

JESD12-5

JEDEC STANDARD

Design for Testability Guidelines

JEDEC Standard No. 12-5

(Addendum No. 5 to JEDEC Standard No. 12)

AUGUST 1988

ELECTRONIC INDUSTRIES ASSOCIATION

ENGINEERING DEPARTMENT



Documents

Document Sharing Hub

N O T I C E

JEDEC Standards and Publications contain material that has been prepared, progressively reviewed, and approved through the JEDEC Council level and subsequently reviewed and approved by the EIA General Counsel.

JEDEC Standards and Publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such standards shall not in any respect preclude any member or nonmember of JEDEC from manufacturing or selling products not conforming to such standards, nor shall the existence of such standards preclude their voluntary use by those other than EIA members whether the standard is to be used either domestically or internationally.

JEDEC Standards or Publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC Standards or Publications.

The information included in JEDEC Standards and Publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC Standard or Publication may be further processed and ultimately become an EIA Standard.

Inquiries, comments, and suggestions relative to the content of this JEDEC Standard or Publication should be addressed to the JEDEC Executive Secretary at EIA Headquarters, 2001 Eye Street, N.W., Washington, D.C. 20006.

COPYRIGHT 1988
ELECTRONIC INDUSTRIES ASSOCIATION

Published by
ELECTRONIC INDUSTRIES ASSOCIATION
Engineering Department
2001 Eye Street, N.W.
Washington, D.C. 20006

PRICE: \$30.00

Printed in U.S.A.

Design For Testability
Guidelines

Prepared by:

JEDEC

Committee on Semicustom Integrated Circuits

DESIGN FOR TESTABILITY GUIDELINES

TABLE OF CONTENTS

1. Introduction
2. Terms and Definitions
3. Structured and Non-structured Approaches to Designing For Testability
4. Isolation Techniques for Megacell/Memory/Analog Structures in Semicustom IC's to Enhance Testability
5. Automatic Test Vector Generation Techniques
6. Test Program Generation From Simulation Files
7. Enhancing DC Parametric and Board Level Testing
8. Summary and Conclusions

Appendix A

Bibliography of Published Literature

NOTE: Two chapters are in the process of completion and will be included in a future edition:

- 1) Packaging Impact on Design For Testability
- 2) Fault Detection, Fault Coverage, Fault Simulation and Testability Analysis

DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 1

INTRODUCTION

(From JEDEC Council Ballot JCB-87-46, formulated under the cognizance of JC44 Committee on Semicustom Integrated Circuits.)

The concern most often voiced by Application Specific Integrated Circuit (ASIC) users is that of testability. Many vendors have taken steps to try to minimize the concerns of their customers but, to date, each company has instituted its own policies and the customers have seen little to ease their confusion. This document is composed of inputs from many IC manufacturers and some IC users. It is intended to bring together a coherent approach to designing for testability. It is not intended as a specification, nor is it to be interpreted as the only way to design. It should, rather, be used as guidance (as the title implies) when designs are being initiated.

As the reader progresses through this guide he will notice that there is no list of authors and no company labels are shown. This was not done to slight the authors or minimize the contributions that they and their companies made. It was done, instead, to try to emphasize the broad base of contributors and companies who support these guidelines.

The second section of this guide contains definitions for most of the terms commonly used with reference to designing for testability. These definitions are the result of industry-wide discussion and, as such, represent the consensus of the industry.

Several topics will be covered. Perhaps the item that leads to the most discussion is Fault Simulation. This technique enables design engineers to evaluate their test patterns (input stimulus and expected output values) to determine whether these patterns will detect faults (errors in the circuitry) that may occur during either the design or processing stages. A fault simulator uses fault models, such as a node shorted to power (stuck-at-one) or a node shorted to ground (stuck-at-zero), and compares the response of a fault-free circuit with the response of a faulty circuit after applying test patterns supplied by the design engineer. If the response of the fault-free circuit is different than the response of the faulty circuit, then the test patterns have detected the fault. By faulting all the nodes in the circuit, the fault simulator will produce the test pattern fault coverage (the percentage of faults detected vs. total faults tested). The higher the fault coverage, the better the



DESIGN FOR TESTABILITY GUIDELINES

test pattern will separate a faulty circuit from a fault-free circuit. By analyzing which faults have not been detected by the current set of test patterns, additional test patterns can be generated by the design engineers in order to detect the faults which were missed. Many vendors recommend greater than 90 percent fault coverage.

