Unit 8: Testability

Objective: At the end of this unit we will be able to understand

- Design for testability (DFT)
- DFT methods for digital circuits:
  - ❑ Ad-hoc methods
  - ❑ Structured methods:
    - Scan
    - Level Sensitive Scan Design
    - Boundary scan
    - Other Scan Techniques

Definition:

Design for testability (DFT) refers to those design techniques that make test generation and test application cost-effective.

Some terminologies:

Input / output (I/O) pads

- Protection of circuitry on chip from damage
- Care to be taken in handling all MOS circuits
- Provide necessary buffering between the environments On & OFF chip
- Provide for the connections of power supply
- Pads must be always placed around the peripheral

Minimum set of pads include:

- VDD connection pad
- GND(VSS) connection pad
- Input pad
- Output pad
- Bidirectional I/O pad

Designer must be aware of:

- nature of circuitry
- ratio/size of inverters/buffers on which output lines are connected
- how input lines pass through the pad circuit (pass transistor/transmission gate)

System delays

Buses:

- convenient concept in distributing data & control through a system
- bidirectional buses are convenient
- in design of datapath
- problems: capacitive load present

- largest capacitance
- sufficient time must be allowed to charge the total bus
- clock ϕ_1 & ϕ_2

Control paths, selectors & decoders

1. select registers and open pass transistors to connect cells to bus
2. Data propagation delay bus
3. Carry chain delay

Faults and Fault Modeling

A fault model is a model of how a physical or parametric fault manifests itself in the circuit Operation. Fault tests are derived based on these models

Physical Faults are caused due to the following reasons:

- Defect in silicon substrate
- Photolithographic defects
- Mask contamination and scratches
- Process variations and abnormalities
- Oxide defects

Physical faults cause Electrical and Logical faults

Logical Faults are:

- Single/multiple stuck-at *(most used)*
- CMOS stuck-open
- CMOS stuck-on
- AND / OR Bridging faults

Electrical faults are due to short, opens, transistor stuck on, stuck open, excessive steady state currents, resistive shorts and open.

Design for Testability

Two key concepts

- Observability
- Controllability

DFT often is associated with design modifications that provide improved access to internal circuit elements such that the local internal state can be controlled (controllability) and/or observed (observability) more easily. The design modifications can be strictly physical in nature (e.g., adding a physical probe point to a net) and/or add active circuit elements to facilitate controllability/observability (e.g., inserting a multiplexer into a net). While controllability and observability improvements for internal circuit elements definitely are important for test, they are not the only type of DFT

What can we do to increase testability?

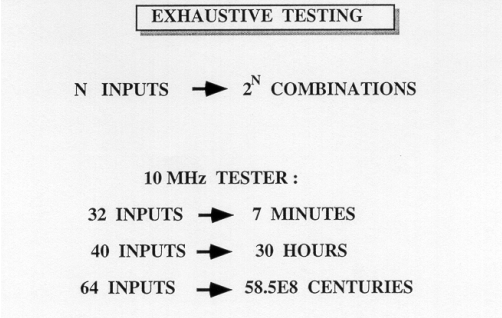
- increase observability
  - ⇨ add more pins (?!)
  - ⇨ add small "probe" bus, selectively enable different values onto bus
  - ⇨ use a hash function to "compress" a sequence of values (e.g., the values of a bus over many clock cycles) into a small number of bits for later read-out
  - ⇨ cheap read-out of all state information

- increase controllability
  - ⇨ use muxes to isolate submodules and select sources of test data as inputs
  - ⇨ provide easy setup of internal state

Testing combinational logic

The solution to the problem of testing a purely combinational logic block is a good set of patterns detecting "all" the possible faults.

The first idea to test an N input circuit would be to apply an N-bit counter to the inputs (controllability), then generate all the 2N combinations, and observe the outputs for checking (observability). This is called "exhaustive testing", and it is very efficient... but only for few- input circuits. When the input number increase, this technique becomes very time consuming.

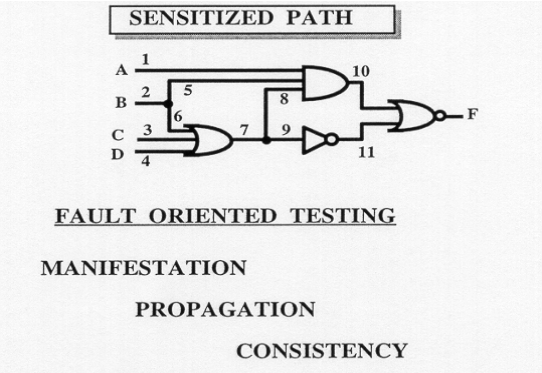


Sensitized Path Testing

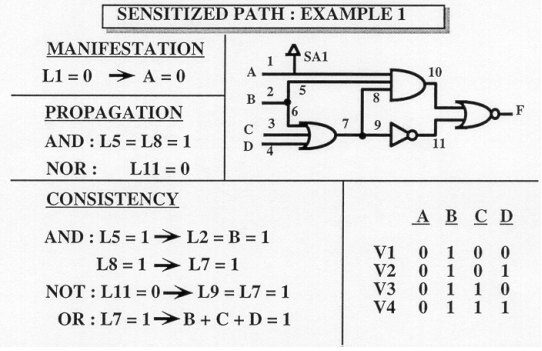
Most of the time, in exhaustive testing, many patterns do not occur during the application of the circuit. So instead of spending a huge amount of time searching for faults everywhere, the possible faults are first enumerated and a set of appropriate vectors are then generated. This is called "single-path sensitization" and it is based on "fault oriented testing".

