CMOS TEST METHODS

7

7.1 The Need for Testing

While in real estate the refrain is "Location! Location! Location!," the comparable advice in IC design should be "Testing! Testing! Testing!" While most problems in VLSI design have been reduced to algorithms in readily available software, the responsibilities for the various levels of testing and testing methodology can be a significant burden on the designer.

In Chapter 4 we noted that the yield of a particular IC was the number of good die divided by the total number of die per wafer. Due to the complexity of the manufacturing process not all die on a wafer correctly operate. Small imperfections in starting material, processing steps, or in photomasking may result in bridged connections or missing features. It is the aim of a test procedure to determine which die are good and should be used in end systems.

Testing a die (chip) can occur:

- at the wafer level.
- at the packaged-chip level.
- at the board level.
- at the system level.
- in the field.

Scanned by CamScanner

By detecting a malfunctioning chip at an earlier level, the manufacturing cost may be kept low. For instance, the approximate cost to a company of detecting a fault at the above levels is[1]:

- wafer $0.01–$.1
- packaged-chip $0.10–$1
- board $1–$10
- system $10–$100
- field $100–$1000.

Obviously, if faults can be detected at the wafer level, the cost of manufacturing is kept the lowest. In some circumstances, the cost to develop adequate tests at the wafer level, mixed signal requirements or speed considerations may require that further testing be done at the packaged-chip level or the board level. A component vendor can only test at the wafer or chip level. Special systems, such as satellite-borne electronics, might be tested exhaustively at the system level.

Tests may fall into two main categories. The first set of tests verifies that the chip performs its intended function; that is, that it performs a digital filtering function, acts as a microprocessor, or communicates using a particular protocol. In other words, these tests assert that all the gates in the chip, acting in concert, achieve a desired function. These tests are usually used early in the design cycle to verify the functionality of the circuit. These will be called *functionality tests* in this book. They may be lumped into the verification activity. The second set of tests verifies that every gate and register in the chip functions correctly. These tests are used after the chip is manufactured to verify that the silicon is intact. They will be called *manufacturing tests* in this book. In many cases these two sets of tests may be one and the same, although the natural flow of design usually has a designer considering function before manufacturing concerns.

It is interesting to note that of most first-time failures of silicon, it is the functionality of the design that is to blame; that is, the chip does exactly what the simulator said it would but for some reason (almost always human error) that function is not what the rest of the system expects.

7.1.1 Functionality Tests

Functionality tests are usually the first tests a designer might construct as part of the design process. Does this adder add? Does this counter count? Does this state-machine yield the right outputs at the right clock cycles?

For most systems, functionality tests involve proving that the circuit is functionally equivalent to some specification. That specification might be a verbal description, a plain-language textual specification, a description in

Scanned by CamScanner

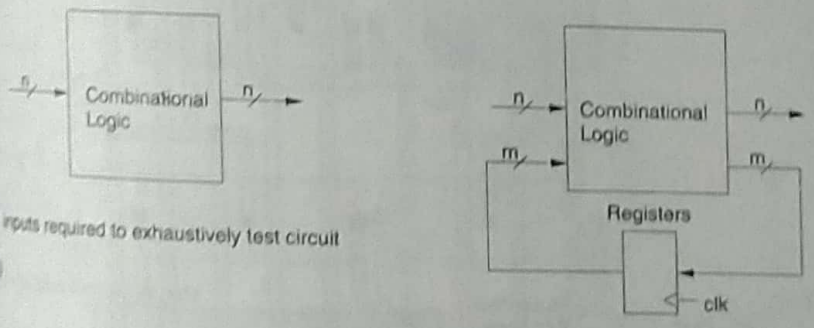
those provided by the test program. If there are any differences, the chip is marked as faulty (with an ink dot) and the miscomparing vectors may be displayed for reference. In the case of a probe card, the card is raised, moved to the next die on the wafer, and lowered, and the test procedure is repeated. In the case of a DUT board with automatic part handling, the tested part is binned into a good or bad bin and a new part is fed to the DUT board, and the test is repeated. In most cases these procedures take a few seconds for each part tested.

The ability to vary the voltage and timing on a per-pin basis with a tester allows a process known as "schmooing" to be carried out. For instance, one might vary the V_{DD} voltage from 3 to 6 volts on a 5-volt part while varying the tester cycle time.This yields a graph that shows the speed sensitivity of the part with respect to voltage. Another "schmoo" test that is frequently exercised is to skew the timing on inputs with respect to the chip clock to look for setup and hold variations.

7.2 Manufacturing Test Principles

A critical factor in all LSI and VLSI design is the need to incorporate methods of testing circuits. This task should proceed concurrently with any architectural considerations and not be left until fabricated parts are available (which is a recurring temptation to designers).

Figure 7.1(a) shows a combinational circuit with n-inputs. To test this circuit exhaustively a sequence of 2^n inputs (or test vectors) must be applied and observed to fully exercise the circuit. This combinational circuit is converted to a sequential circuit with addition of m-storage registers, as shown in Fig. 7.1(b). The state of the circuit is determined by the inputs and the pre-



2^n inputs required to exhaustively test circuit

(a)

2^{n+m} inputs required to exhaustively test circuit

For n = 25 m = 50, 1μS/test, the test time is over 1 billion years (Williams)

(b)

FIGURE 7.1 The combinational explosion in test vectors

Scanned by CamScanner

those provided by the test program. If there are any differences, the chip is marked as faulty (with an ink dot) and the miscomparing vectors may be displayed for reference. In the case of a probe card, the card is raised, moved to the next die on the wafer, and lowered, and the test procedure is repeated. In the case of a DUT board with automatic part handling, the tested part is binned into a good or bad bin and a new part is fed to the DUT board, and the test is repeated. In most cases these procedures take a few seconds for each part tested.

The ability to vary the voltage and timing on a per-pin basis with a tester allows a process known as "schmooing" to be carried out. For instance, one might vary the V_{DD} voltage from 3 to 6 volts on a 5-volt part while varying the tester cycle time.This yields a graph that shows the speed sensitivity of the part with respect to voltage. Another "schmoo" test that is frequently exercised is to skew the timing on inputs with respect to the chip clock to look for setup and hold variations.

7.2 Manufacturing Test Principles

A critical factor in all LSI and VLSI design is the need to incorporate methods of testing circuits. This task should proceed concurrently with any architectural considerations and not be left until fabricated parts are available (which is a recurring temptation to designers).

Figure 7.1(a) shows a combinational circuit with n-inputs. To test this circuit exhaustively a sequence of 2^n inputs (or test vectors) must be applied and observed to fully exercise the circuit. This combinational circuit is converted to a sequential circuit with addition of m-storage registers, as shown in Fig. 7.1(b). The state of the circuit is determined by the inputs and the pre-

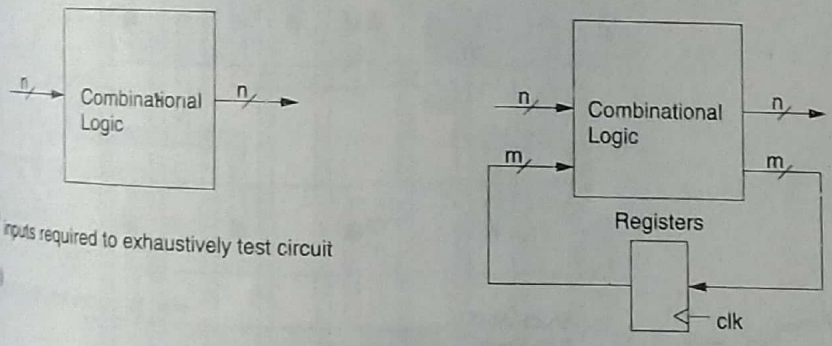


FIGURE 7.1 The combinational explosion in test vectors

Scanned by CamScanner

vious state. A minimum of $2^{(n+m)}$ test vectors must be applied to exhaustively test the circuit. To quote Williams[3]:

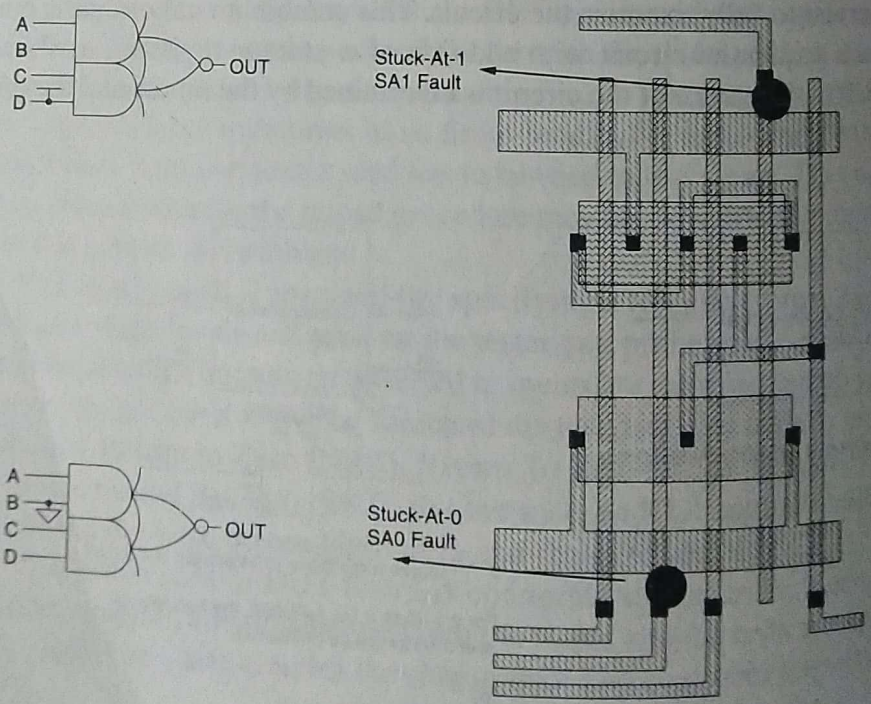
> With LSI, this may be a network with n = 25 and m = 50, or 2^{75} patterns, which is approximately 3.8×10^{22}. Assuming one had the patterns and applied them at an application rate of 1 μs per pattern, the test time would be over a billion years (10^9).

Clearly, this is an important area of design that has to be well understood.

