of the IEEE Computer Society, Document P1149.1/D5 (Draft), June 20, 1989.

[JTAG 1988] Technical Subcommittee of Joint Test Action Group (JTAG), "Boundary-Scan Architecture Standard Proposal," Version 2.0, March, 1988.

[Kobayashi *et al.* 1968] T. Kobayashi, T. Matsue, and H. Shiba, "Flip-Flop Circuit with FLT Capability," *Proc. IECEO Conf.*, in Japanese, p. 692, 1968.

[Kovijanic 1979] P. G. Kovijanic, "Computer Aided Testability Analysis," *Proc. IEEE Automatic Test Conf.*, pp. 292-294, 1979.

[Kovijanic 1981] P. G. Kovijanic, "Single Testability Figure of Merit," *Proc. Intn'l. Test Conf.*, pp. 521-529, October, 1981.

[Lagemaat and Bleeker 1987] D. Van de Lagemaat and H. Bleeker, "Testing a Board with Boundary-Scan," *Proc. Intn'l. Test Conf.*, pp. 724-729, September, 1987.

[Lien and Breuer 1989] J. C. Lien and M. A. Breuer, "A Universal Test and Maintenance Controller for Modules and Boards," *IEEE Trans. on Industrial Electronics*, Vol. 36, No. 2, pp. 231-240, May, 1989.

[Lioy and Mezzalama 1978] A. Lioy and M. Mezzalama, "On Parameters Affecting ATPG Performance," *Proc. COMPEURO Conf.*, pp. 394-397, May, 1987.

[Lippman and Donn 1979] M. D. Lippman and E. S. Donn, "Design Forethought Promotes Easier Testing of Microcomputer Boards," *Electronics International*, Vol. 52, No. 2, pp. 113-119, January, 1979.

[Ma *et al.* 1988] H.-K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, "An Incomplete Scan Design Approach to Test Generation for Sequential Machines," *Proc. Intn'l. Test Conf.*, pp. 730-734, September, 1988.

[Maunder and Beenker 1987] C. Maunder and F. Beenker, "Boundary Scan: A Framework for Structured Design-for-Test," *Proc. Intn'l. Test Conf.*, pp. 714-723, September, 1987.

[Ratiu *et al.* 1982] I. M. Ratiu, A. Sangiovanni-Vincentelli, and D. O. Peterson, "VICTOR: A Fast VLSI Testability Analysis Program," *Proc. Intn'l. Test Conf.*, pp. 397-401, November, 1982.

[Rutman 1972] R. A. Rutman, "Fault Detection Test Generation for Sequential Logic Heuristic Tree Search," *IEEE Computer Repository Paper* No. R-72-187, 1972.

[Savir 1983] J. Savir, "Good Controllability and Observability Do Not Guarantee Good Testability," *IEEE Trans. on Computers*, Vol. C-32, pp. 1198-1200, December, 1983.

[Stephenson and Grason 1976] J. E. Stephenson and J. Grason, "A Testability Measure for Register Transfer Level Digital Circuits," *Proc. Intn'l. Symp. on Fault-Tolerant Computing*, pp. 101-107, June, 1976.

[Stewart 1977] J. H. Stewart, "Future Testing of Large LSI Circuit Cards," *Digest of Papers 1977 Semiconductor Test Symp.*, pp. 6-15, October, 1977.

[Stewart 1978] J. H. Stewart, "Application of Scan/Set for Error Detection and Diagnostics," *Digest of Papers 1978 Semiconductor Test Conf.*, pp. 152-158, 1978.

[Trischler 1980] E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," *Digest of Papers 1980 Test Conf.*, pp. 153-162, November, 1980.

[Wagner 1987] P. T. Wagner, "Interconnect Testing with Boundary-Scan," *Proc. Intn'l. Test Conf.*, pp. 52-57, September, 1987.

[Whetsel 1988a] L. Whetsel, "A Proposed Standard Test Bus and Boundary-Scan Architecture," *Proc. Intn'l. Conf. on Computer Design*, pp. 330-333, 1988.

