UNIT-V

CAD TOOLS FOR DESIGN AND SIMULATION

As the design of very large systems is concerned, it is essential to have computer aids to design so that the design can be completed in a reasonable time.

The designer's 'tool box' should include:

1. **Physical design layout and editing capabilities**, either through textual or graphical entry of information;

2. **Structure generation/system composition capabilities**

3. **Physical verification.**

The tools here should include, design rule checking (DRC), circuit extractors, ratio rule and other static checks, and a capability to plot out and/or display for visual checking;

4. **Behavioral verification.**

Simulation at various levels will be required to check out the design before one embarks on the expense of turning out the design in silicon.

Simulators are available for logic (switch level) simulation and timing simulation.

Circuit simulation via such programs as SPICE is also possible, but may be expensive in terms of computing time and therefore impractical for other than small subsystems.

Recent advances in simulators have made it possible to use the software as 'a probe' to examine the simulated responses on various parts of the circuit to input stimuli also provided via the simulator.

ASPECTS OF DESIGN TOOLS

Graphical Entry Layout

1. Textual entry of layouts was at one time quite widely used and special textual entry editors are in existence and may well be used for small subsystem layout.
2. However, such tools have been virtually swept aside by a much more convenient and highly interactive method of producing layouts for which monochrome or color graphics terminals are used, and on which the layout is built up and displayed during the design process.
3. Such systems are mostly 'menu driven', in that menus of possible actions at various stages of the design are displayed on the screen beside the display of the current layout detail.
4. Some form of cursor allows selection and/or placement of geometric features, etc., and the cursor may also allow selection of menu items or, alternatively, these may also be selected from a keyboard.
5. Positioning of the cursor may be effected from the keyboard in simple systems and/or cursor position may be controlled from a bit pad digitizer or from a 'mouse ', etc

6. Two of the earliest available graphical entry layout packages were **KIC** developed at the University of California, Berkeley, and **PLAN**, originally developed at the University of Adelaide.
7. PLAN makes use of low-cost monochrome as well as color graphics terminals and is marketed by Integrated Silicon Design Pty Ltd, Adelaide.
8. The use of an early version of PLAN to generate layouts illustrated in Figures and it is hoped that the inclusion of these figures, which show various stages of design, is sufficient to convey an idea of the nature of the layout process using this class of software tools.

Design Verification Prior to Fabrication

1. It is not enough to have good design tools for producing mask and system layout detail.
2. It is essential that such tools be complemented by equally effective verification software capable of handling large systems and with reasonable computing power requirements.
3. The nature of the tools required will depend on the way in which an integrated circuit design is represented in the computer.
4. Two basic approaches are:
   1. **Mask level layout languages**, such as CIF, which are well suited to physical layout description but not for capturing the design intent.
   2. **Circuit description languages** where the primitives are circuit elements such as transistors, wires, and 'nodes.
   In general, such languages capture the design intent but do not directly describe the physical layout associated with the design.

Design Rule Checkers (DRC)

A. The cost in time and facilities in mask-making and in fabricating a chip from those masks is such that all possible errors must be eliminated before mask-making proceeds.
B. Once a design has been turned in to silicon there is little that can be done if it doesn't work.

The wise designer will check for errors at all stages of the design, namely:

1. At the pencil and paper stage of the design of leaf-cells;

2. At the leaf-cell level once the layout is complete (e.g. when the CIF code for that leaf-cell has been generated);

3. At the subsystem level to check that butting together and wiring up of leaf-cells is correctly done;

4. Once the entire system layout has been completed.

The nature of physical layout verification 'design rule checking (DRC)' software may depend on whether the design rules are absolute or lambda-based, or on whether or not the layout is on a fixed or virtual grid.

Circuit Extractors

Circuit extractor can describe mask layout in a form suitable to a simulator.

The circuit description contains information about circuit components and their interconnections.

(An example of a circuit extractor program is NET from Integrated Silicon Design Pty Ltd.)

Simulators

The circuit description is subsequently transformed into a set of equations by the simulator from which the predictions of behavior are made.

The topology of the circuit determines two sets of equations:

• Kirchhoff's Current Law-determining the branch currents; and

• Kirchhoffs Voltage Law-determining node voltages.

The electrical behavior is defined by mathematical modeling, the accuracy of which determines two key factors:

• the accuracy of the simulation; and

• the computing power and time needed for the simulation.

Various **types of simulators** are available but generally they fall into the following groups:

• Circuit simulators;

• Timing simulators;

• Logic level (switch level) (functional) simulators;

• System level (functional) simulators.

Circuit simulators are concerned with the electrical behavior of the various parts of the circuit to be implemented in silicon.

Simulation programs such as SPICE can do this quite well, but take a lot of computing time to simulate even relatively small sections of a system.

Timing simulators (PROBE) have attempted to improve matters in these respects by concentrating on active nodes and ignoring quiescent nodes in simulation.

Timing simulators are becoming increasingly important during the design phase because of their speed and consequent interactive qualities.

The structure of these tools ensures that run times are strictly linearly related to the number of devices and nodes being simulated.

Logic level simulators the performance is assessed in terms of logic levels with no or little timing information.

When considering complete systems, logic simulators may be replaced by **system level simulators** which operate at the register transfer level.

Design for Testability*

Design for testability (DFT) makes it possible to:

1. Assure the detection of all faults in a circuit
2. Reduce the cost and time associated with test development
3. Reduce the execution time of performing test on fabricated chips

There are two key concepts underlying all considerations for testability.

They are:

1. Controllability;
2. Observability.

The controllability of a circuit is a measure of the ease (or difficulty) with which the controller (test engineer) can establish a specific signal value at each node by setting values at the circuit input terminals.

(OR)

Controllability of a digital circuit is defined as the difficulty of setting a particular logic signal to 0 or 1.

The observability is a measure of the ease (or difficulty) with which one can determine the signal value at any logic node in the circuit by controlling its primary input and observing the primary output.

(OR)

Observability for a digital circuit is defined as the difficulty of observing the state of a logic signal

The degree of controllability and observability and, thus, the degree of testability of a circuit, can be measured with respect to whether test vectors are generated deterministically or randomly.

These concepts ensure that the designer considers the provision of means of setting or resetting key nodes in the system and of observing the response at key points.

The effects of testability or lack of it are such that it has been predicted that testability will soon become the main design criterion for VLSI circuits.

Design for testability (observability and controllability) is then reduced to a set of design rules or guidelines which, if obeyed, will facilitate test.

A failure during testing at the chip level may be due to a design defect or a poorly controlled fabrication process.