The basic idea is to select a path from the site of a fault, through a sequence of gates leading to an output of the combinational logic under test. The process is composed of three steps :

- **Manifestation** : gate inputs, at the site of the fault, are specified as to generate the opposite value of the faulty value (0 for SA1, 1 for SA0).
- **Propagation** : inputs of the other gates are determined so as to propagate the fault signal along the specified path to the primary output of the circuit. This is done by setting these inputs to "1" for AND/NAND gates and "0" for OR/NOR gates.
- **Consistency** : or justification. This final step helps finding the primary input pattern that will realize all the necessary input values. This is done by tracing backward from the gate inputs to the primary inputs of the logic in order to receive the test patterns.



Example1 - SA1 of line1 (L1) : the aim is to find the vector(s) able to detect this fault.

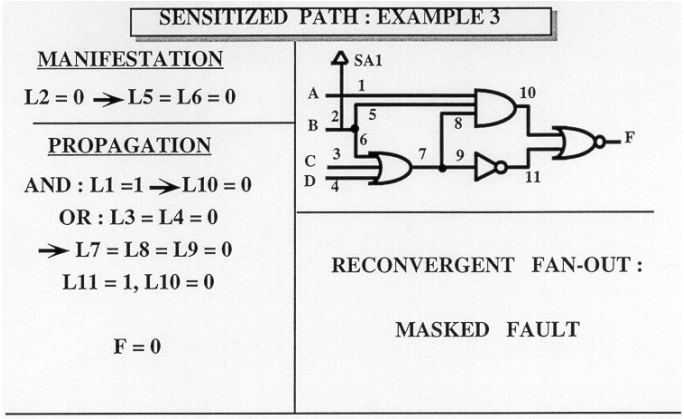


- **Manifestation:** L1 = 0 , then input A = 0. In a fault-free situation, the output F changes with A if B,C and D are fixed : for B,C and D fixed, L1 is SA1 gives F = 0, for instance, even if A = 0 (F = 1 for fault-free).
- **Propagation:** Through the AND-gate : L5 = L8 = 1, this condition is necessary for the propagation of the " L1 = 0 ". This leads to L10 = 0. Through the NOR-gate, and since L10 = 0, then L11 = 0, so the propagated manifestation can reach the primary output F. F is then read and compared with the fault-free value: F = 1.

- **Consistency:** From the AND-gate : L5=1, and then L2=B=1. Also L8=1, and then L7=1. Until now we found the values of A and B. When C and D are found, then the test vectors are generated, in the same manner, and ready to be applied to detect L1= SA1. From the NOT-gate, L11=0, so L9=L7=1 (coherency with L8=L7). From the OR-gate L7=1, and since L6=L2=B=1, so B+C+D=L7=1, then C and D can have either 1 or 0.

These three steps have led to four possible vectors detecting L1=SA1.

Example 2 - SA1 of line8 (L8) : The same combinational logic having one internal line SA1



- **Manifestation :** L8 = 0
- **Propagation:** Through the AND-gate: L5 = L1 = 1, then L10 = 0 Through the NOR-gate: we want to have L11 = 0, not to mask L10 = 0.
- **Consistency:** From the AND-gate L8 = 0 leads to L7 = 0. From the NOT-gate L11 = 0 means L9 = L7 = 1, L7 could not be set to 1 and 0 at the same time. This incompatibility could not be resolved in this case, and the fault "L8 SA1" remains undetectable.

D – Algorithm:

Given a circuit comprising combinational logic, the algorithm aims to find an assignment of input values that will allow detection of a particular internal fault by examining the output conditions. Using this algorithm the system can either be said as good or faulty. 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 time-intensive and computing intensive for large circuits.

Practical design for test guidelines

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

- facilitate test generation
- facilitate test application
- avoid timing problems

These matters are discussed as below:

Improve Controllability and Observability

All "design for test" methods ensure that a design has enough observability and controllability to provide for a complete and efficient testing. When a node has difficult access from primary inputs or outputs (pads of the circuit), a very efficient method is to add internal pads acceding to this kind of node in order, for instance, to control block B2 and observe block B1 with a probe.

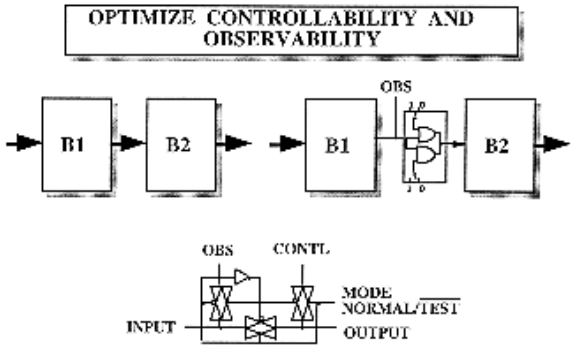


Figure 8.1 Improve Controllability and Observability

It is easy to observe block B1 by adding a pad just on its output, without breaking the link between the two blocks. The control of the block B2 means to set a 0 or a 1 to its input, and also to be transparent to the link B1-B2. The logic functions of this purpose are a NOR- gate, transparent to a zero, and a NAND-gate, transparent to a one. By this way the control of B2 is possible across these two gates.

Another implementation of this cell is based on pass-gates multiplexers performing the same function, but with less transistors than with the NAND and NOR gates (8 instead of 12).