[Whetsel 1988b] L. Whetsel, "A View of the JTAG Port and Architecture," *ATE Instrumentation Conf. West*, pp. 385-410, January, 1988.

[Williams and Angell 1973] M. J. Y. Williams and J. B. Angell, "Enhancing Testability of Large Scale Integrated Circuits Via Test Points and Additional Logic," *IEEE Trans. on Computers*, Vol. C-22, pp. 46-60, January, 1973.

[Writer 1975] P. L. Writer, "Design for Testability," *Proc. IEEE Automated Support Systems Conf.*, pp. 84-87, 1975.

PROBLEMS

9.1 Assume that it costs \$1.00 to test an IC chip, \$10.00 to locate a faulty IC chip on a PCB, the test sets provide 100 percent fault coverage, and a board has 100 ICs. Option 1 is to test every chip, and thus all assembled boards have only good ICs. Option 2 is not to test any ICs before assembly but to test and locate faulty ICs at the board level. At what value of IC yield is it cost-effective to switch from option 1 to option 2?

Hint: This is a hard problem. Note that under option 2 PCBs may contain several bad ICs, and a bad IC may be replaced by another bad IC. Make simplifying assumptions if necessary.

9.2 Partition the circuit shown in Figure 9.54 using the partitioning scheme illustrated in Figure 9.11. Attempt to keep the number of inputs to C_1 and C_2 close to the same, and try to minimize the number of signals between the two partitions.

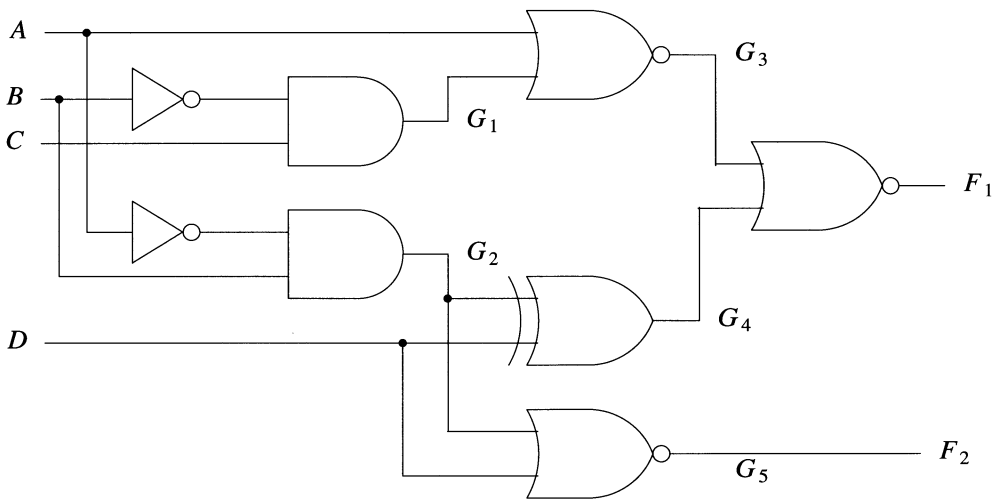


Figure 9.54

9.3 The Am 29818 is a general-purpose pipeline register with parallel load and an on-board shadow register for performing serial shadow register diagnostics and/or writable control store loading. The block diagram for this device is shown in Figure 9.55, and its function table is shown in Figure 9.56. The functional I/Os are defined below. Show how this device can be used in making circuits scan testable. Which generic scan architecture is best suited for implementation by this device, and why?