The inputs of the device under test (DUT) are subjected to a test pattern (or test vector) which supplies a set of binary values, in combination and/or in sequence, to detect faults.

The specification of the test vector sequences must involve the designer, while the generation and application of test patterns to a DUT are the problems faced by the test engineer.

Test pattern generation is assisted by using automatic test pattern generators (ATPG), but they are complicated to use properly and ATPG costs tend to rise rapidly with circuit size.

Once the application of a test pattern has revealed a fault, the process of diagnosis must be invoked to localize the fault.

Test coverage

Detecting all the possible faults in a DUT corresponds to 100% 'test coverage'.

In general it is relatively easy to detect the first 80% of faults using various classical test strategies, but when more than an 80% coverage is required, appropriate test strategies must be developed.

In any case, it is not generally possible to anticipate 100% of all faults, so that we tend to talk about a set of fault hypotheses which may then be covered 100%.

Faults may be classified using different **models** and three such are:

- **Mathematical model;**
- **Logical model (stuck-at);**
- **Physical model**.

The latter two are most commonly used.

The 'stuck-at' model has been widely used and was originally developed in the testing of p.c. boards but is not in itself sufficient to test actual VLSI CMOS circuits.

A further set of physical fault models is also used:

- Class· 0: A single physical defect such as a faulty contact or via, a transistor stuck on or stuck off, an interconnection through any layer open circuit.
- Class 1: Class 0 with a short circuit between metal lines or diffusion lines.
- Class 2: Class 1 with short circuit(s) between two lines on any layer.

Testing Combinational Logic

The solution to the problem of testing combinational logic is to generate a set of test patterns which will detect all possible fault conditions.

The first approach to testing an N input circuit is to generate all the possible 2^N input signal combinations by means of, say, an N-bit counter (controllability) and observe the output(s) for checking (observability).

This is called exhaustive testing and is very effective, but is only practicable where N is relatively small.

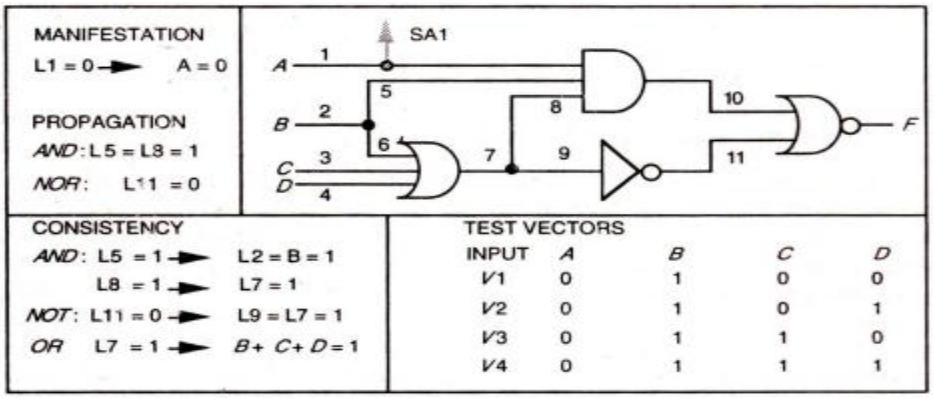
Sensitized path-based testing

The basic idea is to select a path from the site of the possible fault, through a sequence of gates leading to an output of the logic circuitry under test(CUT).

The process comprises three steps:

1. Manifestation:
   Gate inputs at the site of an assumed fault, say a 'stuck at' (SA) fault, are specified to generate the opposite value to the assumed SA value (0 f9r SAl, 1 for SAO).
2. Propagation:
   Inputs of other gates are determined so as to propagate the fault signal along the selected path to the primary output of the circuit. This is done by setting And/Nand inputs to '1' and Or/Nor inputs to '0'.
3. Consistency (or justification):
   This final step finds the primary input patterns to realize all the necessary values. This is done by tracing backward from the gate inputs to the primary input of the logic.·
   Examples will help explain the process.

Example 1: Take an SA1 fault on line 1(L 1) in Figure , then



The D-algorithm

The algorithm aims to find an assignment of input values that will allow detection of a particular internal fault by examining the output conditions.

In order to do this the algorithm is based on the hypothesis of the existence of two machines a good machine and a faulty machine.

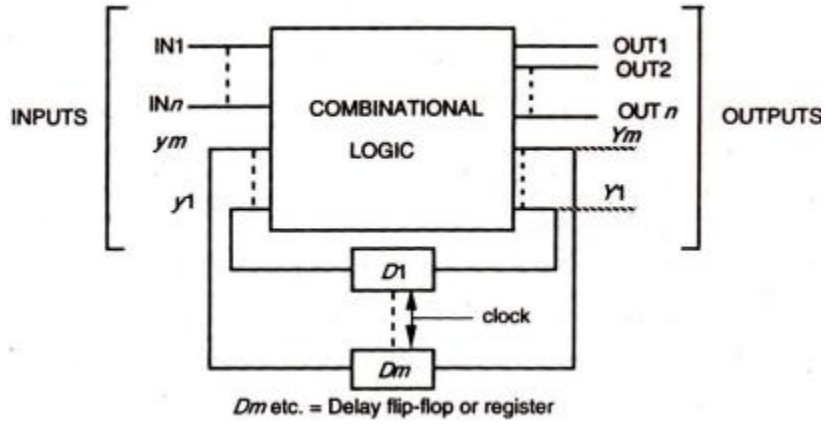
The existence of a fault in the faulty machine will cause a discrepancy between its behavior and that of the good machine for some particular values of inputs.

The D-algorithm provides a systematic means of assigning input values for that particular design so that the discrepancy is driven to an output where it may be observed and thus detected.

The algorithm is extremely time intensive and computing intensive for large circuits and has been the subject of several adoptions, modifications and improvements.

LASAR (Logic Automated Stimulus and Response), PODEM (Path Oriented Decision Making) and FAN (FAN-out oriented test generation) are all improvements on the D-algorithm.

Testing Sequential Logic



Sequential circuits, which may be generally represented as finite state machines, may be modeled as combinational logic with a set of delays and feedback from output to input as shown in Figure.

The 'm' feedback variables constitute the state vector and determine the maximum number of finite states which may be assumed by the circuit.

In the most general case, the next state and the output are both functions of the present state and the independent inputs.

.

Scan Design Techniques

The scan design techniques are structured approaches to designing sequential circuits so that testability is 'designed in' from the outset.

The major difficulty in sequential circuit testing is in determining the internal state of the circuit.