The simple optimization of observation and control is not enough to guarantee a full testability of the blocks B1 and B2. This technique has to be completed with some other techniques of testing depending on the internal structures of blocks B1 and B2.

Use Multiplexers

This technique is an extension of the precedent, while multiplexers are used in case of limitation of primary inputs and outputs.

In this case the major penalties are extra devices and propagation delays due to multiplexers. Demultiplexers are also used to improve observability. Using multiplexers and demultiplexers allows internal access of blocks separately from each other, which is the basis of techniques based on partitioning or bypassing blocks to observe or control separately other blocks.

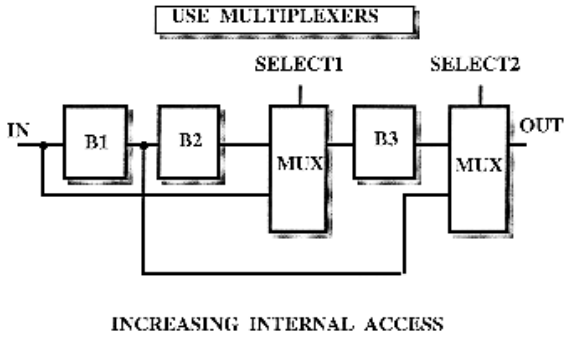


Figure 8.2: Use multiplexers

Partition Large Circuits

Partitioning large circuits into smaller sub-circuits reduces the test-generation effort. The test- generation effort for a general purpose circuit of n gates is assumed to be proportional to somewhere between n2 and n3. If the circuit is partitioned into two sub-circuits, then the amount of test generation effort is reduced correspondingly.

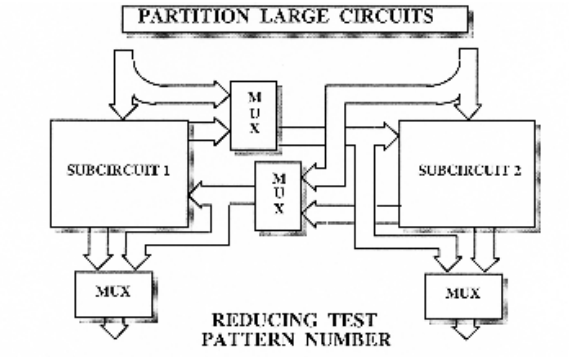


Figure 8.3: Partition Large Circuits

Logical partitioning of a circuit should be based on recognizable sub-functions and can be achieved physically by incorporating some facilities to isolate and control

clock lines, reset lines and power supply lines. The multiplexers can be massively used to separate sub-circuits without changing the function of the global circuit.

Divide Long Counter Chains

Based on the same principle of partitioning, the counters are sequential elements that need a large number of vectors to be fully tested. The partitioning of a long counter corresponds to its division into sub-counters.

The full test of a 16-bit counter requires the application of 216 + 1 = 65537 clock pulses. If this counter is divided into two 8-bit counters, then each counter can be tested separately, and the total test time is reduced 128 times (27). This is also useful if there are subsequent requirements to set the counter to a particular count for tests associated with other parts of the circuit: pre-loading facilities.

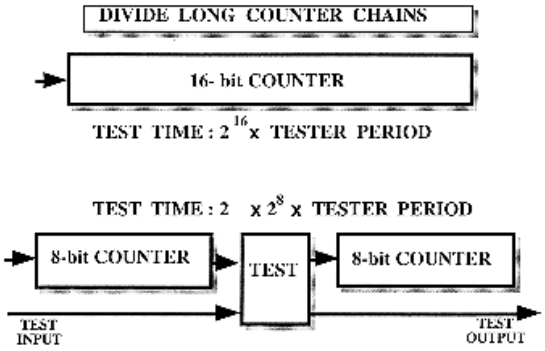


Figure 8.4: Divide Long Counter Chains

Initialize Sequential Logic

One of the most important problems in sequential logic testing occurs at the time of power-on, where the first state is random if there were no initialization. In this case it is impossible to start a test sequence correctly, because of memory effects of the sequential elements.

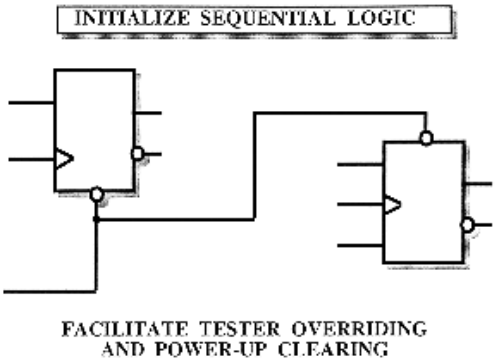


Figure 8.5: Initialize Sequential Logic

The solution is to provide flip-flops or latches with a set or reset input, and then to use them so that the test sequence would start with a known state.

Ideally, all memory elements should be able to be set to a known state, but practically this could be very surface consuming, also it is not always necessary to initialize all the sequential logic. For example, a serial-in serial-out counter could have its first flip-flop provided with an initialization, then after a few clock pulses the counter is in a known state.

Overriding of the tester is necessary some times, and requires the addition of gates before a Set or a Reset so the tester can override the initialization state of the logic.

Avoid Asynchronous Logic

Asynchronous logic uses memory elements in which state-transitions are controlled by the sequence of changes on the primary inputs. There is thus no way to determine easily when the next state will be established. This is again a problem of timing and memory effects.