Designing testability into any circuit will affect the hardware to some degree. Additional logic will most probably have to be included for any of the methods outlined in this guide. This additional logic will increase the amount of silicon required to implement the design and therefore increase the cost. The savings from enhanced testability do not usually show up until the testing cost of the part and its end system are analyzed. These costs include the labor required to generate the test vectors, the computer time to evaluate the vectors, and the actual time required to test the part. Depending on how the additional circuitry is implemented, the ac performance of the part may be degraded. Each of these factors will also be discussed.

The final section of this guide is a bibliography of documents that the reader may want to investigate. While this list is extensive, it is by no means all-inclusive. The reader is encouraged to do some follow-up investigation after finishing this document.



DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 2

TERMS AND DEFINITIONS

1. Node:

The end-point of a branch in a network or a point at which two or more branches meet.

2. Fault:

A defect that may cause a failure in the circuit operation and/or timing.

NOTE:

Subclassifications of faults may not be mutually exclusive.

2.1 Parametric Fault:

A fault in a circuit that causes failure to meet ac or dc specifications but does not cause functional failure.

2.2 Functional Fault:

A fault that causes improper logical operation of a circuit.

2.2.1 Combinational Fault:

A functional fault that is not affected by the sequence of the input stimuli.

2.2.2 Sequential Fault:

A functional fault that is affected by the sequence of the input stimuli.

3. Stuck-at-0 Fault:

A fault in a digital circuit characterized by a node remaining at a logic low (0) state regardless of changes in input stimuli.

3.1 Stuck-at-1 Fault:

A fault in a digital circuit characterized by a node remaining at a logic high (1) state regardless of changes in input stimuli.

DESIGN FOR TESTABILITY GUIDELINES

4. Short Circuit Fault:

A fault in a circuit that alters the number of nodes by connecting two or more nodes together.

4.1 Open Circuit Fault:

A fault in a circuit that alters the number of nodes by breaking a node into two or more nodes.

5. Test Vector:

A single instance of input stimuli and expected output responses.

6. Test Pattern:

A set of test vectors.

7. Test Program:

A test pattern and instructions suitable for use on Automatic Test Equipment.

Note:

A test program may be used to perform functional and parametric (ac, dc, or other) tests.

8. Detectable Fault:

A functional fault for which a test pattern can be created that will always cause the effects of the fault to be observable at an externally accessible node.

8.1 Detected Fault:

A functional fault that causes effects that are observed at an externally accessible node when the circuit is exercised by the existing test pattern.

8.2 Undetected Fault:

A functional fault that causes effects that are not observed at an externally accessible node when the circuit is exercised by the existing test pattern.

DESIGN FOR TESTABILITY GUIDELINES

9. Undetectable Fault:

A functional fault for which no test pattern can be created that will cause the effects of the fault to be observable at an externally accessible node.

10. Test-Pattern Fault Coverage:

The ratio of the total number of detected faults to the total number of detectable faults.

11. Fault detectability Ratio:

The ratio of detectable faults to the sum of detectable and undetectable faults.

12. Fault Grading:

The process of determining the test-pattern fault coverage of a circuit.

13. Fault-Tolerant Design:

A design approach to enhance the ability of a circuit to remain operational after the occurrence of a fault.

Note:

Fault-tolerant design techniques may impact fault detection.

14. Controllability:

The ability of a node to be established at specific logic state(s) by applying stimuli to the circuit's externally accessible node(s).

15. Observability:

The ability to determine the logic state(s) of a node at the circuit's externally accessible node(s).

16. Circuit Initialization:

A sequence of stimuli that set internal nodes of a circuit to a predictable state.

JEDEC Standard No. 12-5
Page 6

DESIGN FOR TESTABILITY GUIDELINES

(Intentionally left blank)



Documents
Document Sharing Hub

DESIGN FOR TESTABILITY GUIDELINES

CHAPTER 3

STRUCTURED AND NON-STRUCTURED
APPROACHES TO DESIGNING FOR TESTABILITY

Design For Testability (DFT) refers to a design approach that enables thorough testing of a system with minimal effort and maximum coverage. A circuit is termed testable when the testing effort involved is minimal compared to the design and manufacturing efforts. The key concepts in DFT are controllability and observability. (See definitions in Chapter 2.) The purpose of the various DFT techniques is to increase the ability to control/observe internal nodes from external inputs/outputs.