$D_7 - D_0$ parallel data input to the pipeline register or parallel data output from the shadow register

DCLK diagnostic clock for loading shadow register (serial or parallel modes)

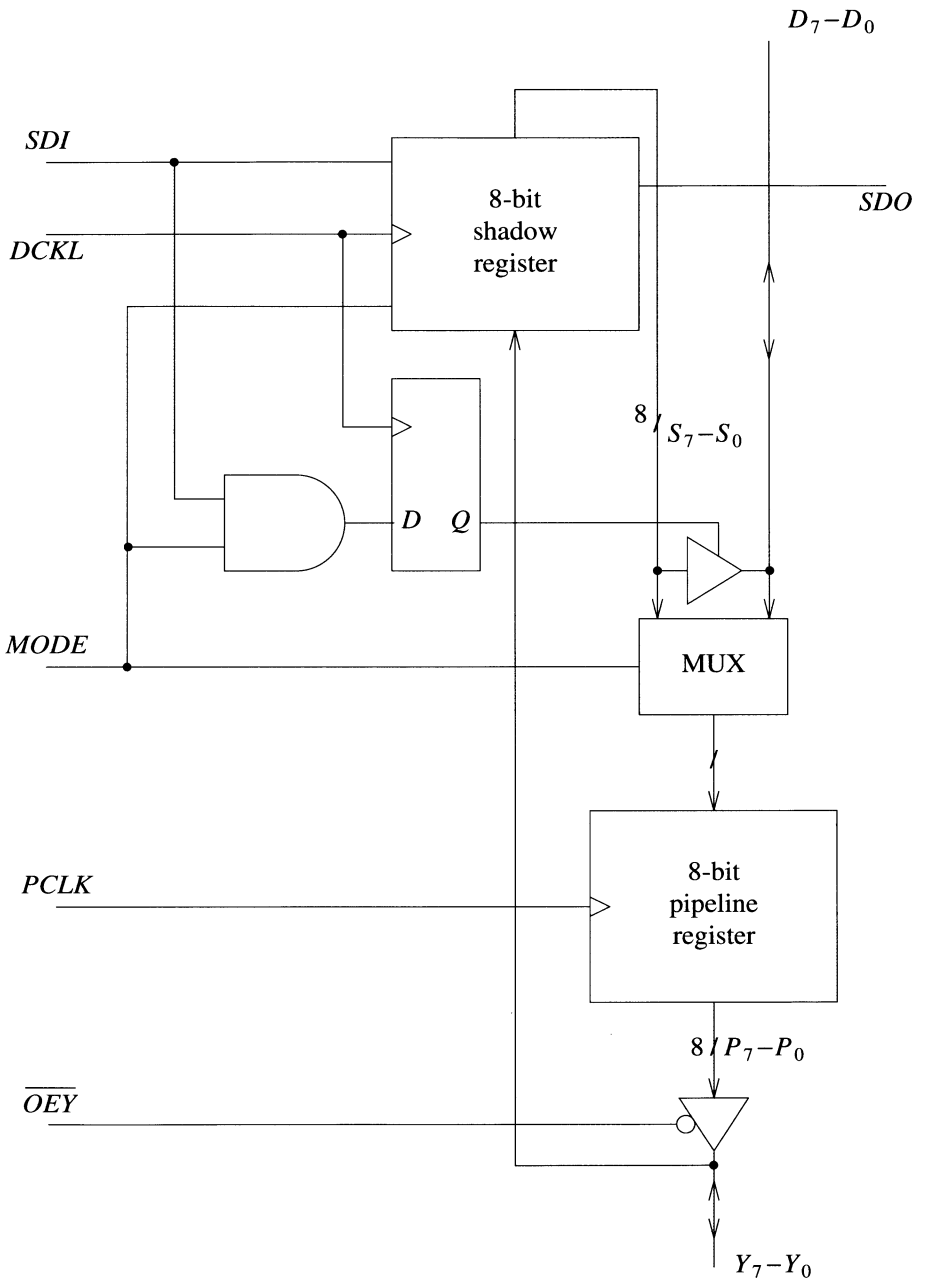


Figure 9.55 The Am 29818

Inputs				Outputs			Operation
SDI	*MODE*	*DCLK*	*PCLK*	*SDO*	Shadow register	Pipeline register	
X	*L*	↑	*X*	S_7	$S_i \leftarrow S_{i-1}$ $S_o \leftarrow SDI$	NA	Serial shift; D_{7-D_0} disabled
X	*L*	*X*	↑	S_7	NA	$P_i \leftarrow D_i$	Normal load pipeline register
L	*H*	↑	*X*	*L*	$S_i \leftarrow Y_i$	NA	Load shadow register from *Y*; $D_7 - D_0$ disabled
X	*H*	*X*	↑	*SDI*	NA	$P_i \leftarrow S_i$	Load pipeline register from shadow register
H	*H*	↑	*X*	*H*	Hold	NA	Hold shadow register; $D_7 - D_0$ enabled