Asynchronous logic is faster than synchronous logic, since the speed in asynchronous logic is only limited by gate propagation delays and interconnects. The design of asynchronous logic is then more difficult than synchronous (clocked) logic and must be carried out with due regards to the possibility of critical races (circuit behavior depending on two inputs changing simultaneously) and hazards (occurrence of a momentary value opposite to the expected value).

Non-deterministic behavior in asynchronous logic can cause problems during fault simulation. Time dependency of operation can make testing very difficult, since it is sensitive to tester signal skew.

Avoid Logical Redundancy

Logical redundancy exists either to mask a static-hazard condition, or unintentionally (design bug). In both cases, with a logically redundant node it is not possible to make a primary output value dependent on the value of the redundant node. This means that certain fault conditions on the node cannot be detected, such as a node SA1 of the function F.

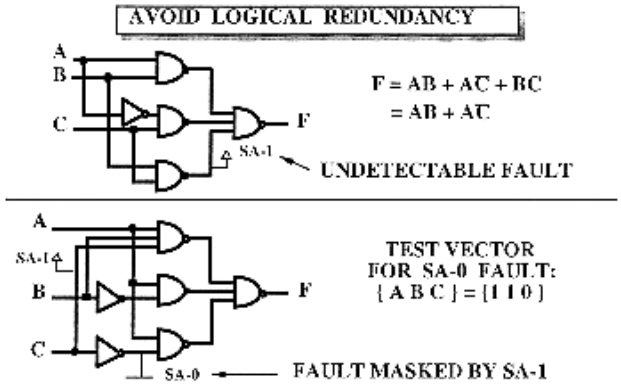


Figure 8.6: Avoid Logical Redundancy

Another inconvenience of logical redundancy is the possibility for a non-detectable fault on a redundant node to mask the detection of a fault normally-detectable, such a SA0 of input C in the second example, masked by a SA1 of a redundant node.

Avoid Delay Dependent Logic

Automatic test pattern generators work in logic domains, they view delay dependent logic as redundant combinational logic. In this case the ATPG will see an AND of a signal with its complement, and will therefore always compute a 0 on the output of the AND-gate (instead of a pulse). Adding an OR-gate after the AND-gate output permits to the ATPG to substitute a clock signal directly.

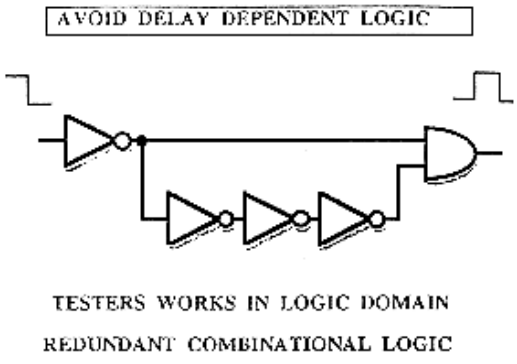


Figure 8.7: Avoid Delay Dependent Logic

Avoid Clock Gating

When a clock signal is gated with any data signal, for example a load signal coming from a tester, a skew or any other hazard on that signal can cause an error on the output of logic.

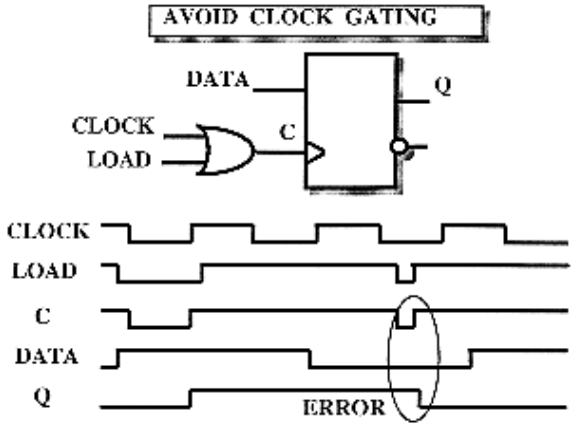


Figure 8.8: Avoid Clock Gating

This is also due to asynchronous type of logic. Clock signals should be distributed in the circuit with respect to synchronous logic structure.

Distinguish Between Signal and Clock

This is another timing situation to avoid, in which the tester could not be synchronized if one clock or more are dependent on asynchronous delays (across D-input of flip-flops, for example).

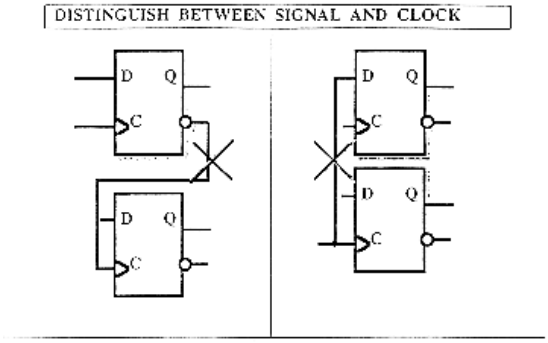


Figure 8.9: Distinguish Between Signal and Clock

Avoid Self Resetting Logic

The self resetting logic is more related to asynchronous logic, since a reset input is independent of clock signal.

Before the delayed reset, the tester reads the set value and continues the normal operation. If a reset has occurred before tester observation, then the read value is erroneous. The solution to this problem is to allow the tester to override by adding an OR-gate, for example, with an inhibition input coming from the tester. By this way the right response is given to the tester at the right time.