Some states of a circuit are buried in logic and cannot easily be controlled or observed by external pins. This makes the generation of test vectors more time consuming and usually results in more test vectors being generated. This problem can be addressed by adding special test pins that increase the controllability or observability of these states. If extra pins are not available, on-chip test logic can be used to test portions of the circuit while it is operating in a test mode. Input combinations that cannot occur during the normal operation of the circuit can be used to place the circuit in the test mode. On-chip test circuitry can also be used to partition the circuit into logical subsystems that can be tested in parallel. Testing smaller logical subsystems simplifies the task of test vector generation. Because the logical subsystems operate in parallel, these test vectors may be merged together, resulting in reduced test time on the tester.

To help solve the testing problem there are two basic approaches prevalent in the industry. The first approach is categorized as ad hoc or nonstructured and the second approach is categorized as a structured approach. The nonstructured techniques are the ones that can be applied to a given product, but are not directed at solving the general problem of testing sequential logic. Non-structured techniques usually provide specific solutions at lower cost than structured approaches would, but without the general applicability. The structured techniques tend to solve the general problem with design methodology and lend themselves more easily to design automation.

The implementation of a structured approach to DFT begins with the design process. Most structured techniques rely on the "shift register" method which connects all memory elements (latches,

DESIGN FOR TESTABILITY GUIDELINES

flip-flops) into a shift register and uses each element as pseudo-input/output for testing purposes. Converting sequential logic into combinational circuits simplifies test pattern generation and reduces test development costs. There are a number of DFT techniques used by VLSI chip designers; three are briefly discussed below.

ISOLATION OF FUNCTIONAL BLOCKS

One of the simplest methods of designing testability into a circuit is to partition the functional blocks such that each may be individually tested as a separate unit. In an integrated circuit, only the nodes at the external pins can be controlled and observed directly. In order to isolate the functional blocks of the chip, internal nodes must be made controllable and observable from the external pins. Additional logic such as multiplexers may have to be added to perform this function. This, in turn, will increase gate count. If the design is implemented in gate arrays, enough extra logic may be required as to force the use of the next larger array. In cell based designs, each added gate will increase die size. The total number of internal signals that must be isolated will dictate the exact amount of additional logic that will have to be added. If extra package pins are available, internal signals may be brought directly to the outside world for observation at no additional cost. If package pins are not available, however, bringing internal signals out will force the use of a larger package. Costs will vary with the package selection.

Any added logic will have an effect on the ac performance of the circuit. If the added logic is to be placed in a critical timing path, a trade off will result between performance and testability.

DESIGN FOR TESTABILITY GUIDELINES

SCAN DESIGN

There are many varieties of scan design and each has its own impact on the hardware. All of these techniques ease the problem of testability by making the circuit appear to be structured as a combinational network. The nature of scan design makes it very easy to control and observe every bi-stable element in the circuit. This discussion will be limited to the following scan design methods:

Level Sensitive Scan Design (LSSD)
Scan Path
Random Access Scan
Two Port Flip-Flop

It should be noted that several of these methods have very restrictive design rules and force a specific design methodology.

LEVEL SENSITIVE SCAN DESIGN (LSSD)

The LSSD method (developed and trademarked by IBM) represents one of the best known and most successful approaches. "Level sensitive" refers to constraints requiring logic subsystems to be insensitive to ac characteristics such as rise time, fall time and minimum circuit delays. LSSD requires that the designer adhere to a very strict set of design rules. These eliminate many possible test problems and ease the automatic generation of test vectors. All bi-stable elements in the circuit must be implemented as level sensitive latches.

The key element in this design is the "Shift Register Latch" (SRL) as shown in Figure 3-1a. The SRL is a level sensitive device that does not contain hazards or race conditions. Figure 3-1b shows that the SRL includes a system input (D), clock (C), external scan-in input (SI), two nonoverlapping clocks (A and B), and scan-out output (SO). Latches L1 and L2 function in a master/slave configuration during the test mode while L1 also functions as a storage element during the system mode.

Interconnecting several SRLs gives the circuit a scan capability and reduces the testing of a sequential circuit to the testing of a combinatorial circuit (Figure 3-1c). The functional operation of combinatorial logic can be tested independent of any timing considerations other than adequately slowing the clock. This method permits partitioning a large network into manageable segments for which tests are easily generated. The number of test

DESIGN FOR TESTABILITY GUIDELINES

patterns required is reduced since multiple portions of the circuit are tested simultaneously.