7.2.1 Fault Models

7.2.1.1 Stuck-At Faults

In order to deal with the existence of good and bad parts it is necessary to propose a "fault model," that is, a model for how faults occur and their impact on circuits. The most popular model is called the "Stuck-At" model. With this model, a faulty gate input is modeled as a "stuck at zero" (Stuck-At-0, S-A-0, SA0) or "stuck at one" (Stuck-At-1, S-A-1, SA1). This model dates from board-level designs where this was determined to be an adequate set of models for modeling faults. Figure 7.2 illustrates how an S-A-0 or S-A-1 fault might occur. These faults most frequently occur due to thin-oxide shorts (the n-transistor gate to V_{SS} or the p-transistor gate to V_{DD}) or metal-to-metal shorts.



Scanned by CamScanner

7.2.1.2 Short-Circuit and Open-Circuit Faults

ther models include "stuck-open"[4] or "shorted" models. Two shorted faults e shown in Fig. 7.3. Considering the faults shown in Fig. 7.3, the short *S1* modeled by an S-A-0 fault at input *A*, while short *S2* modifies the function f the gate. What becomes evident is that to ensure the most accurate mod- ing, faults should be modeled at the transistor level, because it is only at is level that the complete circuit structure is known. For instance, in the se of a simple NAND gate, the intermediate node in the series n-pair is idden" by the schematic. What this implies is that test generation must be ne in such a way as to take account of possible shorts and open circuits at e switch level.[5] Although the switch level may be the most appropriate vel, expediency dictates that most existing systems rely on Boolean logic presentations of circuits and S-A-0 and S-A-1 fault modeling.

A particular problem that arises with CMOS is that it is possible for a ult to convert a combinational circuit into a sequential circuit. This is illus- ated for the case of a 2-input NOR gate in which one of the transistors is ndered ineffective (stuck open or stuck closed) in Fig. 7.4. This might be e to a missing source, drain, or gate connection. If one of the n-transistors connected to gate) is stuck open, then the function displayed by the gate ill be

$$F = (not\ (A + B)) + (A\ .\ (not\ B)\ .\ F_n),$$

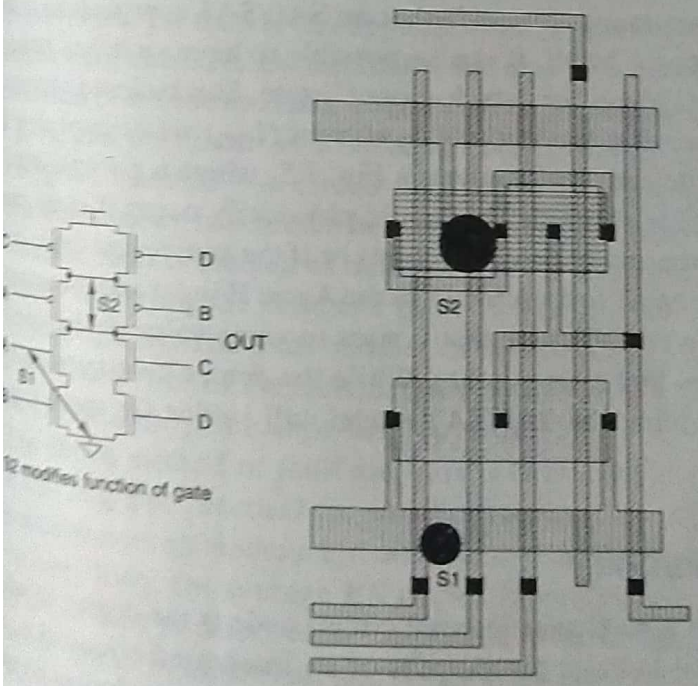


FIGURE 7.3 CMOS bridging faults

Scanned by CamScanner

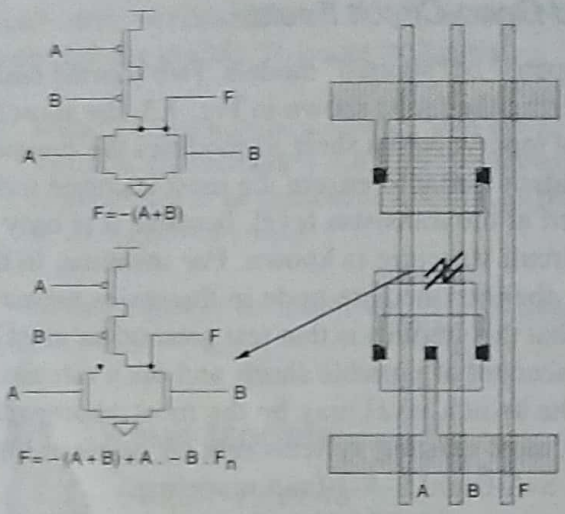


FIGURE 7.4 A CMOS open fault that causes sequential faults

where F_n is the previous state of the gate. Similarly if the B n-transistor drain connection is missing, the function is

$$F = (not\ (A + B)) + ((not\ A)\ .\ B\ .\ F_n).$$

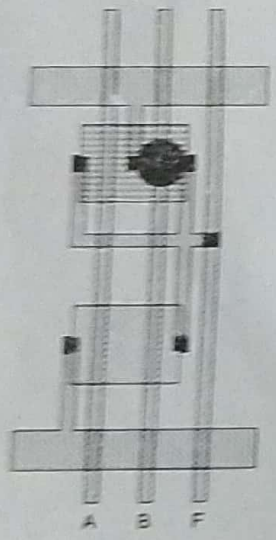


FIGURE 7.5 A defect that causes static I_{DD} current

If either p-transistor is open, the node would be arbitrarily charged (i.e., it might be high due to some weird charging sequence) until one of the n-transistors discharged the node. Thereafter it would remain at zero, bar charge-leakage effects. This problem has caused researchers to search for new methods of test generation to detect such behavior.[6]

Currently debate ranges over whether an SA0/SA1 approach to testing is adequate for testing CMOS. It is also possible to have switches (transistors) exhibit a "stuck-open" or "stuck-closed" state. Stuck-closed states can be detected by observing the static V_{DD} current (I_{DD}) while applying test vectors. Consider the gate fault shown in Fig. 7.5, where a p-transistor in a 2-input NAND gate is shorted. This could physically occur if stray metal overlapped the source and drain connections or if the source and drain diffusions shorted. If we apply test vector 11 to the A and B input and measure the static I_{DD} current, we will notice that it rises to some value determined by the β ratios of the n- and p-transistors. While the debate continues and test cycles are at a premium, the SA0/SA1 model will suffice for some time to come.

7.2.2 Observability

The observability of a particular internal circuit node is the degree to which one can observe that node at the outputs of an integrated circuit (i.e., the pins). This measure is of importance when a designer/tester desires to measure the output of a gate within a larger circuit to check that it operates cor-

Scanned by CamScanner

rectly. Given a limited number of nodes that may be directly observed, it is the aim of well-designed chips to have easily observed gate outputs, and the adoption of some basic test design techniques can aid tremendously in this respect. Ideally, one should be able to observe directly or with moderate indirection (i.e., one may have to wait a few cycles) every gate output within an integrated circuit. While at one time this aim was hindered by limited gate-count processes and a lack of design methodology, current design practices and processes allow one to approach this ideal. Section 7.3 examines a range of methods for increasing observability.

7.2.3 Controllability

The controllability of an internal circuit node within a chip is a measure of the ease of setting the node to a 1 or 0 state. This measure is of importance when assessing the degree of difficulty of testing a particular signal within a circuit. An easily controllable node would be directly settable via an input pad. A node with little controllability might require many hundreds or thousands of cycles to get it to the right state. Often one finds it impossible to generate a test sequence to set a number of poorly controllable nodes into the right state. It should be the aim of a well-designed circuit to have all nodes easily controllable. In common with observability, the adoption of some simple design for test techniques can aid tremendously in this respect.

7.2.4 Fault Coverage

A measure of goodness of a test program is the amount of fault coverage it achieves; that is, for the vectors applied, what percentage of the chip's internal nodes were checked. Conceptually, the way in which the fault coverage is calculated is as follows. Each circuit node is taken in sequence and held to 0 (S-A-0), and the circuit is simulated, comparing the chip outputs with a known "good machine"—a circuit with no nodes artificially set to 0 (or 1). When a discrepancy is detected between the "faulty machine" and the good machine, the fault is marked as detected and the simulation is stopped. This is repeated for setting the node to 1 (S-A-1). In turn, every node is stuck at 1 and 0, sequentially. The total number of nodes that, when set to 0 or 1, do result in the detection of the fault, divided by the total number of nodes in the circuit, is called the percentage-fault coverage.

The above method of fault analysis is called sequential fault grading. While this might be practical for small circuits, or by using hardware simulation accelerators on medium circuits, the time to complete the fault grading may be very long. On average KN cycles (assuming that, on average, N/2 cycles are needed to detect each fault) need to be simulated, where K is the number of nodes in the circuit and N is the length of the test sequence. For $K = 1000$ and $N = 12{,}000$, 12 million cycles are required. At 1 ms per cycle, this yields 12,000 seconds or 3 hrs 20 minutes. For $K = 100{,}000$ and

Scanned by CamScanner

$N = 360{,}000$, 3.6×10^9 cycles are required. At 1 s per cycle, 1040 years would be required to do sequential fault grading.

To overcome these long simulation times many ingenious techniques have been invented to deal with fault simulation.

Scanned by CamScanner

tistically inferred from the number of faults that are detected in the fault set and the size of the set. As with all probabalistic methods it is important that the randomly selected faults be unbiased. Although this approach does not yield a specific level of fault coverage, it will determine whether the fault coverage exceeds a desired level. The level of confidence may be increased by increasing the number of samples.

7.3 Design Strategies for Test

7.3.1 Design for Testability

The key to designing circuits that are testable are the two concepts that we have introduced called controllability and observability. Restated, controllability is the ability to set (to 1) and reset (to 0) every node internal to the circuit. Observability is the ability to observe either directly or indirectly the state of any node in the circuit.

We will first cover three main approaches to what is commonly called *Design for Testability*. These may be categorized as:

- ad-hoc testing.
- scan-based approaches.
- self-test and built-in testing.

Following this we will look at the application of these techniques to particular types of circuits. In this treatment we will look at:

- random logic (multilevel standard-cell, two-level PLA).
- regular logic arrays (datapaths).
- memories (RAM, ROM).

7.3.2 Ad-Hoc Testing

Ad-hoc test techniques, as their name suggests, are collections of ideas aimed at reducing the combinational explosion of testing. Common techniques involve:

- partitioning large sequential circuits.
- adding test points.
- adding multiplexers.
- providing for easy state reset.