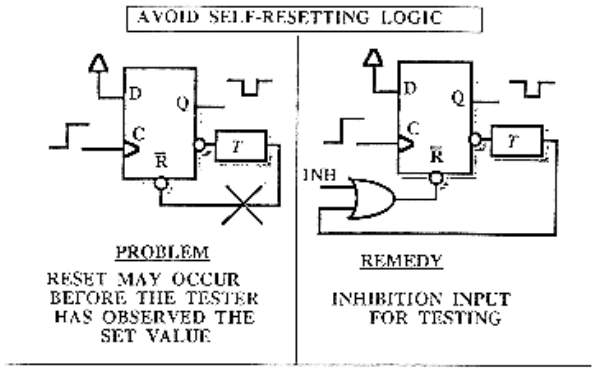


Figure 8.10: Avoid Self Resetting Logic

Use Bused Structure

This approach is related, by structure, to partitioning technique. It is very useful for microprocessor-like circuits. Using this structure allows the external tester the access of three buses, which go to many different modules.

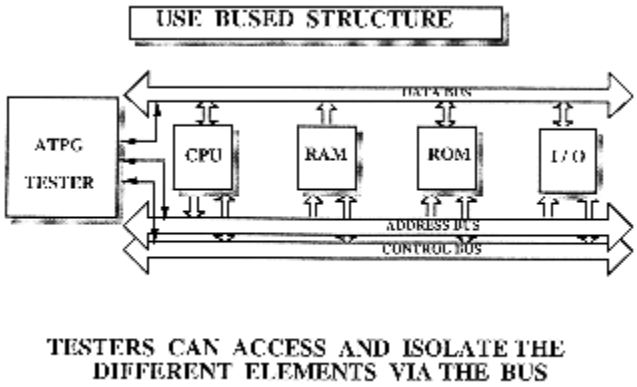


Figure 8.11: Use Bused Structure

The tester can then disconnect any module from the buses by putting its output into a high- impedance state. Test patterns can then be applied to each module separately.

Separate Analog and Digital Circuits

Testing analog circuit requires a completely different strategy than for digital circuit. Also the sharp edges of digital signals can cause cross-talk problem to the analog lines, if they are close to each other.

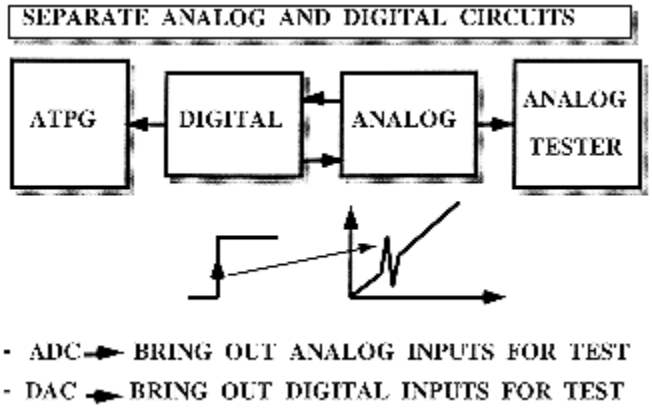


Figure 8.12: Separate Analog and Digital Circuits

If it is necessary to route digital signals near analog lines, then the digital lines should be properly balanced and shielded. Also, in the cases of circuits like Analog-Digital converters, it is better to bring out analog signals for observation before conversion. For Digital-Analog converters, digital signals are to be brought out also for observation before conversion.

Ad-Hoc DFT Method

- Good design practices learnt through experience are used as guidelines:
  - Avoid asynchronous (unclocked) feedback.
  - Make flip-flops initializable.
  - Avoid redundant gates. Avoid large fan-in gates.
  - Provide test control for difficult-to-control signals.
  - Avoid gated clocks.
  - Avoid delay dependant logic.
  - Avoid parallel drivers.
  - Avoid monostable and self-resetting logic.
- Design Reviews
  - ❑ Manual analysis
    - Conducted by experts
  - ❑ Programmed analysis
    - Using design auditing tools
  - ❑ Programmed enforcement
    - Must use certain design practices and cell types.

Objective: Adherence to design guidelines and testability improvement techniques with little impact on performance and area.

- Disadvantages of ad-hoc DFT methods:
  - Experts and tools not always available.
  - Test generation is often manual with no guarantee of high fault coverage.
  - Design iterations may be necessary.

Scan Design Techniques

The set of design for testability guidelines presented above is a set of ad hoc methods to design random logic in respect with testability requirements. The scan design techniques are a set of structured approaches to design (for testability) the sequential circuits.

The major difficulty in testing sequential circuits is determining the internal state of the circuit. Scan design techniques are directed at improving the controllability and observability of the internal states of a sequential circuit. By this the problem of testing a sequential circuit is reduced to that of testing a combinational circuit, since the internal states of the circuit are under control.

Scan Path

The goal of the scan path technique is to reconfigure a sequential circuit, for the purpose of testing, into a combinational circuit. Since a sequential circuit is based on a combinational circuit and some storage elements, the technique of scan path consists in connecting together all the storage elements to form a long serial shift register. Thus the internal state of the circuit can be observed and controlled by shifting (scanning) out the contents of the storage elements. The shift register is then called a scan path.