There are a number of negative factors associated with the LSSD approach. First, the SRLs are 2-3 times as complex as simple latches. The silicon overhead to implement the SRLs is in the range of 4 to 20 percent. However, this overhead can be reduced if the L2 latches can also be utilized to perform system functions. Up to four additional pins are required to implement LSSD, which could affect the circuit size and the package selection. By making functional use of some primary input/outputs required to operate the SRLs, the number of extra pins can be reduced.

The LSSD structured design approach for DFT eliminates or alleviates some of the problems in designing and testing VLSI chips at a reasonable cost.

SCAN PATH

The scan path method is very similar to the LSSD approach, but differs in the latch design and clock mechanism. The basic concept involves serially shifting test data into the internal memory elements and scanning out the test results. This allows initialization of all the internal memory elements to desired states and permits the contents of the buried registers to be easily examined.

The storage element consists of a latch or D flip-flop clocked by CP with a multiplexer at the input, as shown in Figure 3-2. The multiplexer inputs D and T1 represent the system and test data inputs selected by the TEST ENABLE (TE) input signal. The Q output of one flip-flop then feeds the SI input of the next flip-flop. This is done until the SCAN ring is formed. Some vendors offer macro cell libraries including latches and flip-flops with built-in scan circuitry. Scan path testing also requires three additional pins (Test Input, Test Enable and Test Output). Automatic test pattern generators exist which can achieve very high fault coverage using any of the methods discussed. There are some disadvantages however. Adding logic into the data path limits the speed at which the circuit can be clocked. The hardware overhead is related to the number of registers, since each requires a two to one multiplexer. Additional routing must be included for the select signal which connects to each register and for the interconnection of the scan data path.

Experimental results show that the scan path method reduces test generation time by an order of magnitude. The additional

DESIGN FOR TESTABILITY GUIDELINES

circuitry required to implement this scheme ranges from 5 to 15 percent.

RANDOM ACCESS SCAN

Random access scan requires a very similar latch to those used in LSSD or scan path methods, with the exception that a decode input has been added. Rather than connect the scan registers into a serial shift register, the scan registers are individually addressable. The scan data input lines are all tied in common. A decoder must be included in the circuit design in order to get the data into the scan registers. A similar approach must be used to get the data from the scan registers to the outside world. The hardware overhead is approximately the same as for other methods. However, because of the large number of decode lines to be routed the routing overhead is quite high. There is no fixed order in which the data must be loaded or unloaded. If only a few register states change from one test vector to another, only those registers that change must be written. A reduction in test vector length may be an advantage.

DUAL PORT FLIP FLOPS

The dual port flip flop is very similar in concept and implementation to Stanford scan path designs. Rather than place a two to one multiplexer at the D input of the register, which adds extra delay to the data path, two data inputs to the latch section are included. No select signal is used as the selection of data is performed by clocking one of two clocks. If the normal system clock is pulsed, data present at the D input is latched into the register. If the scan clock is pulsed, data at the scan data input is latched into the register. The actual circuit area for implementation is approximately the same as for Stanford scan path. Routing penalty is also similar because a second clock must be routed rather than a test mode pin. The advantage of no AC speed penalty is a plus for this method.

BUILT-IN LOGIC BLOCK OBSERVATION (BILBO)

The previously mentioned techniques require external equipment to generate and apply the test vectors to the circuit. The BILBO technique, shown in Figure 3-3, includes test vector generation and response evaluation on the chip. Such an approach allows a minimum of both tester intervention and test overhead in terms of dedicated test I/O pins. The response of a circuit to a given stimulus gets fed to a Linear Feedback Shift Register (LFSR). The

DESIGN FOR TESTABILITY GUIDELINES

input data is divided by the response and the remainder corresponds to the cyclic code. The cyclic code represents the signature that corresponds to the test input, which enables comparing the code with a predetermined correct signature value.

BILBO can operate in four different modes:

- a) Latch mode
- b) Shift mode
- c) Multiple-input Signature register
- d) Reset mode

The BILBO approach permits functionally converting the system flip-flops to a linear feedback shift register, keeping the overhead low. This technique saves both off-line test generation time and test application time. BILBO needs, at most, two pins: one for setting the chip in the test mode and another for reading the go/no-go information from the signature register.