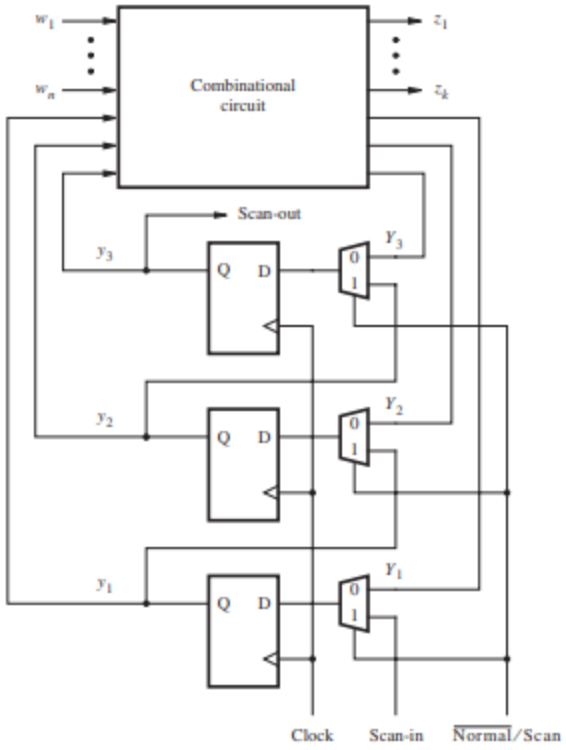
Scan design techniques are directed at improving the controllability and observability of the internal states.

The approach aims to reduce the problem of testing a sequential circuit to that of testing combinational logic.

Scan-Path Technique

A popular technique, called the scan path, uses multiplexers on flip-flop inputs to allow the flip-flops to be used either independently during normal operation of the sequential circuit, or as a part of a shift register for testing purposes.

Figure presents the general scan-path structure for a circuit with three flip-flops.



A 2-to-1 multiplexer connects the D input of each flip-flop either to the corresponding next-state variable or to the serial path that connects all flip-flops into a shift register.

The control signal Normal/Scan selects the active input of the multiplexer.

During the normal operation the flip-flop inputs are driven by the next-state variables, Y1, Y2, and Y3. For testing purposes the shift-register connection is used to scan in the portion of each test vector that involves the present-state variables, y1, y2, and y3.

This connection has Qi connected to Di+1.

The input to the first flip-flop is the externally accessible pin Scan-in.

The output comes from the last flip-flop, which is provided on the Scan-out pin.

The scan-path technique involves the following steps:

1. The operation of the flip-flops is tested by scanning into them a pattern of 0s and 1s, for example, 01011001, in consecutive clock cycles, and observing whether the same pattern is scanned out.

2. The combinational circuit is tested by applying test vectors on w1w2 ···wny1y2y3 and observing the values generated on z1z2 ···zmY1Y2Y3. This is done as follows:

• The y1y2y3 portion of the test vector is scanned into the flip-flops during three clock cycles, using Normal/Scan = 1.

• The w1w2 ···wn portion of the test vector is applied as usual and the normal operation of the sequential circuit is performed for one clock cycle, by setting Normal/Scan = 0. The outputs z1z2 ···zm are observed. The generated values of Y1Y2Y3 are loaded into the flip-flops at this time.

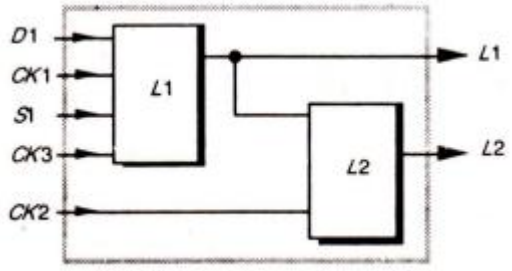
• The select input is changed to Normal/Scan = 1, and the contents of the flip-flops are scanned out during the next three clock cycles, which makes the Y1Y2Y3 portion of the test result observable externally.

Level-sensitive scan design (LSSD)

This is a technique, initially developed by IBM,

which incorporates two aspects- level sensitivity and a scan path approach.

The general arrangement is indicated in Figure.



The **level-sensitive aspect** means that the sequential network is designed so that when an input change occurs, the response is independent of the component and wiring delays within the network.

The **scan path aspect** is due to the use of shift register latches (SRL) employed as storage elements. In the test mode they are connected as a long serial shift register.

Each SRL has a specific design similar to a master-slave flip-flop.

It is driven by two non-overlapping clocks which can be controlled readily from the primary inputs to the circuit.

Input DI is the normal data input to the SRL, clocks CKI and CK2 control the normal operation of the SRL while clocks CK3 and CK2 control scan path movements through the SRL.

The SRL output is derived at L2 in both modes of operation, the mode depending on which clocks are activated. The following advantages are claimed for the LSSD approach:

• The circuit operation is independent of the dynamic characteristics of the logic elements-rise- and fall-times and propagation delays.

• A TP generation is simplified since tests need only be generated for a combinational circuit.

• LSSD methods, when adopted in design, eliminate hazards and races; greatly simplifies test generation and fault simulation.

Built-In-Self-Test (BIST)

As the complexity of individual VLSI circuits and as overall system complexity increase, test generation and application becomes an expensive, and not always very effective, means of testing.

Further, there are also very difficult problems associated with the high speeds at which many VLSI systems are designed to operate.

Such problems require the use of very sophisticated, but not always affordable, test equipments. Consequently, BIST .

BIST techniques aim to effectively integrate an automatic test system into the chip design.

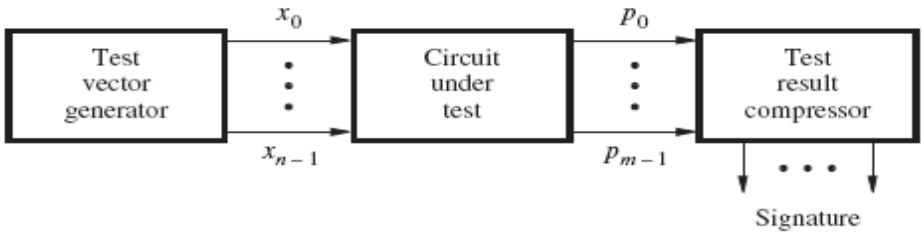
BIST objectives are:

1. to reduce test pattern generation costs;

2. to reduce the volume of test data;

3. to reduce test time.

Figure shows a possible BIST arrangement in which a test vector generator produces the test vectors that must be applied to the circuit under test.



Randomly chosen test vectors give good results, with the fault coverage depending on the number of tests performed.