Scanned by CamScanner

Long counters are good examples of circuits that can be tested by ad-hoc techniques. For instance imagine you have designed an 8-bit counter and want to test it. Figure 7.11(a) shows a naive implementation in which the counter only has a *RESET* and a *CLOCK* input, with the terminal count (*TC*) being observable. The designer probably thought that a reset and 256 clock cycles, followed by the observation of *TC*, would be adequate for testing purposes. Apart from the nonobservability of the count value (*Q*<7:0>), the

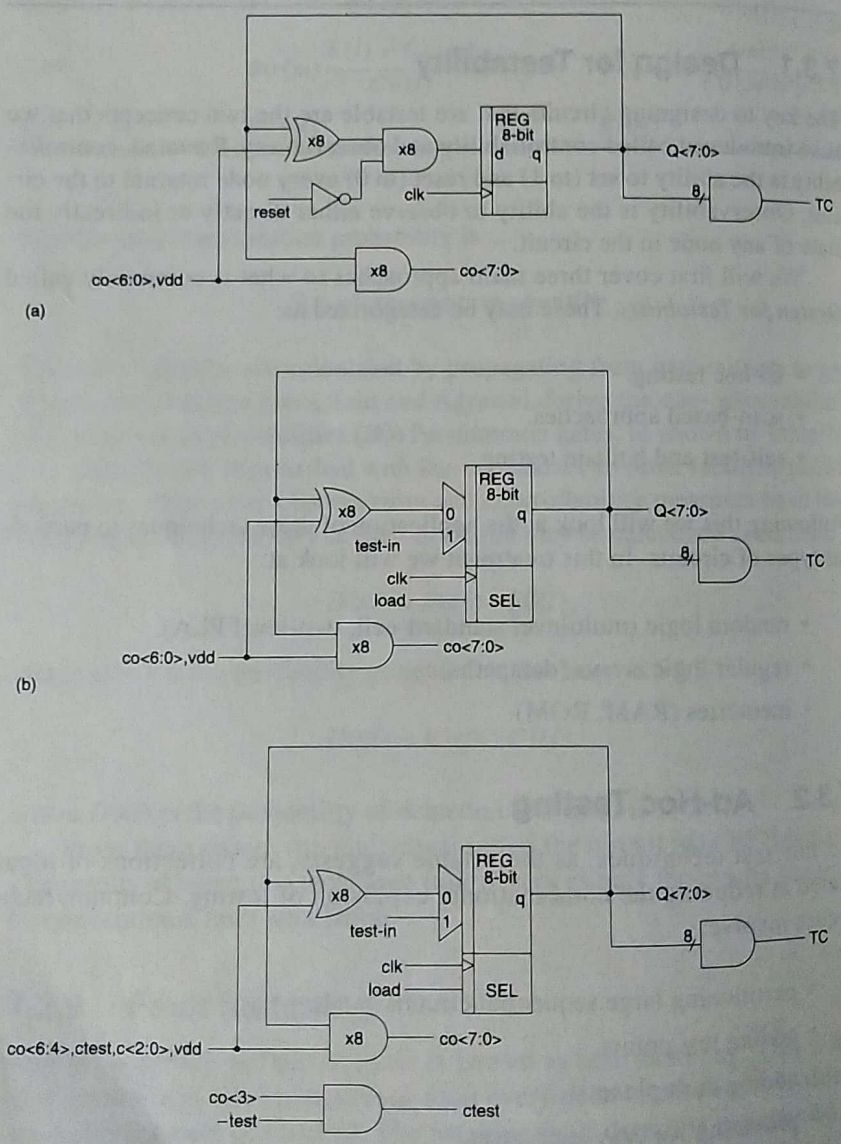


FIGURE 7.11 Ad-hoc test techniques applied to a counter

Scanned by CamScanner

main problem is the number of cycles required to test a single counter. Possible ad-hoc test techniques are shown in Fig. 7.11(b) and Fig. 7.11(c). In Fig. 7.11(b), a parallel-load feature is added to the counter. This enables the counter to be preloaded with appropriate values to check the carry propagation within the counter. Another technique is to reduce the length of each counter to, say, 4 bits, as shown in Fig. 7.11(c). This is achieved by having the test signal block the carry propagate at every 4-bit boundary. With this method 16 vectors exhaustively can test each 4-bit section. The carry propagate between 4-bit sections may be tested with a few additional vectors.

Another technique classified in this category is the use of the bus in a bus-oriented system for test purposes. This is shown on Fig. 7.12(a) for a very simple accumulator. Each register has been made loadable from the bus and capable of being driven onto the bus. Here the internal logic values that exist on a data bus are enabled onto the bus for testing purposes. A more general scheme is illustrated in Fig. 7.12(b), where the normally inaccessible inputs are set and the outputs are observed via the bus.

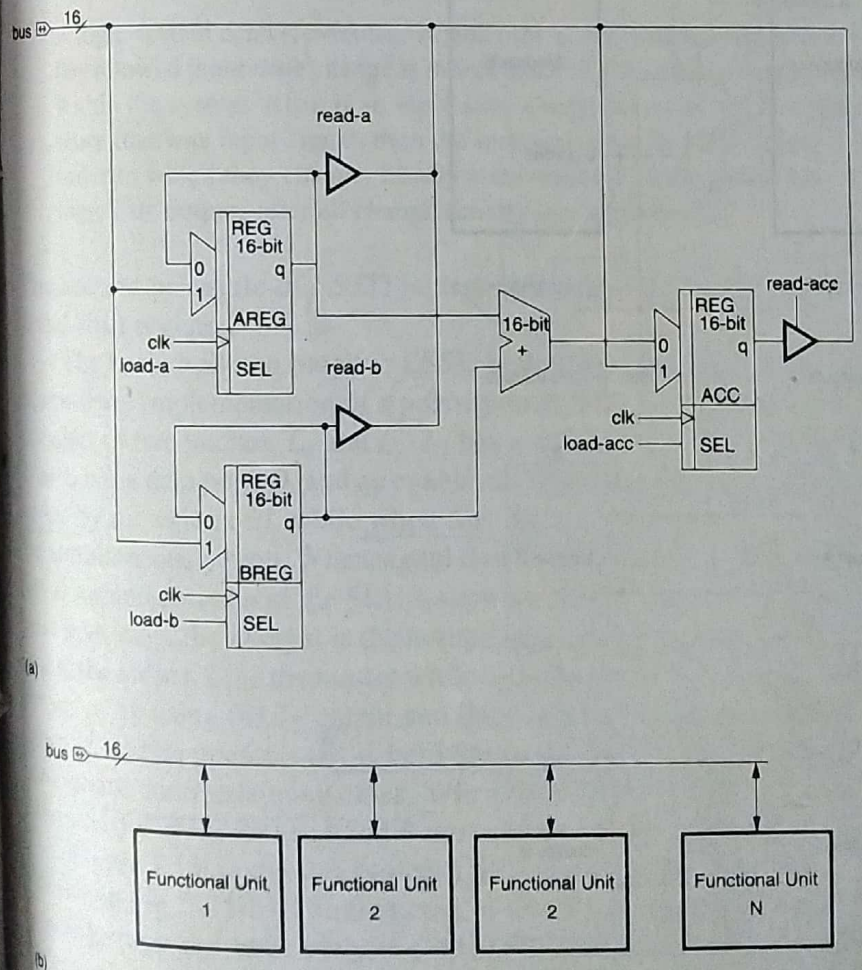


FIGURE 7.12 Bus-oriented test techniques

Scanned by CamScanner

Frequently, multiplexers may be used to provide alternative signal paths during testing. In CMOS, transmission gate multiplexers provide low area and speed overhead. Figure 7.13(a) shows a scheme called a Design for Autonomous Test[23], which uses multiplexers. Figure 7.13(b) shows the circuit configured for normal use, while Fig. 7.13(c) shows the circuit configured to test module A.

Any design should always have a method of resetting the internal state of the chip within a single cycle or at most a few cycles. Apart from making testing easier, this also makes simulation faster because a few cycles are required to initialize the chip.

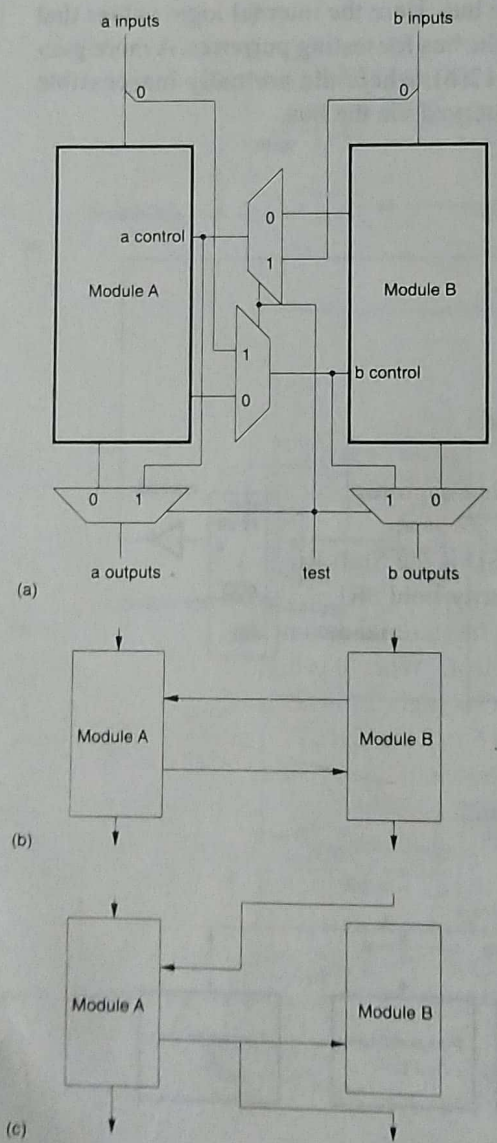


FIGURE 7.13 Multiplexer based testing

Q44. Explain about Testing and Fault Simulation in VLSI Design.

April-11, Set-2, Q3

Ans:

Testing

For answer refer Unit-V, Q36.

Fault Simulation

For answer refer Unit-V, Q42.

5.2.3 Design Strategies for Test, Chip Level Test Techniques

Q45. What are the different categories of DFT techniques? Explain.