The overhead in terms of silicon real estate is about 5 to 15 percent and there is some loss of performance due to delays in the Exclusive-OR gates.

BUILT-IN TEST (BIT) AND BUILT-IN SELF-TEST (BIST)

There are many types of built in test and built in self tests circuits. Some of the methods use portions of the system circuitry for self oscillation test or polynomial generation. These require very little hardware overhead, but their test coverage is based upon probability rather than exhaustive coverage. An advantage is that these methods allow in-system test with little difficulty.

GOOD ENGINEERING PRACTICES TO ENHANCE TESTABILITY

Logic Design With Testability in mind

Test program development and debugging can be very time consuming and costly unless designers consider test issues during the logic design phase. Due to the increased complexity of VLSI chips, it is not feasible for test engineers to independently develop and debug test programs. The responsibility now lies with

DESIGN FOR TESTABILITY GUIDELINES

logic designers to design logic such that it is easy to test and requires a reduced set of vectors to thoroughly test the device.

Various techniques can be applied to ensure that a design is testable. Such techniques essentially improve controllability, observability and predictability of a design. Some of these techniques are fairly simple and can be implemented with minimum overhead, and they aid in the development and debugging of the circuit and its test program. A brief description of some testability guidelines is given below.

1. Initialize the circuit to a known state

The entire circuit should be initializable to a known state without requiring knowledge of the current state. In order to easily predict the output of a circuit, all the latches and flip-flops must have set or reset capability. Circuits that never need to be set to a particular state during normal operation, such as frequency dividers (Figure 3-4a), still require a set/reset capability. The tester needs to know how many pulses to apply before the first pulse appears on the output, and this depends on the initial state of the circuit. A power-on clear mechanism, which initializes the circuit in the actual system configuration, may not be an adequate test initialization implementation. Although this function must also be tested, the tester loses many of its utility programs if initialization is performed only in this manner and overall testing suffers dramatically. By adding the special set or reset signal, the circuit can be directly set into a known state and the tester is guaranteed that the first pulse will appear after a known number of cycles and all of the tester's utility programs will still be intact. (Figure 3-4b).

2. Avoid asynchronous circuitry

For asynchronous systems, changes in the output state depend on circuit delays that are a function of wafer processing. Due to the variation in process parameters, there is no guarantee as to when an output will change state. (It may be at the end of one cycle or at the beginning of another.) Since the test vectors must be determined before the test is started, it is very difficult to keep the tester synchronized with the circuit. Therefore, testing of asynchronous circuitry is difficult and costly (and in some cases, impossible).

In an asynchronous SR Latch, for example Figure 3-5a, set and reset signals can change state at any time, causing the output to change state at any time during any period. The output transition

DESIGN FOR TESTABILITY GUIDELINES

is dependent on the delays that create the set and reset signals. Addition of the clock (Figure 3-5b) ensures that if the set or reset signal changes state, the output will change only after the next clock transition.

3. Allow internal clocks to be bypassed from the circuit's inputs

On-chip oscillator circuitry or other periodic pulse generating circuitry makes synchronization of the tester and the circuit very difficult (Figure 3-6a). By allowing these clock signals to be bypassed and generated by the tester, the tester is automatically synchronized with the circuit (Figure 3-6b). It is desirable to output any bypassed clock signal as a circuit test.

4. Allow counters and dividers to be bypassed

Bypassing sequential circuits decreases the number of test vectors required to test the device and reduces the test time. The sequential circuitry is tested once and then is bypassed while the rest of the circuit is being tested.

When 10 latches are used to divide a 1 MHz clock signal down to 1 kHz, the tester has to apply 1000 test vectors for each 1 kHz pulse (Figure 3-7a). However, when bypass circuitry is available, the divider can be tested once and then bypassed. The remainder of the circuit can be tested with each test vector creating a 1 kHz pulse (Figure 3-7b).

5. Partition circuitry that has more than 8 bits of cascaded latches

The number of vectors required to step through all the bits in cascaded latches is two to the power of the number of bits that are cascaded. A 24 bit counter would require 16 million vectors to count through its sequence (after initialization), resulting in a ridiculous amount of time and cost for testing (Figure 3-8a). After partitioning the counter into 3 groups of 8 bits, 768 vectors are required to test the counter, each group requiring 256 vectors, (Figure 3-8b). The circuit used to partition the counter can pass the count signal forward to the output or can be used directly to set the second or the third section of the partitioned circuitry.