NOTATION

INPUTS

H = high
L = low
X = don't care
↑ = low-to-high clock transition

OUTPUTS

$S_7 - S_o$ = shadow register outputs
$P_7 - P_0$ = pipeline register outputs
$D_7 - D_0$ = data I/O port
$Y_7 - Y_0$ = *Y* I/O port
NA = not applicable; output is not a function of the specified input combinations

Figure 9.56 Am 29818 function table description

MODE control input for pipeline register multiplexer and shadow register control

$\overline{OEY}$ active LOW output enable for *Y*-port

PCLK pipeline register clock input loads *D*-port or shadow register contents on low to high transition

SDI serial data input to shadow register

SDO serial data output from shadow register

$Y_7 - Y_0$ data outputs from the pipeline register and parallel inputs to the shadow register

9.4 Consider an asynchronous circuit having many inputs, outputs, and feedback lines. Assume that during testing, line 1 requires 0-injection only; line 2 requires 1-injection only; line 3 requires both 0 and 1 injection; lines 4 and 5 require observation only.

a. Modify the design so that it has the desired test features. Minimize the amount of hardware you add.

b. Reduce your I/O requirements by sharing functional and test I/O pins.

c. Modify your results to part a. by using a scan register so that observation points and control points do not have to be tied to I/O pins. Assume that during the scan operation the normal state of the circuit should not be disturbed.

9.5 Consider IC1 and IC2, where each IC contains boundary scan registers as shown in Figure 9.18(b). Assume the outputs of IC1 drive the inputs to IC2, and the outputs of IC2 drive the inputs to IC1. Show how the scan registers of these ICs can be interconnected. Describe how the interconnections between these ICs can be tested by means of the boundary scan registers. Explain the control sequencing required for the signal T1 as well as the signal $\overline{N/T}$ that controls the scan registers.

Are these control signals also tied to scan flip-flops? If not, how are these lines tested? If so, does the scan operation interfere with their values?

9.6 Consider a scan register implemented using MD-FF elements as shown in Figure 9.23(b). Show how clock skew can lead to erroneous operation during shift operations.

9.7 Consider a random-access scan architecture. How would you organize the test data to minimize the total test time? Describe a simple heuristic for ordering these data.

9.8 Figure 9.29 shows the double-latch LSSD design. Assume there are 100 SRLs and the circuit requires 1000 test vectors. How many clock cycles are required to test this circuit?

9.9 Consider the single-latch LSSD design shown in Figure 9.30. For each case listed below, determine the best way to organize the testing of the blocks N_1 and N_2 to minimize test time. Let $n = 10$ and $m = 30$.

a. Assume N_1 requires 10 test vectors, and N_2 requires 30.

b. Assume N_1 requires 30 test vectors and N_2 requires only 10.

9.10 Single-latch LSSD designs appear to require a significant amount of overhead with respect to other types of LSSD designs (see Figure 9.33). What advantages, if any, do such designs have over the other designs?

9.11 For each of the circuit configurations shown in Figure 9.57 state whether it satisfies the LSSD design rules and which rules, if any, it violates.

9.12 Assume that the normal function of a register in a large circuit is only to act as a shift register. To make this circuit scan testable by either the LSSD design methodology or the shift register modification technique requires some extra logic and interconnect since both a normal-mode and test-mode shift register will exist in the final design. Comment on the pros and cons of such an approach, and suggest how the normal-mode shift register can be used in the scan mode to reduce hardware overhead. What problems may this create?