April-11, Set-4, Q8(a)

Ans:

Design for testability normally fall into three general categories. They are,

(i) Adhoc design methods
(ii) Structured design methods
(iii) Self -test.

(i) **Adhoc Design Methods**

Adhoc design methods are used to improve testability of specific circuits. The best way of achieving this is, to incorporate additional control and observation points in a circuit.

The commonly used techniques for Ad-Hoc testing are,

1. Partitioning large sequential circuits
2. Adding test points
3. Adding multiplexers
4. Providing for easy state reset.

(ii) **Structured Design Methods**

Structured design method incorporates serial scan and parallel scan.

For answer refer Unit-V, Q45, Topics: Serial Scan, Parallel Scan.

(iii) **Self-test**

For answer refer Unit-V, Q46.

Q45. Explain the Scan-based Test Techniques.

Ans:

There are three types of Scan-based Techniques. They are as follows,

1. Level Sensitive Scan Design (LSSD)
2. Serial Scan
3. Parallel Scan.

1. **Level Sensitive Scan Design (LSSD)**

The most widely used Scan-based test technique is level sensitive scan design, or the LSSD. This is based on two principles. First that the circuit is level sensitive and the second principle of LSSD is that each register may be converted to a serial shift register.

From the first principle, any logic system is level sensitive if and only if it satisfies the following rule,

(a) It steady state response of any allowed input change is independent of circuit delays.

(b) The response is independent of the order in changing the inputs.

The block level and gate level implementation of Shift Register Latch (SRL) is shown in figures (a), (b) and (c) respectively. The block level arrangement consists of,

(a) Two latches L_1 and L_2 where L_1 consists of serial data port, enable, data port and enable C.

(b) L_2 consists of data port D and signal B.

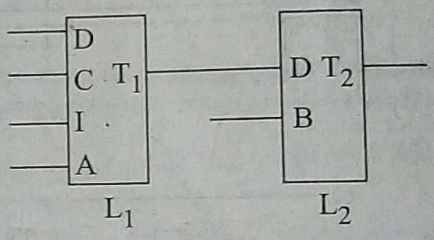
If enable 'A' is high value of T_1 is high. If L_1 is set by D, then 'C' becomes high. It is not possible for 'A' and 'C' at a time become high.

If signal 'B' in 'L_2' is high, then T_1 is transmitted to T_2.

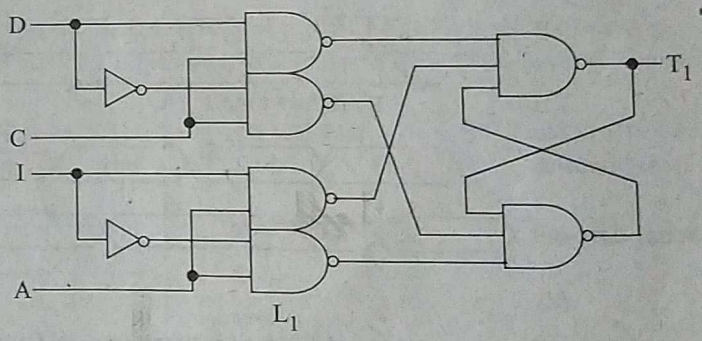
In gate level approach, 'D' is normal input to register and T_2 is the output signal while 'L_1' and 'L_2' acts as master and slave respectively.

The typical and expanded view of a LSSD system is shown in figures 2 (a) and 2(b) respectively. In these, the inputs are taken from preceding stages and outcomes are QA1, QA2 and QA3. These outcomes are connected to logic circuits. These logic circuit outcomes are connected to second rank of SRL. The outcomes of these are QB1 and QB2 and QB3.

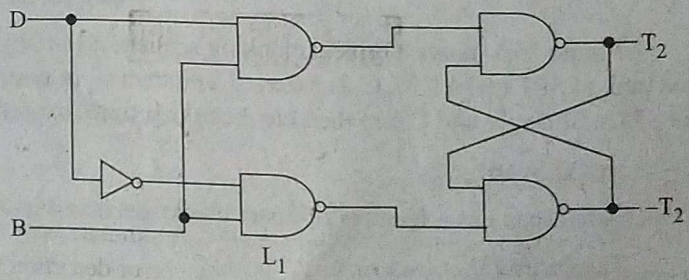
In this testing process, serial data-in is clocking to right point which is controlled and serial data out is for observation. Hence its concluded that input is controlled and output is observed.



(a)



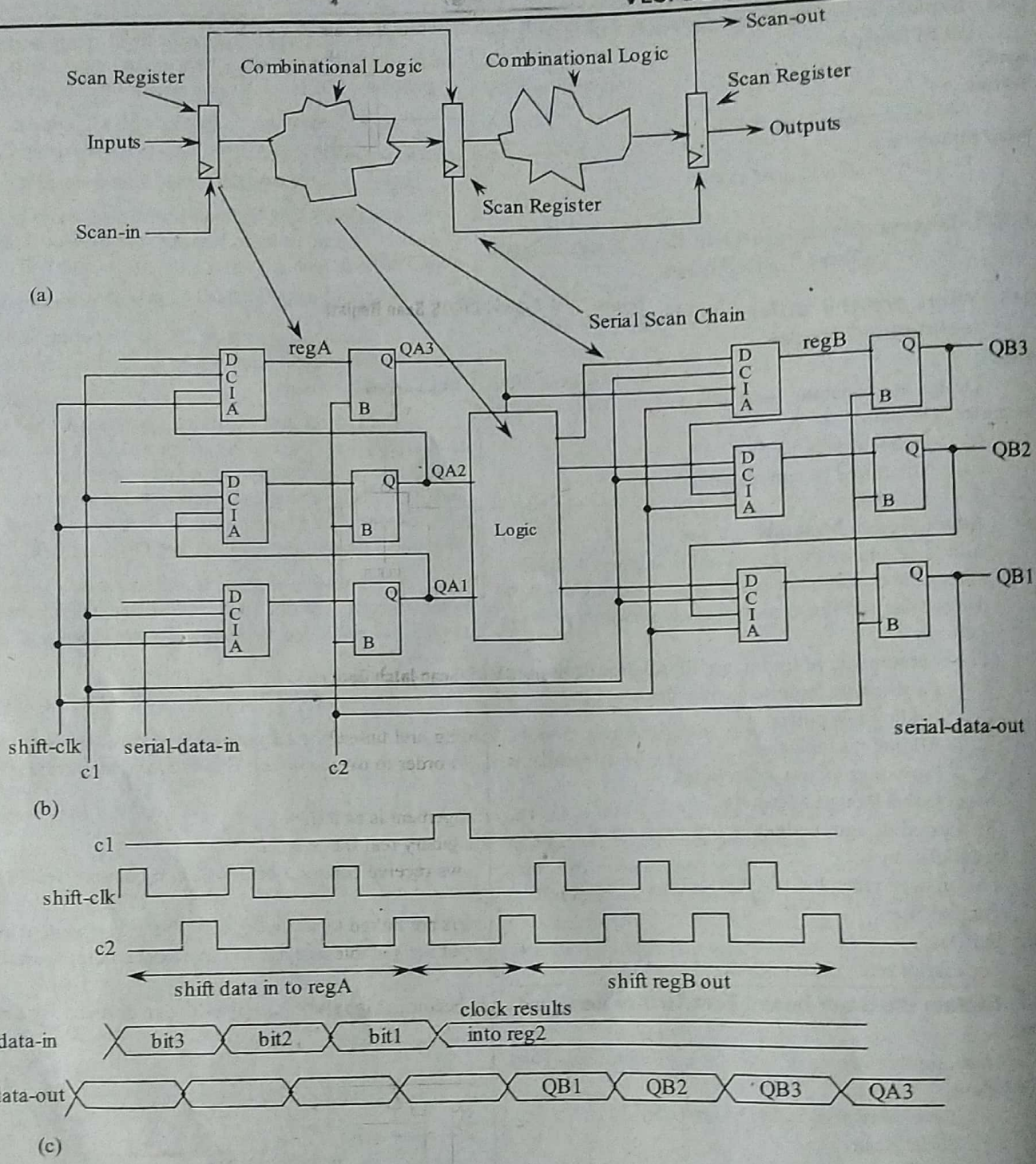
(b)



(c)

Figure (1): A Shift Register Latch

Scanned by CamScanner



(a) Basic Architecture ; (b) Example Circuit ; (c) Example Timing

Figure (2): An LSSD Scan Chain

Figure 2(c) shows a typical clocking sequence. Initially the shift-clk and C_2 are clocked three times to shift data in to the first rank of SRLs (QA1-3). C_1 is asserted, and then C_2 is asserted, clocking the output of the logic block into the second rank of the SRLs. Shift-clk and C_2 are then clocked three times to shift QB1, QB2, and QB3 out via the serial - data - outline.

2. Serial Scan

Serial scan has a features of faster clock speed and small registers structure which is shown in figure (3).

The figure (3) consists of,

(a) Multiplexer which consists of D register, Test Input pin (TI) and Test Enable pin (TE).

(b) If TE is high, the TI undergoes into rising edge clock.

Look for the SIA GROUP LOGO on the TITLE COVER before you buy

Scanned by CamScanner

The following diagrams shows various implementations of circuit diagram of CMOS SRL implementations.