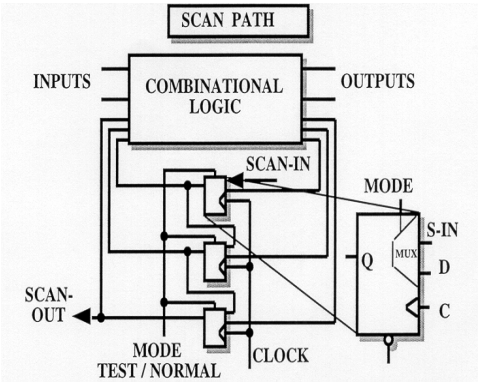


Figure 8.13: Scan Path

The storage elements can either be D, J-K, or R-S types of flip-flops, but simple latches cannot be used in scan path. However, the structure of storage elements is slightly different than classical ones. Generally the selection of the input source is achieved using a multiplexer on the data input controlled by an external mode signal. This multiplexer is integrated into the D-flip-flop, in our case; the D-flip-flop is then called MD-flip-flop (multiplexed-flip-flop).

The sequential circuit containing a scan path has two modes of operation: a normal mode and a test mode which configure the storage elements in the scan path.

As analyzed from figure 8.13, in the normal mode, the storage elements are connected to the combinational circuit, in the loops of the global sequential circuit, which is considered then as a finite state machine.

In the test mode, the loops are broken and the storage elements are connected together as a serial shift register (scan path), receiving the same clock signal. The input of the scan path is called scan-in and the output scan-out. Several scan paths can be implemented in one same complex circuit if it is necessary, though having several scan-in inputs and scan-out outputs.

A large sequential circuit can be partitioned into sub-circuits, containing combinational sub-circuits, associated with one scan path each. Efficiency of the test pattern generation for a combinational sub-circuit is greatly improved by partitioning, since its depth is reduced.

Before applying test patterns, the shift register itself has to be verified by shifting in all ones i.e. 111...11, or zeros i.e. 000...00, and comparing.

The method of testing a circuit with the scan path is as follows:

1. Set test mode signal, flip-flops accept data from input scan-in
2. Verify the scan path by shifting in and out test data
3. Set the shift register to an initial state
4. Apply a test pattern to the primary inputs of the circuit
5. Set normal mode, the circuit settles and can monitor the primary outputs of the circuit
6. Activate the circuit clock for one cycle
7. Return to test mode
8. Scan out the contents of the registers, simultaneously scan in the next pattern

Level sensitivity scan design (LSSD)

`

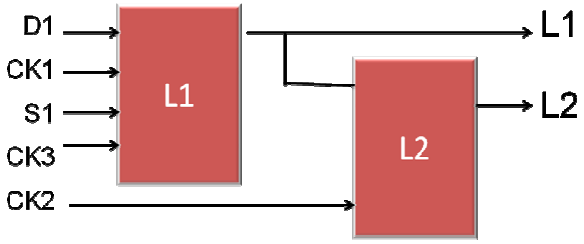


Figure 8.14: Level sensitivity scan design

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 (Figure 8.14).

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 shidt register. Each SRL has a specific design similar to a master-slave FF. it is driven by two non-overlapping clocks which can be controlled readily from the primary inputs to the circuit. Input D1 is the normal data input to the SRL; clocks CK1 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.

Advantages:

- Circuit operation is independent of dynamic characteristics of the logic elements
- ATP generation is simplified
- Eliminate hazards and races
- Simplifies test generation and fault simulation

Boundary Scan Test (BST)

Boundary Scan Test (BST) is a technique involving scan path and self-testing techniques to resolve the problem of testing boards carrying VLSI integrated circuits and/or surface mounted devices (SMD).

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

BST (figure 8.15) consists in placing a scan path (shift register) adjacent to each component pin and to interconnect the cells in order 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 through the board.

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 chip
- Deliver test data to the chips on board for self-testing
- Test the chips themselves with internal self-test

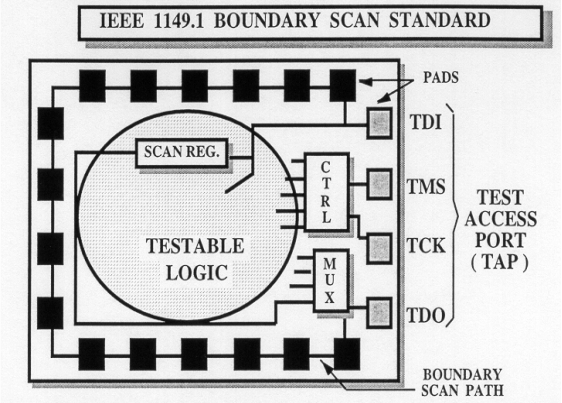


Figure 8.15: Boundary Scan Test (BST)

The advantages of Boundary scan techniques are as follows :

- No need for complex testers in PCB testing
- Test engineers work is simplified and more efficient
- Time to spend on test pattern generation and application is reduced
- Fault coverage is greatly increased.

Other scan techniques:

Partial Scan Method

- A subset of flip-flops is scanned.
- Objectives:
  - Minimize area overhead and scan sequence length, yet achieve required fault coverage
  - Exclude selected flip-flops from scan:
    - Improve performance
    - Allow limited scan design rule violations
  - Allow automation:
    - In scan flip-flop selection
    - In test generation
  - Shorter scan sequences – reduce application time

Random Access Scan Method

- The scan function is implemented like a random-access memory (RAM)
- All flip-flops form a RAM in scan mode
- A subset of flip-flops can be included in the RAM if partial scan is desired
- In scan mode, any flip-flop can be read or written

Procedure:

- Set test inputs to all test points
- Apply the master reset signal to initialize all memory elements
- Set scan-in address & data, then apply the scan clock
- Repeat the above step until all internal test inputs are scanned
- Clock once for normal operation
- Check states of the output points
- Read the scan-out states of all memory elements by applying the address

Built-in-self test

Objectives:

1. To reduce test pattern generation cost
2. To reduce volume of test data
3. To reduce test time

Built-in Self Test, or BIST, is the technique of designing additional hardware and software features into integrated circuits to allow them to perform self-testing, i.e., testing of their own operation (functionally, parametrically, or both) using their own circuits, thereby reducing dependence on an external automated test equipment (ATE).