9.13 Assume that the normal function of a register (made up of D flip-flops) in a large circuit is both to load data in parallel as well as shift. A multiplexer is used to select the input appropriate to each flip-flop. Show how in a scan design it may be possible to use

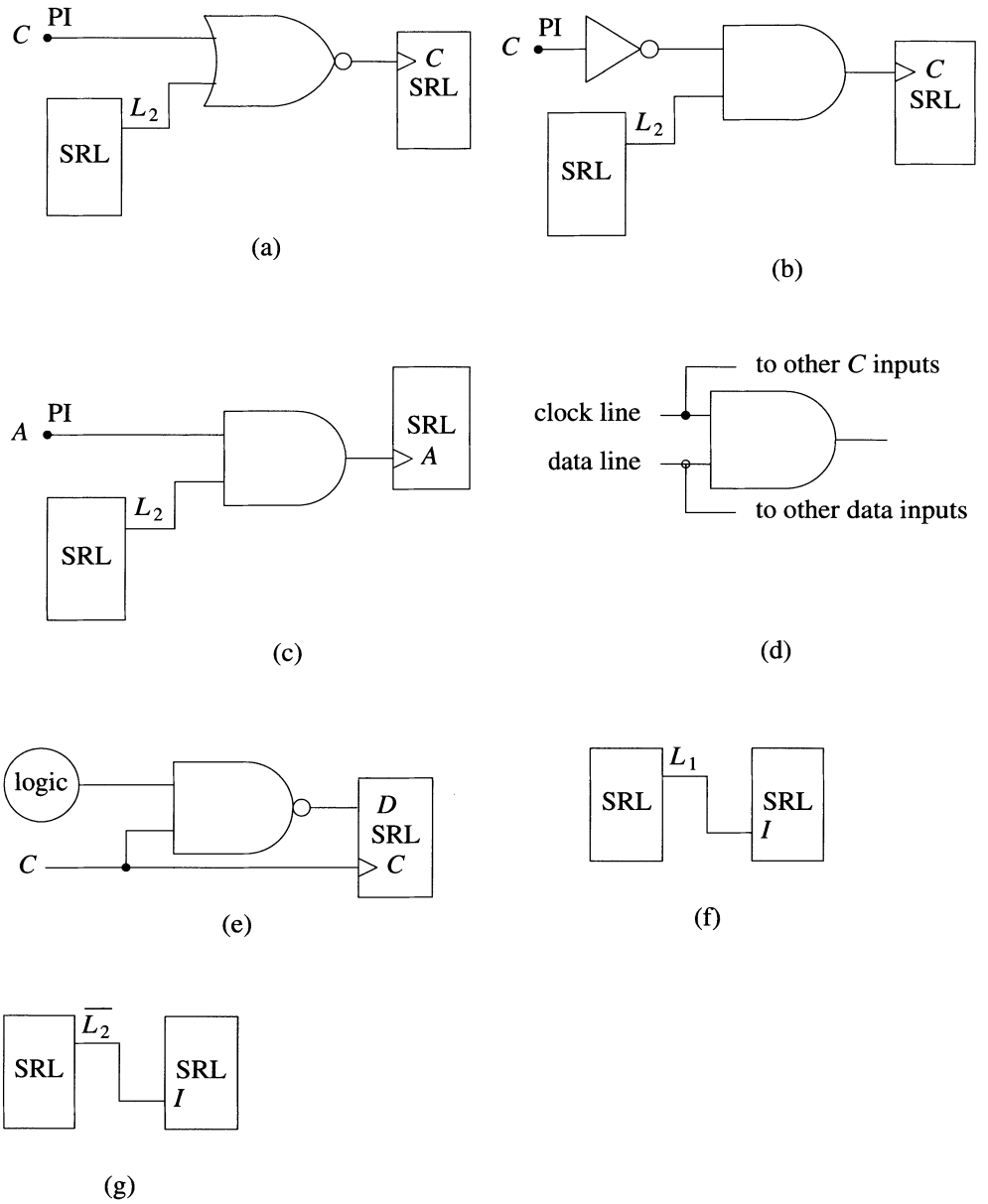


Figure 9.57 Legal and illegal LSSD circuit configurations

the scan operation for both functional and test mode operation and thus eliminate the multiplexers referred to above. What problems may exist because of this dual use of the scan registers?