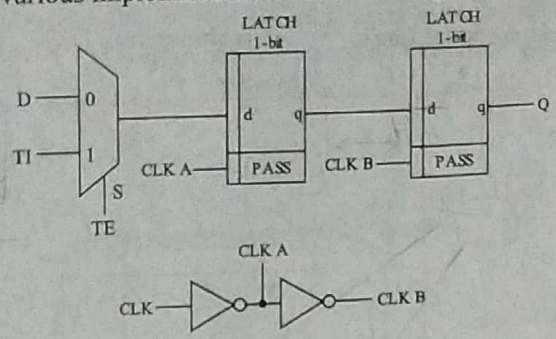


Figure (3): A Typical CMOS Scan Register

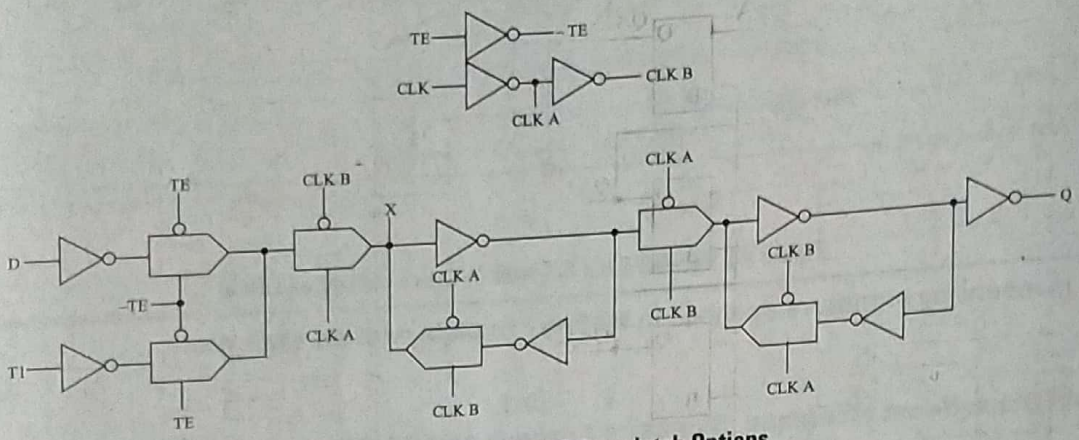


Figure (4): Various CMOS Scan-latch Options

3. Parallel Scan

The chains of serial-scan become quite long, and the loading and unloading sequence can dominate testing time. An extension of serial scan is called random-access or parallel scan. In order to overcome this disadvantage by reducing the long scan chains, we can use parallel scan.

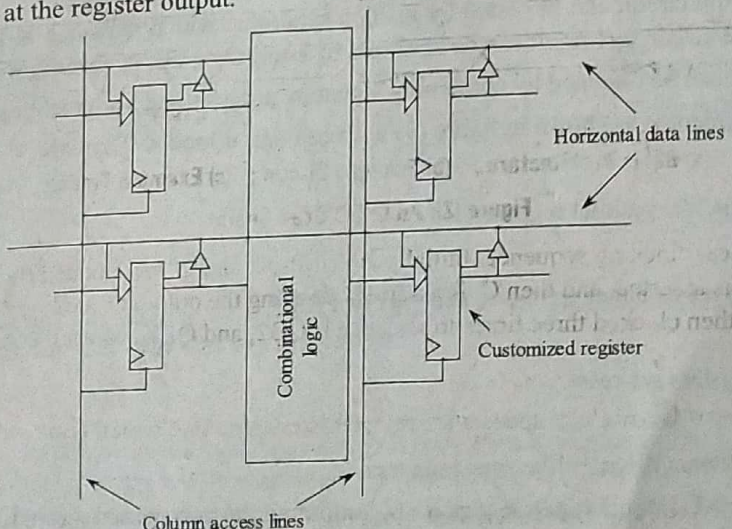
Figure (5) shows basic structure of a parallel scan. The arrangement is as follows,

(a) Every individual register in design is packaged on imaginary real frame work.

(b) With in these registers, the registers with common rows receive common data lines and common read and write control signals are received by the common columns.

From the figure, it is clear that the D and Q signals of registers are linked to normal circuit. The observation of a register output can be possible by enable the porper column read line and put the suitable address on output data multiplexer. The data also written to any register correspondingly.

Similarly, if test-write enable is low or 0. Probe [J] and CLK becomes high and sense [i] value can be driven towards node 'y'. These changes are visible at the register output.



Figure(5): Parallel Scan-basic Structure

Scanned by CamScanner

Figure (6) shows a D register implementation called a cross-controlled catch. It consists of a normal CMOS master-slave edge triggered register augmented by two small n-transistors, N_1, and N_2. When-test-write-enable is high, probe[j] is high, and CLK is low, the value of node Y(D) may be sensed on sense[i] via transistor N_2.

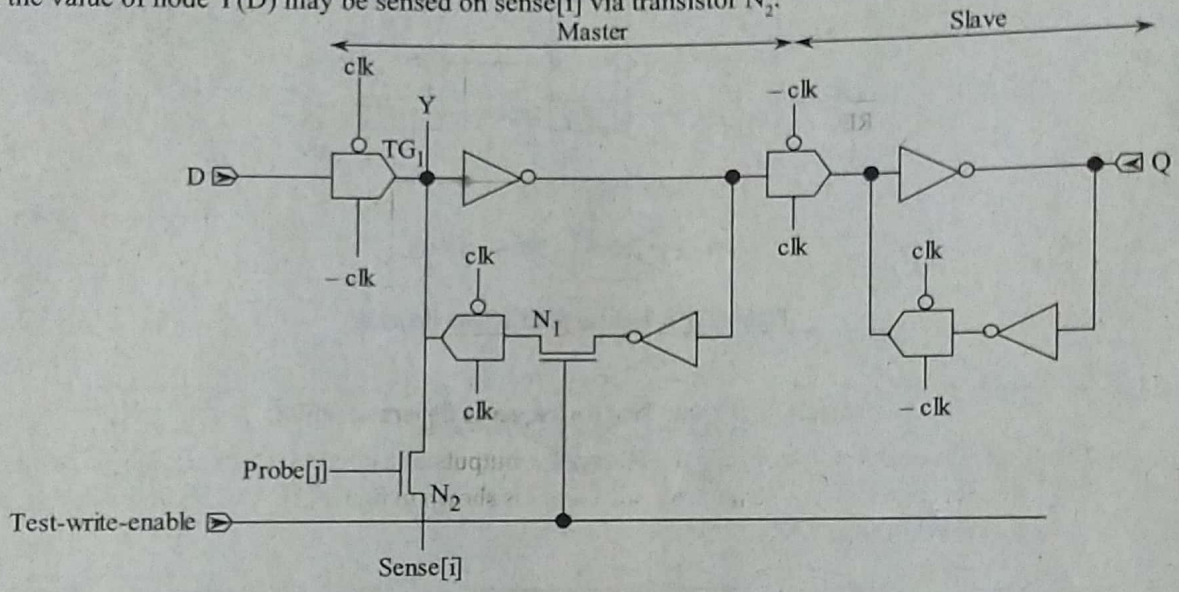


Figure (6): Parallel Scan Register (A Cross-controlled Latch)

Scanned by CamScanner

te-enable is low,
an be driven onto
er. The net effect
ors is to slightly
act for an ASIC-

ry register in the
7.3.4.1). The large
t-fault simulation.

7.3.4 Self-Test Techniques

Self-test and built-in test techniques, as their names suggest, rely on augmenting circuits to allow them to perform operations on themselves that prove correct operation.

7.3.4.1 *Signature Analysis and BILBO*

One method of incorporating a built-in test module is to use signature analysis[32,33] or cyclic-redundancy checking. This involves the use of a pseudo–

Scanned by CamScanner

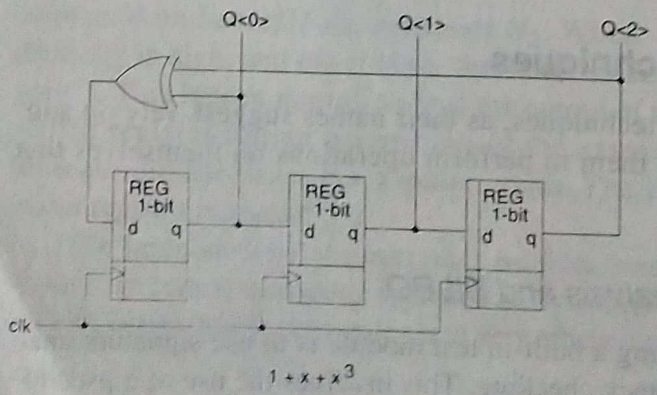
random sequence generator (PRSG) to generate the input signals for a section of combinational circuitry and then using a signature analyzer to observe the output signals.

A PRSG implements a polynomial of some length N. It is constructed from a linear feedback shift register (LFSR), which is constructed, in turn, from a number of 1-bit registers connected in a serial fashion, as shown in Fig. 7.21. The outputs of certain shift bits are XORed and fed back to the input of the LFSR to calculate the required polynomial. For instance, in Fig. 7.21, the 3-bit shift register is computing the polynomial $f(x) = 1 + x + x^3$. For an n-bit LFSR, the output will cycle through $2^n - 1$ states before repeating the sequence. Tables for determining suitable shift registers may be found in Golumb.[34] A complete feedback shift register (CFSR) includes the zero state, which may be required in some test situations. Methods for designing these may be found in Wang and McCluskey.[35]

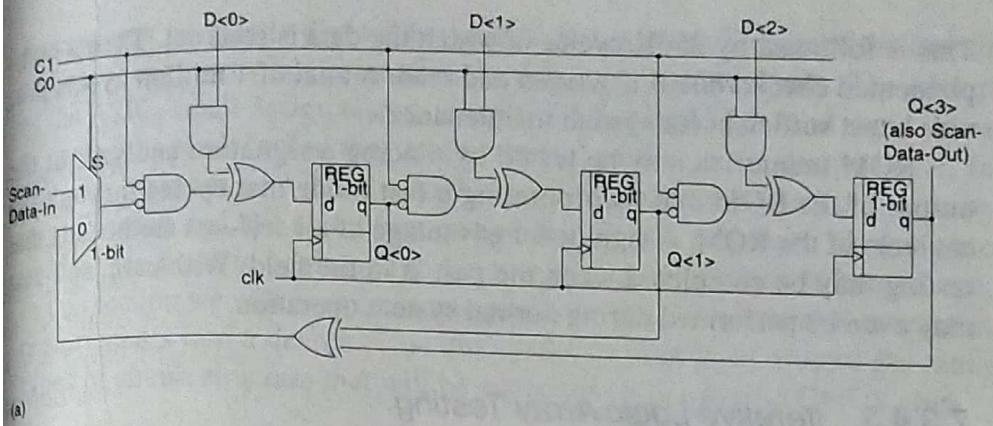
A signature analyzer is constructed by cyclically adding the outputs of a circuit to a shift register or an LFSR if successive logic blocks are to be tested in a like manner. A typical circuit is shown in Fig. 7.22(a). As each test vector is run, the incoming data is XORed with the contents of the LFSR. At the end of a test sequence, the LFSR contains a number, known as the *syndrome*, which is a function of the current output and all previous outputs. This can be compared with the correct syndrome (derived by running a test program on the good logic) to determine whether the circuit is good or bad.