6. Provide direct access to the control inputs of each functional module

DESIGN FOR TESTABILITY GUIDELINES

By applying a stimulus directly to a module without passing the simulator through complex circuitry, the number of test vectors can be minimized.

To stimulate functional module 3 (Figure 3-9a), signals must propagate through modules 1 and 2. If modules 1 and 2 contain sequential circuitry, a large number of vectors will be required to stimulate the inputs of module 3. By allowing module 3 to receive its signals either from module 2 (bypassing module 1) or directly from the circuit inputs, the number of test vectors required to test the device can be reduced (Figure 3-9b).

7. Provide direct access to observe the outputs of each functional module.

The response for a particular module can be observed at the circuit output without propagating it through other complex modules.

When trying to observe the response of functional module 1 (Figure 3-10a), the response must propagate through modules 2 and 3. If modules 2 and 3 contain sequential circuitry, a large number of vectors will be required to observe the response from module 1 at the output of the circuit. By allowing module 1 to feed directly to the circuit output, the response values can be observed with a reduced set of test vectors (Figure 3-10b).

8. Avoid nesting of sequential circuits

In sequential circuits, the next state is a function of the present state and current stimulus while the present state is determined by the stimulus applied previously. Therefore, multiple patterns need to be applied to control or observe the sequential circuit. The length of the test pattern increases dramatically if one sequential circuit feeds back to another sequential circuit. This is also known as nesting of sequential circuitry.

In the example, (Figure 3-11a) many vectors are required to stimulate the modules or to pass the response to the rest of the circuit. By partitioning the sequential circuits and breaking the feedback paths, controllability and observability can be improved (Figure 3-11b).

DESIGN FOR TESTABILITY GUIDELINES

9. Allow redundant circuitry to be tested

Redundant circuitry is often included to improve reliability or fabrication yields. Access must be provided to allow both portions of circuitry to be tested separately to ensure that they both work.

Figure 3-12a shows a circuit with redundancy. To insure that input B is noticed at the output whenever it goes high, this circuit was designed with multiple paths for signal B to reach the output. This redundancy makes it impossible to test for the short to ground, as shown. By adding steering logic, both signal paths can be stimulated separately to insure that they are functioning correctly (Figure 3-12b).

10. Watch out for signals that reconverge

Signals that fan out in multiple directions from a single source and later reconverge at the inputs to a functional module can create logic hazards and untestable circuits.

The circuit shown in Figure 3-13a is hazard-free but has an untestable fault. When signal B or C is HIGH, signal A can change state without causing a spike on the output. If B and C are both HIGH, the output of G4 will remain HIGH regardless of the value of G2. Therefore a short at the output of G2 will not be detected. By breaking the reconvergent path of the signal A, as shown in Figure 3-13b, and by making the test input LOW, the output of G2 can be tested. This additional circuitry greatly enhances the testability of the circuit.

DESIGN FOR TESTABILITY GUIDELINES

SUMMARY

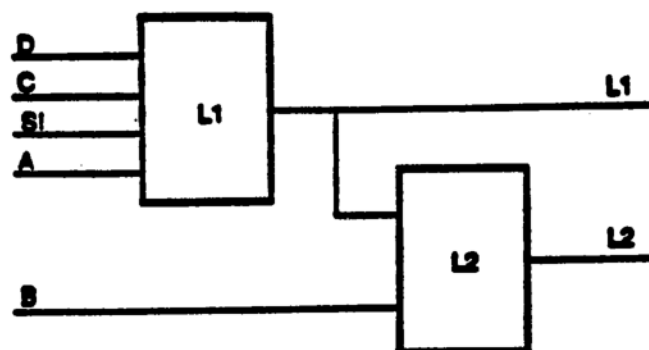
This chapter has presented several methods of designing testability into circuits. An attempt was made to focus on the subjects of chip size and performance impacts. Only general methods were discussed, as methods vary from vendor to vendor, causing each to have a different effect on the hardware.