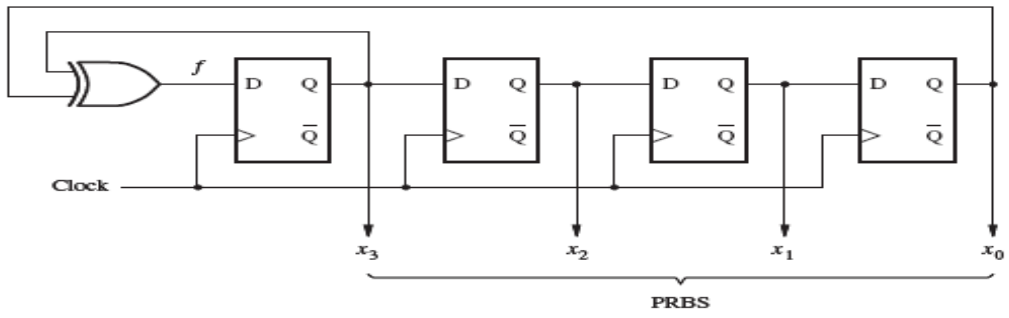
For each test vector applied to the circuit, it is necessary to determine the required response of the circuit.

The response of a good circuit may be determined using the simulator tool of a CAD system.

The expected responses to the applied tests must be stored on the chip so that a comparison can be made when the circuit is being tested.

The generator for pseudorandom tests is easily constructed using a feedback shift-register circuit.

A small example of a possible generator is given in Figure 11.14.

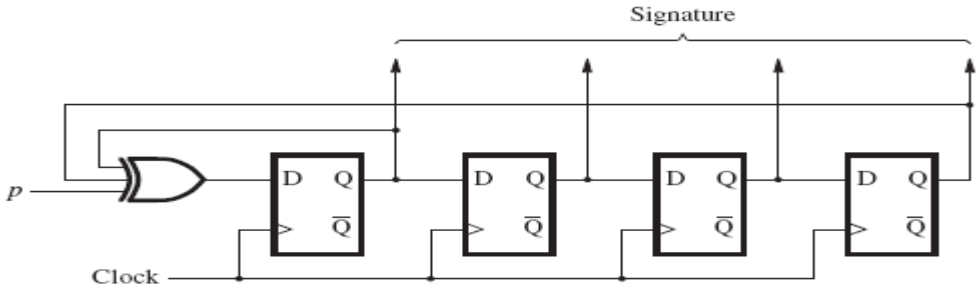


x_3	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	...
x_2	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0	...
x_1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	...
x_0	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	...
f	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1	...

A four-bit shift register, with the signals from the first and fourth stages fed back through an XOR gate, generates 15 different patterns during successive clock cycles.

A compressor circuit includes the output signals produced by the circuit under test.

Figure shows a single-input compressor circuit (SIC), which uses the same feedback connections as the PRBSG of Figure



The input p is the output of a circuit under test.

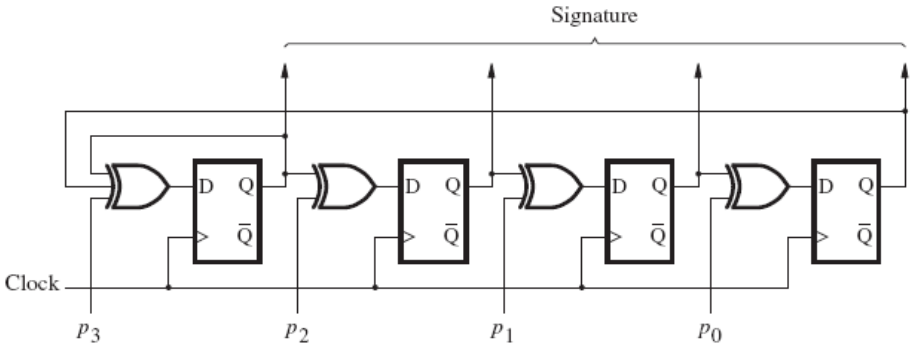
After applying a number of test vectors, the resulting values of p drive the SIC and, coupled with the LFSR functionality, produce a four-bit pattern called a signature.

The signature represents a single pattern that may be interpreted as a result of all the applied tests.

It can be compared against a predetermined pattern to see if the tested circuit is working properly.

If the circuit under test has more than one output, then an LSFR with multiple inputs can be used.

Figure 11.16 illustrates how four inputs, p0 through p3,



.Again the four-bit signature provides a good mechanism for distinguishing among different sequences of four-bit patterns that may appear on the inputs of this multiple-input compressor circuit (MIC).

BIST for Sequential Circuit:

A complete BIST scheme for a sequential circuit may be implemented as indicated in Figure.

The scan-path approach is used to provide a testable circuit.

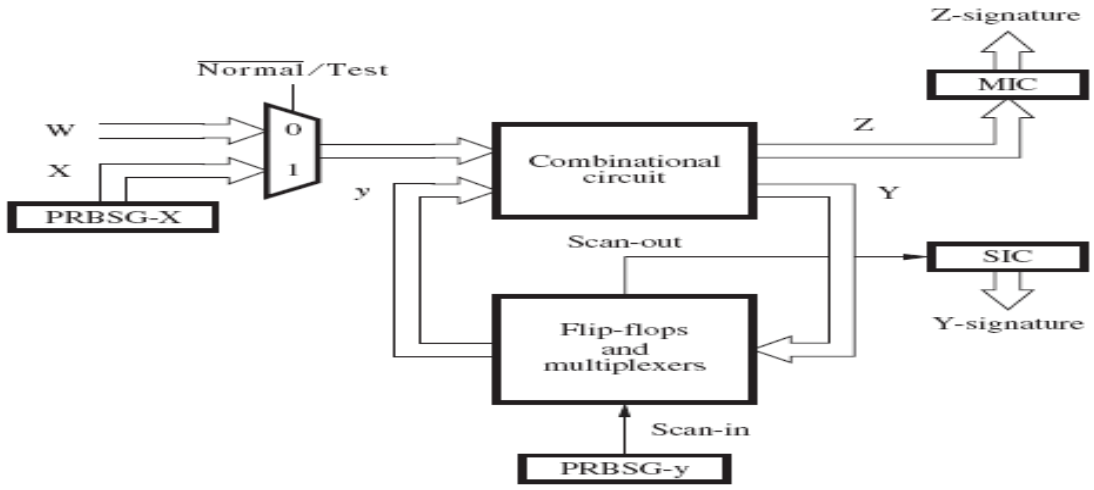
The test patterns that would normally be applied on the primary inputs W = w1w2 ⋯wn are generated internally as the patterns on X = x1x2 ⋯ xn.