Signature analysis can be merged with the scan technique to create a structure known as BILBO—for Built-In Logic Block Observation.[36]

A 3-bit register is shown with the associated circuitry. In mode D ($C0 = C1 = 1$), the registers act as conventional parallel registers. In mode A ($C0 = C1 = 0$), the registers act as scan registers. In mode C ($C0 = 1$, $C1 = 0$), the registers act as a signature analyzer or pseudo–random sequence generator (PRSG). The registers are reset if $C0 = 0$ and $C1 = 1$. Thus a complete test-generation and observation arrangement can be implemented, as shown in Fig. 7.22(b). In this case two sets of registers have been added in addition to some random logic to effect the test structure.

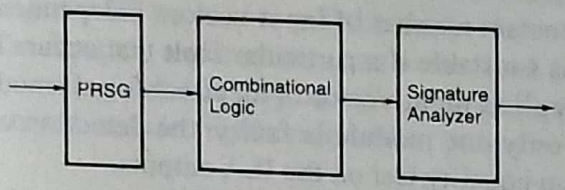


7.21 Pseudo–sequence generator



(a)

MODE	C_0	C_1	
A	0	0	Scan Mode
B	0	1	Reset
C	1	0	PRSG or Signature Analyzer
D	1	1	Parallel Registers



(b)

FIGURE 7.22 Built-in logic block observation (BILBO): (a) individual register; (b) use in a system

A chip set for FFT applications was designed with local testing based on pseudo–random pattern generation and signature analysis.[37] With a 28-bit pattern generator and a 17-bit signature at 10 MHz it took 26 seconds to test the part.

7.3.4.2 *Memory Self-Test*

Embedding self-test circuits for memories in higher-speed circuits not only may be the way of testing the structures at speed but can save on the number of external test vectors that have to be run. A typical read/write memory (RAM) test program for an M-bit address memory might be as follows[38,39]:

```
FOR i=0 to M-1 write( data )
FOR i=0 to M-1 read( data ) then write(data)
FOR i=0 to M-1 read(data) then write( data )
FOR i=M-1 to 0 read( data ) then write(data)
FOR i=M-1 to 0 read(data) then write( data )
```

data is 1 and $\overline{\texttt{data}}$ is 0 for a 1-bit memory or a selected set of patterns for an n-bit word. For an 8-bit memory data, might be x00, x55, x33, and x0F. An address counter, some multiplexers, and a simple-state machine result in a fairly low overhead self-test structure for read/write memories. Oshawa et al.[40] describe a 4-Mbit RAM with self-test. The self-test consists of 256K cycles that input a checkerboard pattern to test for cell-to-cell interference.

Scanned by CamScanner

This is followed by 256K cycles in which the data is read out. Then a complemented checkerboard is written and read. A total of 1 million cycles provide a test sufficient for system maintenance.

ROM memories may be tested by placing a signature analyzer at the output of the ROM and incorporating a test mode that cycles through the contents of the ROM. A significant advantage of all self-test methods is that testing may be completed when the part is in the field. With care, self-test may even be performed during normal system operation.

7.3.4.3 *Iterative Logic Array Testing*

Arrays of logic[41,42] present an interesting problem to the test architect because the replication can be used to advantage in reducing the number of tests. In addition, by augmenting the logic extremely high fault coverage rates are possible. An iterative logic array (ILA) is a collection of identical logic modules (such as an n-bit adder). An ILA is C-testable if it can be tested with a constant number of input vectors independent of the iteration count. An ILA is I-testable if a particular fault that occurs in any module as a result of an applied input vector is identical for all modules in the ILA. Assuming that only one module is faulty, the detection of a fault may be made by using an equality test on the ILA outputs.

7.3.5 IDDQ Testing

An increasingly popular method of testing for bridging faults is called IDDQ (V_{DD} supply current Quiescent) or current-supply monitoring.[43,44] This relies on the fact that when a complementary CMOS logic gate is not switching, it draws no DC current (except for leakage). When a bridging fault occurs, for some combination of input conditions a measurable DC I_{DD} will flow. Testing consists of applying the normal vectors, allowing the signals to settle, and then measuring I_{DD}. To be effective any circuits that draw DC power such as pseudo-nMOS gates or analog circuits have to be disabled. Because many circuits now require SLEEP modes to reduce power, this may not be a substantial additional overhead.

Because current measuring is slow, the tests must be run slower than normal, thus increasing the test time. However, this technique gives a form of indirect massive observability at little circuit overhead.

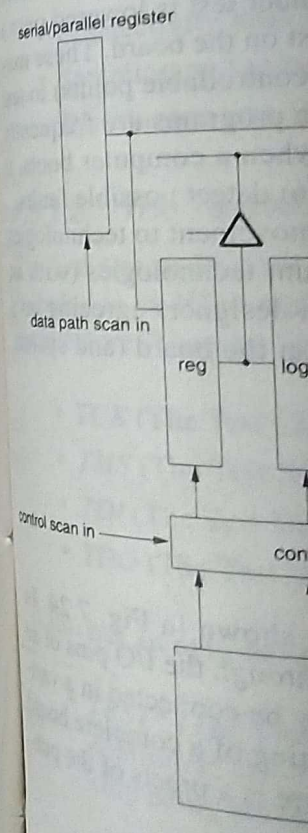
7.4 Chip-Level Test Techniques

In this chapter we have discussed the principles behind testing ICs, and covered some techniques aimed at making testing easier. In the past the design

process was frequently divide
and a test engineer who design
of the ASIC, small design tear
to market have all forced the
fact, the designer who is only
implemented and not about
product deadlines to be slippe
In this section we will examin
requirements into a design.
types of circuit structure that

7.4.1 Regular Logi

Partial serial scan or paralle
tures such as datapaths. One
cessor is shown in Fig. 7.2
serially loaded register. The
path registers. The datapath
may be loaded into a registe
signals to the datapath are a



computing the polynomial $f(x) = 1 + x + x^3$.
TITLE COVER before you buy

Scanned by CamScanner

process was frequently divided between a designer who designed the circuit and a test engineer who designed the test to apply to that circuit. The advent of the ASIC, small design teams, the desire for reliable ICs, and rapid times to market have all forced the "test problem" earlier in the design cycle. In fact, the designer who is only thinking about what functionality has to be implemented and not about how to test the circuit will quite likely cause product deadlines to be slipped and in extreme cases products to be stillborn. In this section we will examine some practical methods of incorporating test requirements into a design. This discussion is structured around the main types of circuit structure that will be encountered in a digital CMOS chip.

7.4.1 Regular Logic Arrays

Partial serial scan or parallel scan is probably the best approach for structures such as datapaths. One approach that has been used in a Lisp microprocessor is shown in Fig. 7.23.[45] Here the input busses may be driven by a serially loaded register. These in turn may be used to load the internal datapath registers. The datapath registers may be sourced onto a bus, and this bus may be loaded into a register that may be serially accessed. All of the control signals to the datapath are also made scannable.

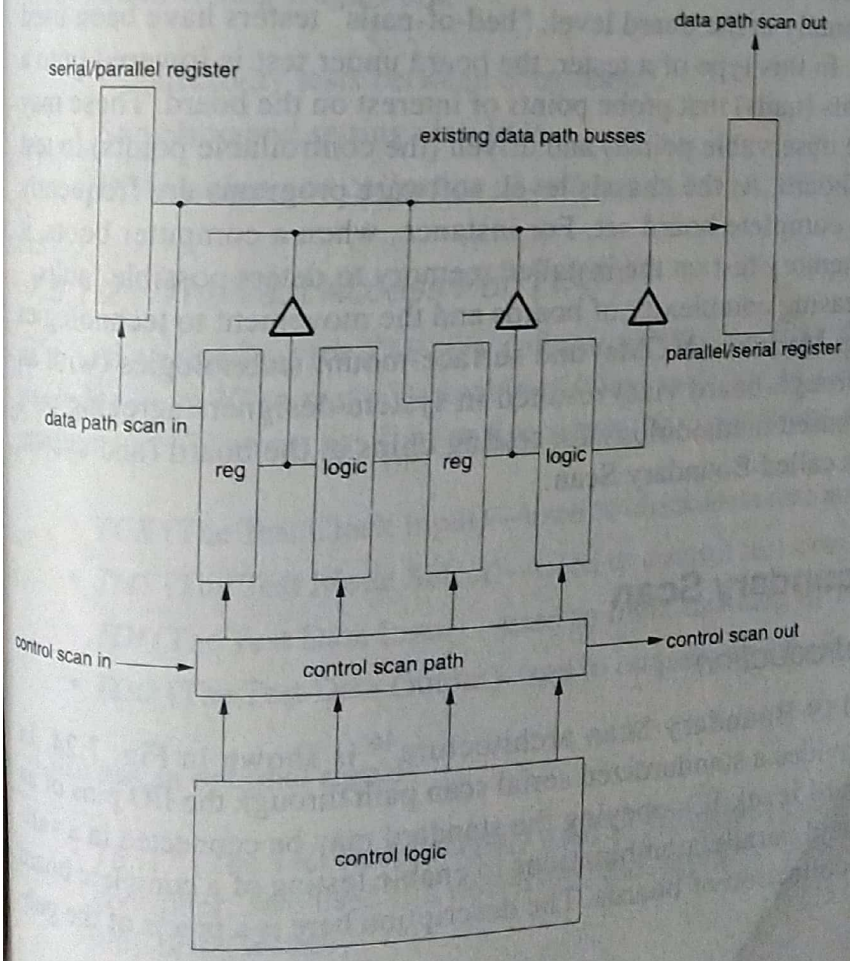


FIGURE 7.23 Datapath test scheme

Scanned by CamScanner

7.4.2 Memories

Memories may use the self-testing techniques mentioned in Section 7.3.4.2. Alternatively, the provision of multiplexers on data inputs and addresses and convenient external access to data outputs enables the testing of embedded memories. It is a mistake to have memories indirectly accessible (i.e., data is written by passing through logic, data is observed after passing through logic, addresses can not be conveniently sequenced). Because memories have to be tested exhaustively, any overhead on writing and reading the memories can substantially increase the test time and, probably more significantly, turn the testing task into an effort in inscrutability.

7.4.3 Random Logic

Random logic is probably best tested via full serial scan or parallel scan.