BIST is a Design-for-Testability (DFT) technique, because it makes the electrical testing of a chip easier, faster, more efficient, and less costly. The concept of BIST is applicable to just about any kind of circuit, so its implementation can vary as widely as the product diversity that it caters to. As an example, a common BIST approach for

DRAM's includes the incorporation onto the chip of additional circuits for pattern generation, timing, mode selection, and go-/no-go diagnostic tests.

Advantages of implementing BIST include:

1) Lower cost of test, since the need for external electrical testing using an ATE will be reduced, if not eliminated
2) Better fault coverage, since special test structures can be incorporated onto the chips
3) Shorter test times if the BIST can be designed to test more structures in parallel
4) Easier customer support and
5) Capability to perform tests outside the production electrical testing environment. The last advantage mentioned can actually allow the consumers themselves to test the chips prior to mounting or even after these are in the application boards.

Disadvantages of implementing BIST include:

1) Additional silicon area and fab processing requirements for the BIST circuits
2) Reduced access times
3) Additional pin (and possibly bigger package size) requirements, since the BIST circuitry need a way to interface with the outside world to be effective and
4) Possible issues with the correctness of BIST results, since the on-chip testing hardware itself can fail.

Techniques are:

- compact test: signature analysis
- linear feedback shift register
- BILBO
- self checking technique

Compact Test: Signature analysis

Signature analysis performs polynomial division that is, division of the data out of the device under test (DUT). This data is represented as a polynomial P(x) which is divided by a characteristic polynomial C(x) to give the signature R(x), so that

$$R(x) = P(x)/C(x)$$

This is summarized as in figure 8.16.

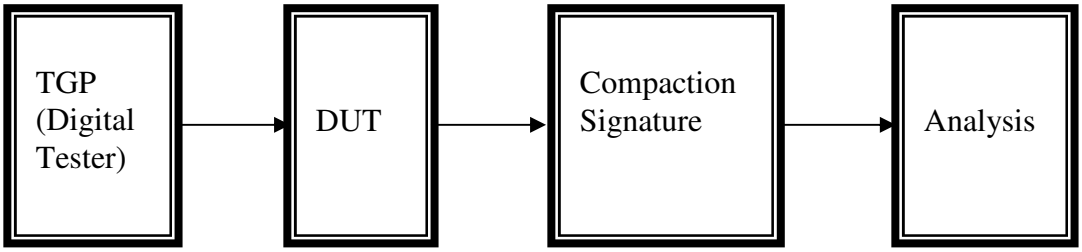


Figure 8.16: BIST – signature analysis

Linear feedback shift register (LFSR):

An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR (Figure 8.16) on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops.

LFSR technique can be applied in a number of ways, including random number generation, polynomial division for signature analysis, and n-bit counting. LFSR can be series or parallel, the differences being in the operating speed and in the area of silicon occupied; Parallel LFSR being faster but larger than serial LFSR.

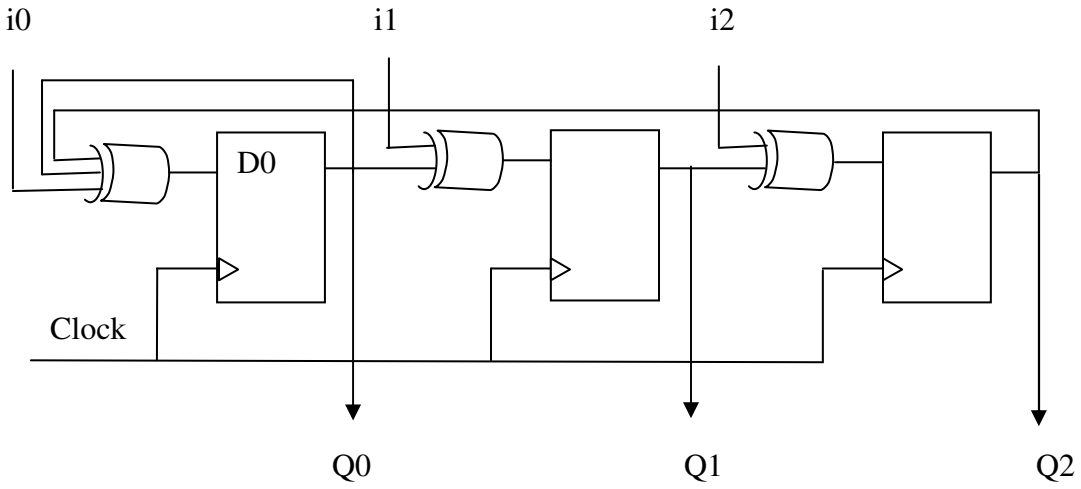


Figure 8.16: Linear feedback shift register

Built-in logic block observer (BILBO):

BILBO is a built-in test generation scheme which uses signature analysis in conjunction with a scan path. The major component of a BILBO is an LFSR with a few gates (Figure 8.17).