Multiplexers are needed to allow switching from W to X , as inputs to the combinational circuit.

A pseudorandom binary sequence generator, PRBSG-X , generates the test patterns for X .

The portion of the tests applied via the next-state signals, y, is generated by the second PRBS generator, PRBSG-y.

These patterns are scanned into the flip-flops as explained .



The test outputs are compressed using the two compressor circuits.

The patterns on the primary outputs, Z = z1z2 ⋯zm, are compressed using the MIC circuit, and those on the next-state wires Y = Y1Y2 ⋯ Yk , by the SIC circuit.

These circuits produce the Z-signature and Y -signature, respectively.

At the end of the testing process the two signatures are compared with the stored patterns.

The effectiveness of the BIST approach depends on the length of the LFSR generator and compressor circuits.

Longer shift registers give better results .

One reason for failing to detect that the circuit under test may be faulty is that the pseudorandomly generated tests do not have perfect coverage of all possible faults.

Another reason is that a signature generated by compressing the outputs of a faulty circuit may coincidentally end up being the same as the signature of the good circuit.

This can occur because the compression process results in a loss of some information, such that two distinct output patterns may be compressed into the same signature. This is known as the aliasing problem

Built-in Logic Block Observer (BILBO)

The essence of BIST is to have internal capability for generation of tests and for compression of the results. Instead of using separate circuits for these two functions, it is possible to design a single circuit that serves both purposes.

Figure shows the structure of a possible circuit, known as the built-in logic block observer (BILBO).

This four-bit circuit has the same feedback connections.

The BILBO circuit has four modes of operation, which are controlled by the mode bits, M1 and M2.

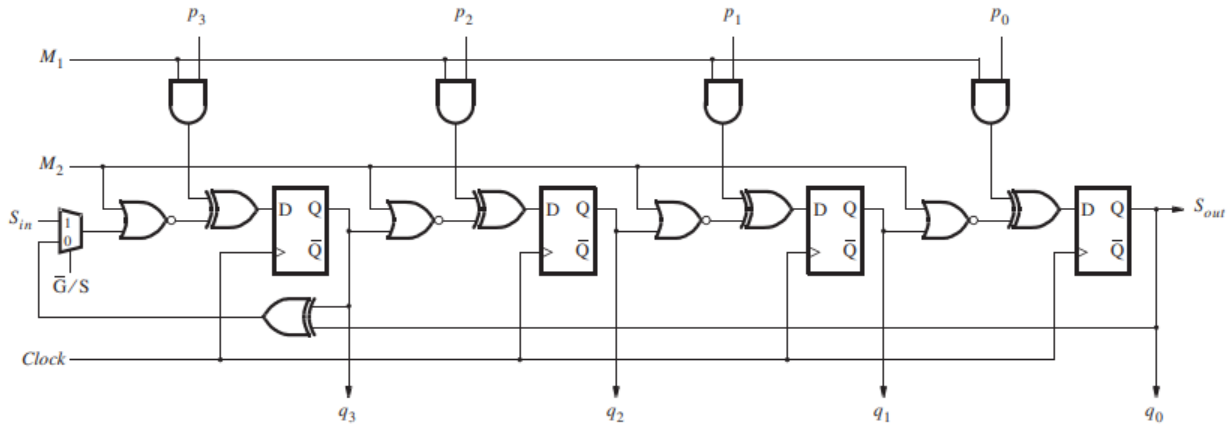
The modes are as follows:

• M1M2 = 11—Normal system mode in which all flip-flops are independently controlled by the signals on inputs p0 through p3. In this mode each flip-flop may be used to implement a state variable of a finite state machine by using p0 to p3 as y0 to y3.

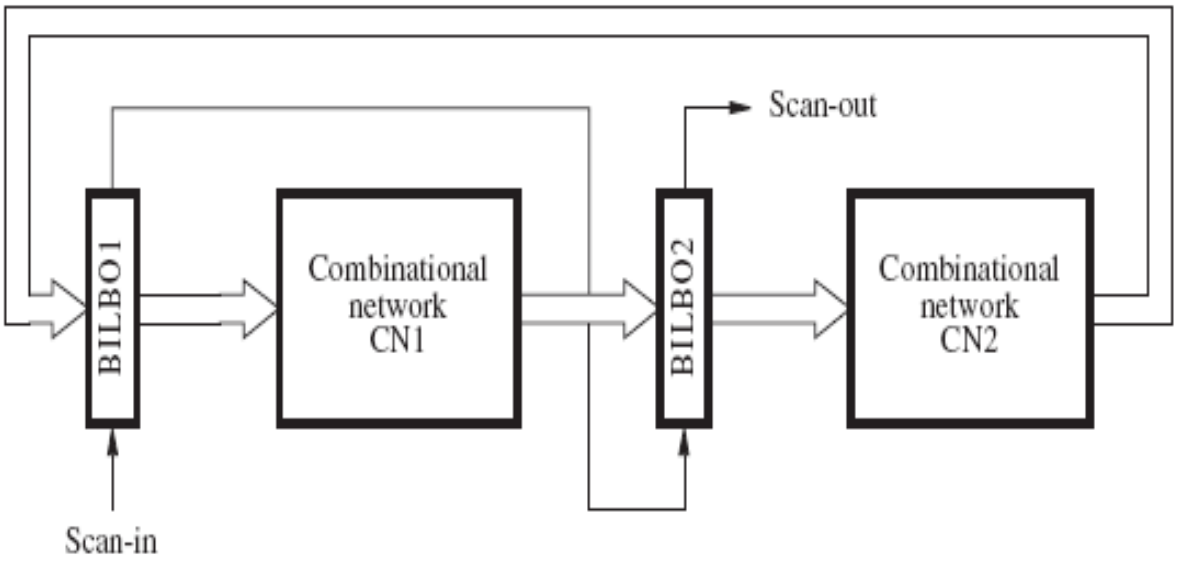
• M1M2 = 00 — Shift-register mode in which the flip-flops are connected into a shift register. This mode allows test vectors to be scanned in, and the results of applied tests to be scanned out, if the control input G/S is equal to 1. If G/S = 0, then the circuit acts as the PRBS generator.

• M1M2 = 10 — Signature mode in which a series of patterns applied on inputs p0 through p3 are compressed into a signature available as a pattern on q0 through q3.

• M1M2 = 01 — Reset mode in which all flip-flops are reset to 0



An efficient way of using BILBO circuits is presented in Figure 11.19.



A combinational circuit can be tested by partitioning it into two (or more) parts.