7.5 System-Level Test Techniques

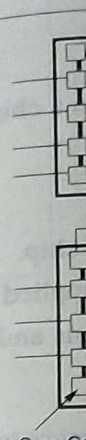
Up to this point we have concentrated on the methods of testing individual chips. Traditionally at the board level, "bed-of-nails" testers have been used to test boards. In this type of a tester, the board under test is lowered onto a set of test points (nails) that probe points of interest on the board. These may be sensed (the observable points) and driven (the controllable points) to test the complete board. At the chassis level, software programs are frequently used to test a complete board set. For instance, when a computer boots, it might run a memory test on the installed memory to detect possible faults.

The increasing complexity of boards and the movement to technologies like Multichip Modules (MCMs) and surface-mount technologies (with an absence of through-board vias) resulted in system designers agreeing on a unified scan-based methodology for testing chips at the board (and system level). This is called Boundary Scan.

7.5.1 Boundary Scan

7.5.1.1 Introduction

The IEEE 1149 Boundary Scan architecture[46] is shown in Fig. 7.24. In essence it provides a standardized serial scan path through the I/O pins of an IC. At the board level, ICs obeying the standard may be connected in a variety of series and parallel combinations to enable testing of a complete board or, possibly, collection of boards. The description here is a precis of the pub-



I/O Pad and Boundary Scan Ce

lished standard. The
in a unified testing f

- Connectivity
- Sampling and
- Distribution a

7.5.1.2 The Tes

The Test Access Por
included in an IC to
architecture. The po

- *TCK* (The Tes
- *TMS* (The Tes
- *TDI* (The Tes
- *TDO* (The Te

It also has an option

- *TRST** (The T
controller; als
chip being tes

COVER before you buy

Scanned by CamScanner

7.4.2 Memories

Memories may use the self-testing techniques mentioned in Section 7.3.4.2. Alternatively, the provision of multiplexers on data inputs and addresses and convenient external access to data outputs enables the testing of embedded memories. It is a mistake to have memories indirectly accessible (i.e., data is written by passing through logic, data is observed after passing through logic, addresses can not be conveniently sequenced). Because memories have to be tested exhaustively, any overhead on writing and reading the memories can substantially increase the test time and, probably more significantly, turn the testing task into an effort in inscrutability.

7.4.3 Random Logic

Random logic is probably best tested via full serial scan or parallel scan.

7.5 System-Level Test Techniques

Up to this point we have concentrated on the methods of testing individual chips. Traditionally at the board level, "bed-of-nails" testers have been used to test boards. In this type of a tester, the board under test is lowered onto a set of test points (nails) that probe points of interest on the board. These may be sensed (the observable points) and driven (the controllable points) to test the complete board. At the chassis level, software programs are frequently used to test a complete board set. For instance, when a computer boots, it might run a memory test on the installed memory to detect possible faults.

The increasing complexity of boards and the movement to technologies like Multichip Modules (MCMs) and surface-mount technologies (with an absence of through-board vias) resulted in system designers agreeing on a unified scan-based methodology for testing chips at the board (and system level). This is called Boundary Scan.

7.5.1 Boundary Scan

7.5.1.1 Introduction

The IEEE 1149 Boundary Scan architecture[46] is shown in Fig. 7.24. In essence it provides a standardized serial scan path through the I/O pins of an IC. At the board level, ICs obeying the standard may be connected in a variety of series and parallel combinations to enable testing of a complete board or, possibly, collection of boards. The description here is a precis of the published standard. The standard al
in a unified testing framework:

- Connectivity tests betwe
- Sampling and setting ch
- Distribution and collect

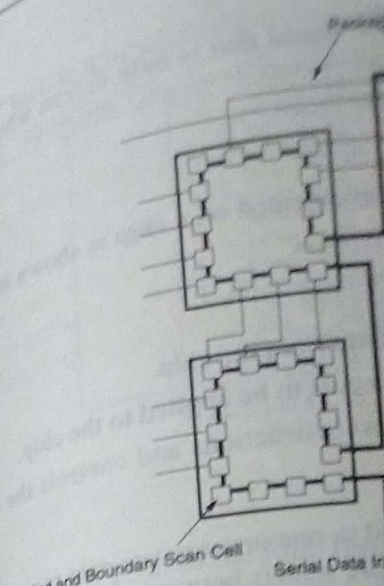
7.5.1.2 The Test Access

The Test Access Port (or TAP
included in an IC to make it
architecture. The port has fou

- *TCK* (The Test Clock I
- *TMS* (The Test Mode
- *TDI* (The Test Data In
- *TDO* (The Test Data

It also has an optional signa

- *TRST** (The Test Res
controller; also used
chip being tested.



The wafer level and system levels tests can be done by component vendor and satellite-borne electronics respectively.

3. I_{DD} test.

igure (a) shows
the function of
at this level tha

Look for the SIA GROUP LOGO on the TITLE COVER before you buy
UM ALL-IN-ONE

Scanned by CamScanner

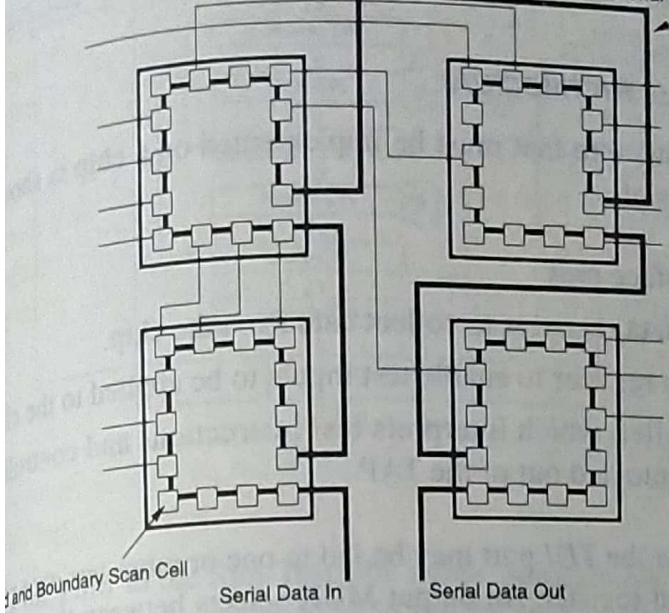


FIGURE 7.24 Boundary scan architecture

ed standard. The standard allows for the following types of tests to be run unified testing framework:

- Connectivity tests between components.
- Sampling and setting chip I/Os.
- Distribution and collection of self-test or built-in-test results.

5.1.2 *The Test Access Port (TAP)*

e Test Access Port (or TAP) is a definition of the interface that needs to be luded in an IC to make it capable of being included in a Boundary-Scan hitecture. The port has four or five single-bit connections, as follows:

- *TCK* (The Test Clock Input)—used to clock tests into and out of chips.
- *TMS* (The Test Mode Select)—used to control test operations.
- *TDI* (The Test Data Input)—used to input test data to a chip.
- *TDO* (The Test Data Output) used to output test data from a chip.

also has an optional signal

- *TRST** (The Test Reset Signal) used to asynchronously reset the TAP controller; also used if a power-up reset signal is not available in the chip being tested.

...ip, and relate t

...ure (a) shows two shorted faults. The short *S1* in figure (a) is modeled by an S-A-0 fault at input *A*, ...the function of the gate. To ensure the most accurate modeling, faults should be modeled at the transisto... ...at this level that the complete circuit structure is known.

u buy

Scanned by CamScanner

The *TDO* signal is defined as a tristate signal that is only driven when the TAP controller is outputting test data.

7.5.1.3 The Test Architecture

The basic test architecture that must be implemented on a chip is shown in Fig. 7.25. It consists of:

- the TAP interface pins
- a set of test-data registers to collect data from the chip.
- an instruction register to enable test inputs to be applied to the chip.
- a TAP controller, which interprets test instructions and controls the flow of data into and out of the TAP.

Data that is input via the *TDI* port may be fed to one or more test data registers or an instruction register. An output MUX selects between the instruction register and the data registers to be output to the tristate *TDO* pin.

7.5.1.4 The TAP Controller

The TAP controller is a 16-state FSM that proceeds from state to state based on the *TCK* and *TMS* signals. It provides signals that control the test data registers, and the instruction register. These include serial-shift clocks and update clocks.

The state diagram is shown in Fig. 7.26. The state adjacent to each state transition is that of the *TMS* signal at the rising edge of *TCK*.

The reader is referred to the standard for complete descriptions of these states. It is probably best to understand them by examining a typical test sequence. Starting initially in the Test-Logic-Reset state, a low on TMS tran-

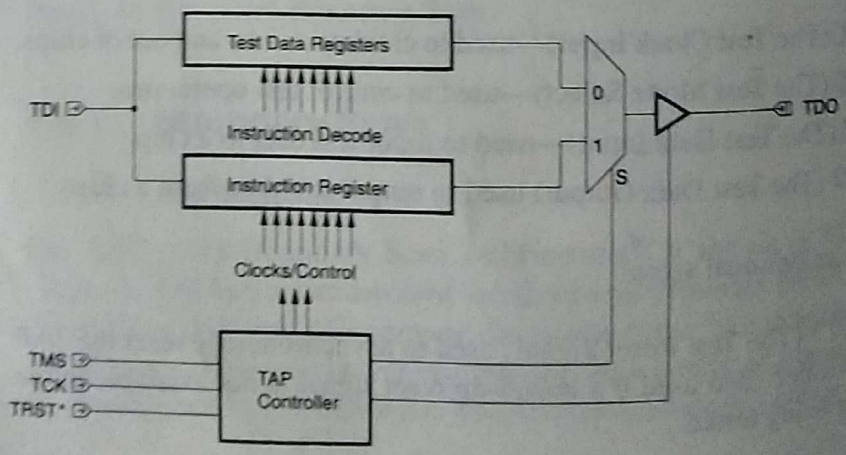


FIGURE 7.25 TAP architecture

Scanned by CamScanner

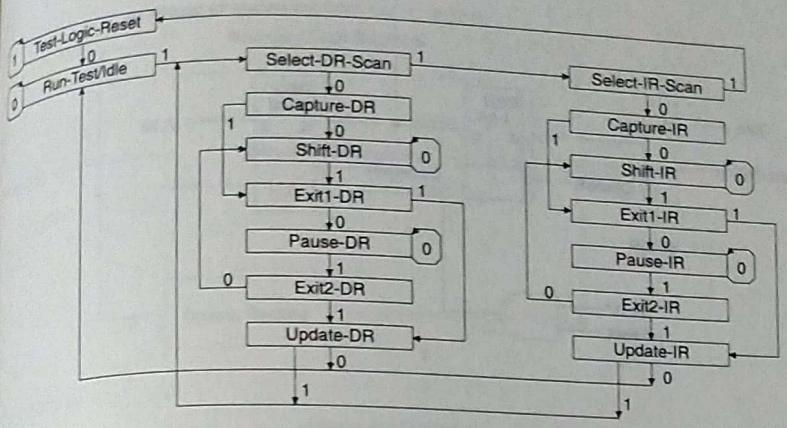


FIGURE 7.26 TAP controller state diagram

sitions the FSM to the Run-Test/Idle mode. Holding *TMS* high for the next three *TCK* cycles places the FSM in the Select-DR-Scan, Select-IR-Scan, and finally Capture-IR mode. In this mode two bits are input to the *TDI* port and shifted into the instruction register. Asserting *TMS* for a cycle allows the instruction register to pause while serially loading to allow tests to be carried out. Asserting *TMS* for two cycles, allows the FSM to enter the Exit2-IR mode on exit from the Pause-IR state and then to enter the Update-IR mode where the Instruction Register is updated with the new IR value. Similar sequencing is used to load the data registers.