A **BILBO** register (**built-in logic block observer**) combines normal flipflops with a few additional gates to provide four different functions. The example circuit shown in the applet realizes a four-bit register. However, the generalization to larger bit-widths should be obvious, with the XOR gates in the LFSR feedback path chosen to implement a good polynomial for the given bit-width.

When the **A** and **B** control inputs are both 1, the circuit functions as a normal parallel D-type register.

When both **A** and **B** inputs are 0, the D-inputs are ignored (due to the AND gate connected to A), but the flipflops are connected as a shift-register via the NOR and XOR gates. The input to the first flipflop is then selected via the multiplexer controlled by the **S** input. If the S input is 1, the multiplexer transmits the value of the external **SIN** shift-in input to the first flipflop, so that the BILBO register works as a normal shift-register. This allows to initialize the register contents using a single signal wire, e.g. from an external test controller.

If all of the **A**, **B**, and **S** inputs are 0, the flipflops are configured as a shift-register, again, but the input bit to the first flipflop is computed by the XOR gates in the LFSR feedback path. This means that the register works as a standard LFSR pseudorandom pattern generator, useful to drive the logic connected to the Q outputs. Note that the start value of the LFSR sequence can be set by shifting it in via the SIN input.

Finally, if **B** and **S** are 0 but **A** is 1, the flipflops are configured as a shift-register, but the input value of each flipflop is the XOR of the D-input and the Q-output of the previous flipflop. This is exactly the configuration of a standard LFSR signature analysis register.

Because a BILBO register can be used as a pattern generator for the block it drives, as well provide signature-analysis for the block it is driven by, a whole circuit can be made self-testable with very low overhead and with only minimal performance degradation (two extra gates before the D inputs of the flipflops).

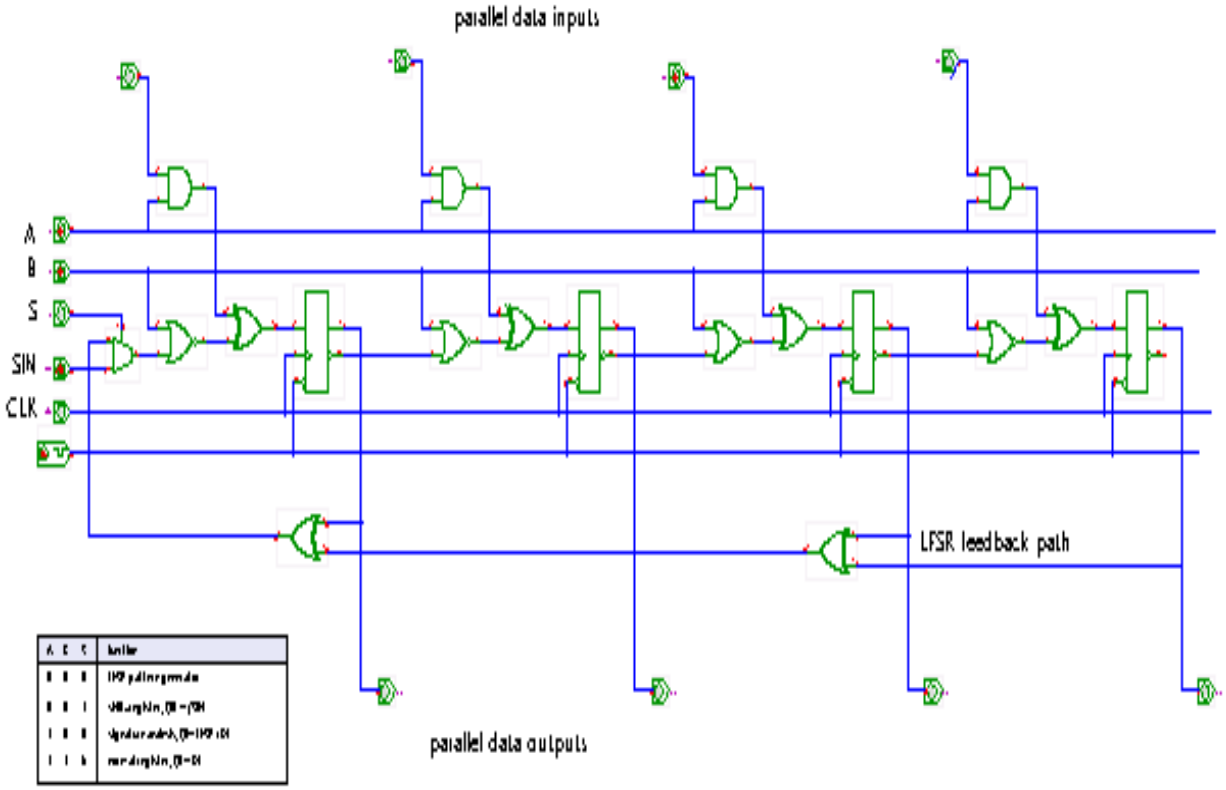


Figure 8.17: BIST – BILBO

Self-checking techniques:

It consists of logic block and checkers should then obey a set of rules in which the logic block is 'strongly fault secure' and the checker 'strongly code disjoint'. The code use in data encoding depends on the type of errors that may occur at the logic block output. In general three types are possible:

- Simple error: one bit only affected at a time.
- Unidirectional error: multiple bits at 1 instead of 0 (or 0 instead of 1)
- Multiple errors: multiple bits affected in any order.

Self-checking techniques are applied to circuits in which security is important so that fault tolerance is of major interest. Such technique will occupy more area in silicon than classical techniques such as functional testing but provide very high test coverage.