A BILBO circuit is used to provide inputs to one part and to accept outputs from the other part.

The testing process involves a two-phase approach.

First, BILBO1 is used as a PRBS generator that provides test patterns for combinational network 1 (CN1). During this time BILBO2 acts as a compressor and produces a signature for the test.

The signature is shifted out by placing BILBO2 into the shift-register mode.

Next, the roles of BILBO1 and BILBO2 are reversed, and the process is repeated to test CN2.

The detailed steps in the testing process are

1. Scan the initial test pattern into BILBO1 and reset all flip-flops in BILBO2.

2. Use BILBO1 as the PRBS generator for a given number of clock cycles and use BILBO2 to produce a signature.

3. Scan out the contents of BILBO2 and externally compare the signature; then scan into it the initial test pattern for testing CN2. Reset the flip-flops in BILBO1.

4. Use BILBO2 as the PRBS generator for a given number of clock cycles and use BILBO1 to produce a signature.

5. Scan out the signature in BILBO1 and externally compare it with the required pattern.

The BILBO circuits are used in this way for testing purposes.

At other times the normal system mode is used

Boundary scan test (BST)

This is a technique involving scan path and self-testing to resolve the problems associated with the testing of boards carrying VLSI circuits and/or surface-mounted devices (SMD).

Printed circuit boards (PCBs) are becoming very dense and complex, especially with SMD circuits, so that most test equipment cannot guarantee a good fault coverage.

BST consists of placing a scan path (shift register) cell adjacent to each component pin and to interconnect the cells so as to form a chain around the border of the circuit.

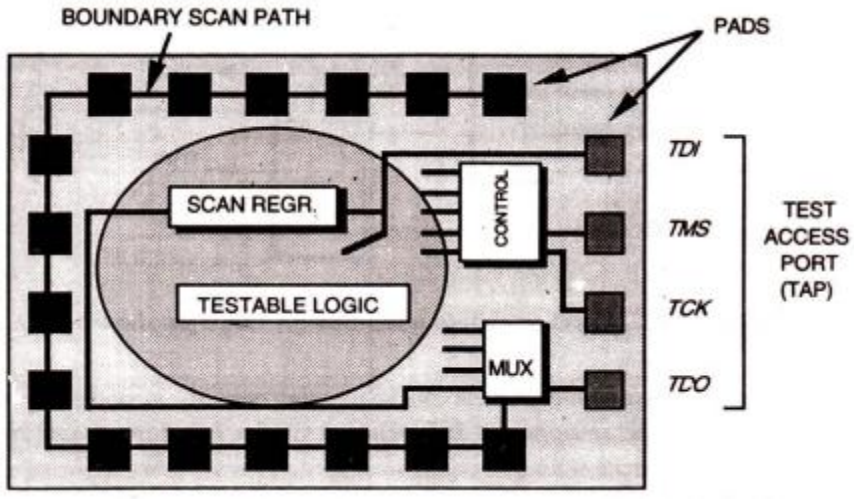
The BST circuits contained on one board are then connected together to form a single path. The general idea is illustrated in Figure .

The boundary scan path is provided with serial input and output pads and appropriate clock pads which make it possible to:

• test the interconnections between the various chips on the board;

• deliver test data to the chips on the board for self-testing;

• test the chips themselves with internal self-test facilities.



BS techniques are grouped by the IEEE standards organization into a 'standard test access port and boundary scan architecture' (namely, IEEE, p. 1149.1-1990).

The advantages of BST are seen as follows:

• no need for complex testers in PCB testing;

• the test engineer's work is simplified and efficient;

• the time spent on test pattern generation and application is reduced;

• fault coverage is increased.

Practical Design for test (OFT) Guidelines

Practical guidelines for testability to facilitate test processes in three main ways:

• facilitate test generation;

• facilitate test application; and

• avoid timing problems.

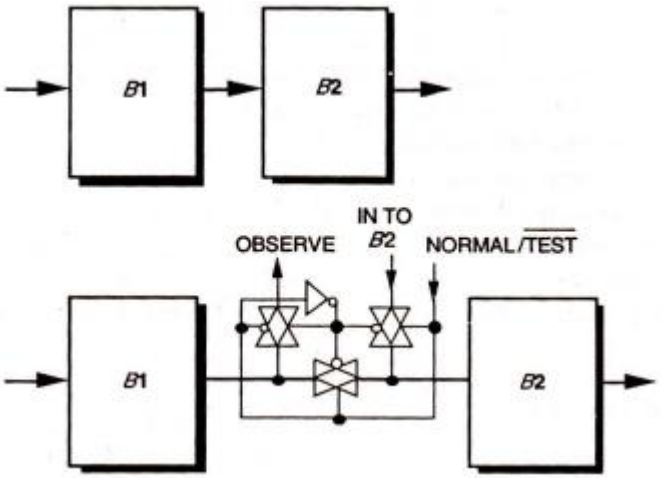
Improve controllability and observability

Design for test methods must ensure that a design is well enough covered to provide for complete and efficient testing.

When a node is difficult to access from primary input or output pads, then a very effective method is to add additional, internal pads to access the desired point.

These additional pads may be accessed using a prober.

If the node is a link between blocks of a circuit, as in Figure, Some additional circuitry will be required and a possible configuration is set out in the figure.



If the Normal/ Test line is set to 1 (Normal) then transmission gates T2 and T3 are open and T1 is closed. Normal transmission between the blocks can take place through T2 but a control input to block 2 can also be applied through T3.

When the Normal/ Test line is set to 0 (Test) then transmission gates T2 and T3 are closed, there will be no transmission between the blocks, and the output (observe) of block I can be monitored through f 1 which is now open.

This solution requires three pads and eight transistors in a CMOS environment.

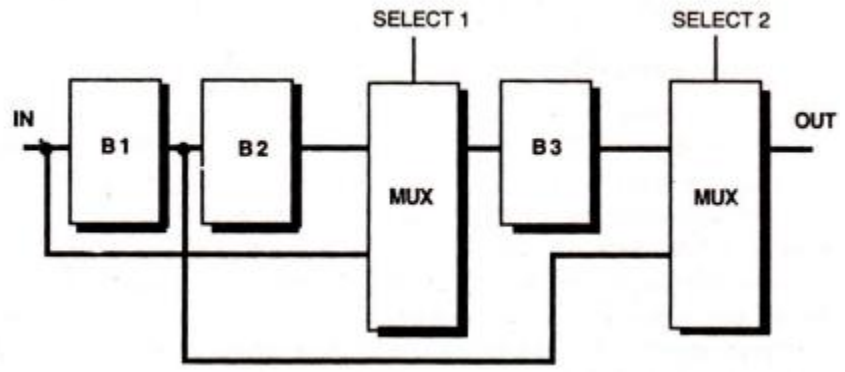
The use of Inter-block multiplexers

Some general attributes are illustrated in Figure.