A CMOS implementation of the Tap Controller based on that in the standard is shown in Fig. 8.89.

7.5.1.5 The Instruction Register (IR)

The instruction register has to be at least two bits long, and logic detecting the state of the instruction register has to decode at least three instructions which are as follows:

- BYPASS—This instruction is represented by an IR having all zeroes in it. It is used to bypass any serial-data registers in a chip with a 1-bit register. This allows specific chips to be tested in a serial-scan chain without having to shift through the accumulated SR stages in all the chips.
- EXTEST—This instruction allows for the testing of off-chip circuitry and is represented by all ones in the IR.
- SAMPLE/PRELOAD—This instruction places the boundary-scan registers (i.e., at the chips' I/O pins) in the DR chain, and samples or preloads the chips I/Os.

Scanned by CamScanner

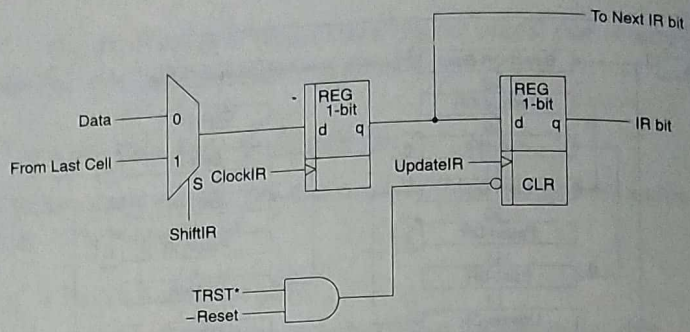


FIGURE 7.27 Instruction-register bit implementation

In addition to these instructions, the following are also recommended:

- INTEST—This instruction allows for single-step testing of internal circuitry via the boundary-scan registers.
- RUNBIST—This instruction is used to run internal self-testing procedures within a chip.

Further instructions may be defined as needed to provide other testing functions.

A typical IR bit is shown in Fig. 7.27.

7.5.1.6 Test-Data Registers (DRs)

The test-data registers are used to set the inputs of modules to be tested, and to collect the results of running tests. The simplest data-register configuration would be a boundary-scan register (passing through all I/O pads) and a bypass register (1-bit long). Figure 7.28 shows a generalized view of the data registers where one internal data register has been added. A multiplexer under the control of the Tap controller selects which particular data register is routed to the *TDO* pin.

7.5.1.7 Boundary Scan Registers

The boundary scan register is a special case of a data register. It allows circuit-board interconnections to be tested, external components tested, and the state of chip digital I/Os to be sampled. Apart from the bypass register, it is the only data register required in a Boundary Scan compliant part.

A single structure (in addition to the existing I/O circuitry) can be used for all I/O pad types, depending on the connections made to the cell. It consists of two multiplexers and two edge-triggered registers. Figure 7.29(a) shows this cell used as an input pad. Two register bits allow the serial shifting of data through the boundary-scan chain and the local storage of a data

Look for the SIA GROUP LOGO on the TITLE COVER before you buy

Scanned by CamScanner

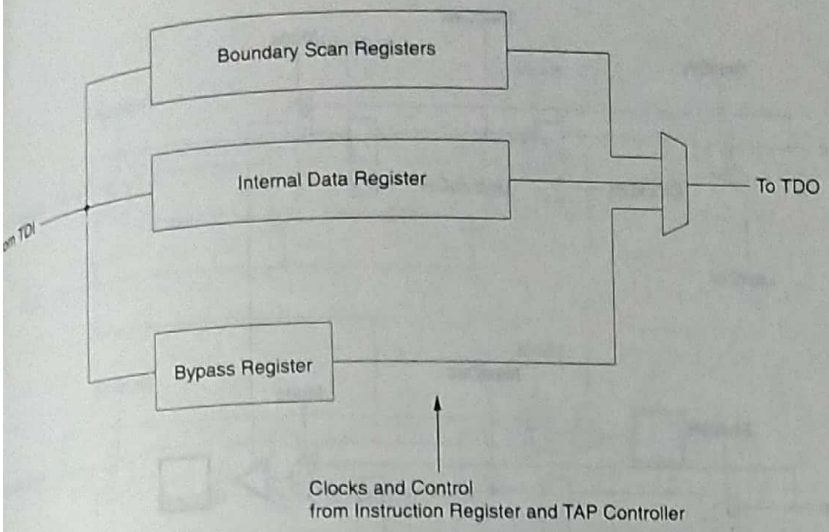


FIGURE 7.28 TAP data registers

it. This data bit may be directed to internal circuitry in the INTEST or RUNBIST modes (*Mode* = 1). When *Mode* = 0, the cell is in EXTEST or SAMPLE/PRELOAD mode. A further multiplexer under the control of *ShiftDR* controls the serial/parallel nature of the cell. The signal *ClockDR* and *UpdateDR* generated by the Tap Controller load the serial and parallel register, respectively.

An output cell is shown in Fig. 7.29(b). When *Mode* = 1, the cell is in EXTEST, INTEST, or RUNBIST modes, communicating the internal data to

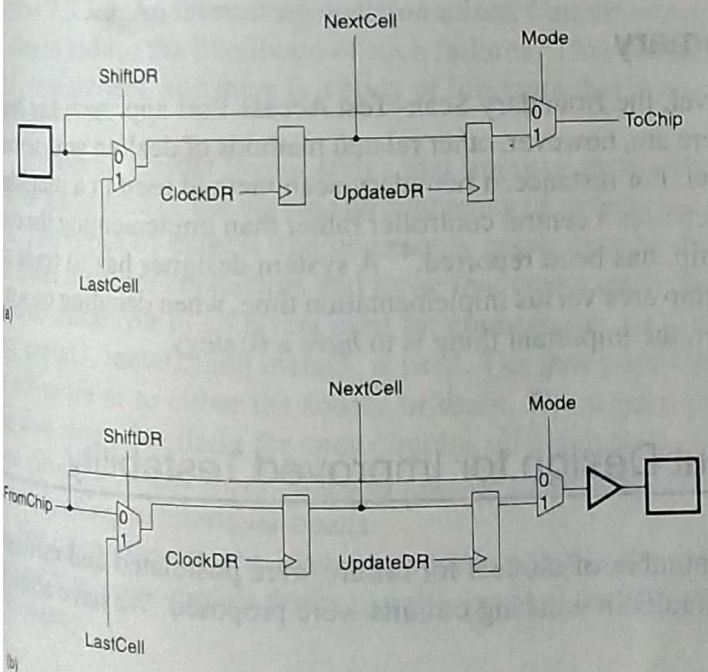


FIGURE 7.29 Boundary scan (a) input and (b) output cells

Scanned by CamScanner

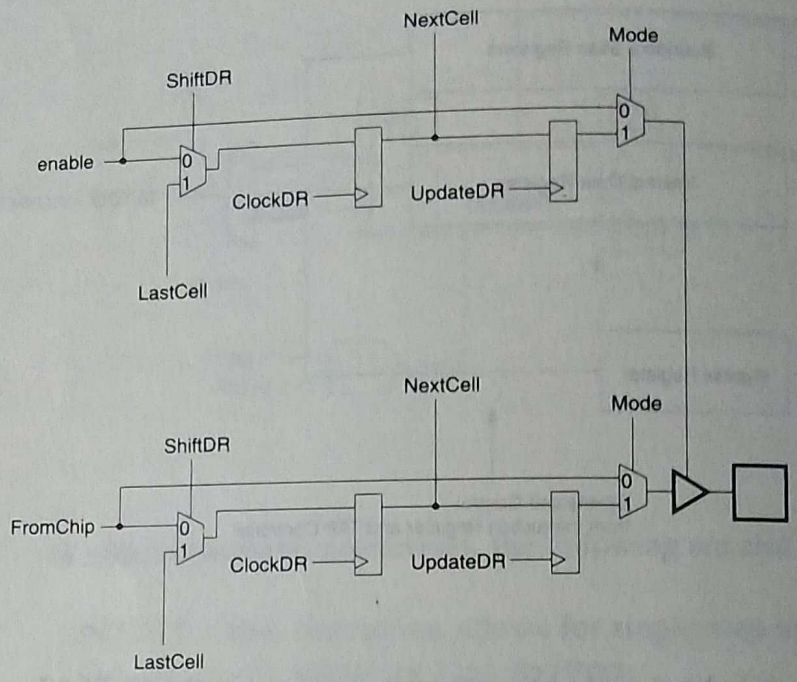


FIGURE 7.30 Boundary scan tristate cell

the output pad. When *Mode* = 0, the cell is in the SAMPLE/PRELOAD mode.

Two output cells may be combined to form a tristate boundary-scan cell, as shown in Fig. 7.30. The output signal and tristate-enable each have their own muxes and registers. The *Mode* control is the same for the output-cell example.

Finally, a bidirectional pin combines an input and tristate cell, as shown in Fig. 7.31.

7.5.2 Summary

At the system level, the Boundary Scan–Test Access Port approach has been summarized. There are, however, other related methods of dealing with testing at the system level. For instance, a boundary-scan method used in a multichip workstation, which uses a central controller rather than implementing the controller in each chip, has been reported.[47] A system designer has to trade off aspects, such as chip area versus implementation time, when deciding on a test strategy. However, the important thing is to *have* a strategy.

7.6 Layout Design for Improved Testability

Scanned by CamScanner