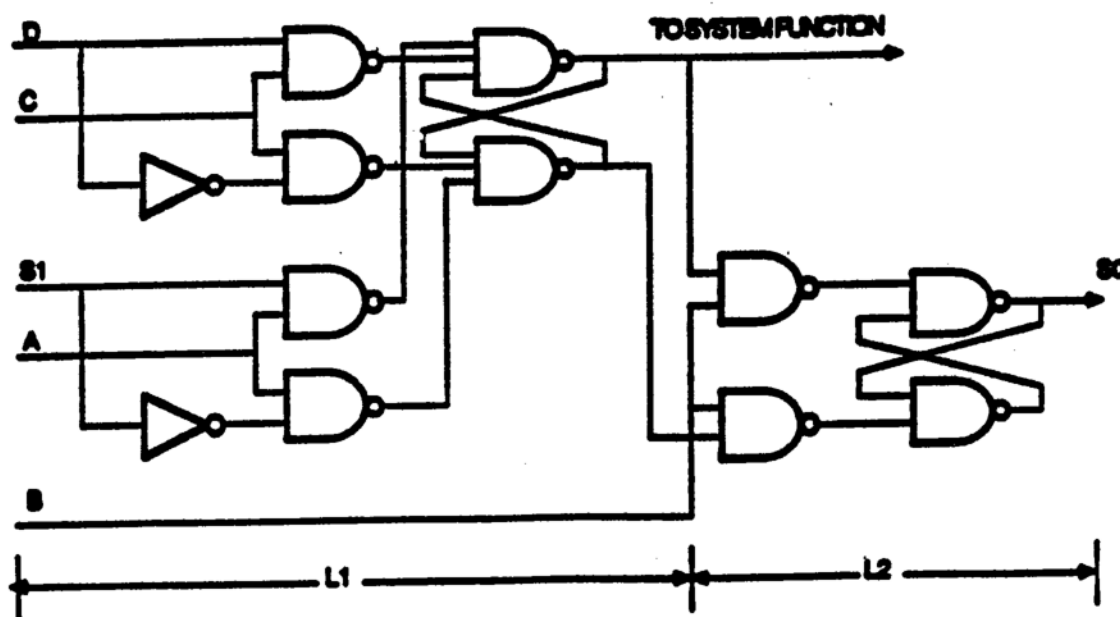
DESIGN FOR TESTABILITY GUIDELINES

Figure 3-1
Shift Register Latch (SRL)