This arrangement allows the bypassing of blocks.

The addition of demultiplexers also improves observability.

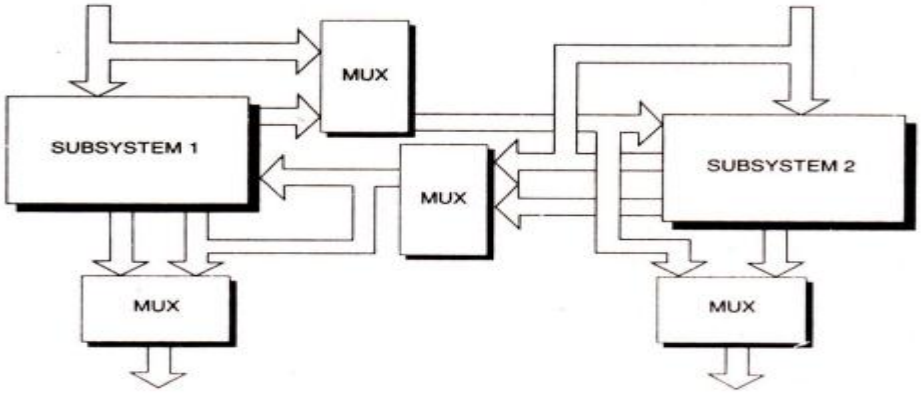
The major penalties incurred here are the numerous extra devices and the added propagation delays through the multiplexers.



The partitioning of large circuits

Partitioning large circuits into smaller subcircuits is an effective way of reducing test generation complexity and test time.

Isolation and control are readily achieved through the use of multiplexers as suggested in Figure .

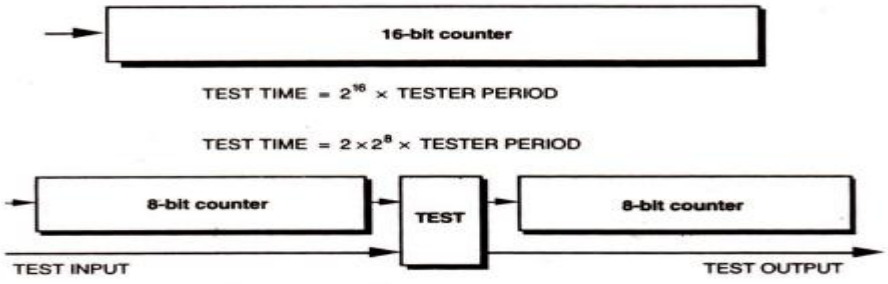


Dividing long counter chains:

Counters are sequential and need a large number of input vectors to be fully tested.

Partitioning into sub-counters can be very effective in reducing test complexity.

For example, the full testing of a 16-bit counter requires the application of 2^{16} = 65,536 clock pulses. Division of the counter into two 8-bit counters, as Figure , reduces this number to 2 x 2^{8} = 512 clock pulses.



Initialization of sequential logic

An important problem in sequential logic testing arises at power-up time where the first state will be quite random if there is no initialization.

In this case it is impossible to start a test sequence correctly.

The remedy is to design the circuit using elements which have a preset and/or clear facility (e.g. JK flip-flop elements with Pr. and Clr. inputs).

Asynchronous sequential logic

Asynchronous logic is driven by self-timing state transitions in response to changes of the primary inputs.

Although asynchronous logic is inherently faster than clocked logic it has several serious disadvantages from the test viewpoint as follows:

• testing is difficult;

• sensitivity to tester skew;

• non-deterministic behavior;

• prone to races and other hazards.

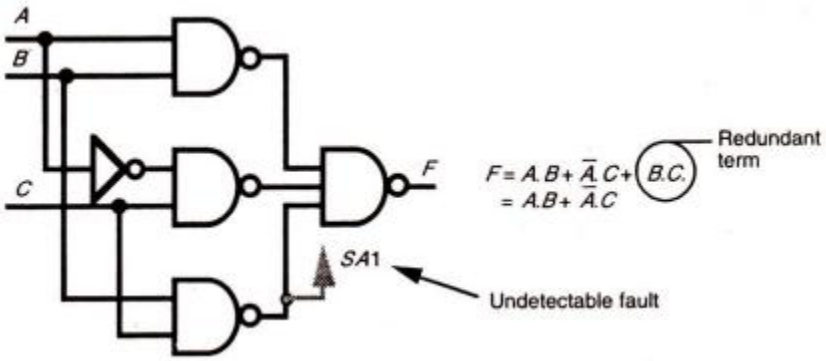
The design processes are more difficult than synchronous logic and must be approached with care, taking due account of critical race and other hazard-generating conditions.

Avoiding logical redundancy

Logical redundancy may be present by design; for example,. in order to mask a static hazard condition, or unintentionally as a design bug.

In both cases it is not possible to make a primary output value dependent on the value of the redundant node.

Thus, there are certain fault conditions associated with the node which cannot be detected, Take, for instance, the two sets of conditions outlined in Figure .



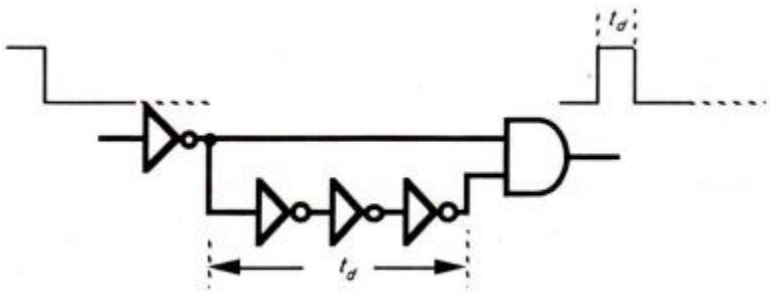
Avoiding delay dependant logic

An example of a delay-dependent circuit is given in Figure.

It will be seen that the presence of a pulse at the AND gate output depends not on the logical performance of the three inverters but rather on their temporal performance.

Automatic test pattern generators (ATPGs) work in the logic domain and view delay-dependent logic as redundant combinational logic.