9.14 Describe and illustrate a linear time algorithm for partitioning a combinational circuit into clouds. Assume no storage cells have been clustered into registers.

9.15 Given a clouded version of a circuit S, describe an algorithm for generating the minimal number of registers from the storage cells consistent with the existing set of clouds.

9.16 Argue for or against the following proposition. Randomly selecting a subset of flip-flops to be made part of a scan path in a partial scan design will reduce ATG cost compared to a nonscan design.

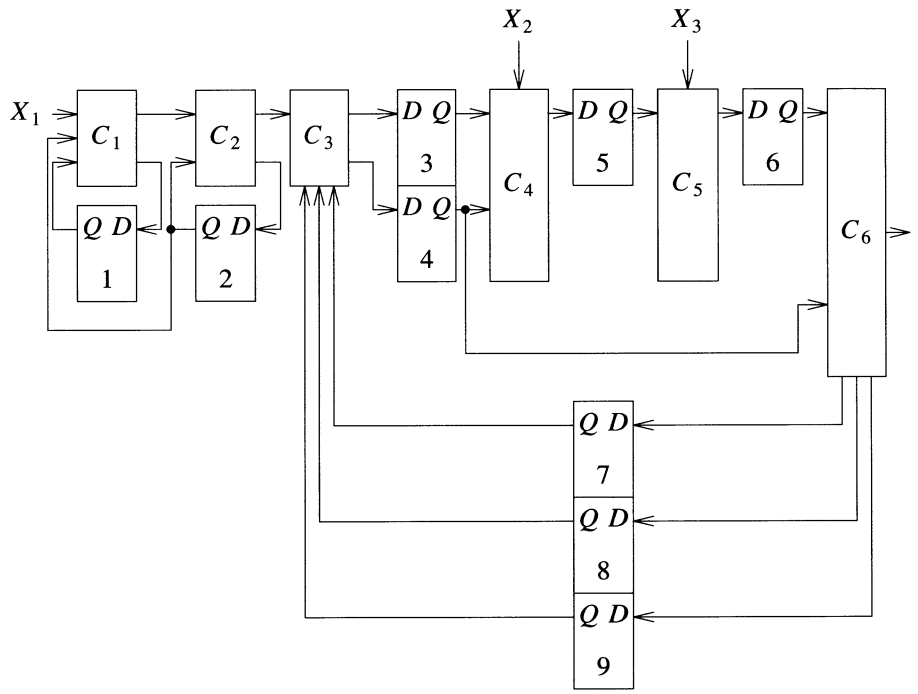


Figure 9.58 Circuit to be made balanced

9.17 Cloud the circuit shown in Figure 9.58, and identify a minimum number of flip-flops to be made part of the scan path so the resulting circuit is a balanced structure.

9.18 Show a design for the boundary-scan cell circuitry for a bidirectional I/O pin in the IEEE 1149.1 methodology.

9.19 Two boundary-scan cells are needed to implement a tristate output pad, one to specify whether or not the pad should be in the high impedance state, and the other to determine the logic value of the pad when it is not in the high impedance state. Indicate the logic associated with such an output pad.

9.20 The following questions deal with IEEE 1149.1.

a) Discuss various applications for the capability to load the instruction register with system status information.

b) Discuss how the on-chip test bus circuitry can be tested.

c) Suggest a format for the information in the instruction register, including fields, their width and functionality.

d) Suggest several uses for an identification test data register.

9.21 Describe a generic test program, consisting of instructions and test data, for use in testing the interconnect between two chips which support IEEE 1149.1. At each step, indicate the level of the *TMS* line and the state of the TAP controller.