3-1a. Symbolic representation

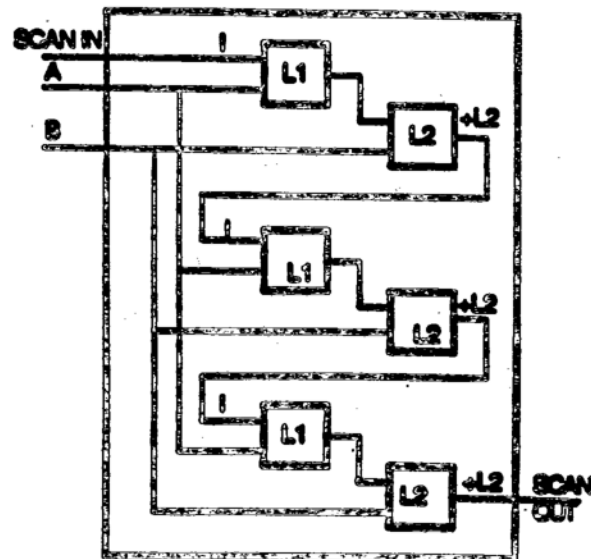


3-1b. Gate level implementation



DESIGN FOR TESTABILITY GUIDELINES

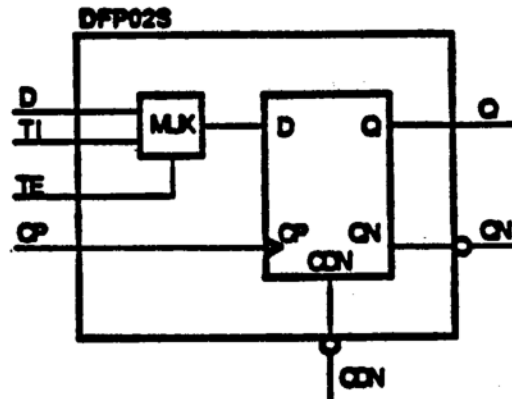
3-1c. Circuit interconnection



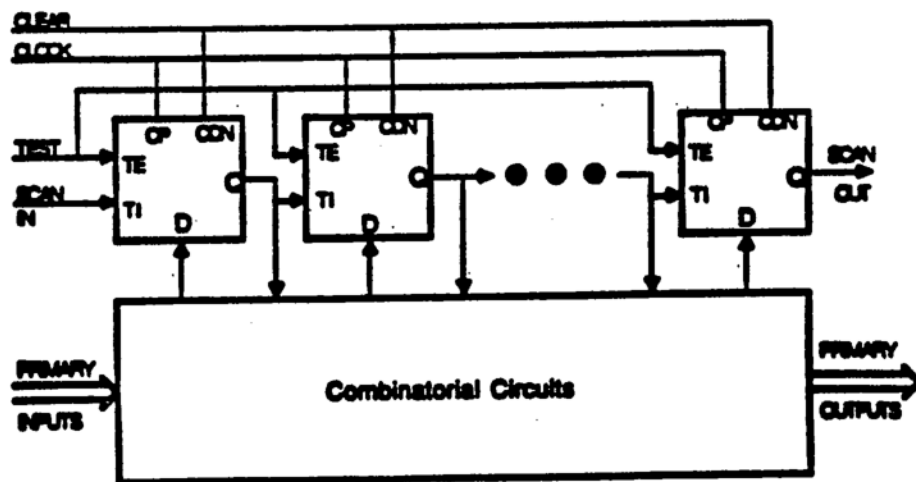
DESIGN FOR TESTABILITY GUIDELINES

Figure 3-2
Scan Path Implementation

3-2a. D Flip-flop with scan



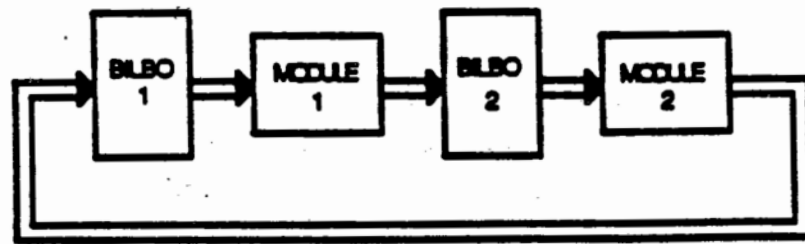
3-2b. Circuit implementation



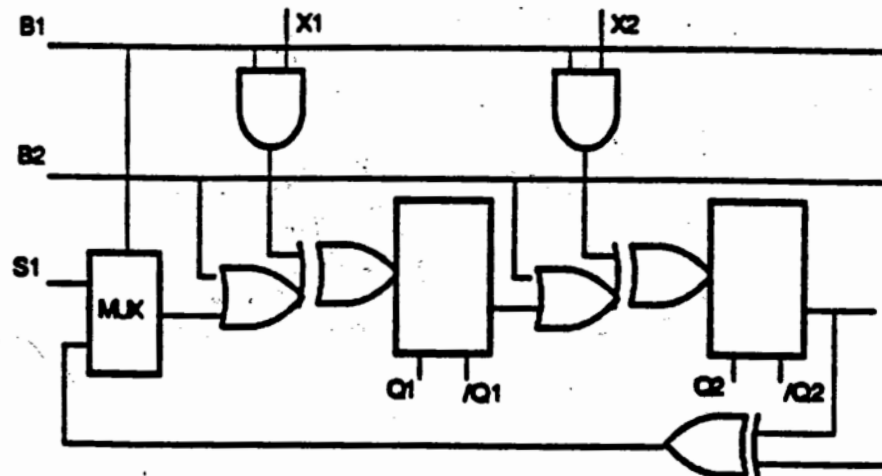
DESIGN FOR TESTABILITY GUIDELINES

Figure 3-3
BILBO Implementation

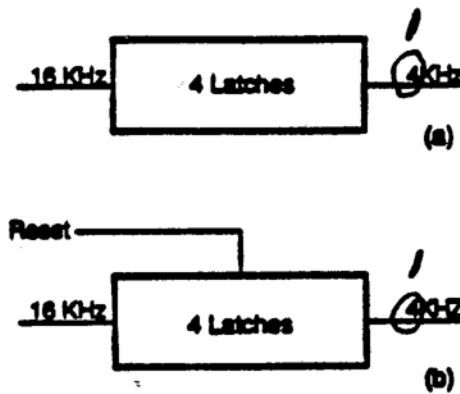
3-3a. BILBO Implementation for combinatorial network



3-3b. BILBO Operating modes



DESIGN FOR TESTABILITY GUIDELINES

Figure 3-4
SET/RESET capabilityFigure 3-5
Asynchronous SR Latch