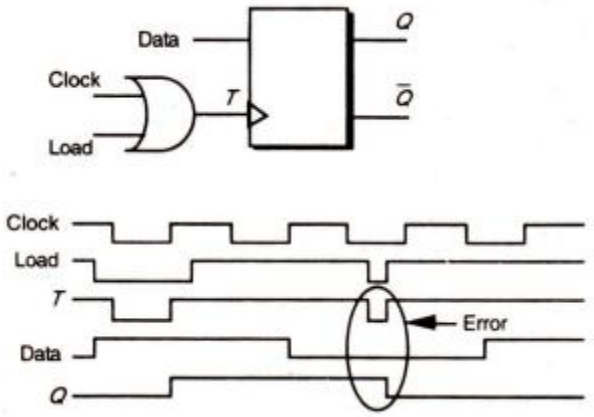
In the case illustrated in Figure , the ATPG will see the Anding of a signal with its complement and will therefore always compute a '0' as the output of the And. gate- rather than a pulse.



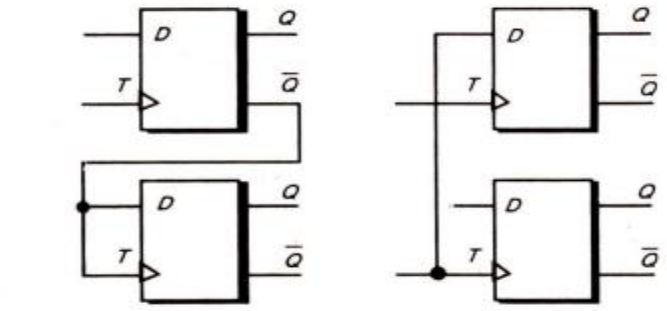
Avoiding gating or asynchronous delays

In the clock line When a clock signal is gated with another signal, such as a load signal coming from a tester, then any skew (or other hazard) on that signal can cause an erroneous output from the associated logic.

This is illustrated in Figure .



Further, another timing situation to avoid is that illustrated in Figure .

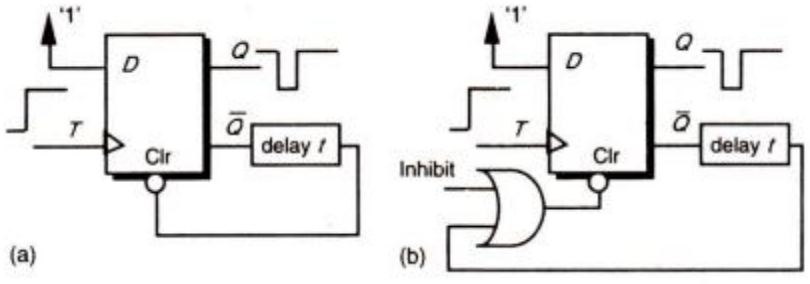


where the tester could not be synchronized if one or more clock is dependent on asynchronous delays (from the D-input to Q-input of the flip-flop in the figure), or when a signal is used both as data and as a clock.

Avoiding self-resetting logic

The problems here are akin to those in asynchronous logic, since the reset input is independent of the system clock. This can result in an erroneous value being read by the tester.

The situation is indicated in Figure(a).

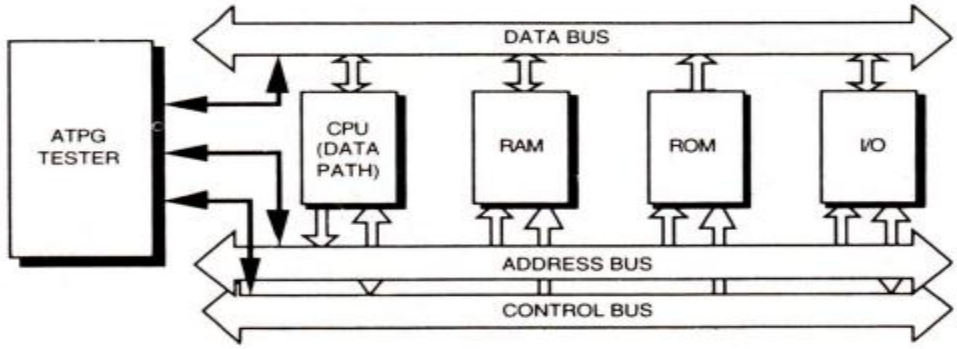


One solution to this problem is to allow the tester to override by adding an Or gate as indicated in Figure (b).

This allows the tester to receive the right response at the right time.

The use of bused structures

This approach is related to the partitioning technique and is very widely used for microprocessor like circuits as illustrated in Figure.



Using this arrangement allows the tester access to all the main subsystems and other modules which the buses interconnect.

The tester can then effectively disconnect any unit or module from the bus by putting its output into the high impedance state. Test patterns can then be applied to each separately.

Separation of analog and digital circuits

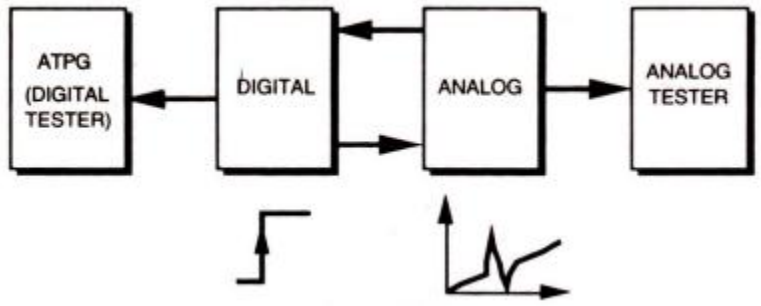
The testing of analog circuits requires a completely different strategy from digital circuits. and is therefore incompatible.

Furthermore, the fast rise and fall times of digital signals can give rise to cross-talk problems · in analog signal lines if they are in close proximity.

Where it is essential to route digital signals near analog lines, then consideration must be given to balancing and shielding the digital signals.

In the case of analog-digital converters, it is better to bring out the analog signals for observation before conversion.

For digital-analog conversion the digital signals may also be brought out for observation prior to the converter as outlined in Figure .



ADC TESTING: BRING OUT ANALOG INPUTS FOR TEST OBSERVE DIGITAL OUTPUT

DAC TESTING: BRING OUT DIGITAL INPUTS FOR TEST OBSERVE ANALOG OUTPUT

Bypassing techniques

Bypassing a subsystem consists of providing the facilities for propagating its inputs directly through to its outputs.

The aim is to bypass the sub-system in order to directly access another subsystem to be tested, and, as with partitioning, wide use is made of multiplexers to achieve the bypassing.

To speed up the testing, some subsystems may be tested simultaneously if the propagation paths are associated with other disjoint or separate subsystems.